JSJS

A Strongly Typed Language for the Web

http://jsjs-lang.org

Jain Bahul	bkj2111
Srivastav Prakhar	ps2894
Jain Ayush	aj2672
Sadekar Gaurang	gss2147

Contents

Introduction

JSJS is a strongly typed language for the web. Taking inspiration from languages such Scala, TypeScript and OCaml, the goal of JSJS is to enable programmers to write type-safe, correct and robust code that compiles down to Javascript. Our target users are programmers who like functional nature of Javascript but want the safety and guarantees that come with a statically typed language without sacrificing its ubiquity.

Motivation

> *Any application that can be written in JavaScript, will eventually be written in JavaScript* - **Jeff Atwood**

Javascript is the *lingua franca* of the web. With its humble beginnings in engine of the browser, Javascript has slowly and steadily has gained wide adoption across the developer community at large. Famed to have gone from conception to a working implementation in just 10 days, Javascript now runs on everything - from massive servers to even the smallest Raspberry Pi. However, Javascript's massive success has often been met with wide bewilderment (even disdain in some cases) from developers.

The members of this group, however, believe that Javascript is an incredibly powerful language. Having spent most of our time programming for the web - both frontend and backend, building apps ranging from trivial browser plugins to standalone cross-platform native ones, we marvel that at the fact that a language could have such a widespread use and applicability. The good news that we are not alone in this assessment. The biggest tech giants of today are spending numerous resources in making the language even better. This includes but is not limited to the monumental advances done by the Google folks on V8, the recent Chakra engine by Microsoft and the numerous tools open-sourced by Facebook.

As a language Javascript is quite interesting. With features such as closures, functions as first class objects, asynchronous programming via callbacks and a prototype based system, Javascript can definitely be touted as a modern language. On the other hand, JS is infamous for weird object rules, global name definitions and hard-to-grok prototype chain. The fact that one of the most popular books on JS is titled "Javascript: The Good Parts" indicates the warts in the language.

Our goal with **JSJS** is to create a functional language that has features such as type safety and immutability but at the same time can be used where Javascript runs. Despite Javascript treating functions as first class objects, Javascript is typically used as an imperative programming language. Its lack of structure has encouraged programmers to think of Javascript objects in terms of state, instead of attempting to transform data. Javascript is a product of rapid evolution, and thus for many people from the functional programming school of thought, it seems broken. Although its core library is small, Javascript's weak typing and object construction system create a broken mix, where functions are first class, yet objects are used imperatively. These are some of the issues that we plan to address with JSJS.

In conclusion, we believe that JSJS's list of features and Javascript's ubiquity is a deadly combination that make it compelling tool for programmers.

Features

- **Strong Type Safety** JavaScript is a dynamically typed language, with no real notion of type safety. A variable in JavaScript can be assigned to data of any type and this assignment can be changed through the course of the program. JSJS is be a strongly typed language with static type checking, like OCaml. Types will not be inferred, so the types must be explicitly stated while declaring values and defining functions.
  A **strong type system** is described as one in which there is no possibility of an unchecked runtime

type error and does not permit any implicit type conversion for operations at runtime.
Since JSJS compiles down to JavaScript, which is a dynamically typed language, types in JSJS merely act as annotations for type safety and will be erased during generation of the JavaScript code.

- **Immutable Values** One of the hallmarks of functional programming is immutability of all values used in a program. Javascript is an imperative language, where state can be changed and variables can be reassigned with new values. JSJS will have no concept of variables and all assignments will be treated as values, and hence cannot be changed or reassigned during the running of the program. We have chosen the keyword `val` for assigning values to identifiers to express the concept of all values being immutable more emphatically.
- **Everything is an Expression** Every statement in JSJS is an expression that evaluates to a result of some type, exactly like in OCaml. We will also provide a Unit type, to allow for side-effects such as printing and logging which don't evaluate to an explicit return type.
- **Immutable Data Structures** An immutable or persistent data structure is one in which previous versions of the data structure are preserved when it is modified. Immutable data structures make programs more robust and easy to reason about. Data structures in JavaScript are completely mutable. JSJS will provide immutable implementations for Lists and Maps in the language, with a standard library for functions like `List.hd` and `List.tl`, and `Map.get` and `Map.set`.
- **Polymorphic Types** Support for polymorphic types is another fundamental feature of functional programming. This allows functions to take arguments of generic types and perform some operation on them, without the function being required to explicitly know the underlying types of the argument. For eg., we can have a function that takes as an argument a list of any type, and performs some operation on it. This function can work with both lists of numbers and lists of strings. JSJS supports polymorphism for the built-in data types.
- **Functions as First-Class Objects** Functions in JSJS are first-class objects. Functions can be passed as arguments to other functions, returned by other functions, and can be assigned to variables. JSJS also supports anonymous functions. JavaScript has support for functions as first-class objects, so JSJS provides some syntactic sugar to make their usage sweeter.

Comparison with Javascript

Feature	JSJS	JS
Type Safety	Yes	No
Type Inference	Yes	No
Immutable values	Yes	No
Immutable Collections	Yes	No
Programming Style	Functional	Imperative
Works in Node?	Yes	Yes
Works in Browser?	Yes	Yes

Language Tutorial

Setup

The language has been developed in OCaml which needs to be installed to be able to use the compiler. The best way is to install OPAM(OCaml Package Manager). Using OPAM, OCaml and related packages and libraries can be installed. Follow the below commands for the basic setup.

```
$ opam init
$ opam switch 4.01.0
$ eval `opam config env`
$ opam install core
$ opam install async yojson core_extended core_bench \
   cohttp async_graphics cryptokit menhir
```

In addition, Node js version 4.4.4 or above that supports ES6 is required to be able to run the generated JS code and run the test suite.

After the initial setup, go to the folder where you want JSJS installed and clone the git repository:

```
$ https://github.com/prakhar1989/JSJS.git
```

Using the Compiler

Inside the JSJS folder, type make. This generates the jsjs.out file, the JSJS compiler. The jsjs.out file can be used to compile the JSJS code using the following:

```
$ ./jsjs.out a.jsjs
```

This generates a .js file containing the JavaScript code equivalent of the JSJS code. The JS file produced can be executed like any JavaScript code.

There are no minimum requirements for a JSJS program, like a main function etc. The only requirement is to follow the syntatical conventions of the language(mentioned in the next section). Here is a simple Hello World statement in JSJS.

```
print("Hello World");
$ Hello World
```

Here is a more complicated program that computer GCD of two given numbers. The following programs uses nested if-then-else contruct to support control flow, lambda expressions and also uses recursion.

```
val gcd = /\(a, b) => {
  // nested if-else expressions
  if a == b then a
  else if a > b
  then gcd((a - b), b)
  else gcd((b - a), a);
};

print(gcd(10, 24));
$ 2

print(gcd(14, 21));
$ 7
```

The following program computes the sum of squares of the elements of a list. It uses the list data type and the associated library functions.

```
// let's define a function
val sq = /\(x) => x * x;

// values and functions both
// can be optionally annotated
val sum: num = List.fold_left(

    // lambda expressions
    /\(x, y) => x + y, 0,

    // the List module has
    // a bunch of useful functions
    List.map(sq, List.range(1, 10))
);

// show result
print(sum);
$ 285
```

The following is a code to implement quicksort.

```
val sort = /\(xs: list T): list T => {
    // for an empty list, return the list itself
    if xs == [] then []
    else {
        // collect the elements smaller than head
        val smaller = List.filter(/\(x) => x < hd(xs), tl(xs));

        // collect the elements larger than head
        val greater = List.filter(/\(x) => x >= hd(xs), tl(xs));

        // recursively sort and concat these lists
        List.concat(sort(smaller), hd(xs) :: sort(greater));
    };
};

// let's test it
val sorted = sort([10, 5, 0, 3, 8]);
val sorted_strs = sort(["c", "z", "a", "e", "y"]);

// printing...
print(sorted);
print(sorted_strs);
$ List [ 0, 3, 5, 8, 10 ]
$ List [ "a", "c", "e", "y", "z" ]
```

JSJS is a strongly typed language for the web. Taking inspiration from languages such as OCaml, Scala and TypeScript, JSJS aims to be a pragmatic and a powerful language that can be used to build real-world applications. Since JSJS compiles down to Javascript, it can run both in the browser and on the server (Node.js). While designing the syntax and semantics of the language our goal has been the following -

- Concise, yet expressive
- Approachable to Javascript users

- Familiar to functional programmers
- Explicit is better than implicit
- Better understandability and readability of code

Comparison with Javascript

Literals

JS	JSJS
`3`	`3`
`3.1415`	`3.1415`
`Hello World!`	`Hello World!`
`'Hello world!'`	Cannot use single quotes for strings
`true`	`true`
`[1,2,3]`	`[1,2,3]`
`{ foo : 1, bar : 2 }`	`{ foo : 1, bar : 2 }`

Strings

JS	JSJS
`'abc' + '123'`	`abc ^ 123`

Maps

JS	JSJS
`map[foo],` `map[bar] = 2,`	`get(map, foo),` `set(map, bar, 2),`

Functions

JS

```
1. function(x,y) { return x + y, }
2. List.map(function(x) { return x*x, }, [1,2,3,4]),
3. Math.max(3, 4)
4. var filter = function(f, xs) { ... }
```

JSJS

```
1. /\(x : num, y : num) : num => x+y,
2. List.map((/\(x : num): num => x * x), [1,2,3,4]),
3. Math.max(3, 4)
4. val filter = /\(f: (num) -> bool, xs: list num): list num => { ... }
```

Assignments

Below are the assignments

JS	JSJS
`var x = 10`	`val x = 10,`
`var x = 'foo',`	`val x = foo,`
`var x = true,`	`val x = true,`
`var x = [1,2,3],`	`val x = [1,2,3],`
`var x = { foo : 1, bar : 2 },`	`val x = {foo: 1,bar: 2 },`
`var x = function(x, y) { return x+y, }`	`val sq = /\(x) => x * x,`

Control Flow

JS	JSJS
`if(x > y) { return x, }`	NA, else construct necessary
`if(x > y) { return x, } else { return y, }`	`if x > y then x else y,`
`return 42,`	No return statements, only expressions

Types

Types are at the forefront of JSJS. Every value declaration, functions, and even compound literals need an explicit type definition (although the user will not have to explicit annotate type information in code, it is automatically inferred by the compiler). This section elaborates more on the different types of data that JSJS supports.

Primitive Types

There are four fundamental types or *primitive* types in JSJS.

Number

The `num` or the `number` type in JSJS corresponds to the Number type in Javascript.

According to the ECMAScript standard, there is only one number type: the double-precision 64-bit binary format IEEE 754 value (number between -(253 -1) and 253 -1). There is no specific type for integers and hence the same type is used to represent floating-point numbers.

String

The `string` type in JSJS is used to represent textual data and corresponds to the String type in Javascript. It is a set of elements of 16-bit unsigned integer values. Each element in the String occupies a position in the String. The first element is at index 0, the next at index 1, and so on. The length of a String is the number of elements in it.

Boolean

The `bool` type in JSJS represents a logical entity and can have two values: `true`, and `false`.

Unit

Unit, written `unit`, is a built-in type that has only one value, i.e. `(-)`. It is mostly used for functions that causes side effects and have no useful return value. In the translated javascript code, a `unit` is converted to `undefined`.

Unit is also used for interopability with functions that have no return value at all. For example, the JSJS function below would have the type `string -> unit` in JSJS.

```
val nothing = /\(x: string): unit => print(x)
```

The literal `unit` has the type `unit`.

Composite Types

There are two primary composite types in JSJS

Lists

Like other functional programming languages, Lists are the fundamental data structures in JSJS. They serve as the primary basis of storing one or more related values together. The type signature of a list is `list T` where `T` is one of the other types - either primitive or composite. Here are a few examples of defining list types -

- `list num`
- `list string`
- `list list bool`

Lists in JSJS are homogenous, i.e they can contain only one type of data. A list whose type is declared as `list num` can only store elements of `num` type. Lists can also contain other lists which have a type `list list T`.

Maps

Maps is another important data structure in JSJS and is treated as a first-class citizen. Since JSJS compiles down to Javascript where Maps (called objects) are used extremely liberally, Javascript programmers will feel right at home in accessing this useful data structure with as much ease as they are used to. Map types follow a different syntax for declaration: `<T: U>` where `T`is the type of the key and `U` is the type of the value stored in the Map. Example declarations of Maps are -

```
// simple maps
val names : <string: num> = { ... }
val friends : <string: list num> = { ... }

// nested maps
val people : <string: <string: bool>> = { ... }
```

Like lists, Maps in JSJS are homogenous. Like other strongly typed languages, the keys of a map should have the same type and so should the values.

Function Types

Like other values, functions are expressions in JSJS. That means that functions also have types and function expression also have a function type. The type of a function, is mapping of types from its formal arguments to its return type. Thus a function that takes `n` arguments has the following type signature - `(T1, T2, T3, ... Tn) -> T`.

```
// type of a function that takes two arguments
// of type num and returns a bool type.
(num, num) -> bool
```

The type of a function is determined at the time of its declaration. Each formal argument in a function definition should have a type attached followed finally by the return type of the function. If the types are not annotated explicitly by the user, the compiler automatically sets a type to the formal arguments and the return type of the function.

```
// a function declaration with explicit type annotations
val pow : (num, num) -> num = /\(x: num, y: num): num = {
   // ...
}
```

Parametric Types

JSJS also supports polymorphic types. Polymorphic function types can contain of type variables. These are like placeholders for the types used when applying the polymorphic function. A type variable has to be defined by an single uppercase alphabet.

```
val map = /\(f: T -> U, xs: list T): list U => {
 // ...
}
```

Type Declarations

Type annotations of all expressions are completely optional. A JSJS program can be written completely without any type declarations anywhere and will still work perfectly with full type safety. Any expression with undefined type information is annotated with a unique parametric type by the compiler in the beginning. These types are then resolved by inferring the type from the rest of the program.

Examples

```
// explicit type definition
val name : string = Foobar,

// types are optional and count is assigned the `num` type.
val count = 10 + 20,

// wrong type annotations will raise type mismatch errors
val happy? : bool = string1 ^   ^ string2,

// functions with explicit type declaration
val square = /\(x: num, y: num): num => x * y,
```

```
// completely unannotated and type-inferred functions.
val prod = /\(x, y) => x * y,
```

Thus, the type system in JSJS is much more strict and explicit that javascript while maintaining the conciseness in code.

Types in AST

In conclusion, the types of JSJS are defined in the AST as below -

```
type primitiveType =
  | T of string
  | TExn
  | TAny
  | TNum
  | TString
  | TBool
  | TUnit
  | TFun of funcType
  | TList of primitiveType
  | TMap of primitiveType * primitiveType
and funcType = primitiveType list * primitiveType
,,
```

Lexical Conventions

Comments

Only single-line comments are allowed in JSJS. Anything followed by `//` on the line will be considered as a comment and will be ignored by the compiler.

Example:

```
// This is a comment

......... // This is a comment too
```

Identifiers

Identifiers are sequences of characters used for naming JSJS entities. All identifiers cannot have the same spelling (character sequence) as a JSJS keyword, JS keyword, or a boolean literals, or else a compile-time error occurs. Lowercase letters and uppercase letter are distinct, such as `isEmpty?` and `isempty?` are two different identifiers.

Value and Function Identifiers

Valid identifier characters for values and functions include ASCII letters, decimal digits, underscore character and the '?' character. The first character must be a small case alphabetic character. The '?' character can

only be used as the last character of the identifier. An identifier can also be named as _ which basically discards the value once evaluated.

Regular Expression:

```
id = ['a'-'z'] ['a'-'z' 'A'-'Z' '0'-'9' '_']* ['?']? | "_"
```

Example:

```
// Valid Identifiers for values and functions
_
x
age
totalAmount
total_amount
isEmpty?
person1

// Invalid Identifiers for values and functions
X
1person
isEmpt?y
&name
Person
```

Module Identifiers

Valid identifier characters for modules include only ASCII letters. The first character must be an upper case alphabetic character.

Regular Expression:

```
module = ['A'-'Z'] ['a'-'z' 'A'-'Z']+
```

Example:

```
// Valid Identifiers for modules
List
StringMap
HashSet

// Invalid Identifiers for modules
stringMap
hash_set
&list
```

Keywords

Keywords are special identifiers reserved for use as part of the programming language itself. Since JSJS code compiles down to Javascript eventually, keywords of Javascript also behave as keywords for JSJS.

You cannot use them for any other purpose. JSJS recognizes the following words as keywords:

\

break	bool	case	class	catch	const	continue
debugger	default	delete	do	else	export	extends
false	finally	for	function	if	import	in
instanceof	list	new	num	return	string	super
switch	this	throw	try	typeof	var	void
while	with	yield	val	then	true	unit

Separators

A separator separates tokens. Separators themselves are simply single-character tokens.

Character	Token
'('	{ LPAREN }
')'	{ RPAREN }
'{'	{ LBRACE }
'}'	{ RBRACE }
'['	{ LSQUARE }
']'	{ RSQUARE }
','	{ SEMICOLON }
','	{ COMMA }
'.'	{ DOT }

Literals

A literal is a source code representation of a value of a primitive type or a composite type.

Number Literals

A number literal has the following parts: a whole-number part, an optional decimal point (represented by an ASCII period character), and a following fraction part. The whole number and fraction parts are defined by a single digit 0 or one digit from 1-9 followed by more ASCII digits from 0 to 9.

Regular Expression:

```
num : ['0'-'9']+ '.'? ['0'-'9']*
```

Example:

```
// Valid number literals
4
4.5
0.0002
42.

// Invalid number literals
.7
1e+3
```

Boolean Literals

The boolean type has two values, represented by the boolean literals true and false, formed from ASCII letters.

Regular Expression:

```
bool : true | false
```

String Literals

A string literal is represented as a sequence of zero or more ASCII characters enclosed in two double quotes. The following characters are represented with an escape sequence, which consists of a backslash and another character:

- \ - backslash
- " - double-quotes
- \n - new line
- \r - carriage return
- \t - tab character

Regular Expression:

```
string : ([' '-'!' '#'-'[' ']'-'~'] | '\\' ['\\' '' 'n' 'r' 't'])*
```

Unit Literal

The unit literal has only one possible value, i.e. `(-)`.

Operators

The following operators are reserved lexical elements in the language. See the expression and operators section for more detail on their defined behavior

Character	Token
'+'	{ PLUS }
'-'	{ MINUS }
'*'	{ MULTIPLY }
'/'	{ DIVIDE }
'%'	{ MODULUS }
'^'	{ CARET }
'<'	{ LT }
'<='	{ LTE }
'>'	{ GT }
'>='	{ GTE }
'=='	{ EQUALS }
'='	{ ASSIGN }
'!'	{ NOT }
'&&'	{ AND }
'||'	{ OR }
'::'	{ CONS }

Functions

All functions in JSJS are Lambda expressions. Since functions are treated as first class citizens, these lambda expressions can be assigned as values to identifiers, passed as arguments to other functions, and returned as values from other functions. To make a named function declaration, a lambda expression is assigned to an identifier using the `val` keyword, just like any other type declaration.

Lambdas or Function Literals

Lambda expressions are denoted by the symbol `/\`, which resembles the upper case Greek letter Lambda.

JSJS Lambda expressions have the following form:

```
/\(argument declarations if any) : return type => {
  Block of expressions, the last of which
  is the value that is returned
};
```

A shorthand syntax is also supported if the body of the Lambda is a single statement:

```
/\(argument declarations, if any) : return type => expression;

// even more concise
/\(argument declarations, if any) => expression
```

The argument declarations *can* be annotated with their types, and a return type of the body of the `/\` expression also *can* be specified but both are optional. For example, the following / expression takes a single number `x` as an argument and evaluates the square of `x`.

```
/\(x : num) : num => x * x;
```

It is also possible to define Lambda expressions that don't take any arguments:

```
/\() : unit => println(hello world);
```

or those that take multiple arguments:

```
// notice type of lname and return type of the /\-expression has been omitted.
/\(fname : string, lname) => Hello  ^   ^ fname ^   ^ lname;
```

The body of a Lambda can also be a block of expressions, where the last expression is the one that is implicitly evaluated and returned. The following `/\` takes a single numeric argument `x` and adds the value `y` - assigned as 10 in the body to `x`.

Example:

```
/\(x : num) : num => {
  val y = 10;
  x + y;
};
```

Function Calls and Usage

Functions are called by invoking the name of the function and passing it actual arguments. These arguments can be any kind of expressions. Every function call is itself an expression, and evaluates to a value which has a type (the return type of function).

Example:

```
// function called with a numeric literal
val sq_5 = sq(5);

// function called with an expression
val x = 8;
val y = sq(x);
val z = cube(x + y);

// function called with a function as an argument
val addOne = /\(x) => x + 1;
val addOneSqr = /\(f, g, x) => f(g(x));
val result = addOneSqr(sq, addOne, 8);
```

Operators

JSJS supports various operators for different data types. Broadly, JSJS includes Arithmetic Operators, Relational Operators, Boolean operators, Assignment Operator and String Operator. While most of these are binary operators, some are unary.

```
  | Binop of expr * op * expr
  | Unop of op * expr
```

The above code excerpt defines two types of expressions. The former defines a binary operator while the latter gives the format of a unary operator.

Arithmetic Operators

JSJS supports the following arithmetic operators: `+, -, *, /, %.`. All arithmetic operators require two operands of `num` data types. These can be either literals or variables or a combination of the two.

Addition (left associative)

```
10 + 7 // -> 17
x + y
```

Subtraction (left associative)

```
10 - 7 // -> 3
x - y
```

Multiplication (left associative)

```
10 * 7 // -> 70
x * y
```

Division (left associative)

```
21 / 7 // -> 3
x / y
```

Modulus (left associative)

```
10 % 7 // -> 3
x % y
```

Negation (left associative)

```
-5 // -> -5
- (5 + 4) // -> -7
```

Relational Operators

The following relational operators are supported by JSJS: `==`, `!=`, `>=`, `<=`, `>`, `<`. Relational operators require two operands. These operands can be of any type as long as they are of the same type. These expressions always return a value of boolean data type. Comparisons between Lists and Maps (w.r.t their keys) are done similar to strings, i.e. lexicographically.

Equals (left associative)

```
5 == 5 // -> true
abc == def // -> false
true == false // -> false
```

Not Equals (left associative)

```
5 != 5 // -> false
abc != defn // -> true
true != false // -> true
```

Less than (left associative)

```
5 < 5 // -> false
abc < defn // -> true
true < false // -> false
```

Less than or Equals (left associative)

```
5 <= 5 // -> true
abc <= defn // -> true
true <= false // -> false
```

Greater than (left associative)

```
5 > 5 // -> false
abc > defn // -> false
true > false // -> true
```

Greater than or Equals (left associative)

```
5 >= 5 // -> true
abc >= defn // -> false
true >= false // -> true
```

Boolean operators

JSJS supports three boolean operators: `&&, ||, !`. `&&` and `||` are binary operators whereas `!` is a unary operator. These act on boolean data types only and return a single value of type boolean.

And (left associative)

```
true && false // -> false
true && true // -> true
false && false // -> false
```

Or (left associative)

```
true || false // -> true
true || true // -> true
false || false // -> false
```

Not (left associative)

```
!true // -> false
!false // -> true
```

Assignment Operator

The assignment operator is used to assign a value to an identifier. The value of the expression on the right side is evaluated and assigned to the identifier on the left hand side, thus it has right associativity.

Example:

```
val x : num = 5 + 3;
val y : string = abc ^ def;
val z : bool = true;
val sq = /\(x) => x * x;
```

String Concatenation Operator

JSJS includes an operator for strings as well. The `^` operator is the string concatenation operator, which takes two strings and returns an output string formed by the concatenation of the two. It is left associative.

Example:

```
abc ^ defg // -> abcdefg
x ^ y
```

Cons Operator

This `::` operator is provided to append a value in the beginning of a list. The operand on the right of `::` has to be of type `list T` whereas the operand on the left of the operator has to be of type `T`. The result of this expression will also be a list of type `list T`. It is right associative.

Example:

```
1 :: [2,3,4,5] // -> [1,2,3,4,5]
1 :: 2 :: 3 :: [4,5] // -> [1,2,3,4,5]
```

Operator precedence

JSJS defines a precedence order in which operations are performed when more than one operators are present in a single expression. The operators sharing the same precedence are evaluated according to their associativity. Operators which are left associative are evaluated from left to right. Similarly, right associative operators are evaluated from right to left. In JSJS, all operators are left associative except for the assign(=) operator. Following is the chart of operator precedence.

Operator Precedence	
`- (negation)`	highest precedence
`*, /, %`	2
`+, -`	3
`<=, >=, <, >, ==, !=`	4
`!`	5
`&&`	6
`||`	7
`^`	8
`::`	9
`=`	lowest precedence

Expressions

Everything in JSJS is an expression. Accordingly, an entire JSJS program can be defined as a list of expressions.

All these expressions are composed of identifiers, operators, literals and function calls. Following is an exhaustive list of different types of expressions in JSJS:

```
type program = expr list;;

type expr =
  | UnitLit
  | NumLit of float
  | BoolLit of bool
  | StrLit of string
```

```
  (* binary operation *)
  | Binop of expr * op * expr
  (* unary operation *)
  | Unop of op * expr
  (* a list literal is a list of expressions *)
  | ListLit of expr list
  (* a map literal is a list of key-value pairs *)
  | MapLit of (expr * expr) list
  (* a block is a list of expression *)
  | Block of expr list
  (* an assignment expression takes a string,
     an annotated type and an expression *)
  | Assign of id * primitiveType * expr
  (* a value is just a id *)
  | Val of id
  (* if-then-else takes 3 expressions - predicate,
     then expr and else expr *)
  | If of expr * expr * expr
  (* a function call takes fn name and a list of arguments (exprs) *)
  | Call of expr * expr list
  (* a fn literal is of func_decl record *)
  | FunLit of id list * expr * primitiveType
  (* a module literal is a module name and expression *)
  | ModuleLit of id * expr
  (*  an exception type that is generated by throw expr *)
  | Throw of expr
  (* a try catch block has two exprs and an identifier that acts as
     a placeholder for the message *)
  | TryCatch of expr * id * expr
;;
```

Primary Expressions

All JSJS expressions are formed by composing the following base expressions:

1. Identifiers
2. Literals
3. Constants

List Literals

A list literal is represented by comma separated expressions that evaluate to literals of the same type enclosed within square brackets.

Example:

```
// Literal for a list of numbers
[1,2,3,4,5,6]

// Literal for a list of strings
[jsjs, is, awesome, !!]
```

```
// Literal for a list of bools
[1 == 1, 2 == 3, 5 <= 4, !true]
```

Map Literals

A map literal is represented by comma separated key-value pairs that are enclosed within curly braces. A key can only be expressions of number, string or a bool type, while values can be expressions of any type.

Example:

```
// A map literal with key as a number and value a string.
{ 1: One, 2: Two, 3: Three, 4: Four }

// A map literal with key as a number and value as a list of strings.
{ 1: [One, Uno], 2: [Two, Dos], 3: [Three, Tres] }
```

Blocks

A block is a list of multiple expressions enclosed in curly braces. The entire block is executed in the order in which expressions appear and the value of the last expression is returned. In JSJS, blocks can be declared inside of `if-then-else`, `try-catch`, and `lambda` expressions only. Hence, blocks are not first-class citizens in JSJS. Using blocks elsewhere would lead to parser errors. Also the last statement of a block cannot be Assignment expression since it does not make any sense semantically.

Example:

```
val block = /\(w) => {
  val x = w + 2;
  val y = x * 2;
  val z = y + 1;
  z / 2;
};
```

If-Then-Else

`if-then-else` behaves similarly to the standard control flow constructs of programming languages. One important point to keep in mind is that the return type of then and else blocks should be same. Otherwise the compiler will throw a type mismatch error.

Example:

```
if x > y
then { print(x); x; }
else { print(y); y; };
```

The curly braces in then and else blocks are required only if the blocks consist of more than one expression but are optional otherwise. They are accordingly handled in the grammar as well.

Throw

A throw expression comes extremely handy in cases when no known value can be returned. Execution of the current function will stop (the expressions after throw won't be executed), and control will be passed to the first catch block in the call stack. If no catch block exists among caller functions, the program will terminate.

The throw expression uses the `throw` keyword followed by only a string. The compiler annotates throw expressions with a special type `TExn` that bypasses the type checking making it usable everywhere.

Example:

```
val divide = /\(numer, denom) => {
  if denom == 0
  then throw "Divide by zero exception."
  else numer / denom;
};
```

Try-Catch

The try statement allows you to define a block of code to be tested for errors while it is being executed. The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

In JSJS `try-catch` expressions, the block that is being tested for errors in enclosed in a block followed by the `try` keyword. This is followed by the `catch` keyword, an identifier for the error message emitted from the `try` block, and the block of code that is to be executed, if an error occurs in the `try` block. Since the entire `try-catch` construct is an expression, it should have a return type and value. The return type of the `try-catch` expression is the same as the type of the `catch` block and the type of the `try` block should be same as that of the `catch` block.

Example:

```
val divide = /\(numer, denom) => {
  if denom == 0
  then throw "Divide by zero exception."
  else numer / denom;
};

try {
  divide(2, 0);
  (-);
} catch (msg) {
  print(msg);
};
```

Module Access

Currently in JSJS, it is not possible to create user defined modules, but the standard library functions are all defined in Modules. To access these functions we use the name of the module followed by the special `.` dot operator and then the name of the function/property. The return type of this expression will be same as that of the function / property being accessed.

Example:

```
// <module_name>.<func / property type>
List.length(arr);
```

Scoping Rules

Unlike the convoluted scoping rules of Javascript, JSJS enforces scoping rules that gives a clear structure to the program and makes writing understandable and readable code. Here are some of the key scoping rules adhered to in JSJS:

Any value defined in the program is only available following its definition not before it.

```
//.....
// y is not available Here
val y = 10;
// y is available from here on
```

Once a name has been assigned to a value, it cannot be redefined in the same scope. This goes well with the immutable nature of the language. If you cannot mutate a variable then why redefine it with some other value.

```
val x = 10;

// this is not allowed
val x = 5;
```

Any value defined in a scope can only be accessed in the same scope or in a nested scope.

```
// x is not available here
val _ = /\() => {
  val x = 10;
  // x is available here
}
// x is not available here again
```

A value name can be shadowed in a nested scope by defining a value with the same name.

```
val x = "outer";

val res = /\() => {
  print(x); // prints "outer"
  val x = "inner";
  print(x); // prints "inner"
};
print(x); // prints "outer"
```

Standard Library Functions

JSJS provides a simple set of standard library functions to perform basic operations on lists, maps and work with standard input/output. By default, all the following libraries are automatically available.

Top Level Functions

The following functions are available in the global scope of all JSJS programs.

Function	Type	Description
`print`	`(T) -> unit`	prints the argument to console
`print_num`	`(num) -> unit`	prints a numeric argument to console
`print_string`	`(string) -> unit`	prints a string argument to console
`print_bool`	`(bool) -> unit`	prints a boolean argument to console
`hd`	`(list T) -> T`	returns the head element of a list
`tl`	`(list T) -> list T`	returns the list without its head element
`empty?`	`(list T) -> bool`	determines if a list is empty or not
`get`	`(<T:U>, T) -> U`	returns a value associated with a key in a map
`set`	`(<T:U>,T,U) -> <T:U>`	sets a key-value pair in a map
`del`	`(<T:U>, T) -> <T:U>`	deletes a key from the map
`keys`	`(<T:U>) -> list T`	returns the list of keys from a map

List Module

The `List` module provides essential functions to do basic operations on lists:

Function	Type	Description
`length`	`(list T) -> num`	returns the number of elements in a list.
`nth`	`(list T,num) -> T`	returns the nth (0-based indexing) element of a list
`rev`	`(list T) -> list T`	returns the list reversed
`filter`	`((T -> bool), list T) -> list T`	returns a list with all elements that satisfy a given predicate.
`map`	`((T -> U), list T) -> list U`	returns a list with the given function applied to every element of the list.
`iter`	`((T -> unit)), list T -> unit`	applies a function that causes side-effects on every element of a list.
`range`	`((num, num) -> list num`	creates a list with elements from a given starting point and ending point in unit steps
`fold_left`	`((T, U) -> T, T, list U) -> U`	reduces the list using the given function and an accumulator
`concat`	`(list T, list T) -> list T`	concatenates two lists
`insert`	`(list T, T, num) -> list T`	inserts an element in the list given the position (0-based indexing)
`remove`	`(list T,num) ->list T`	removes an element from the list at the given position (0-based indexing)
`sort`	`((T,T)->bool, list T) -> list T`	sorts the list based on the comparator.

Map Module

The `Map` module provides essential functions to do basic operations on maps:

Function	Type	Description
`count`	`(<T:U>) -> num`	returns the number of elements in a list.

Function	Type	Description
`values`	`(<T:U>) -> list U`	returns the list of values in the map.
`merge`	`(<T:U>,<T:U>)-> <T:U>`	merges two maps, duplicate keys from second map are ignored.

Project Planning

Planning Process

Our motto throughout the project has been to work early and work consistently. Fortunately, Friday turned out to the be the day of week where none of the team members had a class so we set that day aside to work on the project. This also fit in well with the meetings scheduled with our TA, David each Monday so that we could share the progress we made over the weekend. As far tools are concerned, we used Github and Slack extensively for planning. All docs that required collaboration were written in Github wiki's, whereas useful lists of resources, documents etc were shared on Slack. Lastly, to track progress we frequently maintained minutes of the meeting and created issues (tagged as feature requests) for anything we discussed that required either immediate or later work.

Specification

Since the initial version of JSJS, languages like OCaml and Scala has served as primary inspiration. As a result, a lot of features and design ideas have been taken from these languages. The specification for the set of features we had originally planned primarily included a high-level idea of the features we want to support and the syntax of our language. The specification was built iteratively and we did not wait for it to be complete before we started to code. Our final concrete specification was laid out in the LRM which we referred out for guidance. Whenever the language started to diverge from the LRM, we ensured that the LRM was kept updated at all times.

Development and Testing

The development process followed the stages of the compiler. Our goal was to finish the scanner and parser quickly, so that the type-checker and the codegen could be worked on. Once we had a whole pipeline ready, we built each feature from end to end, ie. from AST to codegen. We also placed tests front and center in our development process and coupled every feature with a set of accompanying test cases comprising of valid executable programs and error messages.

Project Planning

Planning Process

Our motto throughout the project has been to work early and work consistently. Fortunately, Friday turned out to the be the day of week where none of the team members had a class so we set that day aside to work on the project. This also fit in well with the meetings scheduled with our TA, David each Monday so that we could share the progress we made over the weekend. As far tools are concerned, we used Github and Slack extensively for planning. All docs that required collaboration were written in Github wiki's, whereas useful lists of resources, documents etc were shared on Slack. Lastly, to track progress we frequently maintained minutes of the meeting and created issues (tagged as feature requests) for anything we discussed that required either immediate or later work.

Specification

Since the initial version of JSJS, languages like OCaml and Scala has served as primary inspiration. As a result, a lot of features and design ideas have been taken from these languages. The specification for the set of features we had originally planned primarily included a high-level idea of the features we want to support and the syntax of our language. The specification was built iteratively and we did not wait for it to be complete before we started to code. Our final concrete specification was laid out in the LRM which we referred out for guidance. Whenever the language started to diverge from the LRM, we ensured that the LRM was kept updated at all times.

Development and Testing

The development process followed the stages of the compiler. Our goal was to finish the scanner and parser quickly, so that the type-checker and the codegen could be worked on. Once we had a whole pipeline ready, we built each feature from end to end, ie. from AST to codegen. We also placed tests front and center in our development process and coupled every feature with a set of accompanying test cases comprising of valid executable programs and error messages.

Test Plan

Testing has been a part of our development process since day one. In the below sections, we what are test suite looks like and how we do automation -

Unit Tests

The **Scanner** and **Parser** portions of our compiler have been unit-tested. This was done so that we could be sure that the grammar codified in `parser.mly` was correct and outputted valid token on entering expression. To test this aspect, we used **Menhir**'s `--interpret` flag to help us in testing. We created a Python program that takes a filename of tokens and generates a parse tree as outputted by Menhir.

Integration Tests

Once we started work on the remaining portions of the compiler (typechecker and codegen), we started writing integration tests to test the compiler end-to-end. Though we did not follow test-driven development in the usual style, we followed every feature commit with a thorough set of test cases. At the time of this writing, our compiler has 110 test cases which check each aspect of the compiler.

The following is a list of test-cases that we have written. As it can be seen, the names of the files have been kept descriptive for each feature they stand for. The sections below give a rough outline of a few test cases that were added in the test suite. The complete list of all the test cases are given below -

Literals

```
- declaration of literals of all types.
- assignment of literals.
- ...
```

Functions

```
- function declaration.
- passing functions as arguments.
- function calls.
- recursive function calls.
- generic functions
-  ...
```

Type System

```
- annotating types explicitly
- incorrect number of function arguments check.
- incorrect type of function arguments.
- type check of primary type expressions.
- type check of composite type expressions.
-  ...
```

Maps

```
- Adding an element gives a new map
- Only a key of valid type should be allowed
- Deleting the key from the map gives a new map
- ...
```

List

```
- List standard library functions work as expected
- List concat works only with correct types
- List equality on lists containing the same elements
- ...
```

Automation

The test suite is automated and is written in OCaml. It works by simply looking at format of the `jsjs` file, compiles it and finally runs it with Node.js. If the file starts with `pass` the test-runner expects the code to compile and produce output as given in the corresponding `.out` file. Instead, if the test starts with `pass`, the file is expected to not compile and generate a warning message as given in the corresponding output file.

Here's a sample output of the test-cases -

```
Test Summary
-------------------------
All testcases complete.
Total Testcases : 110
Total Passing   : 110
Total Failed    : 0
Execution time  : 8.023073s
```

Lastly, we also have continuous integration with Travis CI setup that automatically checks and runs all our test cases whenever a commit is pushed or a pull-request is opened on our repository.

Testcases

A list of all test-cases

```
make run-test

    pass-unit_equality2.jsjs
    pass-unit_equality.jsjs
    pass-unaryops.jsjs
    pass-semi_annotated_map.jsjs
    pass-semi_annotated_function.jsjs
    pass-scoping2.jsjs
    pass-scoping.jsjs
    pass-moreexception.jsjs
    pass-modulenamescoping.jsjs
    pass-math.jsjs
    pass-mapvalues.jsjs
    pass-mapmerge.jsjs
    pass-mapinit.jsjs
    pass-mapequality1.jsjs
    pass-mapcount.jsjs
    pass-map_set.jsjs
    pass-map_keys_values.jsjs
    pass-map_keys.jsjs
    pass-map_has.jsjs
    pass-map_get.jsjs
    pass-map_del.jsjs
    pass-map_booleanops.jsjs
    pass-listequality3.jsjs
    pass-listequality1.jsjs
    pass-list_sort_str.jsjs
    pass-list_sort_num.jsjs
    pass-list_sort_custom_comp.jsjs
    pass-list_rev.jsjs
    pass-list_remove.jsjs
    pass-list_range.jsjs
    pass-list_nthexception.jsjs
    pass-list_map.jsjs
    pass-list_length.jsjs
    pass-list_iter.jsjs
    pass-list_insertexception.jsjs
    pass-list_insert.jsjs
    pass-list_foldleft.jsjs
    pass-list_filter.jsjs
    pass-list_concat.jsjs
    pass-list_booleanops.jsjs
    pass-ifelse.jsjs
    pass-identityfn.jsjs
    pass-helloworld.jsjs
    pass-generic_annotated_map.jsjs
    pass-generic_annotated_list.jsjs
    pass-generic_annotated_function.jsjs
    pass-function_args.jsjs
    pass-fn_compose.jsjs
```

```
pass-exception.jsjs
pass-empty_list.jsjs
pass-disposablevar.jsjs
pass-cons.jsjs
pass-booleanprecedence.jsjs
pass-booleanops2.jsjs
pass-booleanops.jsjs
pass-assign2.jsjs
pass-anonymousfnapp.jsjs
pass-annotated_type.jsjs
fail-unaryops.jsjs
fail-scoping_order.jsjs
fail-recursivelistfnapp.jsjs
fail-recursivefnapp.jsjs
fail-recursive3.jsjs
fail-recursive2.jsjs
fail-recursive.jsjs
fail-mapvalues.jsjs
fail-mapmerge.jsjs
fail-mapinit.jsjs
fail-map_value_type2.jsjs
fail-map_value_type.jsjs
fail-map_set2.jsjs
fail-map_set1.jsjs
fail-map_key_type2.jsjs
fail-map_key_type.jsjs
fail-map_has.jsjs
fail-map_get.jsjs
fail-map_del.jsjs
fail-map_booleanops.jsjs
fail-listequality2.jsjs
fail-list_sort_comptype.jsjs
fail-list_sort_comp.jsjs
fail-list_lit.jsjs
fail-list_insert.jsjs
fail-list_foldleft.jsjs
fail-list_filter.jsjs
fail-list_concat.jsjs
fail-list_booleanops.jsjs
fail-js_keyword.jsjs
fail-invalid_return.jsjs
fail-invalid_module.jsjs
fail-incorrect_block_usage.jsjs
fail-ifelse3.jsjs
fail-ifelse2.jsjs
fail-ifelse1.jsjs
fail-genericfns.jsjs
fail-faulty_iter.jsjs
fail-equality_type.jsjs
fail-duplicate_vars_in_functions.jsjs
fail-duplicate_variables.jsjs
fail-duplicate_args.jsjs
fail-disposablevar.jsjs
fail-declare_keyword.jsjs
```

```
      fail-cons.jsjs
      fail-booleanops.jsjs
      fail-assign3.jsjs
      fail-assign1.jsjs
      fail-anonymousfnapp2.jsjs
      fail-anonymousfnapp.jsjs
      fail-annotated_num.jsjs
      fail-annotated_lists.jsjs

    Test Summary
    -------------------------
    All testcases complete.
    Total Testcases : 110
    Total Passing   : 110
    Total Failed    : 0
    Execution time  : 8.123073s
```

The testing scripts used for unit and integration testing are copied in the appendix.

Source to Target

GCD

Below is a example codegen output of the GCD program written in JSJS.

JSJS ->

```
val gcd = /\(a, b) => {
  if a == b then a
  else {
    if a > b
    then gcd((a - b), b)
    else gcd((b - a), a);
  };
};

print_num(gcd(12, 8));
```

Codegen ->

```
'use strict'

var Immutable = (this,function(){"use strict";function
t(t,e){e&&(t.prototype=Object.create(e.prototype)),t.prototype.constructor=t}fun
ction e(t){return o(t)?t:O(t)}function r(t){return u(t)?t:x(t)}function
n(t){return s(t)?t:k(t)}function i(t){return o(t)&&!a(t)?t:A(t)}function
o(t){return!(!t||!t[ar])}function u(t){return!(!t||!t[hr])}function
s(t){return!(!t||!t[fr])}function a(t){return u(t)||s(t)}function
h(t){return!(!t||!t[cr])}function f(t){return t.value=!1,t}function
c(t){t&&(t.value=!0)}function _(){}function p(t,e){e=e||0;for(var
r=Math.max(0,t.length-e),n=Array(r),i=0;r>i;i++)n[i]=t[i+e];return n}function
v(t){return void 0===t.size&&(t.size=t.__iterate(y)),t.size}function
l(t,e){if("number"!=typeof e){var r=e>>>0;if(""+r!==e||4294967295===r)return
```

```
NaN;e=r}return 0>e?v(t)+e:e}function y(){return!0}function
d(t,e,r){return(0===t||void 0!==r&&-r>=t)&&(void 0===e||void
0!==r&&e>=r)}function m(t,e){return w(t,e,0)}function g(t,e){return
w(t,e,e)}function w(t,e,r){return void 0===t?r:0>t?Math.max(0,e+t):void
0===e?t:Math.min(e,t)}function S(t){this.next=t}function z(t,e,r,n){var
i=0===t?e:1===t?r:[e,r];return n?n.value=i:n={value:i,done:!1},n}function
I(){return{value:void 0,done:!0}}function b(t){return!!M(t)}function
q(t){return t&&"function"==typeof t.next}function D(t){var e=M(t);return
e&&e.call(t)}function M(t){var e=t&&(zr&&t[zr]||t[Ir]);return"function"==typeof
e?e:void 0}function E(t){return t&&"number"==typeof t.length}function
O(t){return null===t||void 0===t?T():o(t)?t.toSeq():C(t)}function x(t){return
null===t||void
0===t?T().toKeyedSeq():o(t)?u(t)?t.toSeq():t.fromEntrySeq():W(t)}function
k(t){return null===t||void
0===t?T():o(t)?u(t)?t.entrySeq():t.toIndexedSeq():B(t)}function
A(t){return(null===t||void
0===t?T():o(t)?u(t)?t.entrySeq():t:B(t)).toSetSeq()}function
j(t){this._array=t,this.size=t.length}function K(t){var
e=Object.keys(t);this._object=t,this._keys=e,
this.size=e.length}function
R(t){this._iterable=t,this.size=t.length||t.size}function
U(t){this._iterator=t,this._iteratorCache=[]}function
L(t){return!(!t||!t[qr])}function T(){return Dr||(Dr=new j([]))}function
W(t){var e=Array.isArray(t)?new j(t).fromEntrySeq():q(t)?new
U(t).fromEntrySeq():b(t)?new R(t).fromEntrySeq():"object"==typeof t?new
K(t):void 0;if(!e)throw new TypeError("Expected Array or iterable object of [k,
v] entries, or keyed object: "+t);return e}function B(t){var e=J(t);if(!e)throw
new TypeError("Expected Array or iterable object of values: "+t);return
e}function C(t){var e=J(t)||"object"==typeof t&&new K(t);if(!e)throw new
TypeError("Expected Array or iterable object of values, or keyed object:
"+t);return e}function J(t){return E(t)?new j(t):q(t)?new U(t):b(t)?new
R(t):void 0}function N(t,e,r,n){var i=t._cache;if(i){for(var
o=i.length-1,u=0;o>=u;u++){var s=i[r?o-u:u];if(e(s[1],n?s[0]:u,t)===!1)return
u+1}return u}return t.__iterateUncached(e,r)}function P(t,e,r,n){var
i=t._cache;if(i){var o=i.length-1,u=0;return new S(function(){var
t=i[r?o-u:u];return u++>o?I():z(e,n?t[0]:u-1,t[1])})}return
t.__iteratorUncached(e,r)}function H(t,e){return
e?V(e,t,"",{"":t}):Y(t)}function V(t,e,r,n){return
Array.isArray(e)?t.call(n,r,k(e).map(function(r,n){return
V(t,r,n,e)})):Q(e)?t.call(n,r,x(e).map(function(r,n){return
V(t,r,n,e)})):e}function Y(t){return
Array.isArray(t)?k(t).map(Y).toList():Q(t)?x(t).map(Y).toMap():t}function
Q(t){return t&&(t.constructor===Object||void 0===t.constructor)}function
X(t,e){if(t===e||t!==t&&e!==e)return!0;if(!t||!e)return!1;if("function"==typeof
t.valueOf&&"function"==typeof
e.valueOf){if(t=t.valueOf(),e=e.valueOf(),t===e||t!==t&&e!==e)return!0;if(!t||!e
)return!1}return"function"==typeof t.equals&&"function"==typeof
e.equals&&t.equals(e)?!0:!1}function F(t,e){if(t===e)return!0;if(!o(e)||void
0!==t.size&&void 0!==e.size&&t.size!==e.size||void 0!==t.__hash&&void
0!==e.__hash&&t.__hash!==e.__hash||u(t)!==u(e)||s(t)!==s(e)||h(t)!==h(e))return!
1;if(0===t.size&&0===e.size)return!0;
var r=!a(t);if(h(t)){var n=t.entries();return e.every(function(t,e){var
i=n.next().value;return i&&X(i[1],t)&&(r||X(i[0],e))})&&n.next().done}var
i=!1;if(void 0===t.size)if(void 0===e.size)"function"==typeof
```

```
t.cacheResult&&t.cacheResult();else{i=!0;var f=t;t=e,e=f}var
c=!0,_=e.__iterate(function(e,n){return(r?t.has(e):i?X(e,t.get(n,yr)):X(t.get(n,
yr),e))?void 0:(c=!1,!1)});return c&&t.size===_}function G(t,e){if(!(this
instanceof G))return new G(t,e);if(this._value=t,this.size=void
0===e?1/0:Math.max(0,e),0===this.size){if(Mr)return Mr;Mr=this}}function
Z(t,e){if(!t)throw Error(e)}function $(t,e,r){if(!(this instanceof $))return
new $(t,e,r);if(Z(0!==r,"Cannot step a Range by 0"),t=t||0,void
0===e&&(e=1/0),r=void
0===r?1:Math.abs(r),t>e&&(r=-r),this._start=t,this._end=e,this._step=r,this.size
=Math.max(0,Math.ceil((e-t)/r-1)+1),0===this.size){if(Er)return
Er;Er=this}}function tt(){throw TypeError("Abstract")}function et(){}function
rt(){}function nt(){}function it(t){return
t>>>1&1073741824|3221225471&t}function ot(t){if(t===!1||null===t||void
0===t)return 0;if("function"==typeof
t.valueOf&&(t=t.valueOf(),t===!1||null===t||void 0===t))return
0;if(t===!0)return 1;var e=typeof t;if("number"===e){var
r=0|t;for(r!==t&&(r^=4294967295*t);t>4294967295;)t/=4294967295,r^=t;return
it(r)}if("string"===e)return t.length>Ur?ut(t):st(t);if("function"==typeof
t.hashCode)return t.hashCode();if("object"===e)return
at(t);if("function"==typeof t.toString)return st(""+t);throw Error("Value type
"+e+" cannot be hashed.")}function ut(t){var e=Wr[t];return void
0===e&&(e=st(t),Tr===Lr&&(Tr=0,Wr={}),Tr++,Wr[t]=e),e}function st(t){for(var
e=0,r=0;t.length>r;r++)e=31*e+t.charCodeAt(r)|0;return it(e)}function at(t){var
e;if(jr&&(e=Or.get(t),void 0!==e))return e;if(e=t[Rr],void 0!==e)return
e;if(!Ar){if(e=t.propertyIsEnumerable&&t.propertyIsEnumerable[Rr],void
0!==e)return e;if(e=ht(t),void 0!==e)return
e}if(e=++Kr,1073741824&Kr&&(Kr=0),jr)Or.set(t,e);else{if(void
0!==kr&&kr(t)===!1)throw Error("Non-extensible objects are not allowed as
keys.");
if(Ar)Object.defineProperty(t,Rr,{enumerable:!1,configurable:!1,writable:!1,valu
e:e});else if(void
0!==t.propertyIsEnumerable&&t.propertyIsEnumerable===t.constructor.prototype.pro
pertyIsEnumerable)t.propertyIsEnumerable=function(){return
this.constructor.prototype.propertyIsEnumerable.apply(this,arguments)},t.propert
yIsEnumerable[Rr]=e;else{if(void 0===t.nodeType)throw Error("Unable to set a
non-enumerable property on object.");t[Rr]=e}}return e}function
ht(t){if(t&&t.nodeType>0)switch(t.nodeType){case 1:return t.uniqueID;case
9:return t.documentElement&&t.documentElement.uniqueID}}function
ft(t){Z(t!==1/0,"Cannot perform this action with an infinite size.")}function
ct(t){return null===t||void
0===t?zt():_t(t)&&!h(t)?t:zt().withMutations(function(e){var
n=r(t);ft(n.size),n.forEach(function(t,r){return e.set(r,t)})})}function
_t(t){return!(!t||!t[Br])}function
pt(t,e){this.ownerID=t,this.entries=e}function
vt(t,e,r){this.ownerID=t,this.bitmap=e,this.nodes=r}function
lt(t,e,r){this.ownerID=t,this.count=e,this.nodes=r}function
yt(t,e,r){this.ownerID=t,this.keyHash=e,this.entries=r}function
dt(t,e,r){this.ownerID=t,this.keyHash=e,this.entry=r}function
mt(t,e,r){this._type=e,this._reverse=r,this._stack=t._root&&wt(t._root)}function
 gt(t,e){return z(t,e[0],e[1])}function
wt(t,e){return{node:t,index:0,__prev:e}}function St(t,e,r,n){var
i=Object.create(Cr);return
i.size=t,i._root=e,i.__ownerID=r,i.__hash=n,i.__altered=!1,i}function
zt(){return Jr||(Jr=St(0))}function It(t,e,r){var n,i;if(t._root){var
```

```
o=f(dr),u=f(mr);if(n=bt(t._root,t.__ownerID,0,void 0,e,r,o,u),!u.value)return
t;i=t.size+(o.value?r===yr?-1:1:0)}else{if(r===yr)return t;i=1,n=new
pt(t.__ownerID,[[e,r]])}return t.__ownerID?(t.size=i,t._root=n,t.__hash=void
0,t.__altered=!0,t):n?St(i,n):zt()}function bt(t,e,r,n,i,o,u,s){return
t?t.update(e,r,n,i,o,u,s):o===yr?t:(c(s),c(u),new dt(e,n,[i,o]))}function
qt(t){return t.constructor===dt||t.constructor===yt}function
Dt(t,e,r,n,i){if(t.keyHash===n)return new yt(e,n,[t.entry,i]);var
o,u=(0===r?t.keyHash:t.keyHash>>>r)&lr,s=(0===r?n:n>>>r)&lr,a=u===s?[Dt(t,e,r+pr
,n,i)]:(o=new dt(e,n,i),
s>u?[t,o]:[o,t]);return new vt(e,1<<u|1<<s,a)}function Mt(t,e,r,n){t||(t=new
_);for(var i=new dt(t,ot(r),[r,n]),o=0;e.length>o;o++){var
u=e[o];i=i.update(t,0,void 0,u[0],u[1])}return i}function Et(t,e,r,n){for(var
i=0,o=0,u=Array(r),s=0,a=1,h=e.length;h>s;s++,a<<=1){var f=e[s];void
0!==f&&s!==n&&(i|=a,u[o++]=f)}return new vt(t,i,u)}function
Ot(t,e,r,n,i){for(var o=0,u=Array(vr),s=0;0!==r;s++,r>>>=1)u[s]=1&r?e[o++]:void
0;return u[n]=i,new lt(t,o+1,u)}function xt(t,e,n){for(var
i=[],u=0;n.length>u;u++){var s=n[u],a=r(s);o(s)||(a=a.map(function(t){return
H(t)})),i.push(a)}return jt(t,e,i)}function kt(t,e,r){return
t&&t.mergeDeep&&o(e)?t.mergeDeep(e):X(t,e)?t:e}function At(t){return
function(e,r,n){if(e&&e.mergeDeepWith&&o(r))return e.mergeDeepWith(t,r);var
i=t(e,r,n);return X(e,i)?e:i}}function jt(t,e,r){return
r=r.filter(function(t){return
0!==t.size}),0===r.length?t:0!==t.size||t.__ownerID||1!==r.length?t.withMutation
s(function(t){for(var n=e?function(r,n){t.update(n,yr,function(t){return
t===yr?r:e(t,r,n)})}:function(e,r){t.set(r,e)},i=0;r.length>i;i++)r[i].forEach(n
)}):t.constructor(r[0])}function Kt(t,e,r,n){var
i=t===yr,o=e.next();if(o.done){var u=i?r:t,s=n(u);return
s===u?t:s}Z(i||t&&t.set,"invalid keyPath");var
a=o.value,h=i?yr:t.get(a,yr),f=Kt(h,e,r,n);return
f===h?t:f===yr?t.remove(a):(i?zt():t).set(a,f)}function Rt(t){return
t-=t>>1&1431655765,t=(858993459&t)+(t>>2&858993459),t=t+(t>>4)&252645135,t+=t>>8
,t+=t>>16,127&t}function Ut(t,e,r,n){var i=n?t:p(t);return i[e]=r,i}function
Lt(t,e,r,n){var i=t.length+1;if(n&&e+1===i)return t[e]=r,t;for(var
o=Array(i),u=0,s=0;i>s;s++)s===e?(o[s]=r,u=-1):o[s]=t[s+u];return o}function
Tt(t,e,r){var n=t.length-1;if(r&&e===n)return t.pop(),t;for(var
i=Array(n),o=0,u=0;n>u;u++)u===e&&(o=1),i[u]=t[u+o];return i}function Wt(t){var
e=Pt();if(null===t||void 0===t)return e;if(Bt(t))return t;var
r=n(t),i=r.size;return 0===i?e:(ft(i),i>0&&vr>i?Nt(0,i,pr,null,new
Ct(r.toArray())):e.withMutations(function(t){t.setSize(i),r.forEach(function(e,r
){return t.set(r,e)})}))}function Bt(t){
return!(!t||!t[Vr])}function Ct(t,e){this.array=t,this.ownerID=e}function
Jt(t,e){function r(t,e,r){return 0===e?n(t,r):i(t,e,r)}function n(t,r){var
n=r===s?a&&a.array:t&&t.array,i=r>o?0:o-r,h=u-r;return
h>vr&&(h=vr),function(){if(i===h)return Xr;var t=e?--h:i++;return
n&&n[t]}}function i(t,n,i){var
s,a=t&&t.array,h=i>o?0:o-i>>n,f=(u-i>>n)+1;return
f>vr&&(f=vr),function(){for(;;){if(s){var t=s();if(t!==Xr)return
t;s=null}if(h===f)return Xr;var o=e?--f:h++;s=r(a&&a[o],n-pr,i+(o<<n))}}}var
o=t._origin,u=t._capacity,s=Gt(u),a=t._tail;return
r(t._root,t._level,0)}function Nt(t,e,r,n,i,o,u){var s=Object.create(Yr);return
s.size=e-t,s._origin=t,s._capacity=e,s._level=r,s._root=n,s._tail=i,s.__ownerID=
o,s.__hash=u,s.__altered=!1,s}function Pt(){return Qr||(Qr=Nt(0,0,pr))}function
Ht(t,e,r){if(e=l(t,e),e!==e)return t;if(e>=t.size||0>e)return
t.withMutations(function(t){0>e?Xt(t,e).set(0,r):Xt(t,0,e+1).set(e,r)});e+=t._or
```

```
igin;var n=t._tail,i=t._root,o=f(mr);return
e>=Gt(t._capacity)?n=Vt(n,t.__ownerID,0,e,r,o):i=Vt(i,t.__ownerID,t._level,e,r,o
),o.value?t.__ownerID?(t._root=i,t._tail=n,t.__hash=void
0,t.__altered=!0,t):Nt(t._origin,t._capacity,t._level,i,n):t}function
Vt(t,e,r,n,i,o){var u=n>>>r&lr,s=t&&t.array.length>u;if(!s&&void 0===i)return
t;var a;if(r>0){var h=t&&t.array[u],f=Vt(h,e,r-pr,n,i,o);return
f===h?t:(a=Yt(t,e),a.array[u]=f,a)}return
s&&t.array[u]===i?t:(c(o),a=Yt(t,e),void
0===i&&u===a.array.length-1?a.array.pop():a.array[u]=i,a)}function
Yt(t,e){return e&&t&&e===t.ownerID?t:new Ct(t?t.array.slice():[],e)}function
Qt(t,e){if(e>=Gt(t._capacity))return t._tail;if(1<<t._level+pr>e){for(var
r=t._root,n=t._level;r&&n>0;)r=r.array[e>>>n&lr],n-=pr;return r}}function
Xt(t,e,r){void 0!==e&&(e=0|e),void 0!==r&&(r=0|r);var n=t.__ownerID||new
_,i=t._origin,o=t._capacity,u=i+e,s=void
0===r?o:0>r?o+r:i+r;if(u===i&&s===o)return t;if(u>=s)return t.clear();for(var
a=t._level,h=t._root,f=0;0>u+f;)h=new Ct(h&&h.array.length?[void
0,h]:[],n),a+=pr,f+=1<<a;f&&(u+=f,i+=f,s+=f,o+=f);for(var
c=Gt(o),p=Gt(s);p>=1<<a+pr;)h=new Ct(h&&h.array.length?[h]:[],n),
a+=pr;var v=t._tail,l=c>p?Qt(t,s-1):p>c?new
Ct([],n):v;if(v&&p>c&&o>u&&v.array.length){h=Yt(h,n);for(var
y=h,d=a;d>pr;d-=pr){var
m=c>>>d&lr;y=y.array[m]=Yt(y.array[m],n)}y.array[c>>>pr&lr]=v}if(o>s&&(l=l&&l.re
moveAfter(n,0,s)),u>=p)u-=p,s-=p,a=pr,h=null,l=l&&l.removeBefore(n,0,u);else
if(u>i||c>p){for(f=0;h;){var
g=u>>>a&lr;if(g!==p>>>a&lr)break;g&&(f+=(1<<a)*g),a-=pr,h=h.array[g]}h&&u>i&&(h=
h.removeBefore(n,a,u-f)),h&&c>p&&(h=h.removeAfter(n,a,p-f)),f&&(u-=f,s-=f)}retur
n
t.__ownerID?(t.size=s-u,t._origin=u,t._capacity=s,t._level=a,t._root=h,t._tail=l
,t.__hash=void 0,t.__altered=!0,t):Nt(u,s,a,h,l)}function Ft(t,e,r){for(var
i=[],u=0,s=0;r.length>s;s++){var
a=r[s],h=n(a);h.size>u&&(u=h.size),o(a)||(h=h.map(function(t){return
H(t)})),i.push(h)}return u>t.size&&(t=t.setSize(u)),jt(t,e,i)}function
Gt(t){return vr>t?0:t-1>>>pr<<pr}function Zt(t){return null===t||void
0===t?ee():$t(t)?t:ee().withMutations(function(e){var
n=r(t);ft(n.size),n.forEach(function(t,r){return e.set(r,t)})})}function
$t(t){return _t(t)&&h(t)}function te(t,e,r,n){var
i=Object.create(Zt.prototype);return
i.size=t?t.size:0,i._map=t,i._list=e,i.__ownerID=r,i.__hash=n,i}function
ee(){return Fr||(Fr=te(zt(),Pt()))}function re(t,e,r){var
n,i,o=t._map,u=t._list,s=o.get(e),a=void 0!==s;if(r===yr){if(!a)return
t;u.size>=vr&&u.size>=2*o.size?(i=u.filter(function(t,e){return void
0!==t&&s!==e}),n=i.toKeyedSeq().map(function(t){return
t[0]}).flip().toMap(),t.__ownerID&&(n.__ownerID=i.__ownerID=t.__ownerID)):(n=o.r
emove(e),i=s===u.size-1?u.pop():u.set(s,void 0))}else
if(a){if(r===u.get(s)[1])return t;n=o,i=u.set(s,[e,r])}else
n=o.set(e,u.size),i=u.set(u.size,[e,r]);return
t.__ownerID?(t.size=n.size,t._map=n,t._list=i,t.__hash=void
0,t):te(n,i)}function
ne(t,e){this._iter=t,this._useKeys=e,this.size=t.size}function
ie(t){this._iter=t,this.size=t.size}function
oe(t){this._iter=t,this.size=t.size}function
ue(t){this._iter=t,this.size=t.size}function se(t){var e=Ee(t);return
e._iter=t,e.size=t.size,e.flip=function(){return t},e.reverse=function(){var
e=t.reverse.apply(this);
```

```
return e.flip=function(){return t.reverse()},e},e.has=function(e){return
t.includes(e)},e.includes=function(e){return
t.has(e)},e.cacheResult=Oe,e.__iterateUncached=function(e,r){var n=this;return
t.__iterate(function(t,r){return
e(r,t,n)!==!1},r)},e.__iteratorUncached=function(e,r){if(e===Sr){var
n=t.__iterator(e,r);return new S(function(){var t=n.next();if(!t.done){var
e=t.value[0];t.value[0]=t.value[1],t.value[1]=e}return t})}return
t.__iterator(e===wr?gr:wr,r)},e}function ae(t,e,r){var n=Ee(t);return
n.size=t.size,n.has=function(e){return t.has(e)},n.get=function(n,i){var
o=t.get(n,yr);return
o===yr?i:e.call(r,o,n,t)},n.__iterateUncached=function(n,i){var o=this;return
t.__iterate(function(t,i,u){return
n(e.call(r,t,i,u),i,o)!==!1},i)},n.__iteratorUncached=function(n,i){var
o=t.__iterator(Sr,i);return new S(function(){var i=o.next();if(i.done)return
i;var u=i.value,s=u[0];return z(n,s,e.call(r,u[1],s,t),i)})},n}function
he(t,e){var r=Ee(t);return r._iter=t,r.size=t.size,r.reverse=function(){return
t},t.flip&&(r.flip=function(){var e=se(t);return e.reverse=function(){return
t.flip()},e}),r.get=function(r,n){return
t.get(e?r:-1-r,n)},r.has=function(r){return
t.has(e?r:-1-r)},r.includes=function(e){return
t.includes(e)},r.cacheResult=Oe,r.__iterate=function(e,r){var n=this;return
t.__iterate(function(t,r){return
e(t,r,n)},!r)},r.__iterator=function(e,r){return t.__iterator(e,!r)},r}function
fe(t,e,r,n){var i=Ee(t);return n&&(i.has=function(n){var i=t.get(n,yr);return
i!==yr&&!!e.call(r,i,n,t)},i.get=function(n,i){var o=t.get(n,yr);return
o!==yr&&e.call(r,o,n,t)?o:i}),i.__iterateUncached=function(i,o){var
u=this,s=0;return t.__iterate(function(t,o,a){return
e.call(r,t,o,a)?(s++,i(t,n?o:s-1,u)):void
0},o),s},i.__iteratorUncached=function(i,o){var u=t.__iterator(Sr,o),s=0;return
new S(function(){for(;;){var o=u.next();if(o.done)return o;var
a=o.value,h=a[0],f=a[1];if(e.call(r,f,h,t))return
z(i,n?h:s++,f,o)}})},i}function ce(t,e,r){var n=ct().asMutable();return
t.__iterate(function(i,o){n.update(e.call(r,i,o,t),0,function(t){
return t+1})}),n.asImmutable()}function _e(t,e,r){var
n=u(t),i=(h(t)?Zt():ct()).asMutable();t.__iterate(function(o,u){i.update(e.call(
r,o,u,t),function(t){return t=t||[],t.push(n?[u,o]:o),t})});var o=Me(t);return
i.map(function(e){return be(t,o(e))})}function pe(t,e,r,n){var i=t.size;if(void
0!==e&&(e=0|e),void 0!==r&&(r=0|r),d(e,r,i))return t;var
o=m(e,i),u=g(r,i);if(o!==o||u!==u)return pe(t.toSeq().cacheResult(),e,r,n);var
s,a=u-o;a===a&&(s=0>a?0:a);var h=Ee(t);return h.size=0===s?s:t.size&&s||void
0,!n&&L(t)&&s>=0&&(h.get=function(e,r){return
e=l(this,e),e>=0&&s>e?t.get(e+o,r):r}),h.__iterateUncached=function(e,r){var
i=this;if(0===s)return 0;if(r)return this.cacheResult().__iterate(e,r);var
u=0,a=!0,h=0;return t.__iterate(function(t,r){return a&&(a=u++<o)?void
0:(h++,e(t,n?r:h-1,i)!==!1&&h!==s)}),h},h.__iteratorUncached=function(e,r){if(0!
==s&&r)return this.cacheResult().__iterator(e,r);var
i=0!==s&&t.__iterator(e,r),u=0,a=0;return new
S(function(){for(;u++<o;)i.next();if(++a>s)return I();var t=i.next();return
n||e===wr?t:e===gr?z(e,a-1,void 0,t):z(e,a-1,t.value[1],t)})},h}function
ve(t,e,r){var n=Ee(t);return n.__iterateUncached=function(n,i){var
o=this;if(i)return this.cacheResult().__iterate(n,i);var u=0;return
t.__iterate(function(t,i,s){return
e.call(r,t,i,s)&&++u&&n(t,i,o)}),u},n.__iteratorUncached=function(n,i){var
o=this;if(i)return this.cacheResult().__iterator(n,i);var
```

```
u=t.__iterator(Sr,i),s=!0;return new S(function(){if(!s)return I();var
t=u.next();if(t.done)return t;var i=t.value,a=i[0],h=i[1];return
e.call(r,h,a,o)?n===Sr?t:z(n,a,h,t):(s=!1,I())})},n}function le(t,e,r,n){var
i=Ee(t);return i.__iterateUncached=function(i,o){var u=this;if(o)return
this.cacheResult().__iterate(i,o);var s=!0,a=0;return
t.__iterate(function(t,o,h){return s&&(s=e.call(r,t,o,h))?void
0:(a++,i(t,n?o:a-1,u))}),a},i.__iteratorUncached=function(i,o){var
u=this;if(o)return this.cacheResult().__iterator(i,o);var
s=t.__iterator(Sr,o),a=!0,h=0;return new S(function(){var
t,o,f;do{if(t=s.next(),t.done)return n||i===wr?t:i===gr?z(i,h++,void
0,t):z(i,h++,t.value[1],t);
var c=t.value;o=c[0],f=c[1],a&&(a=e.call(r,f,o,u))}while(a);return
i===Sr?t:z(i,o,f,t)})},i}function ye(t,e){var
n=u(t),i=[t].concat(e).map(function(t){return
o(t)?n&&(t=r(t)):t=n?W(t):B(Array.isArray(t)?t:[t]),t}).filter(function(t){retur
n 0!==t.size});if(0===i.length)return t;if(1===i.length){var
a=i[0];if(a===t||n&&u(a)||s(t)&&s(a))return a}var h=new j(i);return
n?h=h.toKeyedSeq():s(t)||(h=h.toSetSeq()),h=h.flatten(!0),h.size=i.reduce(functi
on(t,e){if(void 0!==t){var r=e.size;if(void 0!==r)return t+r}},0),h}function
de(t,e,r){var n=Ee(t);return n.__iterateUncached=function(n,i){function
u(t,h){var
f=this;t.__iterate(function(t,i){return(!e||e>h)&&o(t)?u(t,h+1):n(t,r?i:s++,f)==
=!1&&(a=!0),!a},i)}var s=0,a=!1;return
u(t,0),s},n.__iteratorUncached=function(n,i){var
u=t.__iterator(n,i),s=[],a=0;return new S(function(){for(;u;){var
t=u.next();if(t.done===!1){var
h=t.value;if(n===Sr&&(h=h[1]),e&&!(e>s.length)||!o(h))return
r?t:z(n,a++,h,t);s.push(u),u=h.__iterator(n,i)}else u=s.pop()}return
I()})},n}function me(t,e,r){var n=Me(t);return
t.toSeq().map(function(i,o){return n(e.call(r,i,o,t))}).flatten(!0)}function
ge(t,e){var r=Ee(t);return
r.size=t.size&&2*t.size-1,r.__iterateUncached=function(r,n){var
i=this,o=0;return
t.__iterate(function(t,n){return(!o||r(e,o++,i)!==!1)&&r(t,o++,i)!==!1},n),o},r.
__iteratorUncached=function(r,n){var i,o=t.__iterator(wr,n),u=0;return new
S(function(){return(!i||u%2)&&(i=o.next(),i.done)?i:u%2?z(r,u++,e):z(r,u++,i.val
ue,i)})},r}function we(t,e,r){e||(e=xe);var
n=u(t),i=0,o=t.toSeq().map(function(e,n){return[n,e,i++,r?r(e,n,t):e]}).toArray(
);return o.sort(function(t,r){return
e(t[3],r[3])||t[2]-r[2]}).forEach(n?function(t,e){o[e].length=2}:function(t,e){o
[e]=t[1]}),n?x(o):s(t)?k(o):A(o)}function Se(t,e,r){if(e||(e=xe),r){var
n=t.toSeq().map(function(e,n){return[e,r(e,n,t)]}).reduce(function(t,r){return
ze(e,t[1],r[1])?r:t});return n&&n[0]}return t.reduce(function(t,r){return
ze(e,t,r)?r:t})}function ze(t,e,r){var n=t(r,e);return 0===n&&r!==e&&(void
0===r||null===r||r!==r)||n>0}function Ie(t,r,n){
var i=Ee(t);return i.size=new j(n).map(function(t){return
t.size}).min(),i.__iterate=function(t,e){for(var
r,n=this.__iterator(wr,e),i=0;!(r=n.next()).done&&t(r.value,i++,this)!==!1;);ret
urn i},i.__iteratorUncached=function(t,i){var o=n.map(function(t){return
t=e(t),D(i?t.reverse():t)}),u=0,s=!1;return new S(function(){var e;return
s||(e=o.map(function(t){return t.next()}),s=e.some(function(t){return
t.done})),s?I():z(t,u++,r.apply(null,e.map(function(t){return
t.value})))})},i}function be(t,e){return L(t)?e:t.constructor(e)}function
qe(t){if(t!==Object(t))throw new TypeError("Expected [K, V] tuple:
```

```
"+t)}function De(t){return ft(t.size),v(t)}function Me(t){return
u(t)?r:s(t)?n:i}function Ee(t){return
Object.create((u(t)?x:s(t)?k:A).prototype)}function Oe(){return
this._iter.cacheResult?(this._iter.cacheResult(),this.size=this._iter.size,this)
:O.prototype.cacheResult.call(this)}function xe(t,e){return
t>e?1:e>t?-1:0}function ke(t){var r=D(t);if(!r){if(!E(t))throw new
TypeError("Expected iterable or array-like: "+t);r=D(e(t))}return r}function
Ae(t,e){var r,n=function(o){if(o instanceof n)return o;if(!(this instanceof
n))return new n(o);if(!r){r=!0;var
u=Object.keys(t);Re(i,u),i.size=u.length,i._name=e,i._keys=u,i._defaultValues=t}
this._map=ct(o)},i=n.prototype=Object.create(Gr);return
i.constructor=n,n}function je(t,e,r){var
n=Object.create(Object.getPrototypeOf(t));return
n._map=e,n.__ownerID=r,n}function Ke(t){return
t._name||t.constructor.name||"Record"}function
Re(t,e){try{e.forEach(Ue.bind(void 0,t))}catch(r){}}function
Ue(t,e){Object.defineProperty(t,e,{get:function(){return
this.get(e)},set:function(t){Z(this.__ownerID,"Cannot set on an immutable
record."),this.set(e,t)}})}function Le(t){return null===t||void
0===t?Ce():Te(t)&&!h(t)?t:Ce().withMutations(function(e){var
r=i(t);ft(r.size),r.forEach(function(t){return e.add(t)})})}function
Te(t){return!(!t||!t[Zr])}function We(t,e){return
t.__ownerID?(t.size=e.size,t._map=e,t):e===t._map?t:0===e.size?t.__empty():t.__m
ake(e)}function Be(t,e){var r=Object.create($r);
return r.size=t?t.size:0,r._map=t,r.__ownerID=e,r}function Ce(){return
tn||(tn=Be(zt()))}function Je(t){return null===t||void
0===t?He():Ne(t)?t:He().withMutations(function(e){var
r=i(t);ft(r.size),r.forEach(function(t){return e.add(t)})})}function
Ne(t){return Te(t)&&h(t)}function Pe(t,e){var r=Object.create(en);return
r.size=t?t.size:0,r._map=t,r.__ownerID=e,r}function He(){return
rn||(rn=Pe(ee()))}function Ve(t){return null===t||void
0===t?Xe():Ye(t)?t:Xe().unshiftAll(t)}function
Ye(t){return!(!t||!t[nn])}function Qe(t,e,r,n){var i=Object.create(on);return
i.size=t,i._head=e,i.__ownerID=r,i.__hash=n,i.__altered=!1,i}function
Xe(){return un||(un=Qe(0))}function Fe(t,e){var
r=function(r){t.prototype[r]=e[r]};return
Object.keys(e).forEach(r),Object.getOwnPropertySymbols&&Object.getOwnPropertySym
bols(e).forEach(r),t}function Ge(t,e){return e}function
Ze(t,e){return[e,t]}function $e(t){return
function(){return!t.apply(this,arguments)}}function tr(t){return
function(){return-t.apply(this,arguments)}}function
er(t){return"string"==typeof t?JSON.stringify(t):t}function rr(){return
p(arguments)}function nr(t,e){return e>t?1:t>e?-1:0}function
ir(t){if(t.size===1/0)return 0;var
e=h(t),r=u(t),n=e?1:0,i=t.__iterate(r?e?function(t,e){n=31*n+ur(ot(t),ot(e))|0}:
function(t,e){n=n+ur(ot(t),ot(e))|0}:e?function(t){n=31*n+ot(t)|0}:function(t){n
=n+ot(t)|0});return or(i,n)}function or(t,e){return
e=xr(e,3432918353),e=xr(e<<15|e>>>-15,461845907),e=xr(e<<13|e>>>-13,5),e=(e+3864
292196|0)^t,e=xr(e^e>>>16,2246822507),e=xr(e^e>>>13,3266489909),e=it(e^e>>>16)}f
unction ur(t,e){return t^e+2654435769+(t<<6)+(t>>2)|0}var
sr=Array.prototype.slice;t(r,e),t(n,e),t(i,e),e.isIterable=o,e.isKeyed=u,e.isInd
exed=s,e.isAssociative=a,e.isOrdered=h,e.Keyed=r,e.Indexed=n,e.Set=i;var
ar="@@__IMMUTABLE_ITERABLE__@@",hr="@@__IMMUTABLE_KEYED__@@",fr="@@__IMMUTABLE_I
NDEXED__@@",cr="@@__IMMUTABLE_ORDERED__@@",_r="delete",pr=5,vr=1<<pr,lr=vr-1,yr=
```

```
{},dr={value:!1},mr={value:!1},gr=0,wr=1,Sr=2,zr="function"==typeof
Symbol&&Symbol.iterator,Ir="@@iterator",br=zr||Ir;
S.prototype.toString=function(){return"[Iterator]"},S.KEYS=gr,S.VALUES=wr,S.ENTR
IES=Sr,S.prototype.inspect=S.prototype.toSource=function(){return""+this},S.prot
otype[br]=function(){return this},t(O,e),O.of=function(){return
O(arguments)},O.prototype.toSeq=function(){return
this},O.prototype.toString=function(){return this.__toString("Seq
{","}")},O.prototype.cacheResult=function(){return!this._cache&&this.__iterateUn
cached&&(this._cache=this.entrySeq().toArray(),this.size=this._cache.length),thi
s},O.prototype.__iterate=function(t,e){return
N(this,t,e,!0)},O.prototype.__iterator=function(t,e){return
P(this,t,e,!0)},t(x,O),x.prototype.toKeyedSeq=function(){return
this},t(k,O),k.of=function(){return
k(arguments)},k.prototype.toIndexedSeq=function(){return
this},k.prototype.toString=function(){return this.__toString("Seq
[","]")},k.prototype.__iterate=function(t,e){return
N(this,t,e,!1)},k.prototype.__iterator=function(t,e){return
P(this,t,e,!1)},t(A,O),A.of=function(){return
A(arguments)},A.prototype.toSetSeq=function(){return
this},O.isSeq=L,O.Keyed=x,O.Set=A,O.Indexed=k;var
qr="@@__IMMUTABLE_SEQ__@@";O.prototype[qr]=!0,t(j,k),j.prototype.get=function(t,
e){return
this.has(t)?this._array[l(this,t)]:e},j.prototype.__iterate=function(t,e){for(va
r r=this._array,n=r.length-1,i=0;n>=i;i++)if(t(r[e?n-i:i],i,this)===!1)return
i+1;return i},j.prototype.__iterator=function(t,e){var
r=this._array,n=r.length-1,i=0;return new S(function(){return
i>n?I():z(t,i,r[e?n-i++:i++])})},t(K,x),K.prototype.get=function(t,e){return
void 0===e||this.has(t)?this._object[t]:e},K.prototype.has=function(t){return
this._object.hasOwnProperty(t)},K.prototype.__iterate=function(t,e){for(var
r=this._object,n=this._keys,i=n.length-1,o=0;i>=o;o++){var
u=n[e?i-o:o];if(t(r[u],u,this)===!1)return o+1}return
o},K.prototype.__iterator=function(t,e){var
r=this._object,n=this._keys,i=n.length-1,o=0;return new S(function(){var
u=n[e?i-o:o];return
o++>i?I():z(t,u,r[u])})},K.prototype[cr]=!0,t(R,k),R.prototype.__iterateUncached
=function(t,e){if(e)return this.cacheResult().__iterate(t,e);
var r=this._iterable,n=D(r),i=0;if(q(n))for(var
o;!(o=n.next()).done&&t(o.value,i++,this)!==!1;);return
i},R.prototype.__iteratorUncached=function(t,e){if(e)return
this.cacheResult().__iterator(t,e);var r=this._iterable,n=D(r);if(!q(n))return
new S(I);var i=0;return new S(function(){var e=n.next();return
e.done?e:z(t,i++,e.value)})},t(U,k),U.prototype.__iterateUncached=function(t,e){
if(e)return this.cacheResult().__iterate(t,e);for(var
r=this._iterator,n=this._iteratorCache,i=0;n.length>i;)if(t(n[i],i++,this)===!1)
return i;for(var o;!(o=r.next()).done;){var
u=o.value;if(n[i]=u,t(u,i++,this)===!1)break}return
i},U.prototype.__iteratorUncached=function(t,e){if(e)return
this.cacheResult().__iterator(t,e);var
r=this._iterator,n=this._iteratorCache,i=0;return new
S(function(){if(i>=n.length){var e=r.next();if(e.done)return
e;n[i]=e.value}return z(t,i,n[i++])})};var
Dr;t(G,k),G.prototype.toString=function(){return 0===this.size?"Repeat
[]":"Repeat [ "+this._value+" "+this.size+" times
]"},G.prototype.get=function(t,e){return
```

```
this.has(t)?this._value:e},G.prototype.includes=function(t){return
X(this._value,t)},G.prototype.slice=function(t,e){var r=this.size;return
d(t,e,r)?this:new
G(this._value,g(e,r)-m(t,r))},G.prototype.reverse=function(){return
this},G.prototype.indexOf=function(t){return
X(this._value,t)?0:-1},G.prototype.lastIndexOf=function(t){return
X(this._value,t)?this.size:-1},G.prototype.__iterate=function(t,e){for(var
r=0;this.size>r;r++)if(t(this._value,r,this)===!1)return r+1;return
r},G.prototype.__iterator=function(t,e){var r=this,n=0;return new
S(function(){return
r.size>n?z(t,n++,r._value):I()})},G.prototype.equals=function(t){return t
instanceof G?X(this._value,t._value):F(t)};var
Mr;t($,k),$.prototype.toString=function(){return 0===this.size?"Range
[]":"Range [ "+this._start+"..."+this._end+(1!==this._step?" by
"+this._step:"")+" ]"},$.prototype.get=function(t,e){return
this.has(t)?this._start+l(this,t)*this._step:e},$.prototype.includes=function(t)
{var e=(t-this._start)/this._step;return e>=0&&this.size>e&&e===Math.floor(e);
},$.prototype.slice=function(t,e){return
d(t,e,this.size)?this:(t=m(t,this.size),e=g(e,this.size),t>=e?new $(0,0):new
$(this.get(t,this._end),this.get(e,this._end),this._step))},$.prototype.indexOf=
function(t){var e=t-this._start;if(e%this._step===0){var
r=e/this._step;if(r>=0&&this.size>r)return
r}return-1},$.prototype.lastIndexOf=function(t){return
this.indexOf(t)},$.prototype.__iterate=function(t,e){for(var
r=this.size-1,n=this._step,i=e?this._start+r*n:this._start,o=0;r>=o;o++){if(t(i,
o,this)===!1)return o+1;i+=e?-n:n}return
o},$.prototype.__iterator=function(t,e){var
r=this.size-1,n=this._step,i=e?this._start+r*n:this._start,o=0;return new
S(function(){var u=i;return
i+=e?-n:n,o>r?I():z(t,o++,u)})},$.prototype.equals=function(t){return t
instanceof
$?this._start===t._start&&this._end===t._end&&this._step===t._step:F(this,t)};va
r Er;t(tt,e),t(et,tt),t(rt,tt),t(nt,tt),tt.Keyed=et,tt.Indexed=rt,tt.Set=nt;var
Or,xr="function"==typeof
Math.imul&&-2===Math.imul(4294967295,2)?Math.imul:function(t,e){t=0|t,e=0|e;var
r=65535&t,n=65535&e;return
r*n+((t>>>16)*n+r*(e>>>16)<<16>>>0)|0},kr=Object.isExtensible,Ar=function(){try{
return
Object.defineProperty({},"@",{}),!0}catch(t){return!1}}(),jr="function"==typeof
WeakMap;jr&&(Or=new WeakMap);var Kr=0,Rr="__immutablehash__";"function"==typeof
Symbol&&(Rr=Symbol(Rr));var
Ur=16,Lr=255,Tr=0,Wr={};t(ct,et),ct.of=function(){var
t=sr.call(arguments,0);return zt().withMutations(function(e){for(var
r=0;t.length>r;r+=2){if(r+1>=t.length)throw Error("Missing value for key:
"+t[r]);e.set(t[r],t[r+1])}})},ct.prototype.toString=function(){return
this.__toString("Map {","}")},ct.prototype.get=function(t,e){return
this._root?this._root.get(0,void
0,t,e):e},ct.prototype.set=function(t,e){return
It(this,t,e)},ct.prototype.setIn=function(t,e){return
this.updateIn(t,yr,function(){return
e})},ct.prototype.remove=function(t){return
It(this,t,yr)},ct.prototype.deleteIn=function(t){return
this.updateIn(t,function(){return
yr})},ct.prototype.update=function(t,e,r){return
```

```
1===arguments.length?t(this):this.updateIn([t],e,r);
},ct.prototype.updateIn=function(t,e,r){r||(r=e,e=void 0);var
n=Kt(this,ke(t),e,r);return n===yr?void
0:n},ct.prototype.clear=function(){return
0===this.size?this:this.__ownerID?(this.size=0,this._root=null,this.__hash=void
0,this.__altered=!0,this):zt()},ct.prototype.merge=function(){return
xt(this,void 0,arguments)},ct.prototype.mergeWith=function(t){var
e=sr.call(arguments,1);return
xt(this,t,e)},ct.prototype.mergeIn=function(t){var
e=sr.call(arguments,1);return
this.updateIn(t,zt(),function(t){return"function"==typeof
t.merge?t.merge.apply(t,e):e[e.length-1]})},ct.prototype.mergeDeep=function(){re
turn xt(this,kt,arguments)},ct.prototype.mergeDeepWith=function(t){var
e=sr.call(arguments,1);return
xt(this,At(t),e)},ct.prototype.mergeDeepIn=function(t){var
e=sr.call(arguments,1);return
this.updateIn(t,zt(),function(t){return"function"==typeof
t.mergeDeep?t.mergeDeep.apply(t,e):e[e.length-1]})},ct.prototype.sort=function(t
){return Zt(we(this,t))},ct.prototype.sortBy=function(t,e){return
Zt(we(this,e,t))},ct.prototype.withMutations=function(t){var
e=this.asMutable();return
t(e),e.wasAltered()?e.__ensureOwner(this.__ownerID):this},ct.prototype.asMutable
=function(){return this.__ownerID?this:this.__ensureOwner(new
_)},ct.prototype.asImmutable=function(){return
this.__ensureOwner()},ct.prototype.wasAltered=function(){return
this.__altered},ct.prototype.__iterator=function(t,e){return new
mt(this,t,e)},ct.prototype.__iterate=function(t,e){var r=this,n=0;return
this._root&&this._root.iterate(function(e){return
n++,t(e[1],e[0],r)},e),n},ct.prototype.__ensureOwner=function(t){return
t===this.__ownerID?this:t?St(this.size,this._root,t,this.__hash):(this.__ownerID
=t,this.__altered=!1,this)},ct.isMap=_t;var
Br="@@__IMMUTABLE_MAP__@@",Cr=ct.prototype;Cr[Br]=!0,Cr[_r]=Cr.remove,Cr.removeI
n=Cr.deleteIn,pt.prototype.get=function(t,e,r,n){for(var
i=this.entries,o=0,u=i.length;u>o;o++)if(X(r,i[o][0]))return i[o][1];return
n},pt.prototype.update=function(t,e,r,n,i,o,u){for(var
s=i===yr,a=this.entries,h=0,f=a.length;f>h&&!X(n,a[h][0]);h++);
var _=f>h;if(_?a[h][1]===i:s)return
this;if(c(u),(s||!_)&&c(o),!s||1!==a.length){if(!_&&!s&&a.length>=Nr)return
Mt(t,a,n,i);var v=t&&t===this.ownerID,l=v?a:p(a);return
_?s?h===f-1?l.pop():l[h]=l.pop():l[h]=[n,i]:l.push([n,i]),v?(this.entries=l,this
):new pt(t,l)}},vt.prototype.get=function(t,e,r,n){void 0===e&&(e=ot(r));var
i=1<<((0===t?e:e>>>t)&lr),o=this.bitmap;return
0===(o&i)?n:this.nodes[Rt(o&i-1)].get(t+pr,e,r,n)},vt.prototype.update=function(
t,e,r,n,i,o,u){void 0===r&&(r=ot(n));var
s=(0===e?r:r>>>e)&lr,a=1<<s,h=this.bitmap,f=0!==(h&a);if(!f&&i===yr)return
this;var c=Rt(h&a-1),_=this.nodes,p=f?_[c]:void
0,v=bt(p,t,e+pr,r,n,i,o,u);if(v===p)return this;if(!f&&v&&_.length>=Pr)return
Ot(t,_,h,s,v);if(f&&!v&&2===_.length&&qt(_[1^c]))return
_[1^c];if(f&&v&&1===_.length&&qt(v))return v;var
l=t&&t===this.ownerID,y=f?v?h:h^a:h|a,d=f?v?Ut(_,c,v,l):Tt(_,c,l):Lt(_,c,v,l);re
turn l?(this.bitmap=y,this.nodes=d,this):new
vt(t,y,d)},lt.prototype.get=function(t,e,r,n){void 0===e&&(e=ot(r));var
i=(0===t?e:e>>>t)&lr,o=this.nodes[i];return
o?o.get(t+pr,e,r,n):n},lt.prototype.update=function(t,e,r,n,i,o,u){void
```

```
0===r&&(r=ot(n));var
s=(0===e?r:r>>>e)&lr,a=i===yr,h=this.nodes,f=h[s];if(a&&!f)return this;var
c=bt(f,t,e+pr,r,n,i,o,u);if(c===f)return this;var
_=this.count;if(f){if(!c&&(_--,Hr>_))return Et(t,h,_,s)}else _++;var
p=t&&t===this.ownerID,v=Ut(h,s,c,p);return
p?(this.count=_,this.nodes=v,this):new
lt(t,_,v)},yt.prototype.get=function(t,e,r,n){for(var
i=this.entries,o=0,u=i.length;u>o;o++)if(X(r,i[o][0]))return i[o][1];return
n},yt.prototype.update=function(t,e,r,n,i,o,u){void 0===r&&(r=ot(n));var
s=i===yr;if(r!==this.keyHash)return
s?this:(c(u),c(o),Dt(this,t,e,r,[n,i]));for(var
a=this.entries,h=0,f=a.length;f>h&&!X(n,a[h][0]);h++);var
_=f>h;if(_?a[h][1]===i:s)return this;if(c(u),(s||!_)&&c(o),s&&2===f)return new
dt(t,this.keyHash,a[1^h]);var v=t&&t===this.ownerID,l=v?a:p(a);return
_?s?h===f-1?l.pop():l[h]=l.pop():l[h]=[n,i]:l.push([n,i]),v?(this.entries=l,this
):new yt(t,this.keyHash,l)},dt.prototype.get=function(t,e,r,n){return
X(r,this.entry[0])?this.entry[1]:n;
},dt.prototype.update=function(t,e,r,n,i,o,u){var
s=i===yr,a=X(n,this.entry[0]);return(a?i===this.entry[1]:s)?this:(c(u),s?void
c(o):a?t&&t===this.ownerID?(this.entry[1]=i,this):new
dt(t,this.keyHash,[n,i]):(c(o),Dt(this,t,e,ot(n),[n,i])))},pt.prototype.iterate=
yt.prototype.iterate=function(t,e){for(var
r=this.entries,n=0,i=r.length-1;i>=n;n++)if(t(r[e?i-n:n])===!1)return!1},vt.prot
otype.iterate=lt.prototype.iterate=function(t,e){for(var
r=this.nodes,n=0,i=r.length-1;i>=n;n++){var
o=r[e?i-n:n];if(o&&o.iterate(t,e)===!1)return!1}},dt.prototype.iterate=function(
t,e){return t(this.entry)},t(mt,S),mt.prototype.next=function(){for(var
t=this._type,e=this._stack;e;){var
r,n=e.node,i=e.index++;if(n.entry){if(0===i)return gt(t,n.entry)}else
if(n.entries){if(r=n.entries.length-1,r>=i)return
gt(t,n.entries[this._reverse?r-i:i])}else if(r=n.nodes.length-1,r>=i){var
o=n.nodes[this._reverse?r-i:i];if(o){if(o.entry)return
gt(t,o.entry);e=this._stack=wt(o,e)}continue}e=this._stack=this._stack.__prev}re
turn I()};var Jr,Nr=vr/4,Pr=vr/2,Hr=vr/4;t(Wt,rt),Wt.of=function(){return
this(arguments)},Wt.prototype.toString=function(){return this.__toString("List
[","]")},Wt.prototype.get=function(t,e){if(t=l(this,t),t>=0&&this.size>t){t+=thi
s._origin;var r=Qt(this,t);return r&&r.array[t&lr]}return
e},Wt.prototype.set=function(t,e){return
Ht(this,t,e)},Wt.prototype.remove=function(t){return
this.has(t)?0===t?this.shift():t===this.size-1?this.pop():this.splice(t,1):this}
,Wt.prototype.insert=function(t,e){return
this.splice(t,0,e)},Wt.prototype.clear=function(){return
0===this.size?this:this.__ownerID?(this.size=this._origin=this._capacity=0,this.
_level=pr,this._root=this._tail=null,this.__hash=void
0,this.__altered=!0,this):Pt()},Wt.prototype.push=function(){var
t=arguments,e=this.size;return
this.withMutations(function(r){Xt(r,0,e+t.length);for(var
n=0;t.length>n;n++)r.set(e+n,t[n])})},Wt.prototype.pop=function(){return
Xt(this,0,-1)},Wt.prototype.unshift=function(){var t=arguments;return
this.withMutations(function(e){Xt(e,-t.length);for(var
r=0;t.length>r;r++)e.set(r,t[r]);
})},Wt.prototype.shift=function(){return
Xt(this,1)},Wt.prototype.merge=function(){return Ft(this,void
0,arguments)},Wt.prototype.mergeWith=function(t){var
```

```
e=sr.call(arguments,1);return
Ft(this,t,e)},Wt.prototype.mergeDeep=function(){return
Ft(this,kt,arguments)},Wt.prototype.mergeDeepWith=function(t){var
e=sr.call(arguments,1);return
Ft(this,At(t),e)},Wt.prototype.setSize=function(t){return
Xt(this,0,t)},Wt.prototype.slice=function(t,e){var r=this.size;return
d(t,e,r)?this:Xt(this,m(t,r),g(e,r))},Wt.prototype.__iterator=function(t,e){var
r=0,n=Jt(this,e);return new S(function(){var e=n();return
e===Xr?I():z(t,r++,e)})},Wt.prototype.__iterate=function(t,e){for(var
r,n=0,i=Jt(this,e);(r=i())!==Xr&&t(r,n++,this)!==!1;);return
n},Wt.prototype.__ensureOwner=function(t){return
t===this.__ownerID?this:t?Nt(this._origin,this._capacity,this._level,this._root,
this._tail,t,this.__hash):(this.__ownerID=t,this)},Wt.isList=Bt;var
Vr="@@__IMMUTABLE_LIST__@@",Yr=Wt.prototype;Yr[Vr]=!0,Yr[_r]=Yr.remove,Yr.setIn=
Cr.setIn,Yr.deleteIn=Yr.removeIn=Cr.removeIn,Yr.update=Cr.update,Yr.updateIn=Cr.
updateIn,Yr.mergeIn=Cr.mergeIn,Yr.mergeDeepIn=Cr.mergeDeepIn,Yr.withMutations=Cr
.withMutations,Yr.asMutable=Cr.asMutable,Yr.asImmutable=Cr.asImmutable,Yr.wasAlt
ered=Cr.wasAltered,Ct.prototype.removeBefore=function(t,e,r){if(r===e?1<<e:0===t
his.array.length)return this;var n=r>>>e&lr;if(n>=this.array.length)return new
Ct([],t);var i,o=0===n;if(e>0){var
u=this.array[n];if(i=u&&u.removeBefore(t,e-pr,r),i===u&&o)return
this}if(o&&!i)return this;var s=Yt(this,t);if(!o)for(var
a=0;n>a;a++)s.array[a]=void 0;return
i&&(s.array[n]=i),s},Ct.prototype.removeAfter=function(t,e,r){if(r===(e?1<<e:0)|
|0===this.array.length)return this;var
n=r-1>>>e&lr;if(n>=this.array.length)return this;var i;if(e>0){var
o=this.array[n];if(i=o&&o.removeAfter(t,e-pr,r),i===o&&n===this.array.length-1)r
eturn this}var u=Yt(this,t);return u.array.splice(n+1),i&&(u.array[n]=i),u};var
Qr,Xr={};t(Zt,ct),Zt.of=function(){return
this(arguments)},Zt.prototype.toString=function(){return
this.__toString("OrderedMap {","}");
},Zt.prototype.get=function(t,e){var r=this._map.get(t);return void
0!==r?this._list.get(r)[1]:e},Zt.prototype.clear=function(){return
0===this.size?this:this.__ownerID?(this.size=0,this._map.clear(),this._list.clea
r(),this):ee()},Zt.prototype.set=function(t,e){return
re(this,t,e)},Zt.prototype.remove=function(t){return
re(this,t,yr)},Zt.prototype.wasAltered=function(){return
this._map.wasAltered()||this._list.wasAltered()},Zt.prototype.__iterate=function
(t,e){var r=this;return this._list.__iterate(function(e){return
e&&t(e[1],e[0],r)},e)},Zt.prototype.__iterator=function(t,e){return
this._list.fromEntrySeq().__iterator(t,e)},Zt.prototype.__ensureOwner=function(t
){if(t===this.__ownerID)return this;var
e=this._map.__ensureOwner(t),r=this._list.__ensureOwner(t);return
t?te(e,r,t,this.__hash):(this.__ownerID=t,this._map=e,this._list=r,this)},Zt.isO
rderedMap=$t,Zt.prototype[cr]=!0,Zt.prototype[_r]=Zt.prototype.remove;var
Fr;t(ne,x),ne.prototype.get=function(t,e){return
this._iter.get(t,e)},ne.prototype.has=function(t){return
this._iter.has(t)},ne.prototype.valueSeq=function(){return
this._iter.valueSeq()},ne.prototype.reverse=function(){var
t=this,e=he(this,!0);return this._useKeys||(e.valueSeq=function(){return
t._iter.toSeq().reverse()}),e},ne.prototype.map=function(t,e){var
r=this,n=ae(this,t,e);return this._useKeys||(n.valueSeq=function(){return
r._iter.toSeq().map(t,e)}),n},ne.prototype.__iterate=function(t,e){var
r,n=this;return this._iter.__iterate(this._useKeys?function(e,r){return
```

```
t(e,r,n)}:(r=e?De(this):0,function(i){return
t(i,e?--r:r++,n)}),e)},ne.prototype.__iterator=function(t,e){if(this._useKeys)re
turn this._iter.__iterator(t,e);var
r=this._iter.__iterator(wr,e),n=e?De(this):0;return new S(function(){var
i=r.next();return
i.done?i:z(t,e?--n:n++,i.value,i)})},ne.prototype[cr]=!0,t(ie,k),ie.prototype.in
cludes=function(t){return
this._iter.includes(t)},ie.prototype.__iterate=function(t,e){var
r=this,n=0;return this._iter.__iterate(function(e){return
t(e,n++,r)},e)},ie.prototype.__iterator=function(t,e){var
r=this._iter.__iterator(wr,e),n=0;
return new S(function(){var e=r.next();return
e.done?e:z(t,n++,e.value,e)})},t(oe,A),oe.prototype.has=function(t){return
this._iter.includes(t)},oe.prototype.__iterate=function(t,e){var r=this;return
this._iter.__iterate(function(e){return
t(e,e,r)},e)},oe.prototype.__iterator=function(t,e){var
r=this._iter.__iterator(wr,e);return new S(function(){var e=r.next();return
e.done?e:z(t,e.value,e.value,e)})},t(ue,x),ue.prototype.entrySeq=function(){retu
rn this._iter.toSeq()},ue.prototype.__iterate=function(t,e){var r=this;return
this._iter.__iterate(function(e){if(e){qe(e);var n=o(e);return
t(n?e.get(1):e[1],n?e.get(0):e[0],r)}},e)},ue.prototype.__iterator=function(t,e)
{var r=this._iter.__iterator(wr,e);return new S(function(){for(;;){var
e=r.next();if(e.done)return e;var n=e.value;if(n){qe(n);var i=o(n);return
z(t,i?n.get(0):n[0],i?n.get(1):n[1],e)}}})},ie.prototype.cacheResult=ne.prototyp
e.cacheResult=oe.prototype.cacheResult=ue.prototype.cacheResult=Oe,t(Ae,et),Ae.p
rototype.toString=function(){return this.__toString(Ke(this)+"
{","}")},Ae.prototype.has=function(t){return
this._defaultValues.hasOwnProperty(t)},Ae.prototype.get=function(t,e){if(!this.h
as(t))return e;var r=this._defaultValues[t];return
this._map?this._map.get(t,r):r},Ae.prototype.clear=function(){if(this.__ownerID)
return this._map&&this._map.clear(),this;var t=this.constructor;return
t._empty||(t._empty=je(this,zt()))},Ae.prototype.set=function(t,e){if(!this.has(
t))throw Error('Cannot set unknown key "'+t+'" on
'+Ke(this));if(this._map&&!this._map.has(t)){var
r=this._defaultValues[t];if(e===r)return this}var
n=this._map&&this._map.set(t,e);return
this.__ownerID||n===this._map?this:je(this,n)},Ae.prototype.remove=function(t){i
f(!this.has(t))return this;var e=this._map&&this._map.remove(t);return
this.__ownerID||e===this._map?this:je(this,e)},Ae.prototype.wasAltered=function(
){return this._map.wasAltered()},Ae.prototype.__iterator=function(t,e){var
n=this;return r(this._defaultValues).map(function(t,e){return
n.get(e)}).__iterator(t,e)},Ae.prototype.__iterate=function(t,e){
var n=this;return r(this._defaultValues).map(function(t,e){return
n.get(e)}).__iterate(t,e)},Ae.prototype.__ensureOwner=function(t){if(t===this.__
ownerID)return this;var e=this._map&&this._map.__ensureOwner(t);return
t?je(this,e,t):(this.__ownerID=t,this._map=e,this)};var
Gr=Ae.prototype;Gr[_r]=Gr.remove,Gr.deleteIn=Gr.removeIn=Cr.removeIn,Gr.merge=Cr
.merge,Gr.mergeWith=Cr.mergeWith,Gr.mergeIn=Cr.mergeIn,Gr.mergeDeep=Cr.mergeDeep
,Gr.mergeDeepWith=Cr.mergeDeepWith,Gr.mergeDeepIn=Cr.mergeDeepIn,Gr.setIn=Cr.set
In,Gr.update=Cr.update,Gr.updateIn=Cr.updateIn,Gr.withMutations=Cr.withMutations
,Gr.asMutable=Cr.asMutable,Gr.asImmutable=Cr.asImmutable,t(Le,nt),Le.of=function
(){return this(arguments)},Le.fromKeys=function(t){return
this(r(t).keySeq())},Le.prototype.toString=function(){return
this.__toString("Set {","}")},Le.prototype.has=function(t){return
```

```
this._map.has(t)},Le.prototype.add=function(t){return
We(this,this._map.set(t,!0))},Le.prototype.remove=function(t){return
We(this,this._map.remove(t))},Le.prototype.clear=function(){return
We(this,this._map.clear())},Le.prototype.union=function(){var
t=sr.call(arguments,0);return t=t.filter(function(t){return
0!==t.size}),0===t.length?this:0!==this.size||this.__ownerID||1!==t.length?this.
withMutations(function(e){for(var
r=0;t.length>r;r++)i(t[r]).forEach(function(t){return
e.add(t)})}):this.constructor(t[0])},Le.prototype.intersect=function(){var
t=sr.call(arguments,0);if(0===t.length)return this;t=t.map(function(t){return
i(t)});var e=this;return
this.withMutations(function(r){e.forEach(function(e){t.every(function(t){return
t.includes(e)})||r.remove(e)})})},Le.prototype.subtract=function(){var
t=sr.call(arguments,0);if(0===t.length)return this;t=t.map(function(t){return
i(t)});var e=this;return
this.withMutations(function(r){e.forEach(function(e){t.some(function(t){return
t.includes(e)})&&r.remove(e)})})},Le.prototype.merge=function(){return
this.union.apply(this,arguments)},Le.prototype.mergeWith=function(t){var
e=sr.call(arguments,1);return this.union.apply(this,e)},
Le.prototype.sort=function(t){return
Je(we(this,t))},Le.prototype.sortBy=function(t,e){return
Je(we(this,e,t))},Le.prototype.wasAltered=function(){return
this._map.wasAltered()},Le.prototype.__iterate=function(t,e){var r=this;return
this._map.__iterate(function(e,n){return
t(n,n,r)},e)},Le.prototype.__iterator=function(t,e){return
this._map.map(function(t,e){return
e}).__iterator(t,e)},Le.prototype.__ensureOwner=function(t){if(t===this.__ownerI
D)return this;var e=this._map.__ensureOwner(t);return
t?this.__make(e,t):(this.__ownerID=t,this._map=e,this)},Le.isSet=Te;var
Zr="@@__IMMUTABLE_SET__@@",$r=Le.prototype;$r[Zr]=!0,$r[_r]=$r.remove,$r.mergeDe
ep=$r.merge,$r.mergeDeepWith=$r.mergeWith,$r.withMutations=Cr.withMutations,$r.a
sMutable=Cr.asMutable,$r.asImmutable=Cr.asImmutable,$r.__empty=Ce,$r.__make=Be;v
ar tn;t(Je,Le),Je.of=function(){return
this(arguments)},Je.fromKeys=function(t){return
this(r(t).keySeq())},Je.prototype.toString=function(){return
this.__toString("OrderedSet {","}")},Je.isOrderedSet=Ne;var
en=Je.prototype;en[cr]=!0,en.__empty=He,en.__make=Pe;var
rn;t(Ve,rt),Ve.of=function(){return
this(arguments)},Ve.prototype.toString=function(){return this.__toString("Stack
[","]")},Ve.prototype.get=function(t,e){var
r=this._head;for(t=l(this,t);r&&t--;)r=r.next;return
r?r.value:e},Ve.prototype.peek=function(){return
this._head&&this._head.value},Ve.prototype.push=function(){if(0===arguments.leng
th)return this;for(var
t=this.size+arguments.length,e=this._head,r=arguments.length-1;r>=0;r--)e={value
:arguments[r],next:e};return
this.__ownerID?(this.size=t,this._head=e,this.__hash=void
0,this.__altered=!0,this):Qe(t,e)},Ve.prototype.pushAll=function(t){if(t=n(t),0=
==t.size)return this;ft(t.size);var e=this.size,r=this._head;return
t.reverse().forEach(function(t){e++,r={value:t,next:r}}),this.__ownerID?(this.si
ze=e,this._head=r,this.__hash=void
0,this.__altered=!0,this):Qe(e,r)},Ve.prototype.pop=function(){return
this.slice(1)},Ve.prototype.unshift=function(){return
this.push.apply(this,arguments)},Ve.prototype.unshiftAll=function(t){
```

```
return this.pushAll(t)},Ve.prototype.shift=function(){return
this.pop.apply(this,arguments)},Ve.prototype.clear=function(){return
0===this.size?this:this.__ownerID?(this.size=0,this._head=void
0,this.__hash=void
0,this.__altered=!0,this):Xe()},Ve.prototype.slice=function(t,e){if(d(t,e,this.s
ize))return this;var r=m(t,this.size),n=g(e,this.size);if(n!==this.size)return
rt.prototype.slice.call(this,t,e);for(var
i=this.size-r,o=this._head;r--;)o=o.next;return
this.__ownerID?(this.size=i,this._head=o,this.__hash=void
0,this.__altered=!0,this):Qe(i,o)},Ve.prototype.__ensureOwner=function(t){return

t===this.__ownerID?this:t?Qe(this.size,this._head,t,this.__hash):(this.__ownerID
=t,this.__altered=!1,this)},Ve.prototype.__iterate=function(t,e){if(e)return
this.reverse().__iterate(t);for(var
r=0,n=this._head;n&&t(n.value,r++,this)!==!1;)n=n.next;return
r},Ve.prototype.__iterator=function(t,e){if(e)return
this.reverse().__iterator(t);var r=0,n=this._head;return new
S(function(){if(n){var e=n.value;return n=n.next,z(t,r++,e)}return
I()})},Ve.isStack=Ye;var
nn="@@__IMMUTABLE_STACK__@@",on=Ve.prototype;on[nn]=!0,on.withMutations=Cr.withM
utations,on.asMutable=Cr.asMutable,on.asImmutable=Cr.asImmutable,on.wasAltered=C
r.wasAltered;var un;e.Iterator=S,Fe(e,{toArray:function(){ft(this.size);var
t=Array(this.size||0);return
this.valueSeq().__iterate(function(e,r){t[r]=e}),t},toIndexedSeq:function(){retu
rn new ie(this)},toJS:function(){return this.toSeq().map(function(t){return
t&&"function"==typeof t.toJS?t.toJS():t}).__toJS()},toJSON:function(){return
this.toSeq().map(function(t){return t&&"function"==typeof
t.toJSON?t.toJSON():t}).__toJS()},toKeyedSeq:function(){return new
ne(this,!0)},toMap:function(){return
ct(this.toKeyedSeq())},toObject:function(){ft(this.size);var t={};return
this.__iterate(function(e,r){t[r]=e}),t},toOrderedMap:function(){return
Zt(this.toKeyedSeq())},toOrderedSet:function(){return
Je(u(this)?this.valueSeq():this)},toSet:function(){return
Le(u(this)?this.valueSeq():this)},toSetSeq:function(){return new oe(this);
},toSeq:function(){return
s(this)?this.toIndexedSeq():u(this)?this.toKeyedSeq():this.toSetSeq()},toStack:f
unction(){return Ve(u(this)?this.valueSeq():this)},toList:function(){return
Wt(u(this)?this.valueSeq():this)},toString:function(){return"[Iterable]"},__toSt
ring:function(t,e){return 0===this.size?t+e:t+"
"+this.toSeq().map(this.__toStringMapper).join(", ")+"
"+e},concat:function(){var t=sr.call(arguments,0);return
be(this,ye(this,t))},includes:function(t){return this.some(function(e){return
X(e,t)})},entries:function(){return
this.__iterator(Sr)},every:function(t,e){ft(this.size);var r=!0;return
this.__iterate(function(n,i,o){return t.call(e,n,i,o)?void
0:(r=!1,!1)}),r},filter:function(t,e){return
be(this,fe(this,t,e,!0))},find:function(t,e,r){var n=this.findEntry(t,e);return
n?n[1]:r},findEntry:function(t,e){var r;return
this.__iterate(function(n,i,o){return t.call(e,n,i,o)?(r=[i,n],!1):void
0}),r},findLastEntry:function(t,e){return
this.toSeq().reverse().findEntry(t,e)},forEach:function(t,e){return
ft(this.size),this.__iterate(e?t.bind(e):t)},join:function(t){ft(this.size),t=vo
id 0!==t?""+t:",";var e="",r=!0;return
this.__iterate(function(n){r?r=!1:e+=t,e+=null!==n&&void
```

```
0!==n?""+n:""}),e},keys:function(){return
this.__iterator(gr)},map:function(t,e){return
be(this,ae(this,t,e))},reduce:function(t,e,r){ft(this.size);var n,i;return
arguments.length<2?i=!0:n=e,this.__iterate(function(e,o,u){i?(i=!1,n=e):n=t.call
(r,n,e,o,u)}),n},reduceRight:function(t,e,r){var
n=this.toKeyedSeq().reverse();return
n.reduce.apply(n,arguments)},reverse:function(){return
be(this,he(this,!0))},slice:function(t,e){return
be(this,pe(this,t,e,!0))},some:function(t,e){return!this.every($e(t),e)},sort:fu
nction(t){return be(this,we(this,t))},values:function(){return
this.__iterator(wr)},butLast:function(){return
this.slice(0,-1)},isEmpty:function(){return void
0!==this.size?0===this.size:!this.some(function(){return!0})},count:function(t,e
){return v(t?this.toSeq().filter(t,e):this)},countBy:function(t,e){return
ce(this,t,e)},equals:function(t){
return F(this,t)},entrySeq:function(){var t=this;if(t._cache)return new
j(t._cache);var e=t.toSeq().map(Ze).toIndexedSeq();return
e.fromEntrySeq=function(){return t.toSeq()},e},filterNot:function(t,e){return
this.filter($e(t),e)},findLast:function(t,e,r){return
this.toKeyedSeq().reverse().find(t,e,r)},first:function(){return
this.find(y)},flatMap:function(t,e){return
be(this,me(this,t,e))},flatten:function(t){return
be(this,de(this,t,!0))},fromEntrySeq:function(){return new
ue(this)},get:function(t,e){return this.find(function(e,r){return X(r,t)},void
0,e)},getIn:function(t,e){for(var r,n=this,i=ke(t);!(r=i.next()).done;){var
o=r.value;if(n=n&&n.get?n.get(o,yr):yr,n===yr)return e}return
n},groupBy:function(t,e){return _e(this,t,e)},has:function(t){return
this.get(t,yr)!==yr},hasIn:function(t){return
this.getIn(t,yr)!==yr},isSubset:function(t){return t="function"==typeof
t.includes?t:e(t),this.every(function(e){return
t.includes(e)})},isSuperset:function(t){return t="function"==typeof
t.isSubset?t:e(t),t.isSubset(this)},keySeq:function(){return
this.toSeq().map(Ge).toIndexedSeq()},last:function(){return
this.toSeq().reverse().first()},max:function(t){return
Se(this,t)},maxBy:function(t,e){return Se(this,e,t)},min:function(t){return
Se(this,t?tr(t):nr)},minBy:function(t,e){return
Se(this,e?tr(e):nr,t)},rest:function(){return
this.slice(1)},skip:function(t){return
this.slice(Math.max(0,t))},skipLast:function(t){return
be(this,this.toSeq().reverse().skip(t).reverse())},skipWhile:function(t,e){retur
n be(this,le(this,t,e,!0))},skipUntil:function(t,e){return
this.skipWhile($e(t),e)},sortBy:function(t,e){return
be(this,we(this,e,t))},take:function(t){return
this.slice(0,Math.max(0,t))},takeLast:function(t){return
be(this,this.toSeq().reverse().take(t).reverse())},takeWhile:function(t,e){retur
n be(this,ve(this,t,e))},takeUntil:function(t,e){return
this.takeWhile($e(t),e)},valueSeq:function(){return
this.toIndexedSeq()},hashCode:function(){return
this.__hash||(this.__hash=ir(this))}});var
sn=e.prototype;sn[ar]=!0,sn[br]=sn.values,
sn.__toJS=sn.toArray,sn.__toStringMapper=er,sn.inspect=sn.toSource=function(){re
turn""+this},sn.chain=sn.flatMap,sn.contains=sn.includes,function(){try{Object.d
efineProperty(sn,"length",{get:function(){if(!e.noLengthWarning){var
t;try{throw Error()}catch(r){t=r.stack}if(-1===t.indexOf("_wrapObject"))return
```

```
console&&console.warn&&console.warn("iterable.length has been deprecated, use
iterable.size or iterable.count(). This warning will become a silent error in a
future version. "+t),this.size}}})}catch(t){}}(),Fe(r,{flip:function(){return
be(this,se(this))},findKey:function(t,e){var r=this.findEntry(t,e);return
r&&r[0]},findLastKey:function(t,e){return
this.toSeq().reverse().findKey(t,e)},keyOf:function(t){return
this.findKey(function(e){return X(e,t)})},lastKeyOf:function(t){return
this.findLastKey(function(e){return X(e,t)})},mapEntries:function(t,e){var
r=this,n=0;return be(this,this.toSeq().map(function(i,o){return
t.call(e,[o,i],n++,r)}).fromEntrySeq())},mapKeys:function(t,e){var
r=this;return be(this,this.toSeq().flip().map(function(n,i){return
t.call(e,n,i,r)}).flip())}});var
an=r.prototype;an[hr]=!0,an[br]=sn.entries,an.__toJS=sn.toObject,an.__toStringMa
pper=function(t,e){return JSON.stringify(e)+":
"+er(t)},Fe(n,{toKeyedSeq:function(){return new
ne(this,!1)},filter:function(t,e){return
be(this,fe(this,t,e,!1))},findIndex:function(t,e){var
r=this.findEntry(t,e);return r?r[0]:-1},indexOf:function(t){var
e=this.toKeyedSeq().keyOf(t);return void
0===e?-1:e},lastIndexOf:function(t){var
e=this.toKeyedSeq().reverse().keyOf(t);return void
0===e?-1:e},reverse:function(){return
be(this,he(this,!1))},slice:function(t,e){return
be(this,pe(this,t,e,!1))},splice:function(t,e){var
r=arguments.length;if(e=Math.max(0|e,0),0===r||2===r&&!e)return
this;t=m(t,0>t?this.count():this.size);var n=this.slice(0,t);return
be(this,1===r?n:n.concat(p(arguments,2),this.slice(t+e)))},findLastIndex:functio
n(t,e){var r=this.toKeyedSeq().findLastKey(t,e);return void
0===r?-1:r},first:function(){return this.get(0)},flatten:function(t){return
be(this,de(this,t,!1));
},get:function(t,e){return t=l(this,t),0>t||this.size===1/0||void
0!==this.size&&t>this.size?e:this.find(function(e,r){return r===t},void
0,e)},has:function(t){return t=l(this,t),t>=0&&(void
0!==this.size?this.size===1/0||this.size>t:-1!==this.indexOf(t))},interpose:func
tion(t){return be(this,ge(this,t))},interleave:function(){var
t=[this].concat(p(arguments)),e=Ie(this.toSeq(),k.of,t),r=e.flatten(!0);return
e.size&&(r.size=e.size*t.length),be(this,r)},last:function(){return
this.get(-1)},skipWhile:function(t,e){return
be(this,le(this,t,e,!1))},zip:function(){var
t=[this].concat(p(arguments));return
be(this,Ie(this,rr,t))},zipWith:function(t){var e=p(arguments);return
e[0]=this,be(this,Ie(this,t,e))}}),n.prototype[fr]=!0,n.prototype[cr]=!0,Fe(i,{g
et:function(t,e){return this.has(t)?t:e},includes:function(t){return
this.has(t)},keySeq:function(){return
this.valueSeq()}}),i.prototype.has=sn.includes,i.prototype.contains=i.prototype.
includes,Fe(x,r.prototype),Fe(k,n.prototype),Fe(A,i.prototype),Fe(et,r.prototype
),Fe(rt,n.prototype),Fe(nt,i.prototype);var
hn={Iterable:e,Seq:O,Collection:tt,Map:ct,OrderedMap:Zt,List:Wt,Stack:Ve,Set:Le,
OrderedSet:Je,Record:Ae,Range:$,Repeat:G,is:X,fromJS:H};return hn})();

let List = {};let Map = {};

List.length = (function(xs152955) { return (
    (function() {
```

```
        let aux987669 = (function(c519347,ys744291) { return ((function() {
return (function() {
        let res_909776
        if ((ys744291).isEmpty()) {
            res_909776 = (function() { return c519347 })()
        } else {
            res_909776 = (function() { return aux987669((c519347 +
1.),(ys744291).delete(0)) })()
        }
        return res_909776
    })() })()) })
        return aux987669(0.,xs152955);
    })()) })
List.nth = (function(xs58804,n909783) { return ((function() { return
(function() {
        let res_96617
        if ((n909783 > List.length(xs58804))) {
            res_96617 = (function() { return (function() { throw
"IndexOutOfBounds" })() })()
        } else {
            res_96617 = (function() { return (function() {
        let res_872215
        if ((n909783 === 0.)) {
            res_872215 = (function() { return (xs58804).get(0) })()
        } else {
            res_872215 = (function() { return
List.nth((xs58804).delete(0),(n909783 - 1.)) })()
        }
        return res_872215
    })() })()
        }
        return res_96617
    })() })()) })
List.filter = (function(pred323060,xs676720) { return ((function() { return
(function() {
        let res_854986
        if ((xs676720).isEmpty()) {
            res_854986 = (function() { return xs676720 })()
        } else {
            res_854986 = (function() { return (function() {
        let res_871586
        if (pred323060((xs676720).get(0))) {
            res_871586 = (function() { return
(List.filter(pred323060,(xs676720).delete(0))).insert(0, (xs676720).get(0)) })()
        } else {
            res_871586 = (function() { return
List.filter(pred323060,(xs676720).delete(0)) })()
        }
        return res_871586
    })() })()
        }
        return res_854986
    })() })()) })
List.range = (function(start189577,end725273) { return ((function() { return
```

```
(function() {
        let res_407063
        if ((start189577 >= end725273)) {
            res_407063 = (function() { return Immutable.List.of() })()
        } else {
            res_407063 = (function() { return (List.range((start189577 +
1.),end725273)).insert(0, start189577) })()
        }
        return res_407063
    })() })()) })
List.rev = (function(xs371692) { return (
    (function() {
        let aux217189 = (function(acc257347,ys782934) { return ((function() {
return (function() {
        let res_350619
        if ((ys782934).isEmpty()) {
            res_350619 = (function() { return acc257347 })()
        } else {
            res_350619 = (function() { return aux217189((acc257347).insert(0,
(ys782934).get(0)),(ys782934).delete(0)) })()
        }
        return res_350619
    })() })()) })
        return aux217189(Immutable.List.of(),xs371692);
    })()) })
List.concat = (function(xs707750,ys227059) { return (
    (function() {
        let aux653859 = (function(as744495,bs867682) { return ((function() {
return (function() {
        let res_198607
        if ((as744495).isEmpty()) {
            res_198607 = (function() { return bs867682 })()
        } else {
            res_198607 = (function() { return
aux653859((as744495).delete(0),(bs867682).insert(0, (as744495).get(0))) })()
        }
        return res_198607
    })() })()) })
        return aux653859(List.rev(xs707750),ys227059);
    })()) })
List.iter = (function(f67879,xs104810) { return ((function() { return
(function() {
        let res_724643
        if ((xs104810).isEmpty()) {
            res_724643 = (function() { return (undefined) })()
        } else {
            res_724643 =
    (function() {
        let dontcare700770 = f67879((xs104810).get(0))
        return List.iter(f67879,(xs104810).delete(0));
    })()
        }
        return res_724643
    })() })()) })
```

```
List.map = (function(fn542827,xs991843) { return ((function() { return
(function() {
        let res_926600
        if ((xs991843).isEmpty()) {
            res_926600 = (function() { return Immutable.List.of() })()
        } else {
            res_926600 = (function() { return
(List.map(fn542827,(xs991843).delete(0))).insert(0,
fn542827((xs991843).get(0))) })()
        }
        return res_926600
    })() })()) })
List.fold_left = (function(fn365047,acc563235,xs989538) { return (
    (function() {
        let aux947577 = (function(acc444668,xs499293) { return ((function() {
return (function() {
        let res_914479
        if ((xs499293).isEmpty()) {
            res_914479 = (function() { return acc444668 })()
        } else {
            res_914479 = (function() { return
aux947577(fn365047(acc444668,(xs499293).get(0)),(xs499293).delete(0)) })()
        }
        return res_914479
    })() })()) })
        return aux947577(acc563235,xs989538);
    })()) })
List.insert = (function(xs574345,x507443,pos39928) { return ((function() {
return (function() {
        let res_844039
        if (((pos39928 > List.length(xs574345)) || (pos39928 < 0.))) {
            res_844039 = (function() { return (function() { throw
"IndexOutOfBounds" })() })()
        } else {
            res_844039 =
    (function() {
        let aux774015 = (function(count147655,acc372920,xs761990) { return
((function() { return (function() {
        let res_979106
        if ((count147655 === pos39928)) {
            res_979106 = (function() { return
List.concat(List.rev(acc372920),(xs761990).insert(0, x507443)) })()
        } else {
            res_979106 = (function() { return aux774015((count147655 +
1.),(acc372920).insert(0, (xs761990).get(0)),(xs761990).delete(0)) })()
        }
        return res_979106
    })() })()) })
        return aux774015(0.,Immutable.List.of(),xs574345);
    })()
        }
        return res_844039
    })() })()) })
List.remove = (function(xs83945,pos446950) { return ((function() { return
```

```
(function() {
        let res_976121
        if (((pos446950 > List.length(xs83945)) || (pos446950 < 0.))) {
            res_976121 = (function() { return (function() { throw
"IndexOutOfBounds" })() })()
        } else {
            res_976121 =
    (function() {
        let aux745678 = (function(count281655,acc176861,xs897437) { return
((function() { return (function() {
        let res_855353
        if ((count281655 === pos446950)) {
            res_855353 = (function() { return
List.concat(List.rev(acc176861),(xs897437).delete(0)) })()
        } else {
            res_855353 = (function() { return aux745678((count281655 +
1.),(acc176861).insert(0, (xs897437).get(0)),(xs897437).delete(0)) })()
        }
        return res_855353
    })() })()) })
        return aux745678(0.,Immutable.List.of(),xs83945);
    })()
        }
        return res_976121
    })() })()) })
List.sort = (function(compare812170,xs14590) { return ((function() { return
(function() {
        let res_507831
        if ((Immutable.is(xs14590, Immutable.List.of()))) {
            res_507831 = (function() { return Immutable.List.of() })()
        } else {
            res_507831 =
    (function() {
        let l1759577 = List.filter((function(x634426) { return ((function() {
return compare812170(x634426,(xs14590).get(0)) })()) }),(xs14590).delete(0));
let l2673199 = List.filter((function(x88462) { return ((function() { return
!compare812170(x88462,(xs14590).get(0)) })()) }),(xs14590).delete(0))
        return
List.concat(List.sort(compare812170,l1759577),(List.sort(compare812170,l2673199)
).insert(0, (xs14590).get(0)));
    })()
        }
        return res_507831
    })() })()) })

Map.count = (function(m41774) { return ((function() { return
List.length(Immutable.fromJS(Array.from((m41774).keys()))) })()) })
Map.values = (function(m434843) { return (
    (function() {
        let aux844523 = (function(ks137159) { return ((function() { return
(function() {
        let res_169927
        if ((ks137159).isEmpty()) {
            res_169927 = (function() { return Immutable.List.of() })()
```

```
        } else {
            res_169927 = (function() { return
(aux844523((ks137159).delete(0))).insert(0,
(m434843).get(((ks137159).get(0)).toString())) })()
        }
        return res_169927
    })() })()) })
        return aux844523(Immutable.fromJS(Array.from((m434843).keys())));
    })()) })
Map.merge = (function(m1906348,m2286175) { return (
    (function() {
        let aux49730 = (function(m22984,mapkeys81025) { return ((function() {
return (function() {
        let res_415856
        if ((mapkeys81025).isEmpty()) {
            res_415856 = (function() { return m22984 })()
        } else {
            res_415856 =
    (function() {
        let key124638 = (mapkeys81025).get(0);
let value855409 = (m2286175).get((key124638).toString());
let newm599965 = (m22984).set((key124638).toString(), value855409)
        return aux49730(newm599965,(mapkeys81025).delete(0));
    })()
        }
        return res_415856
    })() })()) })
        return
aux49730(m1906348,Immutable.fromJS(Array.from((m2286175).keys())));
    })()) })

var num_to_string = function (x) { return x.toString(); };
var print_me = function(m) { console.log(m); };
// printing utils
var print_string = print_me;
var print_num = print_me;
var print_bool = print_me;
var print = print_me;
// generated code follows
let gcd42828 = (function(a96344,b37922) { return ((function() { return
(function() {
        let res_522993
        if ((a96344 === b37922)) {
            res_522993 = (function() { return a96344 })()
        } else {
            res_522993 = (function() { return (function() {
        let res_796446
        if ((a96344 > b37922)) {
            res_796446 = (function() { return gcd42828((a96344 -
b37922),b37922) })()
        } else {
            res_796446 = (function() { return gcd42828((b37922 -
a96344),a96344) })()
        }
```

```
        return res_796446
    })() })()
        }
        return res_522993
    })() })()) });
print_num(gcd42828(12.,8.))
```

Testing Roles

Prakhar wrote the testing infrastructure for both unit and integration tests including test reporting, file structure, automation and Travis CI setup. Gaurang and Bahul worked on writing out test cases for the type-checker and lastly, Ayush worked on testing plan and testing out the standard library.

Roles and Responsibilities

There were no specific roles assigned among us. We set up a room in one of our rooms where we used to meet once or sometimes twice a week and code on person's laptop. We didnt work individually on any part of the project. It was a collaborative effort.

Github Usernames

- Ayush Jain - ayushjain7
- Bahul Jain - bahuljain
- Gaurang Sadekar - gaurangsadekar
- Prakhar Srivastav - prakhar1989

Software Development Environment

For the duration of the project we used the following development environment:

- Operating System - We used both Ubuntu 14.04 and Mac OS systems for the development of our project.
- OCaml - We used the OPAM to install OCaml and the associated packages including utop, core, batteries and menhir.
- Slack - We used slack for communication and planning for our meetings and offline discussions and follow-ups.
- Github - Github was used for version control throughout our project. Since, all of us our familiar with github it made collaboration easy. We maintained separate branches to maintain our own versions of code.
- Vim/Merlin - We used vim with merlin extension for OCaml as our text editor. Merlin is really helpful. It helps with suggestions, auto-completion and any syntactic or semantic errors in the code.
- Latex/Pandoc/Markdown - We used these softwares for documentation so that quality is maintained.

Project Timeline

Date	Projected Milestone
February 5	Project Proposal & LRM
February 15	Scanner
February 31	Parser
March 15	Semantic Analysis
March 31	Codegen & Hello World
April 15	Test Suite
April 22	Bug fixes
May 5	Lists, Maps and supported library
May 11	Final Report

Architectural Design

The JSJS compiler runs an input program through the following major components one after the other.

- **Scanner**
- **Parser**
- **Type Inference and Semantic Checking**
- **Code Generation**

All these components are implemented purely in OCaml. The entry point to the JSJS compiler is `driver.ml`, which reads input from a file and invokes the above components in order.
The scanner (`scanner.mll`) tokenizes the input, and the parser (`parser.mly`) checks for it syntax errors. If there are no syntax errors, the parser emits an Abstract Syntax Tree (AST) structure, using types defined in `ast.mli`. The AST is then type inferred and semantically checked in `typecheck.ml`. If there are no semantic errors in the program, the inferred AST is passed to code generation (`codegen.ml`), which converts it to Javascript code expression by expression.

Once the Javascript code is generated in `out.js`, it can be run in the terminal using `node out.js` or in the browser as well.

Scanner:

The scanner takes the raw program file as input, and outputs the tokens present in the program. These include variable identifiers, keywords operators, literals and important symbols. It also removes any additional white spaces and ignores user comments. The scanner gives an error if any unrecognized symbols or incorrect patterns are used in the program.

Parser:

The parser takes the tokenized program from the scanner and matches it token by token against the grammar defined. If there is any mismatch, the parser raises a syntax error and causes the compiler to exit. If there are no syntax errors, the parser emits an Abstract Syntax Tree for the program. In JSJS, a program is a list of expressions. The types that compose the Abstract Syntax Tree are all defined in `ast.mli`. This AST structure is then passed ahead for semantic analysis.

Type Inference and Semantic Checking:

The semantic analyzer is the most sophisticated component of the JSJS compiler. The code is implemented in `typecheck.ml`. It takes an AST and executes the following semantic checks on it: - checks if values are

not redefined in the current scope. - checks if Javascript or JSJS keywords are redefined in the code. - checks if correct module names and member expressions are referenced in code. - runs Hindley - Milner Type Inference (Algorithm W) on the AST, to concretely infer types and enforce semantic constraints.

Hindley - Milner Type Inference

Hindley - Milner Type Inference or Algorithm W is an algorithm which has the ability to deduce the type of a given program AST without the need of any annotations or information provided by the programmer about the types involved.
The type inference algorithm we have implemented in JSJS infers all types in the AST of an expression with the following 4 steps: - **Annotate**
In this step, we either annotate each expression and sub-expression in the AST with any type annotations that the programmer provides, or with a new placeholder type to be unified or resolved later.

- **Collect**
  In this step, we enforce all the semantic constraints associated with each type. Each sub-expression within an expression also has to be collected recursively. We impose constraints on the enclosing type first as at any point, it gives the most information about what type its subexpression could possibly have. An example of collect is if we get the AST for the expression `x + y`, we need to enforce that the result of the this operation is of type `num` and the operands `x` and `y` are of type `num`. Constraints are always applied from outwards in.
  If we have 3 placeholder types `C`, `B` and `A` associated with `x + y`, `x` and `y` respectively, then we get constraints in collect of the form `[(A, num), (B, num), (C, num)]`. We do this because the substitution and unification algorithm work from right to left.

Once we have the constraints ready, we pass these list of constraints to the Type Unification Algorithm, which comprises of 2 steps, substitute and unify.

- **Substitute**
  In this step we take a substitution rule that tells us to replace all occurrences of type placeholder x with u, in a primitive type t. We are required to apply this substitution to t recursively, so if t is a composite type that contains multiple occurrences of x then at every occurrence of x, a u is to be substituted. We apply the substitutions to the list of constraints, folding from right to left.
- **Unification**
  Type unification is done using 2 mutually recursive methods, `unify` and `unify_one`.
  - Unify: The unify function takes a bunch of constraints it obtained from the collect method and turns them into substitutions. Since constraints on the right have more precedence we start from the rightmost constraint and unify it by calling the unify_one function. unify_one transforms the constraint to a substitution.
  - Unify One: In this method, a constraint is converted to a substitution using type unification rules.

Using these 2 steps, we infer all the types when we unify all the constraints in the list passed from the Collect step.

If the unification passes for all constraints, the program is semantically correct. We then pass the AST with all inferred types to code generation. Inferring all types correctly using HMTI is a mechanism to type check the program correctly, as type information is not useful in code generation to Javascript.

Codegen:

Using this component of the compiler, we generate Javascript code from the AST of the program. Since Javascript is an **untyped** language we don't need any of the inferred type information for code generation. Translating JSJS to JS requires some tricks in code generation because we implemented have JSJS as a completely expression oriented language.

In JS, `if-else` is a statement, but in JSJS `if-then-else` is an expression. The same applies with blocks as well. To deal with this, we wrap all our `if-then-else` and blocks within anonymous function calls, so that they always return a result.

Another issue we noticed with doing a naive Javascript translation was Javascript's inconsistent scoping rules. In Javascript, 2 types of declarations are allowed, `let` and `var`. `let` binds a variable to the nearest enclosing block (essentially a local definition) and `var` binds to the nearest function block. Which means it is accessible everywhere in the function in which it is defined, rather than being accessible *after* its declaration. Since we want proper global and local scoping behavior, we need to do name mangling with some enviroment information. The environment for codegen is simply a map of variable names and their aliases, to replace the names with aliases where they are referenced while generating JS.
This allowed us to replicate typical scoping behavior present in statically scoped languages.

Lessons Learnt

Prakhar

Try to work with people who are accustomed to working with. Start early and try to grok MicroC's implementation as early in the course as you can. Have a strong test suite, invest in your Makefile and use Github issues for planning. Oh, and never submit a pull-request without attaching a cat gif.

Ayush

Working on such a big project over the semester made me realize how important it is to be consistent in your work. We met up at least once a week to work and once with our TA to make sure that we were on track. We always knew what we were going to work on and when we left we decided what we were to study before meeting the next time so that everybody had context. This helped save precious time.

It is also very important to choose a good team. People you can work with. Having a good team can go a long way in achieving the goal of your project. You will really enjoy the class and the project if you choose the team well. Also, I really liked the language. The stuff you can do with OCaml and it's type inference and pattern matching is really cool. I can totally imagine how bloated the project would be if were to do this in C++ or Java. So learn the language as early into the semester as you can.

Gaurang

Working on JSJS was an extremely enlightening experience. Having never done functional programming before, implementing functional features in our language allowed me to learn these concepts in real depth. Particularly, when we implemented Hindley - Milner Type Inference, we realized how reliant we were on OCaml's type system. Throughout the project, I developed a deeper appreciation for programming languages in general, and the complexities involved in providing many of the features in them that we take for granted. Definitely the best project I've worked on at Columbia, in large part due to my team. We were all passionate about JSJS from day 1. I think its really important to have a team that pushes you to work harder and who are all motivated to work on the project regularly. Also, its best to work with people you have teamed up with in the past and people who have similar schedules as you, so that there aren't logistic problems. Besides that, start early and work together on the project at least once a week.

Bahul

Learning and using OCaml for the course and project was the perhaps one of the most enlightening experience in I have had in Columbia. Besides that I really enjoyed working on a different type of software development project. Having a concrete vision for the project right from the beginning really helped us a lot. Collective intelligence goes a long way in not just making decisions but also in writing code. Our way of working on the project was quite different but it worked out really well. Sitting together in front of a big monitor while coding every week helped us tackle a lot of the problems with ease. A comprehensive test suite is a blessing in disguise. Ok Bye!

Appendix

Code

```
.
  Makefile
  README.md
  _tags
  examples
      99bottles.jsjs
      README.md
      countPrimes.jsjs
      infer.jsjs
      infer.jsjs~
      just_exprs.jsjs
      list.jsjs
      list_take.jsjs
      many_anons.jsjs
      map.jsjs
      math.jsjs
      newinfer.jsjs
      operators.jsjs
      sort.jsjs
      testcodegen.jsjs
      typecheck.jsjs
  lib
      list.jsjs
      map.jsjs
  log.txt
  logo.png
  out.js
  run-tests.out
  src
      ast.mli
      codegen.ml
      driver.ml
      exceptions.ml
      lib.ml
      parser.mly
      scanner.mll
      semantic.ml
      stringify.ml
```

```
    typecheck.ml
temp.out
test
    compiler-tests
        fail-annotated_lists.jsjs
        fail-annotated_lists.out
        fail-annotated_num.jsjs
        fail-annotated_num.out
        fail-anonymousfnapp.jsjs
        fail-anonymousfnapp.out
        fail-anonymousfnapp2.jsjs
        fail-anonymousfnapp2.out
        fail-assign1.jsjs
        fail-assign1.out
        fail-assign3.jsjs
        fail-assign3.out
        fail-booleanops.jsjs
        fail-booleanops.out
        fail-cons.jsjs
        fail-cons.out
        fail-declare_keyword.jsjs
        fail-declare_keyword.out
        fail-disposablevar.jsjs
        fail-disposablevar.out
        fail-duplicate_args.jsjs
        fail-duplicate_args.out
        fail-duplicate_variables.jsjs
        fail-duplicate_variables.out
        fail-duplicate_vars_in_functions.jsjs
        fail-duplicate_vars_in_functions.out
        fail-equality_type.jsjs
        fail-equality_type.out
        fail-faulty_iter.jsjs
        fail-faulty_iter.out
        fail-genericfns.jsjs
        fail-genericfns.out
        fail-ifelse1.jsjs
        fail-ifelse1.out
        fail-ifelse2.jsjs
        fail-ifelse2.out
        fail-ifelse3.jsjs
        fail-ifelse3.out
        fail-incorrect_block_usage.jsjs
        fail-incorrect_block_usage.out
        fail-invalid_module.jsjs
        fail-invalid_module.out
        fail-invalid_return.jsjs
        fail-invalid_return.out
        fail-js_keyword.jsjs
        fail-js_keyword.out
        fail-list_booleanops.jsjs
        fail-list_booleanops.out
        fail-list_concat.jsjs
        fail-list_concat.out
```

```
fail-list_filter.jsjs
fail-list_filter.out
fail-list_foldleft.jsjs
fail-list_foldleft.out
fail-list_insert.jsjs
fail-list_insert.out
fail-list_lit.jsjs
fail-list_lit.out
fail-list_sort_comp.jsjs
fail-list_sort_comp.out
fail-list_sort_comptype.jsjs
fail-list_sort_comptype.out
fail-listequality2.jsjs
fail-listequality2.out
fail-map_booleanops.jsjs
fail-map_booleanops.out
fail-map_del.jsjs
fail-map_del.out
fail-map_get.jsjs
fail-map_get.out
fail-map_has.jsjs
fail-map_has.out
fail-map_key_type.jsjs
fail-map_key_type.out
fail-map_key_type2.jsjs
fail-map_key_type2.out
fail-map_set1.jsjs
fail-map_set1.out
fail-map_set2.jsjs
fail-map_set2.out
fail-map_value_type.jsjs
fail-map_value_type.out
fail-map_value_type2.jsjs
fail-map_value_type2.out
fail-mapinit.jsjs
fail-mapinit.out
fail-mapmerge.jsjs
fail-mapmerge.out
fail-mapvalues.jsjs
fail-mapvalues.out
fail-recursive.jsjs
fail-recursive.out
fail-recursive2.jsjs
fail-recursive2.out
fail-recursive3.jsjs
fail-recursive3.out
fail-recursivefnapp.jsjs
fail-recursivefnapp.out
fail-recursivelistfnapp.jsjs
fail-recursivelistfnapp.out
fail-scoping_order.jsjs
fail-scoping_order.out
fail-unaryops.jsjs
fail-unaryops.out
```

```
pass-annotated_type.jsjs
pass-annotated_type.out
pass-anonymousfnapp.jsjs
pass-anonymousfnapp.out
pass-assign2.jsjs
pass-assign2.out
pass-booleanops.jsjs
pass-booleanops.out
pass-booleanops2.jsjs
pass-booleanops2.out
pass-booleanprecedence.jsjs
pass-booleanprecedence.out
pass-cons.jsjs
pass-cons.out
pass-disposablevar.jsjs
pass-disposablevar.out
pass-empty_list.jsjs
pass-empty_list.out
pass-exception.jsjs
pass-exception.out
pass-fn_compose.jsjs
pass-fn_compose.out
pass-function_args.jsjs
pass-function_args.out
pass-generic_annotated_function.jsjs
pass-generic_annotated_function.out
pass-generic_annotated_list.jsjs
pass-generic_annotated_list.out
pass-generic_annotated_map.jsjs
pass-generic_annotated_map.out
pass-helloworld.jsjs
pass-helloworld.out
pass-identityfn.jsjs
pass-identityfn.out
pass-ifelse.jsjs
pass-ifelse.out
pass-list_booleanops.jsjs
pass-list_booleanops.out
pass-list_concat.jsjs
pass-list_concat.out
pass-list_filter.jsjs
pass-list_filter.out
pass-list_foldleft.jsjs
pass-list_foldleft.out
pass-list_insert.jsjs
pass-list_insert.out
pass-list_insertexception.jsjs
pass-list_insertexception.out
pass-list_iter.jsjs
pass-list_iter.out
pass-list_length.jsjs
pass-list_length.out
pass-list_map.jsjs
pass-list_map.out
```

```
pass-list_nthexception.jsjs
pass-list_nthexception.out
pass-list_range.jsjs
pass-list_range.out
pass-list_remove.jsjs
pass-list_remove.out
pass-list_rev.jsjs
pass-list_rev.out
pass-list_sort_custom_comp.jsjs
pass-list_sort_custom_comp.out
pass-list_sort_num.jsjs
pass-list_sort_num.out
pass-list_sort_str.jsjs
pass-list_sort_str.out
pass-listequality1.jsjs
pass-listequality1.out
pass-listequality3.jsjs
pass-listequality3.out
pass-map_booleanops.jsjs
pass-map_booleanops.out
pass-map_del.jsjs
pass-map_del.out
pass-map_get.jsjs
pass-map_get.out
pass-map_has.jsjs
pass-map_has.out
pass-map_keys.jsjs
pass-map_keys.out
pass-map_keys_values.jsjs
pass-map_keys_values.out
pass-map_set.jsjs
pass-map_set.out
pass-mapcount.jsjs
pass-mapcount.out
pass-mapequality1.jsjs
pass-mapequality1.out
pass-mapinit.jsjs
pass-mapinit.out
pass-mapmerge.jsjs
pass-mapmerge.out
pass-mapvalues.jsjs
pass-mapvalues.out
pass-math.jsjs
pass-math.out
pass-modulenamescoping.jsjs
pass-modulenamescoping.out
pass-moreexception.jsjs
pass-moreexception.out
pass-scoping.jsjs
pass-scoping.out
pass-scoping2.jsjs
pass-scoping2.out
pass-semi_annotated_function.jsjs
pass-semi_annotated_function.out
```

```
          pass-semi_annotated_map.jsjs
          pass-semi_annotated_map.out
          pass-unaryops.jsjs
          pass-unaryops.out
          pass-unit_equality.jsjs
          pass-unit_equality.out
          pass-unit_equality2.jsjs
          pass-unit_equality2.out
      parser-tests
          menhir-test.txt
          menhir.py
          test_parser.ml
      run.ml

6 directories, 260 files
```

src/scanner.mll

```
{
    open Parser
    open Lexing

    (* increments the line number *)
    let incr_lineno lexbuf =
        let curr_pos = lexbuf.lex_curr_p in
        lexbuf.lex_curr_p <- { curr_pos with
            pos_lnum = curr_pos.pos_lnum + 1;
            pos_bol = curr_pos.pos_cnum;
        }
    ;;
}

let digit = ['0'-'9']
let id = ['a'-'z'] ['a'-'z' 'A'-'Z' '0'-'9' '_']* ['?']? | "_"
let ws = [' ' '\r' '\t']
let number = digit+ '.'? digit*
let module_lit = ['A'-'Z'] ['a'-'z' 'A'-'Z']+
let string_lit = (([' '-'!' '#'-'[' ']'-'~'] | '\\' ['\\' '"' 'n' 'r' 't'])* as
s)

rule token =
    parse
    | ws                        { token lexbuf; }
    | '\n'                      { incr_lineno lexbuf; token lexbuf; }
    | '+'                       { PLUS }
    | '-'                       { MINUS }
    | '*'                       { MULTIPLY }
    | '/'                       { DIVIDE }
    | '%'                       { MODULUS }
    | '<'                       { LT }
    | '>'                       { GT }
    | '='                       { ASSIGN }
```

```
    | '!'                       { NOT }
    | '^'                       { CARET }
    | "/\\"                     { LAMBDA }
    | "<="                      { LTE }
    | ">="                      { GTE }
    | "=="                      { EQUALS }
    | "!="                      { NEQ }
    | "&&"                      { AND }
    | "||"                      { OR }
    | "//"                      { comment lexbuf; }
    | "val"                     { VAL }
    | "throw"                   { THROW }
    | "try"                     { TRY }
    | "catch"                   { CATCH }
    | "if"                      { IF }
    | "else"                    { ELSE }
    | "then"                    { THEN }
    | "true"                    { TRUE }
    | "false"                   { FALSE }
    | "num"                     { NUM }
    | "bool"                    { BOOL }
    | "unit"                    { UNIT }
    | "string"                  { STRING }
    | "list"                    { LIST }
    | "(-)"                     { UNIT_LIT }
    | "::"                      { CONS }
    | ':'                       { COLON }
    | '('                       { LPAREN }
    | ')'                       { RPAREN }
    | '.'                       { DOT }
    | '{'                       { LBRACE }
    | '}'                       { RBRACE }
    | ';'                       { SEMICOLON }
    | '['                       { LSQUARE }
    | ']'                       { RSQUARE }
    | ','                       { COMMA }
    | "=>"                      { FATARROW }
    | "->"                      { THINARROW }
    | ['A'-'Z'] as c            { GENERIC(Char.escaped c) }
    | number as num             { NUM_LIT(float_of_string num); }
    | '"' (string_lit as s) '"' { STR_LIT(s); }
    | id as ident               { ID(ident); }
    | module_lit  as m_lit      { MODULE_LIT(m_lit); }
    | _ as c                    { let line_no = lexbuf.lex_curr_p.pos_lnum + 1 i
n
                                  let char_no = lexbuf.lex_curr_p.pos_cnum - lex
buf.lex_curr_p.pos_bol in
                                  raise (Exceptions.IllegalCharacter(Char.escape
d c, line_no, char_no)) }
    | eof                       { EOF }

and comment =
    parse
```

```
    | '\n'                      { token lexbuf }
    | _                         { comment lexbuf }
```

src/parser.mly

```
%{
open Ast
%}

/* Tokens */
%token EOF
%token PLUS MINUS MULTIPLY DIVIDE MODULUS
%token LT LTE GT GTE EQUALS NEQ
%token AND OR NOT
%token TRY CATCH THROW
%token LPAREN RPAREN LBRACE RBRACE RSQUARE LSQUARE
%token ASSIGN LAMBDA
%token CARET CONS
%token VAL IF THEN ELSE TRUE FALSE
%token NUM LIST BOOL STRING UNIT
%token COLON SEMICOLON DOT FATARROW COMMA THINARROW

%token UNIT_LIT
%token <float>  NUM_LIT
%token <string> STR_LIT
%token <string> MODULE_LIT
%token <string> ID
%token <string>   GENERIC

/* associativity rules */
%nonassoc THROW
%nonassoc SINGLE
%nonassoc DOUBLE
%nonassoc ANON
%nonassoc DOT
%right ASSIGN
%right CONS
%left CARET
%left OR
%left AND
%left NOT
%left LTE GTE LT GT EQUALS NEQ
%left PLUS MINUS
%left MULTIPLY DIVIDE MODULUS
%left NEG
/* entry point */
%start program
%type <Ast.program> program

%%

program:
```

```
    | expr_list EOF                              { $1 }

delimited_expr:
    | expr SEMICOLON                             { $1 }

block:
    | LBRACE expr_list RBRACE                    { $2 }

expr_list:
    | exprs = nonempty_list(delimited_expr)      { exprs }

formals_opt:
    | opts = separated_list(COMMA, opt)          { opts }

opt:
    | ID                                         { ($1, TAny) }
    | ID COLON primitive                         { ($1, $3) }

primitive:
    | NUM                                        { TNum }
    | BOOL                                       { TBool }
    | STRING                                     { TString }
    | UNIT                                       { TUnit }
    | LPAREN args RPAREN THINARROW primitive     { TFun($2, $5) }
    | LIST primitive                             { TList($2) }
    | LT primitive COLON primitive GT            { TMap($2, $4) }
    | GENERIC                                    { T($1) }

args:
    | args = separated_list(COMMA, primitive) { args }

fun_literals:
    /* Return type not annotated */
    | LAMBDA LPAREN formals_opt RPAREN FATARROW expr %prec ANON {
        let formal_types = List.map snd $3 and ids = List.map fst $3 in
        FunLit(ids, Block([$6]), TFun(formal_types, TAny))
    }
    | LAMBDA LPAREN formals_opt RPAREN FATARROW block {
        let formal_types = List.map snd $3 and ids = List.map fst $3 in
        FunLit(ids, Block($6), TFun(formal_types, TAny))
    }
    /* Return type annotated */
    | LAMBDA LPAREN formals_opt RPAREN COLON primitive FATARROW expr %prec ANON
{
        let formal_types = List.map snd $3 and ids = List.map fst $3 in
        FunLit(ids, Block([$8]), TFun(formal_types, $6))
    }
    | LAMBDA LPAREN formals_opt RPAREN COLON primitive FATARROW block {
        let formal_types = List.map snd $3 and ids = List.map fst $3 in
        FunLit(ids, Block($8), TFun(formal_types, $6))
    }

literals:
```

```
    | NUM_LIT                                              { NumLit($1) }
    | TRUE                                                 { BoolLit(true) }
    | FALSE                                                { BoolLit(false) }
    | STR_LIT                                              { StrLit($1) }
    | ID                                                   { Val($1) }
    | UNIT_LIT                                             { UnitLit }
    | LSQUARE actuals_opt RSQUARE                          { ListLit($2) }
    | LBRACE kv_pairs RBRACE                               { MapLit($2) }
    | fun_literals                                         { $1 }

kv_pairs:
    | kv = separated_list(COMMA, kv_pair)                  { kv }

kv_pair:
    | expr COLON expr                                      { $1, $3 }

expr:
    | literals                                             { $1 }
    | assigns                                              { $1 }
    | LPAREN fun_literals SEMICOLON
      RPAREN LPAREN actuals_opt RPAREN                     { Call($2, $6) }
    | expr PLUS expr                                       { Binop($1, Add, $3) }
    | expr MINUS expr                                      { Binop($1, Sub, $3) }
    | expr MULTIPLY expr                                   { Binop($1, Mul, $3) }
    | expr DIVIDE expr                                     { Binop($1, Div, $3) }
    | expr MODULUS expr                                    { Binop($1, Mod, $3) }
    | expr CARET expr                                      { Binop($1, Caret, $3)
}
    | expr AND expr                                        { Binop($1, And, $3) }
    | expr OR expr                                         { Binop($1, Or, $3) }
    | expr LTE expr                                        { Binop($1, Lte, $3) }
    | expr LT expr                                         { Binop($1, Lt, $3) }
    | expr GTE expr                                        { Binop($1, Gte, $3) }
    | expr GT expr                                         { Binop($1, Gt, $3) }
    | expr EQUALS expr                                     { Binop($1, Equals, $3)
 }
    | expr NEQ expr                                        { Binop($1, Neq, $3) }
    | expr CONS expr                                       { Binop($1, Cons, $3) }
    | LPAREN expr RPAREN { $2 }
    | NOT expr                                             { Unop(Not, $2) }
    | MINUS expr %prec NEG                                 { Unop(Neg, $2) }
    | THROW expr                                           { Throw($2) }
    /* Yes, we aren't proud of this either */
    | TRY expr CATCH LPAREN ID RPAREN expr %prec SINGLE  { TryCatch(Block([$2]),
 $5, Block([$7])) }
    | TRY expr CATCH LPAREN ID RPAREN block              { TryCatch(Block([$2]),
 $5, Block($7)) }
    | TRY block CATCH LPAREN ID RPAREN expr %prec DOUBLE { TryCatch(Block($2), $
5, Block([$7])) }
    | TRY block CATCH LPAREN ID RPAREN block             { TryCatch(Block($2), $
5, Block($7)) }
    | IF expr THEN expr ELSE expr %prec SINGLE           { If($2, Block([$4]), B
lock([$6])) }
```

```
    | IF expr THEN block ELSE expr %prec DOUBLE          { If($2, Block($4), Blo
ck([$6])) }
    | IF expr THEN expr ELSE block                       { If($2, Block([$4]), B
lock($6)) }
    | IF expr THEN block ELSE block                      { If($2, Block($4), Blo
ck($6)) }
    | ID LPAREN actuals_opt RPAREN                       { Call(Val($1), $3) }
    | MODULE_LIT DOT expr                                { ModuleLit($1, $3)}

actuals_opt:
    | opts = separated_list(COMMA, expr)       { opts }

assigns:
    | VAL ID COLON primitive ASSIGN expr       { Assign($2, $4, $6) }
    | VAL ID ASSIGN expr                       { Assign($2, TAny, $4) }
```

src/ast.mli

```
type id = string

(* Operators in JSJS *)
type op =
  (* num operators *)
  | Add | Sub | Mul | Div | Mod | Neg
  (* string operators *)
  | Caret
  (* boolean operators *)
  | And | Or | Not
  (* relational operators *)
  | Lte | Gte | Neq | Equals | Lt | Gt
  (* list operators *)
  | Cons

(* Types in JSJS are either a primitive type
   or a function type. both of these types are
   defined in a mutually recursive fashion *)
type primitiveType =
  (* a generic type *)
  | T of string
  (* a general type. used to define
     empty lists or empty maps *)
  | TAny
  | TNum
  | TString
  | TBool
  | TUnit
  (* type of exception associated with a message *)
  | TExn
  | TFun of funcType
  (* list type - a list of primitive types *)
  | TList of primitiveType
  (* map type - a tuple of key type, value type *)
```

```
  | TMap of primitiveType * primitiveType

(* a function type takes a list of primitive types
   and returns a primitive type *)
and funcType = primitiveType list * primitiveType

(* everything is an expression in JSJS *)
type expr =
  | UnitLit
  | NumLit of float
  | BoolLit of bool
  | StrLit of string
  (* binary operation *)
  | Binop of expr * op * expr
  (* unary operation *)
  | Unop of op * expr
  (* a list literal is a list of expressions *)
  | ListLit of expr list
  (* a map literal is a list of key-value pairs *)
  | MapLit of (expr * expr) list
  (* a block is a list of expression *)
  | Block of expr list
  (* an assignment expression takes a string,
     an annotated type and an expression *)
  | Assign of id * primitiveType * expr
  (* a value is just a id *)
  | Val of id
  (* if-then-else takes 3 expressions - predicate,
     then expr and else expr *)
  | If of expr * expr * expr
  (* a function call takes fn name and a list of arguments (exprs) *)
  | Call of expr * expr list
  (* a fn literal is of func_decl record *)
  | FunLit of id list * expr * primitiveType
  (* a module literal is a module name and expression *)
  | ModuleLit of id * expr
  (*  an exception type that is generated by throw expr *)
  | Throw of expr
  (* a try catch block has two exprs and an identifier that acts as
     a placeholder for the message *)
  | TryCatch of expr * id * expr
;;

(* annotated expression *)
type aexpr =
  | AUnitLit of primitiveType
  | ANumLit of float * primitiveType
  | ABoolLit of bool * primitiveType
  | AStrLit of string * primitiveType
  | ABinop of aexpr * op * aexpr * primitiveType
  | AUnop of op * aexpr * primitiveType
  | AListLit of aexpr list * primitiveType
  | AMapLit of (aexpr * aexpr) list * primitiveType
```

```
  | ABlock of aexpr list * primitiveType
  | AAssign of id * primitiveType * aexpr * primitiveType
  | AVal of id * primitiveType
  | AIf of aexpr * aexpr * aexpr * primitiveType
  | ACall of aexpr * aexpr list * primitiveType
  | AFunLit of id list * aexpr * primitiveType * primitiveType
  | AModuleLit of id * aexpr * primitiveType
  | AThrow of aexpr * primitiveType
  | ATryCatch of aexpr * id * aexpr * primitiveType
;;


(* a JSJS program is a list of expressions *)
type program = expr list;;
```

src/typecheck.ml

```
open Ast
open Lexing
open Exceptions
open Parsing

module NameMap = Map.Make(String)
module GenericMap = Map.Make(Char)
module ModuleMap = Map.Make(String)
module KeywordsSet = Set.Make(String)

(* types *)
type typesTable = Ast.primitiveType NameMap.t
type environment = typesTable * typesTable
type substitutions = (id * primitiveType) list
type constraints = (primitiveType * primitiveType) list


(* mutable state *)
let type_variable = (ref ['A'; 'A'; 'A']);;

let modules = Lib.modules;;

(* maintains a set of js keywords *)
let keywords = ["break"; "case"; "class"; "catch"; "const"; "continue";
                "debugger"; "default"; "delete"; "do"; "else";
                "export"; "extends"; "finally"; "for"; "function"; "if";
                "import"; "in"; "instanceof"; "new"; "return"; "super";
                "switch"; "this"; "throw"; "try"; "typeof"; "var"; "void";
                "while"; "with"; "yield"; "val"; ];;

let jsjs_toplevel = ["print"; "print_num"; "print_string"; "print_bool";
"num_to_string"; "hd"; "empty?"; "tl"; "get"; "set"; "has?"; "del"; "keys"]

let keywords_set = List.fold_left (fun acc x -> KeywordsSet.add x acc)
    KeywordsSet.empty (jsjs_toplevel @  keywords);;
```

```
let get_new_type () =
  let rec aux (cs: char list): char list =
    let ys = match List.rev cs with
      | [] -> ['A']
      | 'Z' :: xs -> 'A' :: List.rev (aux (List.rev xs))
      | x :: xs -> (Char.chr ((Char.code x) + 1)) :: xs
    in List.rev ys
  in
  let curr_type_var = !type_variable in
  type_variable := aux (curr_type_var);
  T(String.concat "" (List.map Char.escaped curr_type_var))
;;

let merge_env (env: environment) : environment =
  let locals, globals = env in
  let merged_globals = NameMap.merge (fun k k1 k2 -> match k1, k2 with
      | Some k1, Some k2 -> Some k1
      | None, k2 -> k2
      | k1, None -> k1)
      locals globals in
  NameMap.empty, merged_globals
;;

let rec annotate_expr (e: expr) (env: environment) : (aexpr * environment) =
  match e with
  | UnitLit -> AUnitLit(TUnit), env
  | NumLit(n) -> ANumLit(n, TNum), env
  | BoolLit(b) -> ABoolLit(b, TBool), env
  | StrLit(s) -> AStrLit(s, TString), env

  | Binop(e1, op, e2) ->
    let ae1, _ = annotate_expr e1 env
    and ae2, _= annotate_expr e2 env
    and new_type = get_new_type () in
    ABinop(ae1, op, ae2, new_type), env

  | Unop(op, e) -> let ae, _ = annotate_expr e env in
    let new_type = get_new_type () in
    AUnop(op, ae, new_type), env

  | Val(id) ->
    let locals, globals = env in
    let typ = if NameMap.mem id locals
      then NameMap.find id locals
      else if NameMap.mem id globals
      then NameMap.find id globals
      else raise (Undefined(id)) in
    AVal(id, typ), env

  | FunLit(ids, e, t) -> begin

      (* check if JS keywords are passed as arguments *)
      List.iter (fun i -> if KeywordsSet.mem i keywords_set
```

```
                then raise (CannotRedefineKeyword(i)) else ()) ids;

      match t with
      | TFun(arg_types, ret_type) ->

        let generified_args = List.map (fun t ->
            match t with
            | TAny -> get_new_type ()
            | _ -> t
          ) arg_types in
        let generified_ret_type = match ret_type with
          | TAny -> get_new_type ()
          | _ -> ret_type in
        let user_fun_lit_type = TFun(generified_args, generified_ret_type) in

        (* annotates arguments with user-defined or new placeholders *)
        let annotated_args = List.map (fun (it, at) ->
            match at with
            | TAny | T(_) -> (it, get_new_type ())
            | _ -> (it, at))
            (List.combine ids arg_types) in

        (* merge locals and globals *)
        let locals, globals = merge_env env in

        (* augment local scope with fn args *)
        let new_locals = ListLabels.fold_left ~init: locals
            annotated_args ~f: (fun map (it, at) ->
                if NameMap.mem it map
                then raise (AlreadyDefined(it))
                else NameMap.add it at map) in

        (* prepare AFunLit *)
        let new_env = (new_locals, globals) in
        let ae, _ = (match e with
            (* if function expr is a block, env is already merged for its
             * annotation, need not be merged again in Block *)
            | Block(es) -> begin
                let aes, _ = ListLabels.fold_left ~init: ([], new_env) es
                    ~f: (fun (aes, env) e -> let ae, env = annotate_expr e env i
n (ae :: aes, env))
                in ABlock(List.rev aes, get_new_type()), new_env
            end
            | _ -> annotate_expr e new_env) in

        let ret_type = if ret_type = TAny then get_new_type () else ret_type in
        let arg_types = List.map snd annotated_args in
        let fun_type = TFun(arg_types, ret_type) in
        AFunLit(ids, ae, user_fun_lit_type, fun_type), env
      | _ -> raise (failwith "unreachable state")
    end

  | Call(fn, args) -> begin
```

```
      let afn, _  = annotate_expr fn env in
      let aargs = List.map (fun arg -> fst (annotate_expr arg env)) args in
      ACall(afn, aargs, get_new_type ()), env
    end

  | Assign(id, t, e) -> begin
      let locals, globals = env in

      (* do not allow reassignment *)
      if NameMap.mem id locals
      then raise (AlreadyDefined(id))
      else if KeywordsSet.mem id keywords_set
      then raise (CannotRedefineKeyword(id))
      (* annotate t with user-provided type or new placeholder *)
      else let t = if t = TAny then get_new_type () else t in
        let new_locals = if id = "_" then locals else NameMap.add id t locals in
        let ae, _ = match e with
          (* to allow recursion, we need to add pass new environment *)
          | FunLit(_) -> annotate_expr e (new_locals, globals)
          | _ -> annotate_expr e env
        in AAssign(id, t, ae, TUnit), (new_locals, globals)
    end

  | ListLit(es) ->
    let aes = List.map (fun e -> fst (annotate_expr e env)) es in
    AListLit(aes, TList(get_new_type ())), env

  | MapLit(kvpairs) ->
    let akvpairs = List.map (fun (k, v) ->
        let ak = fst (annotate_expr k env)
        and av = fst (annotate_expr v env) in ak, av)
        kvpairs
    in AMapLit(akvpairs, TMap(get_new_type (), get_new_type ())), env

  | If(p, e1, e2) ->
    let ap = fst (annotate_expr p env)
    and ae1 = fst (annotate_expr e1 env)
    and ae2 = fst (annotate_expr e2 env)
    in AIf(ap, ae1, ae2, get_new_type ()), env

  | Block(es) -> begin
      let new_env = merge_env env in
      let aes, new_env = ListLabels.fold_left ~init: ([], new_env) es
          ~f: (fun (aes, env) e -> let ae, env = annotate_expr e env in (ae :: a
es, env))
      in ABlock(List.rev aes, get_new_type ()), env
    end

  | Throw(e) -> AThrow(fst (annotate_expr e env), TExn), env

  | TryCatch(t, s, c) ->
    let at, _ = annotate_expr t env in
    let locals, globals = merge_env env in
```

```
    let new_locals = NameMap.add s TString locals in
    let ct, _ = annotate_expr c (new_locals, globals) in
    ATryCatch(at, s, ct, get_new_type ()), env

  | ModuleLit(id, e) ->
    if ModuleMap.mem id modules
    then
      let definitions = ModuleMap.find id modules in
      let ae = (match e with
          | Val(prop) ->
            let prop_type = if NameMap.mem prop definitions
              then NameMap.find prop definitions
              else raise (UndefinedProperty(id, e)) in
            AVal(prop, prop_type)
          | Call(fn, args) -> let afn = (match fn with
              | Val(prop) ->
                let prop_type = if NameMap.mem prop definitions
                  then NameMap.find prop definitions
                  else raise (UndefinedProperty(id, e)) in
                AVal(prop, prop_type)
              | _ -> raise (failwith "unreachable state reached")) in
            let aargs = List.map (fun arg -> fst (annotate_expr arg env)) args i
n
            ACall(afn, aargs, get_new_type ())
          | _ -> raise (UndefinedProperty(id, e))) in
      AModuleLit(id, ae, get_new_type ()), env
    else raise (ModuleNotFound(id))
;;

let rec type_of (aexpr: aexpr): primitiveType =
  match aexpr with
  | AUnitLit(t)           -> t
  | ANumLit(_, t)         -> t
  | ABoolLit(_, t)        -> t
  | AStrLit(_, t)         -> t
  | ABinop(_, _, _, t)    -> t
  | AUnop(_, _, t)        -> t
  | AListLit(_, t)        -> t
  | AMapLit(_, t)         -> t
  | ABlock(_, t)          -> t
  | AAssign(_, _, _, t)   -> t
  | AVal(_, t)            -> t
  | AIf(_, _, _, t)       -> t
  | ACall(_, _, t)        -> t
  | AFunLit(_, _, _, t)   -> t
  | AModuleLit(_, _, t)   -> t
  | AThrow(_, t)          -> t
  | ATryCatch(_, _, _, t) -> t
;;


let rec collect_expr (ae: aexpr): constraints =
  match ae with
```

```
  | AUnitLit(_) | ANumLit(_) | ABoolLit(_) | AStrLit(_) -> []
  | AVal(_) -> []

  | ABinop(ae1, op, ae2, t) ->
    let et1 = type_of ae1 and et2 = type_of ae2 in

    let opc = match op with
      | Add | Sub | Mul | Div | Mod -> [(et1, TNum); (et2, TNum); (t, TNum)]
      | Caret -> [(et1, TString); (et2, TString); (t, TString)]
      | And | Or  -> [(et1, TBool); (et2, TBool); (t, TBool)]
      | Lte | Gte | Neq | Equals | Lt | Gt -> [(et1, et2); (t, TBool)]
      | Cons -> (match et2 with
          (* write something here *)
          | TList(x) -> [(et1, x); (t, et2)]
          | T(_) -> [(et2, TList(et1)); (t, TList(et1))]
          | _ -> raise (NonUniformTypeContainer(et1, et2)))
      | _ -> raise (InvalidOperation(et1, op)) in
    (collect_expr ae1) @ (collect_expr ae2) @ opc

  | AUnop(op, ae, t) ->
    let et = type_of ae in
    let opc = (match op with
        | Not -> [(et, TBool); (t, TBool)]
        | Neg -> [(et, TNum); (t, TNum)]
        | _ -> raise (InvalidOperation(et, op))) in
    (collect_expr ae) @ opc

  | AIf(ap, ae1, ae2, t) ->
    let pt = type_of ap and et1 = type_of ae1 and  et2 = type_of ae2 in
    let cons = [(pt, TBool); (et1, et2); (t, et1)] in
    (collect_expr ap) @ (collect_expr ae1) @ (collect_expr ae2) @ cons

  | AListLit(aes, t) ->
    let list_type = match t with
      | TList(x) -> x
      | _ -> raise (failwith "unreachable state reached in listlit")
    in
    let elem_conts = List.map (fun ae -> (list_type, type_of ae)) aes in
    (List.flatten (List.map collect_expr aes)) @ elem_conts

  (* remember to restrict key types to be TNum, TBool and TString,
     will require another type check *)
  | AMapLit(kvpairs, t) ->
    let kt, vt = match t with
      | TMap(kt, vt) -> kt, vt
      | _ -> raise (failwith "unreachable state reached in maplit")
    in
    let klist = List.map fst kvpairs in
    let vlist = List.map snd kvpairs in
    let k_conts = List.map (fun k -> (kt, type_of k)) klist in
    let v_conts = List.map (fun v -> (vt, type_of v)) vlist in
    [(t, TMap(kt, vt))] @ (List.flatten (List.map collect_expr klist)) @
    (List.flatten (List.map collect_expr vlist)) @ k_conts @ v_conts
```

```
  | ABlock(aes, t) ->
    let last_type = (match List.hd (List.rev aes) with
    | AAssign(id, _, _, _) -> raise (InvalidReturnExpression(id))
    | ae -> type_of ae ) in
    (List.flatten (List.map collect_expr aes)) @ [(t, last_type)]

  | AAssign(id, t, ae, _) -> (collect_expr ae) @ [(t, type_of ae)]

  | AFunLit(_, ae, _, t) -> (match t with
      | TFun(_, ret_type) -> (collect_expr ae) @ [(type_of ae, ret_type)]
      | _ -> raise (failwith "unreachable state reached in funlit"))

  | ACall(afn , aargs, t) ->
    let typ_afn = (match afn with
        | AVal(_) | AFunLit(_) -> type_of afn
        | _ -> raise (failwith "unreachable state reached in call")) in
    let sign_conts = (match typ_afn with
        | TFun(arg_types, ret_type) -> begin
            let l1 = List.length aargs and l2 = List.length arg_types in
            if l1 <> l2
            then raise (MismatchedArgCount(l1, l2))
            else let arg_conts = List.map2 (fun ft at -> (ft, type_of at)) arg_t
ypes aargs in
              arg_conts @ [(t, ret_type)]
          end
        | T(_) -> [(typ_afn, TFun(List.map type_of aargs, t))]
        | _ -> raise (MismatchedTypes(typ_afn, TFun(List.map type_of aargs, t)))
) in
    (collect_expr afn) @ (List.flatten (List.map collect_expr aargs)) @ sign_con
ts

  | AModuleLit(id, ae, t) ->
    (match ae with
     | AVal(prop, ts) ->
       (collect_expr ae) @ [(t, ts)]
     | ACall(afn, aargs, call_t) ->
       let prop, prop_type = (match afn with
           | AVal(s, t) -> s, t
           |  _ -> raise (failwith "unreachable state 1")) in
       (match prop_type with
        | TFun(arg_types, ret_type) ->
          let sign_conts =
            let l1 = List.length aargs and l2 = List.length arg_types in
            if l1 <> l2 then raise (MismatchedArgCount(l1, l2))
            else
              let arg_conts = List.map2
                  (fun ft at -> (ft, type_of at)) arg_types aargs in
              arg_conts
          in
          (List.flatten (List.map (collect_expr) aargs)) @  sign_conts @ [(t, ca
ll_t); (t, ret_type)]
        | _ -> raise(failwith "unreachable state 2"))
     | _ -> raise (failwith "unreachable state 3"))
```

```
  | AThrow(ae, t) -> collect_expr ae @ [(t, TExn)]

  | ATryCatch(atry, id, acatch, t) ->
    let ttry = type_of atry and tcatch = type_of acatch in
    if tcatch = TExn then raise (failwith "Can't throw in a catch block") else
    (collect_expr atry) @ (collect_expr acatch) @ [(ttry, tcatch); (t, tcatch)]

;;

let rec substitute (u: primitiveType) (x: id) (t: primitiveType) : primitiveType
 =
  match t with
  | TNum | TBool | TString | TUnit | TAny | TExn -> t
  | T(c) -> if c = x then u else t
  | TFun(t1, t2) -> TFun(List.map (substitute u x) t1, substitute u x t2)
  | TList(t) -> TList(substitute u x t)
  | TMap(kt, vt) -> TMap(substitute u x kt, substitute u x vt)
;;

let apply (subs: substitutions) (t: primitiveType) : primitiveType =
    List.fold_right (fun (x, u) t -> substitute u x t) subs t
;;

(* this function check if a placeholder (x) occurs within another
   type t. this is used to produce errors such as (fun z -> z z)*)
let rec resolve_type (x: id) (t: primitiveType) : bool =
  match t with
  | T(c) when x = c -> false
  | TList(tlist) -> resolve_type x tlist
  | TMap(kt, vt) -> (resolve_type x kt) && (resolve_type x vt)
  | TFun(argst, ret_t) ->
    List.map (resolve_type x) (ret_t :: argst)
    |> List.fold_left ( && ) true
  | _ -> true
;;

let rec unify (constraints: constraints) : substitutions =
  match constraints with
  | [] -> []
  | (x, y) :: xs ->
    (* generate substitutions of the rest of the list *)
    let t2 = unify xs in
    (* resolve the LHS and RHS of the constraints from the previous substitution
s *)
    let t1 = unify_one (apply t2 x) (apply t2 y) in
    t1 @ t2

and unify_one (t1: primitiveType) (t2: primitiveType) : substitutions =
  match t1, t2 with
  | TNum, TNum | TBool, TBool | TString, TString | TUnit, TUnit -> []
  | TExn, _ | _, TExn -> []
  | T(x), z | z, T(x) ->
      if (t1 = t2 || resolve_type x z)
```

```
      then [(x, z)]
      else raise (MismatchedTypes(t1, t2))
  | TList(t1), TList(t2) -> unify_one t1 t2
  | TMap(kt1, vt1), TMap(kt2, vt2) ->
          let _ = (match kt1 with
            | TNum | TBool | TString | T(_) -> ()
            | _ -> raise (InvalidKeyType(kt1))) in
          unify [(kt1, kt2) ; (vt1, vt2)]
  (* This case is particularly useful when you are calling a function that retur
ns a function *)
  | TFun(a, b), TFun(x, y) ->
      let l1 = List.length a and l2 = List.length x in
      if l1 = l2
      then unify ((List.combine a x) @ [(b, y)])
      else raise (MismatchedArgCount(l1, l2))
  | _ -> raise (MismatchedTypes(t1, t2))
;;

let rec apply_expr (subs: substitutions) (ae: aexpr): aexpr =
  match ae with
  | ABoolLit(b, t)                -> ABoolLit(b, apply subs t)
  | ANumLit(n, t)                 -> ANumLit(n, apply subs t)
  | AStrLit(s, t)                 -> AStrLit(s, apply subs t)
  | AUnitLit(t)                   -> AUnitLit(apply subs t)
  | AVal(s, t)                    -> AVal(s, apply subs t)
  | AAssign(id, t, ae, _)         -> AAssign(id, apply subs t, apply_expr subs a
e, TUnit)
  | ABinop(ae1, op, ae2, t)       -> ABinop(apply_expr subs ae1, op, apply_expr
subs ae2, apply subs t)
  | AUnop(op, ae, t)              -> AUnop(op, apply_expr subs ae, apply subs t)
  | AListLit(aes, t)              -> AListLit(List.map (apply_expr subs) aes, ap
ply subs t)
  | AMapLit(kvpairs, t)           -> AMapLit(List.map (fun (k, v) -> (apply_expr
 subs k, apply_expr subs v)) kvpairs, apply subs t)
  | AIf(ap, ae1, ae2, t)          -> AIf(apply_expr subs ap, apply_expr subs ae1
, apply_expr subs ae2, apply subs t)
  | ABlock(aes, t)                -> ABlock(List.map (apply_expr subs) aes, appl
y subs t)
  | AFunLit(ids, ae, t1, t2)      -> AFunLit(ids, apply_expr subs ae, t1, apply
subs t2)
  | ACall(afn, aargs, t)          -> ACall(apply_expr subs afn, List.map (apply_
expr subs) aargs, apply subs t)
  | AModuleLit(id, ae, t)         -> AModuleLit(id, apply_expr subs ae, apply su
bs t)
  | AThrow(ae, t)                 -> AThrow(apply_expr subs ae, apply subs t)
  | ATryCatch(atry, s, acatch, t) -> ATryCatch(apply_expr subs atry, s, apply_ex
pr subs acatch, apply subs t)
;;

let infer (expr: expr) (env: environment) : aexpr * environment =
  let aexpr, env = annotate_expr expr env in
  let constraints = collect_expr aexpr in
  let subs = unify constraints in
```

```
  let inferred_expr = apply_expr subs aexpr in
  inferred_expr, env
;;

let type_check (program: program) : (aexpr list) =

  (* setting the predefined environment *)
  let predefined = Lib.predefined in
  let env = (NameMap.empty, predefined) in

  (* build the inferred program by inferred each expression one by one
     and updating the environment as we go along *)
  let inferred_program, _ = ListLabels.fold_left program ~init: ([], env)

    ~f: (fun (acc, env) expr ->

          (* get the inferred expression and the updated environment *)
          let inferred_expr, env =
            try infer expr env
            with e -> handle_error e
          in

          (* if expression is assignment, update the environment *)
          let inferred_expr, env = match inferred_expr with
            | AAssign(id, t, ae, _) ->
              let ae, subs = (match ae with
                  (* in this step, we check if the inferred types for a function
                     literal matches the user annotation. *)
                  | AFunLit(id, body, user_type, inferred_type) ->
                    (* get substitutions for user annotated and inferred type *)
                    let subs = unify_one user_type inferred_type in

                    (* apply the substitutions on the user provided type *)
                    let user_unified_t = apply subs user_type in

                    (* apply the substitutions in function body*)
                    let unified_ae = apply_expr subs body in

                    (* return the annotated expression *)
                    AFunLit(id, unified_ae, user_type, user_unified_t), subs
                  | _ -> ae, []) in
               (* since this is an assignment statement, we
                  update the environment with the type of ae *)
              let locals, globals = env and aet = type_of ae in

              (* if id is "_", then don't store the result *)
              let locals = if id = "_" then locals else NameMap.add id aet local
s in

              (* build the original assign expression with new aexpr and applied
 substitutions *)
              let ret_ae = AAssign(id, apply subs t, ae, TUnit) in
              ret_ae, (locals, globals)
```

```
            | _ -> inferred_expr, env in

          (* save the inferred_expr and return it alongwith the env *)
          (inferred_expr :: acc, env))
  in

  (* returning the inferred program *)
  List.rev inferred_program
;;
```

src/semantic.ml

```
open Ast
open Lexing
open Parsing
open Exceptions
open Stringify

module NameMap = Map.Make(String);;
module GenericMap = Map.Make(Char);;
module ModuleMap = Map.Make(String);;
module KeywordsSet = Set.Make(String);;

type typesTable = Ast.primitiveType NameMap.t;;
type typeEnv = typesTable * typesTable;;

(* maintains a set of js keywords *)
let keywords = ["break"; "case"; "class"; "catch"; "const"; "continue";
                "debugger"; "default"; "delete"; "do"; "else";
                "export"; "extends"; "finally"; "for"; "function"; "if";
                "import"; "in"; "instanceof"; "new"; "return"; "super";
                "switch"; "this"; "throw"; "try"; "typeof"; "var"; "void";
                "while"; "with"; "yield" ];;
let js_keywords_set = List.fold_left (fun acc x -> KeywordsSet.add x acc)
    KeywordsSet.empty keywords;;

let build_map (formals: (string * primitiveType) list) =
  List.fold_left
    (fun acc_map (id, t) -> NameMap.add id t acc_map)
    (NameMap.empty) formals
;;

(* Resolves generic types at function call
   1. Checks whether all generic types have been defined
   2. Associates concrete types with generic types
   3. Raises exception for inconsistent generic type resolution
   4. Raises exception for generic function arguments
   5. Cannot declare nested generic functions

Fancy way of saying: Rank-1 Polymorphism \o/ *)
let rec resolve map ft at =
  match ft with
```

```
  | T(c) -> if GenericMap.mem c map
    then (match GenericMap.find c map with
        | TAny -> GenericMap.add c at map
        | t -> if t = at then map else raise (MismatchedTypes(t, at)))
    else raise (UndefinedType(c))
  | TFun(formals_types, ret_type) -> (match at with
      | TFun(actual_types, actual_return_type) ->
        let l1 = List.length formals_types and l2 = List.length actual_types in
        if l1 <> l2 then raise (MismatchedArgCount(l1, l2)) else
          List.fold_left2 resolve map (ret_type :: formals_types) (actual_return
_type :: actual_types)
      | TFunGeneric(x, y) -> raise (InvalidArgumentType(TFunGeneric(x, y)))
      | _ -> raise (MismatchedTypes(ft, at)))
  | TFunGeneric(x, y) -> raise (InvalidArgumentType(TFunGeneric(x, y)))
  | TList(t) -> (match at with
      | TList(a) -> resolve map t a
      | _ -> raise (MismatchedTypes(ft, at)))
  | TMap(kt, vt) -> (match at with
      | TMap(akt, avt) -> let map = resolve map kt akt in
        resolve map vt avt
      | _ -> raise (MismatchedTypes(ft, at)))
  | TNum | TString | TBool | TUnit -> if ft = at then map else raise (Mismatched
Types(ft, at))
  | TAny | TExn -> raise (InvalidArgumentType(ft))
;;

let rec is_generic_type = function
  | TString | TExn | TNum | TBool | TUnit | TAny | TFun(_) -> false
  | T(_) -> true
  | TList(t) -> is_generic_type t
  | TMap(kt, vt) -> is_generic_type kt || is_generic_type vt
  | TFunGeneric(_) -> true
;;

(* TODO: add documentation for this *)
let rec generate_ret_types map = function
  | T(c) -> if GenericMap.mem c map
    then (GenericMap.find c map)
    else raise (UndefinedType(c))
  | TFun(formals_types, ret_type) ->
    let ftypes = List.map (generate_ret_types map) (ret_type :: formals_types) i
n
    TFun(List.tl ftypes, List.hd ftypes)
  | TList(t) -> TList(generate_ret_types map t)
  | TMap(kt, vt) ->
    let keytypes = generate_ret_types map kt
    and valuetypes = generate_ret_types map vt in
    TMap(keytypes, valuetypes)
  | TFunGeneric(x, y) -> raise (InvalidArgumentType(TFunGeneric(x,y)))
  | t -> t
;;

(* checks if two types are equal and returns an option type *)
```

```
let rec validate_types type1 type2 : (Ast.primitiveType option) =
  match (type1, type2) with
  | TList(t1), TList(t2) ->
    (match (validate_types t1 t2) with
    | Some(t) -> Some(TList(t))
    | None -> None)
  | TMap(k1, v1), TMap(k2, v2) ->
    (match validate_types k1 k2 with
     | Some(k) -> (match validate_types v1 v2 with
        | Some(v) -> Some(TMap(k, v))
        | None -> None)
     | None -> None)
  | t1, t2 when t1 = t2 -> Some(t2)
  | TExn, t | t, TExn -> Some(t)
  | TAny, t | t, TAny -> Some(t)
  | t1, t2 -> None
;;

let modules = Lib.modules;;

let rec type_of_expr (env: typeEnv) = function
  | UnitLit -> TUnit, env
  | NumLit(_) -> TNum, env
  | BoolLit(_) -> TBool, env
  | StrLit(_) -> TString, env

  | Binop(e1, op, e2) ->
    let t1, _ = type_of_expr env e1 and t2, _ = type_of_expr env e2 in
    (* for the cons operator, check if the element being added
       has the same type as the rest of the list. In case the list
       is of type TAny, just emit the type of element *)
    if op = Cons
    then (match t1, t2 with
        | x, TList(y) ->
          (match validate_types x y with
           | Some(t) -> TList(t), env
           | None -> raise (MismatchedOperandTypes(op, t1, t2)))
        | _, _ -> raise (MismatchedOperandTypes(op, t1, t2)))
    else if t1 <> t2 then raise (MismatchedOperandTypes (op, t1, t2))
    else begin
      match op with
      | Caret -> if t1 = TString then TString, env
        else raise (InvalidOperation(t1, Caret))
      | And | Or -> if t1 = TBool then TBool, env
        else raise (InvalidOperation(t1, op))
      | Add | Sub | Mul
      | Div | Mod -> if t1 = TNum then TNum, env
        else raise (InvalidOperation(t1, op))
      | Lte | Gte | Neq
      | Equals | Lt | Gt -> TBool, env
      | _ -> raise (InvalidOperation(t1, op))
    end
```

```
  | Unop(op, e) -> begin
      let t, _ = type_of_expr env e in
      match (op, t) with
      | Not, TBool -> TBool, env
      | Neg, TNum -> TNum, env
      | _, _ -> raise (InvalidOperation(t, op))
    end

  | ListLit(es) -> begin
      let ts = List.map
          (fun x -> let t, _ = type_of_expr env x in t)
          es
      in
      if List.length ts = 0 then TList(TAny), env
      else let list_type = List.fold_left
          (fun acc t -> if acc = t then acc
            else raise (NonUniformTypeContainer(acc, t)))
          (List.hd ts) (List.tl ts)
        in
        TList(list_type), env
    end

  | Block(es) -> begin
      let locals, globals = env in
      let merged_globals = NameMap.merge (fun k k1 k2 -> match k1, k2 with
          | Some k1, Some k2 -> Some k1
          | None, k2 -> k2
          | k1, None -> k1)
          locals globals in
      let env = NameMap.empty, merged_globals in
      match es with
      | [] -> TAny, env (* unreachable state since {} is a maplit and not a bloc
k *)
      | Assign(id, _, _) :: [] -> raise (InvalidReturnExpression(id))
      | x :: [] -> type_of_expr env x
      | x :: xs -> (* two or more expressions *)
        (match List.rev(xs) with
         | [] -> raise (failwith("unreachable state reached"))
         | Assign(id, _, _) :: _ -> raise (InvalidReturnExpression(id))
         | _ -> List.fold_left
                  (fun (t, acc_env) e ->
                     let newt, newenv = type_of_expr acc_env e in
                     (newt, newenv))
                  (type_of_expr env x) xs)
    end

  | If(p, e1, e2) -> begin
      let pt, _ = type_of_expr env p in
      if pt <> TBool then raise (MismatchedTypes(TBool, pt))
      else
        let t1, _ = type_of_expr env e1 and t2, _ = type_of_expr env e2 in
        (match validate_types t1 t2 with
         | Some(t) -> t, env
```

```
        | None -> raise (MismatchedTypes(t1, t2)))
    end

  | MapLit(kvpairs) -> begin
      match kvpairs with
      | [] -> TMap(TAny, TAny), env
      | (key, value) :: xs ->
        (* we find the type of key by folding over
           all k-v pairs by taking the type of first
           k-v pair as reference *)

        (* figuring out the type of key *)
        let start_key_type, _ = type_of_expr env key in
        let key_type = List.fold_left (fun acc (k, _) ->
            let t, _ = type_of_expr env k in
            match validate_types t acc with
            | Some(resolved_type) -> resolved_type
            | None -> raise (MismatchedTypes(acc, t)))
            start_key_type xs
        in
        let _ = match key_type with
          | TNum | TBool | TString -> ()
          | _ -> raise (InvalidKeyType(key_type))
        in
        (* figuring out the type of value *)
        let start_value_type, _ = type_of_expr env value in
        let value_type = List.fold_left (fun acc (_, v) ->
            let t, _ = type_of_expr env v in
            match validate_types t acc with
            | Some(resolved_type) -> resolved_type
            | None -> raise (MismatchedTypes(acc, t)))
            start_value_type xs
        in
        TMap(key_type, value_type), env
    end

  | Assign(id, annotated_type, e) -> begin
      (* 1. Get the type of expression
         2. Check if it matches with t
         3. Update locals if all is well.  *)
      let locals, globals = env in
      if NameMap.mem id locals then raise (AlreadyDefined(id))
      else if KeywordsSet.mem id js_keywords_set then raise (CannotRedefineKeywo
rd(id))
      else
        match e with
        | Assign(s, _, _) -> raise (InvalidReturnExpression(s))

        (* In case of function literals, to get the type of expression
           we need to populate the local scope with the types of the
           formals arguments so that the body can be correctly typechecked *)
        | FunLit(fdecl) ->
          let formaltypes = List.map (fun (_, x) -> x) fdecl.formals in
```

```
          let functype = if fdecl.is_generic
            then TFunGeneric((formaltypes, fdecl.return_type), fdecl.generic_typ
es)
            else TFun(formaltypes, fdecl.return_type)
          in
          (* check if the annotated type is same as what computed *)
          let val_type = (match validate_types annotated_type functype with
            | Some(t) -> t
            | None -> raise (MismatchedTypes(annotated_type, functype))) in
          (* update the locals environment with the new value *)
          let locals = (NameMap.add id val_type locals) in
          let etype, _ = type_of_expr (locals, globals) e  in
          if etype = functype
          then TUnit, (locals, globals)
          else raise (MismatchedTypes(functype, etype))

        (* For every other kind of expression, simply determine its type
           and check if it same as the annotated type (if any). *)
        | _ ->
          let etype, _ = type_of_expr env e in
          (* check if the annotated type is same as what computed *)
          let val_type = (match validate_types annotated_type etype with
            | Some(t) -> t
            | None -> raise (MismatchedTypes(annotated_type, etype))) in
          (* update the locals environment with the new value *)
          let locals = (NameMap.add id val_type locals) in
          TUnit, (locals, globals)
    end

  | Throw(e) -> begin
      let t, _ = type_of_expr env e in
      match t with
      | TString -> TExn, env
      | _ -> raise (failwith "exception")
    end

  | TryCatch(e1, id, e2) -> begin
      let t1, _ = type_of_expr env e1 in
      let locals, globals = env in
      let merged_globals = NameMap.merge (fun k k1 k2 -> match k1, k2 with
          | Some k1, Some k2 -> Some k1
          | None, k2 -> k2
          | k1, None -> k1)
          locals globals in
      let locals = NameMap.add id TString NameMap.empty in
      let t2, _ = type_of_expr (locals, merged_globals) e2 in
      (match validate_types t1 t2 with
       | Some(t) -> t, env
       | None -> raise (MismatchedTypes(t1, t2)))
    end

  | Val(s) -> begin
      let locals, globals = env in
```

```
    if NameMap.mem s locals
    then NameMap.find s locals, env
    else if NameMap.mem s globals
    then NameMap.find s globals, env
    else raise (Undefined(s))
  end

| FunLit(fdecl) -> begin
    if fdecl.is_generic
    then begin
      (* in case of a generic function, check if
         parameteric types exist in generic types field of function *)
      let formaltype = List.map (fun (_, x) -> x) fdecl.formals in
      TFunGeneric((formaltype, fdecl.return_type), fdecl.generic_types), env
    end
    (* when the function is not generic *)
    else begin
      let locals, globals = env in
      (* build a map for formals. string -> type of formals *)
      let type_formals = List.fold_left
          (fun acc (id, t) ->
             (* check if none of the formals are generic in which
                case raise an exception *)
            match is_generic_type t with
            | true -> raise InvalidGenericFunctionDefinition
            | false -> NameMap.add id t acc)
          (NameMap.empty) (fdecl.formals) in
      (* merge the global and local maps into one such that
         local definitions take priority *)
      let merged_globals = NameMap.merge (fun k k1 k2 -> match k1, k2 with
          | Some k1, Some k2 -> Some k1
          | None, k2 -> k2
          | k1, None -> k1)
          locals globals in
      (* augment the environment with merged locals and globals *)
      let env = type_formals, merged_globals in
      (* get the type of the body *)
      let block_type, _ = match fdecl.body with
        (* check if the body is a list of expressions or just one expr *)
        | Block (es) -> begin
            match es with
            | [] -> TAny, env
            | x :: [] -> type_of_expr env x
            | x :: xs ->
              List.fold_left
                (fun (t, acc_env) e ->
                   let newt, newenv = type_of_expr acc_env e in
                   (newt, newenv))
                (type_of_expr env x) xs
          end
        | e -> type_of_expr env e
      in
      (* check if the type of last block expr is
```

```
           same as function's return type *)
        let return_type =
          (match validate_types block_type fdecl.return_type with
           | Some(t) -> t
           | None -> raise (MismatchedTypes(fdecl.return_type, block_type))) in
        let formaltype = List.map (fun (_, x) -> x) fdecl.formals in
        TFun(formaltype, return_type), env
      end
    end

  | Call(id, es) -> begin
      let t, _ = type_of_expr env (Val(id)) in
      (* a list of argument types *)
      let args_type = List.map (fun e -> let t, _ = type_of_expr env e in t) es
in
      (match t with
      | TFun(formals_type, return_type) ->
        (* check if the lengths of the formal and actual args match *)
        let l1 = List.length args_type and l2 = List.length formals_type in
        if l1 <> l2 then raise (MismatchedArgCount(l2, l1))
        (* type of each pair of formal and actual args should match *)
        else List.iter2
            (fun ft at ->
               match validate_types ft at with
               | Some(t) -> ()
               | None -> raise (MismatchedTypes(ft, at)))
          formals_type args_type;
        return_type, env

      (* Type-checking generic function calls.
         1. First, resolve types (thanks to the resolve function above)
         2. Next type check the body of the function
         3. Return the "resolved" return type of the call expression *)
      | TFunGeneric((formals_type, return_type), generic_types) -> begin
            (* create an map of all generic types to TAny. Goal is to resolve
               each of these types uniquely and correctly *)
            let genMap = List.fold_left (fun map t -> GenericMap.add t TAny map)
                GenericMap.empty generic_types in
            (* check if the lengths of the formal and actual args match *)
            let l1 = List.length args_type and l2 = List.length formals_type in
            if l1 <> l2 then raise (MismatchedArgCount(l2, l1))
            else
              let genMap = List.fold_left2 resolve genMap formals_type args_type
 in
              (generate_ret_types genMap return_type), env
        end

      | _ -> raise (failwith "unreacheable state reached"))
    end

  (* 1. check if id exists in modules
     2. then check if e exists in that map
     3. return type of e *)
```

```
  | ModuleLit(id, e) -> begin
      let defs = if ModuleMap.mem id modules
        then ModuleMap.find id modules
        else raise (ModuleNotFound(id)) in
      match e with
      | Call(prop, args) ->
        let prop_type = if NameMap.mem prop defs
          then NameMap.find prop defs
          else raise (UndefinedProperty(id, prop)) in
        (match prop_type with
         | TFun(formals_type, return_type) ->
          let args_type = List.map
              (fun e -> let t, _ = type_of_expr env e in t) args
          in
          let l1 = List.length args_type and l2 = List.length formals_type in
          if l1 <> l2 then raise (MismatchedArgCount(l2, l1))
          else List.iter2 (fun ft at -> let _ = validate_types ft at in ())
              formals_type args_type;
          return_type, env
         | TFunGeneric((formals_type, return_type), generic_types) -> begin
            let genMap = List.fold_left (fun map t -> GenericMap.add t TAny map
)
                GenericMap.empty generic_types in
            let args_type = List.map
                (fun e -> let t, _ = type_of_expr env e in t) args in
            let l1 = List.length args_type and l2 = List.length formals_type in
            if l1 <> l2 then raise (MismatchedArgCount(l2, l1))
            else
              let genMap = List.fold_left2 resolve genMap formals_type args_typ
e in
              (generate_ret_types genMap return_type), env
          end
         | _ -> raise (failwith "unreacheable state reached"))
      | Val(prop) -> begin
          if NameMap.mem prop defs
          then NameMap.find prop defs, env
          else raise (UndefinedProperty(id, prop))
        end
      | _ -> raise (UndefinedProperty(id, string_of_expr e))
    end
;;

let type_check (program: Ast.program) =
  let predefined = Lib.predefined in
  List.fold_left
    (fun env expr ->
       try
         let _, env = type_of_expr env expr in env
       with
       | InvalidGenericFunctionDefinition ->
         raise (TypeError (Printf.sprintf "Error: Generic type found. Use [Type]
 to declare generic functions."))
       | ModuleNotFound(s) ->
```

```
        raise (TypeError (Printf.sprintf "Type error: Module '%s' not defined"
s))
      | InvalidReturnExpression(s) ->
        raise (TypeError (Printf.sprintf "Type error: Assignment expressions ca
nnot be returned"))
      | InvalidOperation(t, op) ->
        let st = string_of_type t and sop = string_of_op op in
        raise (TypeError (Printf.sprintf "Type error: Invalid operation '%s' on
 type '%s'" sop st))
      | MismatchedTypes(t1, t2) ->
        let st1 = string_of_type t1 and st2 = string_of_type t2 in
        raise (TypeError (Printf.sprintf "Type error: expected value of type '%
s', got a value of type '%s' instead" st1 st2))
      | MismatchedOperandTypes(op, t1, t2) ->
        let st1 = string_of_type t1 and st2 = string_of_type t2 and sop = strin
g_of_op op in
        raise (TypeError (Printf.sprintf "Type error: Cannot have types '%s' an
d '%s' for operator '%s'" st1 st2 sop))
      | NonUniformTypeContainer(t1, t2) ->
        let st1 = string_of_type t1 and st2 = string_of_type t2 in
        raise (TypeError (Printf.sprintf "Type error: Lists can only contain on
e type. Expected '%s', got a '%s' instead" st1 st2))
      | Undefined(s) ->
        raise (TypeError (Printf.sprintf "Error: value '%s' was used before it
was defined" s))
      | AlreadyDefined(s) ->
        raise (TypeError (Printf.sprintf "Error: '%s' cannot be redefined in th
e current scope" s))
      | UndefinedProperty(module_name, prop) ->
        raise (TypeError (Printf.sprintf "Error: property '%s' is not defined i
n module '%s'" prop module_name))
      | CannotRedefineKeyword(keyword) ->
        raise (TypeError (Printf.sprintf "Error: Cannot define Javascript keywo
rd '%s'" keyword))
      | MismatchedArgCount(l1, l2) ->
        raise (TypeError (Printf.sprintf "Error: Expected number of argument(s)
: %d, got %d instead." l1 l2))
      | UndefinedType(c) ->
        raise (TypeError (Printf.sprintf "Error: Type '%c' not found." c))
      | InvalidArgumentType(_) ->
        raise (TypeError (Printf.sprintf "Error: Invalid argument type. Cannot
pass generic functions as arguments."))
      | InvalidKeyType(t) ->
        raise (TypeError (Printf.sprintf "Type Error: Cannot have '%s' as key t
ype in a Map." (string_of_type t)))
      | e -> raise (TypeError (Printexc.to_string e)))
    (predefined, NameMap.empty)
    program
;;
```

src/codegen.ml

```
open Ast
open Stringify

module NameMap = Map.Make(String);;

(* name environment for program *)
type aliasTable = string NameMap.t
type environment = aliasTable * aliasTable

let merge_env (env: environment) : environment =
  let locals, globals = env in
  let merged_globals = NameMap.merge (fun k k1 k2 -> match k1, k2 with
      | Some k1, Some k2 -> Some k1
      | None, k2 -> k2
      | k1, None -> k1)
      locals globals in
  NameMap.empty, merged_globals
;;

(* helper functions to generate templates *)
let block_template (return_expr: string) (body: string option) : string =
  match body with
  | None ->
    let template = format_of_string "(function() { return %s })()"
    in Printf.sprintf template return_expr
  | Some(exprs) ->
    let template = format_of_string "
    (function() {
        %s
        return %s;
    })()"
    in Printf.sprintf template exprs return_expr
;;

let if_template (pred_expr: string) (if_expr: string) (else_expr: string): strin
g =
  let id = "res_" ^ string_of_int(Random.int 1000000)
  and template = format_of_string "(function() {
        let %s
        if (%s) {
            %s = %s
        } else {
            %s = %s
        }
        return %s
    })()"
  in
  Printf.sprintf template id pred_expr id if_expr id else_expr id
;;

let try_catch_template (try_expr: string) (msg: string) (catch_expr: string) =
```

```
  let id = "res_" ^ string_of_int(Random.int 1000000)
  and template = format_of_string "(function() {
        let %s
        try {
            %s = %s
        } catch(%s) {
            %s = %s
        }
        return %s
    })()"
  in
  Printf.sprintf template id id try_expr msg id catch_expr id
;;

(* removes all ? from string and replaces with __. used for codegen
 * since JS doesnt support ? in var names *)
let remove_qmark (s: string) =
  let s = if s = "_" then s ^ string_of_int (Random.int 100000) else s in
  Str.global_replace (Str.regexp_string "?") "__" s
;;

(* generates a js version of an annotated expression
   takes the name of the module and a module map along with the expression *)
let rec js_of_aexpr (module_name: string) (map:'a NameMap.t) (env: environment)
(aexpr: aexpr) =
  match aexpr with
  | ANumLit(x, _) -> (Printf.sprintf "%s" (string_of_float x)), env
  | AStrLit(s, _) -> (Printf.sprintf "\"%s\"" s), env
  | ABoolLit(b, _) -> (Printf.sprintf "%s" (string_of_bool b)), env
  | AUnitLit(_) -> "(undefined)", env

  | AUnop(o, e, _) ->
    let s1 = string_of_op o and s2, _ = js_of_aexpr module_name map env e in
    (Printf.sprintf "%s%s" s1 s2), env

  | ABinop(e1, o, e2, _) ->
    let s2, _ = js_of_aexpr module_name map env e1
    and s3, _ = js_of_aexpr module_name map env e2 in
    (match o with
      | Cons -> (Printf.sprintf "(%s).insert(0, %s)" s3 s2)
      | Caret -> (Printf.sprintf "(%s + %s)" s2 s3)
      | Equals -> (match (Typecheck.type_of e1) with
          | TList(_) | TMap(_) -> (Printf.sprintf "(Immutable.is(%s, %s))" s2 s3
)
          | _ -> (Printf.sprintf "(%s %s %s)" s2 (string_of_op o) s3))
      | _ -> (Printf.sprintf "(%s %s %s)" s2 (string_of_op o) s3)), env

  | AThrow(e, t) -> (Printf.sprintf "(function() { throw %s })()" (fst (js_of_ae
xpr module_name map env e))), env

  | ATryCatch(e1, s, e2, t) ->
    let s1, _ = js_of_aexpr module_name map env e1 in
    let locals, globals = merge_env env in
```

```
    let new_locals = NameMap.add s s locals in
    let s2, _ = js_of_aexpr module_name map (new_locals, globals) e2 in
    (try_catch_template s1 s s2), env

  | AVal(s, _) ->
      (* top level check if vals are global definitions *)
      (match s with
        | "print_num" | "print_bool" | "print"
        | "print_string" | "hd" | "tl" | "empty__"
        | "get" | "set" | "has__" | "keys" | "del" -> s, env
        | _ -> if NameMap.mem s map
               then (Printf.sprintf "%s.%s" module_name (remove_qmark s)), env
               else begin
                   let s = remove_qmark s in
                   let locals, globals = env in
                   if NameMap.mem s locals
                   then (NameMap.find s locals), env
                   else if NameMap.mem s globals
                       then (NameMap.find s globals), env
                       else raise (failwith ("not found in globals " ^ s))
               end)

  | AAssign(id, _, e, _) ->
  (* add map here *)
    if NameMap.mem id map
    (* then case occurs only for stdlib codegen, no env info required *)
    then (Printf.sprintf "%s.%s = %s" module_name (remove_qmark id) (fst (js_of_
aexpr module_name map env e))), env
    else begin
        let locals, globals = env in
        let name = remove_qmark id in
        let alias = name ^ string_of_int(Random.int 1000000) in
        let new_locals = NameMap.add name alias locals in
        let new_env = new_locals, globals in
        (Printf.sprintf "let %s = %s" alias (fst (js_of_aexpr module_name map ne
w_env e))), new_env
    end

  | ABlock(es, _) ->
    let new_env = merge_env env in
    let es, _ = ListLabels.fold_left ~init: ([], new_env) es
    ~f: (fun (acc, env) e ->
        let js_expr, new_env = (js_of_aexpr module_name map env e) in
        (js_expr :: acc), new_env)
    in
    (match es with
    | [] -> "" (* will never be reached *)
    | x :: [] -> block_template x None
    | x :: xs ->
        let es = String.concat ";\n" (List.rev xs) in
        block_template x (Some es)), env

  | AIf(p, e1, e2, _) ->
```

```
    let pred_s, _ = js_of_aexpr module_name map env p
    and s1, _ = js_of_aexpr module_name map env e1
    and s2, _ = js_of_aexpr module_name map env e2 in
    (if_template pred_s s1 s2), env

  | AFunLit(ids, body, t, _) ->
    let locals, globals = merge_env env in
    let aliases = List.map
    (fun id -> (remove_qmark id) ^ string_of_int(Random.int 1000000)) ids in
    let new_locals = ListLabels.fold_left2 ~init: locals ids aliases
    ~f: (fun locals id alias -> NameMap.add id alias locals) in
    let string_forms = String.concat "," aliases in

    let new_env = new_locals, globals in
    let string_body, _ = (match body with
        (* codegen for function blocks *)
        | ABlock(aes, _) -> begin
            let es, _ = ListLabels.fold_left ~init: ([], new_env) aes
            ~f: (fun (acc, env) e ->
                let js_expr, new_env = (js_of_aexpr module_name map env e) in
                (js_expr :: acc), new_env)
            in
            (match es with
            | [] -> "" (* will never be reached *)
            | x :: [] -> block_template x None
            | x :: xs ->
              let es = String.concat ";\n" (List.rev xs) in
              block_template x (Some es)), new_env
        end
        | _ -> js_of_aexpr module_name map new_env body) in
    let template = format_of_string "(function(%s) { return (%s) })" in
    (Printf.sprintf template string_forms string_body), env

  | ACall(e, es, _) ->
    let id = match e with
      | AVal(s, _) -> let name = (remove_qmark s) in
      (match name with
        | "print_num" | "print_bool" | "print"
        | "print_string" | "hd" | "tl" | "empty__"
        | "get" | "set" | "has__" | "keys" | "del" -> name
        | _ -> fst (js_of_aexpr module_name map env e))
      | AFunLit(_) -> fst (js_of_aexpr module_name map env e)
      | _ -> raise (failwith "not a function call") in
    let es = List.map (fun e -> fst (js_of_aexpr module_name map env e)) es in
    let fn_call = (match id with
     | "print_num" | "print_bool" | "print"
     | "print_string" -> Printf.sprintf "%s(%s)" id (String.concat "," es)
     | "hd" -> Printf.sprintf "(%s).get(0)" (List.hd es)
     | "tl" -> Printf.sprintf "(%s).delete(0)" (List.hd es)
     | "empty__" -> Printf.sprintf "(%s).isEmpty()" (List.hd es)
     | "get" -> Printf.sprintf "(%s).get((%s).toString())" (List.hd es) (List.nt
h es 1)
     | "set" -> Printf.sprintf "(%s).set((%s).toString(), %s)" (List.hd es) (Lis
```

```
t.nth es 1) (List.nth es 2)
     | "has__" -> Printf.sprintf "(%s).has((%s).toString())" (List.hd es) (List.
nth es 1)
     | "keys" -> Printf.sprintf "Immutable.fromJS(Array.from((%s).keys()))" (Lis
t.hd es)
     | "del" -> Printf.sprintf "(%s).remove((%s).toString())" (List.hd es) (List
.nth es 1)
     | _ -> Printf.sprintf "%s(%s)" id (String.concat "," es)) in
    fn_call, env

  | AListLit(es, _) -> let es = String.concat ", " (List.map (fun e ->
         fst (js_of_aexpr module_name map env e)) es) in
    (Printf.sprintf "Immutable.List.of(%s)" es), env

  | AModuleLit(id, e, _) ->
    let js_e = (match e with
         | AVal(prop, _) -> prop
         | ACall(prop, aargs, _) -> let js_args = List.map
           (fun aarg -> fst (js_of_aexpr module_name map env aarg)) aargs in
           let js_args = String.concat "," js_args in
           let prop = (match prop with
           | AVal(id, _) -> id
           | _ -> raise (failwith "mod call can't be funlit or anything else"))
in
           Printf.sprintf "%s(%s)" prop js_args
         | _ -> raise (failwith "unreachable state in module codegen")) in
    (Printf.sprintf "%s.%s" id js_e), env

  | AMapLit(kvpairs, _) ->
    let pairs = List.map (fun (k, v) ->
        Printf.sprintf "%s:%s" (fst (js_of_aexpr module_name map env k))
          (fst (js_of_aexpr module_name map env v))) kvpairs in
    (Printf.sprintf "Immutable.Map({ %s })" (String.concat "," pairs)), env
;;
```

src/exceptions.ml

```
(* Exceptions used in the JSJS compiler *)
open Ast
open Stringify

(* Scanner *)
exception IllegalCharacter of string * int * int

(* Typechecker *)
exception Undefined of string
exception CannotRedefineKeyword of string
exception InvalidOperation of primitiveType * op
exception MismatchedTypes of primitiveType * primitiveType
exception AlreadyDefined of string
exception NonUniformTypeContainer of primitiveType * primitiveType
exception MismatchedArgCount of int * int
```

```
exception UndefinedProperty of string * expr
exception ModuleNotFound of string
exception InvalidReturnExpression of string
exception InvalidKeyType of primitiveType

(* Driver *)
exception TypeError of string

let handle_error (e: exn) =
  match e with
  | ModuleNotFound(s) ->
    raise (TypeError (Printf.sprintf "Type error: Module '%s' not defined" s))
  | InvalidReturnExpression(s) ->
    raise (TypeError (Printf.sprintf "Type error: Assignment expressions cannot
be returned"))
  | InvalidOperation(t, op) ->
    let st = string_of_type t and sop = string_of_op op in
    raise (TypeError (Printf.sprintf "Type error: Invalid operation '%s' on type
 '%s'" sop st))
  | MismatchedTypes(t1, t2) ->
    let st1 = string_of_type t1 and st2 = string_of_type t2 in
    raise (TypeError (Printf.sprintf "Type error: expected value of type '%s', g
ot a value of type '%s' instead" st1 st2))
  | NonUniformTypeContainer(t1, t2) ->
    let st1 = string_of_type t1 and st2 = string_of_type t2 in
    raise (TypeError (Printf.sprintf "Type error: Lists can only contain one typ
e. Expected '%s', got a '%s' instead" st1 st2))
  | Undefined(s) ->
    raise (TypeError (Printf.sprintf "Error: value '%s' was used before it was d
efined" s))
  | AlreadyDefined(s) ->
    raise (TypeError (Printf.sprintf "Error: '%s' cannot be redefined in the cur
rent scope" s))
  | UndefinedProperty(module_name, e) ->
    let prop = string_of_expr e in
    raise (TypeError (Printf.sprintf "Error: property '%s' is not defined in mod
ule '%s'" prop module_name))
  | CannotRedefineKeyword(keyword) ->
    raise (TypeError (Printf.sprintf "Error: Cannot redefine keyword '%s'" keywo
rd))
  | MismatchedArgCount(l1, l2) ->
    raise (TypeError (Printf.sprintf "Error: Expected number of argument(s): %d,
 got %d instead." l1 l2))
  | InvalidKeyType(t) ->
    raise (TypeError (Printf.sprintf "Type Error: Cannot have '%s' as key type i
n a Map." (string_of_type t)))
  | e -> raise (TypeError (Printexc.to_string e))
;;
```

src/stringify.ml

```
(* utility functions for stringifying the AST *)

open Ast

module CharMap = Map.Make(String)

type genericMap = int CharMap.t

let string_of_op = function
  | Add       -> "+"
  | Mul       -> "*"
  | Neg | Sub -> "-"
  | Div       -> "/"
  | Mod       -> "%"
  | Caret     -> "^"
  | And       -> "&&"
  | Or        -> "||"
  | Not       -> "!"
  | Lte       -> "<="
  | Gte       -> ">="
  | Neq       -> "!="
  | Equals    -> "==="
  | Lt        -> "<"
  | Gt        -> ">"
  | Cons      -> "::"
;;

let string_of_type (t: Ast.primitiveType) =
  let rec aux (t: primitiveType) (chr: int) (map: genericMap) =
    match t with
    | TNum    -> "num", chr, map
    | TString -> "string", chr, map
    | TBool   -> "bool", chr, map
    | TAny    -> "any", chr, map
    | TExn    -> "exn", chr, map
    | TUnit   -> "unit", chr, map
    | T(x)    ->
      let gen_chr, new_chr, new_map = if CharMap.mem x map
        then Char.escaped (Char.chr (CharMap.find x map)), chr, map
        else
          let c = Char.escaped (Char.chr chr) in
          c, (chr + 1), CharMap.add x chr map
      in
      Printf.sprintf "%s" gen_chr, new_chr, new_map
    | TList(t) ->
      let st, c, m = aux t chr map in
      (Printf.sprintf "list %s" st), c, m
    | TMap(kt, vt) ->
      let st1, c1, m1 = aux kt chr map in
      let st2, c2, m2 = aux vt c1 m1 in
      (Printf.sprintf "<%s:%s>" st1 st2), c2, m2
```

```
    | TFun(args_type, ret_type) ->
      let sargs, (c, m) = ListLabels.fold_left args_type ~init: ([], (chr, map))
          ~f: (fun (ts, (c, m)) arg ->
              let argt, c1, m1 = aux arg c m in
              (argt :: ts, (c1, m1))
            ) in
      let rs, c, m = aux ret_type c m in
      let sargs = String.concat ", " sargs in
      Printf.sprintf "(%s) -> %s" sargs rs, c, m
  in
  let s, _, _ = aux t 65 CharMap.empty in s
;;

(* returns a stringified version of an expression *)
let rec string_of_expr = function
  | Binop(e1, o, e2) ->
    String.concat " " [string_of_expr e1; string_of_op o; string_of_expr e2]
  | Unop(o, e1) ->
    String.concat " " [string_of_op o; string_of_expr e1]
  | NumLit(x) -> string_of_float x
  | Throw(e)  -> let s = string_of_expr e in "throw " ^ s
  | TryCatch(e1, s, e2) -> let s1 = string_of_expr e1 and s2 = string_of_expr e2
 in
    Printf.sprintf "try %s catch(%s) %s" s1 s s2
  | UnitLit   -> "()"
  | Block(es) -> let ss = String.concat ";\n" (List.map string_of_expr es) in
    "{" ^ ss ^ "}"
  | StrLit(s) -> Printf.sprintf "'%s'" s
  | BoolLit(b) -> if b then "true" else "false"
  | Assign(s, t, e) ->
    let type_str = string_of_type t in
    let id_str = if type_str = "" then type_str else " : " ^ type_str in
    String.concat " " ["val"; s; id_str; "="; string_of_expr e]
  | Val(s) -> s
  | If(e1, b1, b2) ->
    let fhalf = string_of_expr b1 in
    let shalf = string_of_expr b2 in
    String.concat " " ["if"; string_of_expr e1; "then"; fhalf; "else"; shalf;]
  | ListLit(l) ->
    let s = String.concat "," (List.map string_of_expr l) in
    "[" ^ s ^ "]"
  | MapLit(l) ->
    let pairs = List.map (fun (k, v) -> string_of_expr k ^ ":" ^ string_of_expr
v) l in
    "{" ^ (String.concat ",\n" pairs) ^ "}"
  | Call(e, args) -> let ss = List.map string_of_expr args in
    String.concat " " [string_of_expr e; "("; (String.concat ", " ss); ")"]
  | ModuleLit(s, e) -> s ^ "." ^ (string_of_expr e)
  | FunLit(ids, body, t) -> begin
      let args_with_types, ret_type = (match t with
          | TFun(args_type, ret_type) -> List.combine ids args_type, ret_type
          | _ -> raise (failwith "not a valid function")) in
      let fargs = String.concat ", " (List.map (fun (id, typ) -> id ^ " : " ^ st
```

```
ring_of_type typ) args_with_types) in
      let fsig = "(" ^ fargs ^ ")" ^ " : " ^ (string_of_type ret_type) in
      let fbody = string_of_expr body in
      String.concat " " ["/\\"; fsig; "="; "{"; fbody; "}"]
    end
;;

let rec string_of_aexpr (ae: aexpr) : string =
  match ae with
  | ABoolLit(b, t) -> Printf.sprintf "(%s: %s)" (string_of_bool b) (string_of_ty
pe t)
  | ANumLit(x, t) -> Printf.sprintf "(%s: %s)" (string_of_float x) (string_of_ty
pe t)
  | AStrLit(s, t) -> Printf.sprintf "(%s: %s)" s (string_of_type t)
  | AUnitLit(t) -> Printf.sprintf "()"
  | AVal(s, t) -> Printf.sprintf "(%s: %s)" s (string_of_type t)
  | ABinop(e1, op, e2, t) ->
    let s1 = string_of_aexpr e1 in let s2 = string_of_aexpr e2 in
    let sop = string_of_op op in let st = string_of_type t in
    Printf.sprintf "(%s %s %s: %s)" s1 sop s2 st
  | AUnop(op, e, t) ->
    let s = string_of_aexpr e in
    let sop = string_of_op op in let st = string_of_type t in
    Printf.sprintf "(%s %s: %s)" s sop st
  | AAssign(id, t1, e, t2) ->
    Printf.sprintf "val %s: %s = %s : %s" id (string_of_type t1) (string_of_aexp
r e) (string_of_type t2)
  | AListLit(aes, t) ->
    let s = String.concat "," (List.map string_of_aexpr aes) in
    Printf.sprintf "[%s]:%s" s (string_of_type t)
  | AMapLit(kvpairs, t) ->  Printf.sprintf "map < >: %s" (string_of_type t)
  | AIf(ap, ae1, ae2, t) ->  Printf.sprintf "if {} then {} else {}: %s" (string_
of_type t)
  | ABlock(aes, t) -> let ss = List.map string_of_aexpr aes in
    Printf.sprintf "{ %s }" (String.concat "\n" ss)
  | AFunLit(ids, body, _, t) -> begin
      let args_with_types, ret_type = (match t with
          | TFun(args_type, ret_type) -> List.combine ids args_type, ret_type
          | _ -> raise (failwith "not a valid function")) in
      let fargs = String.concat ", " (List.map (fun (id, typ) -> id ^ " : " ^ st
ring_of_type typ) args_with_types) in
      let fsig = "(" ^ fargs ^ ")" ^ " : " ^ (string_of_type ret_type) in
      let fbody = string_of_aexpr body in
      String.concat " " ["/\\"; fsig; "="; "{"; fbody; "}"]
    end
  | ACall(afn, aargs, t) ->
    let sfn = string_of_aexpr afn
    and sargs = String.concat "," (List.map string_of_aexpr aargs) in
    Printf.sprintf "%s(%s) : %s" sfn sargs (string_of_type t)
  | AModuleLit(id, e, t) ->
    let se = string_of_aexpr e in
    Printf.sprintf "%s.(%s) : %s" id se (string_of_type t)
  | AThrow(ae, t) ->
```

```
    let se = string_of_aexpr ae in
    Printf.sprintf "throw %s: %s" se (string_of_type t)
  | ATryCatch(aetry, id, aecatch, t) ->
    let stry = string_of_aexpr aetry
    and scatch = string_of_aexpr aecatch in
    Printf.sprintf "try%s catch(%s)%s" stry id scatch
;;
```

src/driver.ml

```
open Ast
open Lexing
open Parsing
open Core.Std
open Codegen

type action = Compile | GenAST

let generate_stdlib file_path module_name =
  let lexbuf = Lexing.from_channel (open_in file_path) in
  let program = try Parser.program Scanner.token lexbuf with
    | Parser.Error -> Printf.printf "Error: There was an error in your syntax.";
 exit 1
  in
  let inferred_program = try Typecheck.type_check program
    with
    | Exceptions.TypeError(s) -> print_endline s; exit 1
    | e -> Printf.printf "%s" (Exn.to_string e); exit 1
  in
  let map = Lib.ModuleMap.find module_name Lib.modules in
  let js_exprs, _  = List.fold_left inferred_program
      ~f:(fun (acc, env) aexpr ->
          let js_expr, new_env = js_of_aexpr module_name map env aexpr in
          (js_expr :: acc, new_env))
      ~init: ([], (NameMap.empty, NameMap.empty))
  in
  let string_of_stdlib = String.concat ~sep:"\n" (List.rev js_exprs) in
  let template = format_of_string "
%s
" in
  Printf.sprintf template string_of_stdlib
;;

(* creates intermediate JS *)
let dump_javascript filename str =
  let template = format_of_string "'use strict'
%s
%s
%s
var num_to_string = function (x) { return x.toString(); };
var print_me = function(m) { console.log(m); };
// printing utils
```

```
var print_string = print_me;
var print_num = print_me;
var print_bool = print_me;
var print = print_me;
// generated code follows
%s" in
  let stdlib = [("List", "lib/list.jsjs"); ("Map", "lib/map.jsjs")] in
  let module_names = String.concat ~sep:"" (List.map
      ~f: (fun (x, _) -> Printf.sprintf "let %s = {};" x) stdlib)
  in
  let js_of_stdlib = List.fold_left ~init: "" stdlib
      ~f: (fun acc (name, path) -> acc ^ (generate_stdlib path name)) in
  let outc = Out_channel.create filename in
  Printf.fprintf outc template Lib.immutable module_names js_of_stdlib str;
  Out_channel.close outc
;;


let driver filename axn =
  let get_inchan = function
    | None | Some "-" -> In_channel.stdin
    | Some filename -> In_channel.create ~binary:true filename
  in
  let lexbuf = try Lexing.from_channel (get_inchan filename) with
    | Sys_error(s) -> Printf.printf "Error: %s\n" s; exit 1
  in
  let program = try Parser.program Scanner.token lexbuf with
    | Sys_error(s) -> Printf.printf "Error: %s\n" s; exit 1
    | Parser.Error -> Printf.printf "Error: There was a syntax error in your fil
e."; exit 1
    | _ -> Printf.printf "Error: Error in parsing file"; exit 1
  in
  (* TODO: Fix this error catching *)
  let inferred_program = try Typecheck.type_check program
    with
    | Exceptions.TypeError(s) -> Printf.printf "%s\n" s; exit 1
    | e -> Printf.printf "Error: %s\n" (Exn.to_string e); exit 1
  in

  (* JS -> AST -> JS *)
  let print_ast () =
    List.iter inferred_program
      ~f:(fun t -> print_endline (Stringify.string_of_aexpr t));
  in

  (* Compile *)
  let compile_to_js () =
    let js_exprs, _ =
      try List.fold_left inferred_program
        ~f:(fun (acc, env) aexpr ->
            let js_expr, new_env = Codegen.js_of_aexpr "" NameMap.empty env aexp
r in
            (js_expr :: acc, new_env))
```

```
        ~init: ([], (NameMap.empty, NameMap.empty))
      with
    | e -> Printf.printf "Error: %s\n" (Exn.to_string e); exit 1 in
    let s = String.concat ~sep:";\n" (List.rev js_exprs) in
    dump_javascript "out.js" s;
    print_endline "JS file ready - out.js";
  in

  match axn with
  | Compile -> compile_to_js ()
  | GenAST -> print_ast ()
;;

let command =
  Command.basic
    ~summary: "The JSJS compiler."
    ~readme: (fun () -> "Learn more at http://github.com/prakhar1989/JSJS")
    Command.Spec.(
      empty
      +> anon (maybe ("filename" %: file))
      +> flag "-s" no_arg ~doc:" JSJS -> AST -> JSJS"
    )
    (fun filename f1 () ->
       let axn = match f1 with
         | true  -> GenAST
         | false -> Compile
       in
       driver filename axn)
;;

let () = Command.run ~version:"0.1" command;;
```

src/lib.ml

```
open Ast

module ModuleMap = Map.Make(String);;
module NameMap = Map.Make(String);;

(**
 *  Copyright (c) 2014-2015, Facebook, Inc.
 *  All rights reserved.
 *
 *  This source code is licensed under the BSD-style license found in the
 *  LICENSE file in the root directory of this source tree. An additional grant
 *  of patent rights can be found in the PATENTS file in the same directory.
 *)
let immutable = "
var Immutable = (this,function(){\"use strict\";function t(t,e){e&&(t.prototype=
Object.create(e.prototype)),t.prototype.constructor=t}function e(t){return o(t)?
t:O(t)}function r(t){return u(t)?t:x(t)}function n(t){return s(t)?t:k(t)}functio
n i(t){return o(t)&&!a(t)?t:A(t)}function o(t){return!(!t||!t[ar])}function u(t)
```

```
{return!(!t||!t[hr])}function s(t){return!(!t||!t[fr])}function a(t){return u(t)
||s(t)}function h(t){return!(!t||!t[cr])}function f(t){return t.value=!1,t}funct
ion c(t){t&&(t.value=!0)}function _(){}function p(t,e){e=e||0;for(var r=Math.max
(0,t.length-e),n=Array(r),i=0;r>i;i++)n[i]=t[i+e];return n}function v(t){return
void 0===t.size&&(t.size=t.__iterate(y)),t.size}function l(t,e){if(\"number\"!=t
ypeof e){var r=e>>>0;if(\"\"+r!==e||4294967295===r)return NaN;e=r}return 0>e?v(t
)+e:e}function y(){return!0}function d(t,e,r){return(0===t||void 0!==r&&-r>=t)&&
(void 0===e||void 0!==r&&e>=r)}function m(t,e){return w(t,e,0)}function g(t,e){r
eturn w(t,e,e)}function w(t,e,r){return void 0===t?r:0>t?Math.max(0,e+t):void 0=
==e?t:Math.min(e,t)}function S(t){this.next=t}function z(t,e,r,n){var i=0===t?e:
1===t?r:[e,r];return n?n.value=i:n={value:i,done:!1},n}function I(){return{value
:void 0,done:!0}}function b(t){return!!M(t)}function q(t){return t&&\"function\"
==typeof t.next}function D(t){var e=M(t);return e&&e.call(t)}function M(t){var e
=t&&(zr&&t[zr]||t[Ir]);return\"function\"==typeof e?e:void 0}function E(t){retur
n t&&\"number\"==typeof t.length}function O(t){return null===t||void 0===t?T():o
(t)?t.toSeq():C(t)}function x(t){return null===t||void 0===t?T().toKeyedSeq():o(
t)?u(t)?t.toSeq():t.fromEntrySeq():W(t)}function k(t){return null===t||void 0===
t?T():o(t)?u(t)?t.entrySeq():t.toIndexedSeq():B(t)}function A(t){return(null===t
||void 0===t?T():o(t)?u(t)?t.entrySeq():t:B(t)).toSetSeq()}function j(t){this._a
rray=t,this.size=t.length}function K(t){var e=Object.keys(t);this._object=t,this
._keys=e,
this.size=e.length}function R(t){this._iterable=t,this.size=t.length||t.size}fun
ction U(t){this._iterator=t,this._iteratorCache=[]}function L(t){return!(!t||!t[
qr])}function T(){return Dr||(Dr=new j([]))}function W(t){var e=Array.isArray(t)
?new j(t).fromEntrySeq():q(t)?new U(t).fromEntrySeq():b(t)?new R(t).fromEntrySeq
():\"object\"==typeof t?new K(t):void 0;if(!e)throw new TypeError(\"Expected Arr
ay or iterable object of [k, v] entries, or keyed object: \"+t);return e}functio
n B(t){var e=J(t);if(!e)throw new TypeError(\"Expected Array or iterable object
of values: \"+t);return e}function C(t){var e=J(t)||\"object\"==typeof t&&new K(
t);if(!e)throw new TypeError(\"Expected Array or iterable object of values, or k
eyed object: \"+t);return e}function J(t){return E(t)?new j(t):q(t)?new U(t):b(t
)?new R(t):void 0}function N(t,e,r,n){var i=t._cache;if(i){for(var o=i.length-1,
u=0;o>=u;u++){var s=i[r?o-u:u];if(e(s[1],n?s[0]:u,t)===!1)return u+1}return u}re
turn t.__iterateUncached(e,r)}function P(t,e,r,n){var i=t._cache;if(i){var o=i.l
ength-1,u=0;return new S(function(){var t=i[r?o-u:u];return u++>o?I():z(e,n?t[0]
:u-1,t[1])})}return t.__iteratorUncached(e,r)}function H(t,e){return e?V(e,t,\"\
\",{\"\":t}):Y(t)}function V(t,e,r,n){return Array.isArray(e)?t.call(n,r,k(e).map
(function(r,n){return V(t,r,n,e)})):Q(e)?t.call(n,r,x(e).map(function(r,n){retur
n V(t,r,n,e)})):e}function Y(t){return Array.isArray(t)?k(t).map(Y).toList():Q(t
)?x(t).map(Y).toMap():t}function Q(t){return t&&(t.constructor===Object||void 0=
==t.constructor)}function X(t,e){if(t===e||t!==t&&e!==e)return!0;if(!t||!e)retur
n!1;if(\"function\"==typeof t.valueOf&&\"function\"==typeof e.valueOf){if(t=t.va
lueOf(),e=e.valueOf(),t===e||t!==t&&e!==e)return!0;if(!t||!e)return!1}return\"fu
nction\"==typeof t.equals&&\"function\"==typeof e.equals&&t.equals(e)?!0:!1}func
tion F(t,e){if(t===e)return!0;if(!o(e)||void 0!==t.size&&void 0!==e.size&&t.size
!==e.size||void 0!==t.__hash&&void 0!==e.__hash&&t.__hash!==e.__hash||u(t)!==u(e
)||s(t)!==s(e)||h(t)!==h(e))return!1;if(0===t.size&&0===e.size)return!0;
var r=!a(t);if(h(t)){var n=t.entries();return e.every(function(t,e){var i=n.next
().value;return i&&X(i[1],t)&&(r||X(i[0],e))})&&n.next().done}var i=!1;if(void 0
===t.size)if(void 0===e.size)\"function\"==typeof t.cacheResult&&t.cacheResult()
;else{i=!0;var f=t;t=e,e=f}var c=!0,_=e.__iterate(function(e,n){return(r?t.has(e
):i?X(e,t.get(n,yr)):X(t.get(n,yr),e))?void 0:(c=!1,!1)});return c&&t.size===_}f
unction G(t,e){if(!(this instanceof G))return new G(t,e);if(this._value=t,this.s
```

```
ize=void 0===e?1/0:Math.max(0,e),0===this.size){if(Mr)return Mr;Mr=this}}functio
n Z(t,e){if(!t)throw Error(e)}function $(t,e,r){if(!(this instanceof $))return n
ew $(t,e,r);if(Z(0!==r,\"Cannot step a Range by 0\"),t=t||0,void 0===e&&(e=1/0),
r=void 0===r?1:Math.abs(r),t>e&&(r=-r),this._start=t,this._end=e,this._step=r,th
is.size=Math.max(0,Math.ceil((e-t)/r-1)+1),0===this.size){if(Er)return Er;Er=thi
s}}function tt(){throw TypeError(\"Abstract\")}function et(){}function rt(){}fun
ction nt(){}function it(t){return t>>>1&1073741824|3221225471&t}function ot(t){i
f(t===!1||null===t||void 0===t)return 0;if(\"function\"==typeof t.valueOf&&(t=t.
valueOf(),t===!1||null===t||void 0===t))return 0;if(t===!0)return 1;var e=typeof
 t;if(\"number\"===e){var r=0|t;for(r!==t&&(r^=4294967295*t);t>4294967295;)t/=42
94967295,r^=t;return it(r)}if(\"string\"===e)return t.length>Ur?ut(t):st(t);if(\
"function\"==typeof t.hashCode)return t.hashCode();if(\"object\"===e)return at(t
);if(\"function\"==typeof t.toString)return st(\"\"+t);throw Error(\"Value type
\"+e+\" cannot be hashed.\")}function ut(t){var e=Wr[t];return void 0===e&&(e=st
(t),Tr===Lr&&(Tr=0,Wr={}),Tr++,Wr[t]=e),e}function st(t){for(var e=0,r=0;t.lengt
h>r;r++)e=31*e+t.charCodeAt(r)|0;return it(e)}function at(t){var e;if(jr&&(e=Or.
get(t),void 0!==e))return e;if(e=t[Rr],void 0!==e)return e;if(!Ar){if(e=t.proper
tyIsEnumerable&&t.propertyIsEnumerable[Rr],void 0!==e)return e;if(e=ht(t),void 0
!==e)return e}if(e=++Kr,1073741824&Kr&&(Kr=0),jr)Or.set(t,e);else{if(void 0!==kr
&&kr(t)===!1)throw Error(\"Non-extensible objects are not allowed as keys.\");
if(Ar)Object.defineProperty(t,Rr,{enumerable:!1,configurable:!1,writable:!1,valu
e:e});else if(void 0!==t.propertyIsEnumerable&&t.propertyIsEnumerable===t.constr
uctor.prototype.propertyIsEnumerable)t.propertyIsEnumerable=function(){return th
is.constructor.prototype.propertyIsEnumerable.apply(this,arguments)},t.propertyI
sEnumerable[Rr]=e;else{if(void 0===t.nodeType)throw Error(\"Unable to set a non-
enumerable property on object.\");t[Rr]=e}}return e}function ht(t){if(t&&t.nodeT
ype>0)switch(t.nodeType){case 1:return t.uniqueID;case 9:return t.documentElemen
t&&t.documentElement.uniqueID}}function ft(t){Z(t!==1/0,\"Cannot perform this ac
tion with an infinite size.\")}function ct(t){return null===t||void 0===t?zt():_
t(t)&&!h(t)?t:zt().withMutations(function(e){var n=r(t);ft(n.size),n.forEach(fun
ction(t,r){return e.set(r,t)})})}function _t(t){return!(!t||!t[Br])}function pt(
t,e){this.ownerID=t,this.entries=e}function vt(t,e,r){this.ownerID=t,this.bitmap
=e,this.nodes=r}function lt(t,e,r){this.ownerID=t,this.count=e,this.nodes=r}func
tion yt(t,e,r){this.ownerID=t,this.keyHash=e,this.entries=r}function dt(t,e,r){t
his.ownerID=t,this.keyHash=e,this.entry=r}function mt(t,e,r){this._type=e,this._
reverse=r,this._stack=t._root&&wt(t._root)}function gt(t,e){return z(t,e[0],e[1]
)}function wt(t,e){return{node:t,index:0,__prev:e}}function St(t,e,r,n){var i=Ob
ject.create(Cr);return i.size=t,i._root=e,i.__ownerID=r,i.__hash=n,i.__altered=!
1,i}function zt(){return Jr||(Jr=St(0))}function It(t,e,r){var n,i;if(t._root){v
ar o=f(dr),u=f(mr);if(n=bt(t._root,t.__ownerID,0,void 0,e,r,o,u),!u.value)return
 t;i=t.size+(o.value?r===yr?-1:1:0)}else{if(r===yr)return t;i=1,n=new pt(t.__own
erID,[[e,r]])}return t.__ownerID?(t.size=i,t._root=n,t.__hash=void 0,t.__altered
=!0,t):n?St(i,n):zt()}function bt(t,e,r,n,i,o,u,s){return t?t.update(e,r,n,i,o,u
,s):o===yr?t:(c(s),c(u),new dt(e,n,[i,o]))}function qt(t){return t.constructor==
=dt||t.constructor===yt}function Dt(t,e,r,n,i){if(t.keyHash===n)return new yt(e,
n,[t.entry,i]);var o,u=(0===r?t.keyHash:t.keyHash>>>r)&lr,s=(0===r?n:n>>>r)&lr,a
=u===s?[Dt(t,e,r+pr,n,i)]:(o=new dt(e,n,i),
s>u?[t,o]:[o,t]);return new vt(e,1<<u|1<<s,a)}function Mt(t,e,r,n){t||(t=new _);
for(var i=new dt(t,ot(r),[r,n]),o=0;e.length>o;o++){var u=e[o];i=i.update(t,0,vo
id 0,u[0],u[1])}return i}function Et(t,e,r,n){for(var i=0,o=0,u=Array(r),s=0,a=1
,h=e.length;h>s;s++,a<<=1){var f=e[s];void 0!==f&&s!==n&&(i|=a,u[o++]=f)}return
new vt(t,i,u)}function Ot(t,e,r,n,i){for(var o=0,u=Array(vr),s=0;0!==r;s++,r>>>=
1)u[s]=1&r?e[o++]:void 0;return u[n]=i,new lt(t,o+1,u)}function xt(t,e,n){for(va
```

```
r i=[],u=0;n.length>u;u++){var s=n[u],a=r(s);o(s)||(a=a.map(function(t){return H
(t)})),i.push(a)}return jt(t,e,i)}function kt(t,e,r){return t&&t.mergeDeep&&o(e)
?t.mergeDeep(e):X(t,e)?t:e}function At(t){return function(e,r,n){if(e&&e.mergeDe
epWith&&o(r))return e.mergeDeepWith(t,r);var i=t(e,r,n);return X(e,i)?e:i}}funct
ion jt(t,e,r){return r=r.filter(function(t){return 0!==t.size}),0===r.length?t:0
!==t.size||t.__ownerID||1!==r.length?t.withMutations(function(t){for(var n=e?fun
ction(r,n){t.update(n,yr,function(t){return t===yr?r:e(t,r,n)})}:function(e,r){t
.set(r,e)},i=0;r.length>i;i++)r[i].forEach(n)}):t.constructor(r[0])}function Kt(
t,e,r,n){var i=t===yr,o=e.next();if(o.done){var u=i?r:t,s=n(u);return s===u?t:s}
Z(i||t&&t.set,\"invalid keyPath\");var a=o.value,h=i?yr:t.get(a,yr),f=Kt(h,e,r,n
);return f===h?t:f===yr?t.remove(a):(i?zt():t).set(a,f)}function Rt(t){return t-
=t>>1&1431655765,t=(858993459&t)+(t>>2&858993459),t=t+(t>>4)&252645135,t+=t>>8,t
+=t>>16,127&t}function Ut(t,e,r,n){var i=n?t:p(t);return i[e]=r,i}function Lt(t,
e,r,n){var i=t.length+1;if(n&&e+1===i)return t[e]=r,t;for(var o=Array(i),u=0,s=0
;i>s;s++)s===e?(o[s]=r,u=-1):o[s]=t[s+u];return o}function Tt(t,e,r){var n=t.len
gth-1;if(r&&e===n)return t.pop(),t;for(var i=Array(n),o=0,u=0;n>u;u++)u===e&&(o=
1),i[u]=t[u+o];return i}function Wt(t){var e=Pt();if(null===t||void 0===t)return
 e;if(Bt(t))return t;var r=n(t),i=r.size;return 0===i?e:(ft(i),i>0&&vr>i?Nt(0,i,
pr,null,new Ct(r.toArray())):e.withMutations(function(t){t.setSize(i),r.forEach(
function(e,r){return t.set(r,e)})}))}function Bt(t){
return!(!t||!t[Vr])}function Ct(t,e){this.array=t,this.ownerID=e}function Jt(t,e
){function r(t,e,r){return 0===e?n(t,r):i(t,e,r)}function n(t,r){var n=r===s?a&&
a.array:t&&t.array,i=r>o?0:o-r,h=u-r;return h>vr&&(h=vr),function(){if(i===h)ret
urn Xr;var t=e?--h:i++;return n&&n[t]}}function i(t,n,i){var s,a=t&&t.array,h=i>
o?0:o-i>>n,f=(u-i>>n)+1;return f>vr&&(f=vr),function(){for(;;){if(s){var t=s();i
f(t!==Xr)return t;s=null}if(h===f)return Xr;var o=e?--f:h++;s=r(a&&a[o],n-pr,i+(
o<<n))}}}var o=t._origin,u=t._capacity,s=Gt(u),a=t._tail;return r(t._root,t._lev
el,0)}function Nt(t,e,r,n,i,o,u){var s=Object.create(Yr);return s.size=e-t,s._or
igin=t,s._capacity=e,s._level=r,s._root=n,s._tail=i,s.__ownerID=o,s.__hash=u,s._
_altered=!1,s}function Pt(){return Qr||(Qr=Nt(0,0,pr))}function Ht(t,e,r){if(e=l
(t,e),e!==e)return t;if(e>=t.size||0>e)return t.withMutations(function(t){0>e?Xt
(t,e).set(0,r):Xt(t,0,e+1).set(e,r)});e+=t._origin;var n=t._tail,i=t._root,o=f(m
r);return e>=Gt(t._capacity)?n=Vt(n,t.__ownerID,0,e,r,o):i=Vt(i,t.__ownerID,t._l
evel,e,r,o),o.value?t.__ownerID?(t._root=i,t._tail=n,t.__hash=void 0,t.__altered
=!0,t):Nt(t._origin,t._capacity,t._level,i,n):t}function Vt(t,e,r,n,i,o){var u=n
>>>r&lr,s=t&&t.array.length>u;if(!s&&void 0===i)return t;var a;if(r>0){var h=t&&
t.array[u],f=Vt(h,e,r-pr,n,i,o);return f===h?t:(a=Yt(t,e),a.array[u]=f,a)}return
 s&&t.array[u]===i?t:(c(o),a=Yt(t,e),void 0===i&&u===a.array.length-1?a.array.po
p():a.array[u]=i,a)}function Yt(t,e){return e&&t&&e===t.ownerID?t:new Ct(t?t.arr
ay.slice():[],e)}function Qt(t,e){if(e>=Gt(t._capacity))return t._tail;if(1<<t._
level+pr>e){for(var r=t._root,n=t._level;r&&n>0;)r=r.array[e>>>n&lr],n-=pr;retur
n r}}function Xt(t,e,r){void 0!==e&&(e=0|e),void 0!==r&&(r=0|r);var n=t.__ownerI
D||new _,i=t._origin,o=t._capacity,u=i+e,s=void 0===r?o:0>r?o+r:i+r;if(u===i&&s=
==o)return t;if(u>=s)return t.clear();for(var a=t._level,h=t._root,f=0;0>u+f;)h=
new Ct(h&&h.array.length?[void 0,h]:[],n),a+=pr,f+=1<<a;f&&(u+=f,i+=f,s+=f,o+=f)
;for(var c=Gt(o),p=Gt(s);p>=1<<a+pr;)h=new Ct(h&&h.array.length?[h]:[],n),
a+=pr;var v=t._tail,l=c>p?Qt(t,s-1):p>c?new Ct([],n):v;if(v&&p>c&&o>u&&v.array.l
ength){h=Yt(h,n);for(var y=h,d=a;d>pr;d-=pr){var m=c>>>d&lr;y=y.array[m]=Yt(y.ar
ray[m],n)}y.array[c>>>pr&lr]=v}if(o>s&&(l=l&&l.removeAfter(n,0,s)),u>=p)u-=p,s-=
p,a=pr,h=null,l=l&&l.removeBefore(n,0,u);else if(u>i||c>p){for(f=0;h;){var g=u>>
>a&lr;if(g!==p>>>a&lr)break;g&&(f+=(1<<a)*g),a-=pr,h=h.array[g]}h&&u>i&&(h=h.rem
oveBefore(n,a,u-f)),h&&c>p&&(h=h.removeAfter(n,a,p-f)),f&&(u-=f,s-=f)}return t._
_ownerID?(t.size=s-u,t._origin=u,t._capacity=s,t._level=a,t._root=h,t._tail=l,t.
```

```
__hash=void 0,t.__altered=!0,t):Nt(u,s,a,h,l)}function Ft(t,e,r){for(var i=[],u=
0,s=0;r.length>s;s++){var a=r[s],h=n(a);h.size>u&&(u=h.size),o(a)||(h=h.map(func
tion(t){return H(t)})),i.push(h)}return u>t.size&&(t=t.setSize(u)),jt(t,e,i)}fun
ction Gt(t){return vr>t?0:t-1>>>pr<<pr}function Zt(t){return null===t||void 0===
t?ee():$t(t)?t:ee().withMutations(function(e){var n=r(t);ft(n.size),n.forEach(fu
nction(t,r){return e.set(r,t)})})}function $t(t){return _t(t)&&h(t)}function te(
t,e,r,n){var i=Object.create(Zt.prototype);return i.size=t?t.size:0,i._map=t,i._
list=e,i.__ownerID=r,i.__hash=n,i}function ee(){return Fr||(Fr=te(zt(),Pt()))}fu
nction re(t,e,r){var n,i,o=t._map,u=t._list,s=o.get(e),a=void 0!==s;if(r===yr){i
f(!a)return t;u.size>=vr&&u.size>=2*o.size?(i=u.filter(function(t,e){return void
 0!==t&&s!==e}),n=i.toKeyedSeq().map(function(t){return t[0]}).flip().toMap(),t.
__ownerID&&(n.__ownerID=i.__ownerID=t.__ownerID)):(n=o.remove(e),i=s===u.size-1?
u.pop():u.set(s,void 0))}else if(a){if(r===u.get(s)[1])return t;n=o,i=u.set(s,[e
,r])}else n=o.set(e,u.size),i=u.set(u.size,[e,r]);return t.__ownerID?(t.size=n.s
ize,t._map=n,t._list=i,t.__hash=void 0,t):te(n,i)}function ne(t,e){this._iter=t,
this._useKeys=e,this.size=t.size}function ie(t){this._iter=t,this.size=t.size}fu
nction oe(t){this._iter=t,this.size=t.size}function ue(t){this._iter=t,this.size
=t.size}function se(t){var e=Ee(t);return e._iter=t,e.size=t.size,e.flip=functio
n(){return t},e.reverse=function(){var e=t.reverse.apply(this);
return e.flip=function(){return t.reverse()},e},e.has=function(e){return t.inclu
des(e)},e.includes=function(e){return t.has(e)},e.cacheResult=Oe,e.__iterateUnca
ched=function(e,r){var n=this;return t.__iterate(function(t,r){return e(r,t,n)!=
=!1},r)},e.__iteratorUncached=function(e,r){if(e===Sr){var n=t.__iterator(e,r);r
eturn new S(function(){var t=n.next();if(!t.done){var e=t.value[0];t.value[0]=t.
value[1],t.value[1]=e}return t})}return t.__iterator(e===wr?gr:wr,r)},e}function
 ae(t,e,r){var n=Ee(t);return n.size=t.size,n.has=function(e){return t.has(e)},n
.get=function(n,i){var o=t.get(n,yr);return o===yr?i:e.call(r,o,n,t)},n.__iterat
eUncached=function(n,i){var o=this;return t.__iterate(function(t,i,u){return n(e
.call(r,t,i,u),i,o)!==!1},i)},n.__iteratorUncached=function(n,i){var o=t.__itera
tor(Sr,i);return new S(function(){var i=o.next();if(i.done)return i;var u=i.valu
e,s=u[0];return z(n,s,e.call(r,u[1],s,t),i)})},n}function he(t,e){var r=Ee(t);re
turn r._iter=t,r.size=t.size,r.reverse=function(){return t},t.flip&&(r.flip=func
tion(){var e=se(t);return e.reverse=function(){return t.flip()},e}),r.get=functi
on(r,n){return t.get(e?r:-1-r,n)},r.has=function(r){return t.has(e?r:-1-r)},r.in
cludes=function(e){return t.includes(e)},r.cacheResult=Oe,r.__iterate=function(e
,r){var n=this;return t.__iterate(function(t,r){return e(t,r,n)},!r)},r.__iterat
or=function(e,r){return t.__iterator(e,!r)},r}function fe(t,e,r,n){var i=Ee(t);r
eturn n&&(i.has=function(n){var i=t.get(n,yr);return i!==yr&&!!e.call(r,i,n,t)},
i.get=function(n,i){var o=t.get(n,yr);return o!==yr&&e.call(r,o,n,t)?o:i}),i.__i
terateUncached=function(i,o){var u=this,s=0;return t.__iterate(function(t,o,a){r
eturn e.call(r,t,o,a)?(s++,i(t,n?o:s-1,u)):void 0},o),s},i.__iteratorUncached=fu
nction(i,o){var u=t.__iterator(Sr,o),s=0;return new S(function(){for(;;){var o=u
.next();if(o.done)return o;var a=o.value,h=a[0],f=a[1];if(e.call(r,f,h,t))return
 z(i,n?h:s++,f,o)}})},i}function ce(t,e,r){var n=ct().asMutable();return t.__ite
rate(function(i,o){n.update(e.call(r,i,o,t),0,function(t){
return t+1})}),n.asImmutable()}function _e(t,e,r){var n=u(t),i=(h(t)?Zt():ct()).
asMutable();t.__iterate(function(o,u){i.update(e.call(r,o,u,t),function(t){retur
n t=t||[],t.push(n?[u,o]:o),t})});var o=Me(t);return i.map(function(e){return be
(t,o(e))})}function pe(t,e,r,n){var i=t.size;if(void 0!==e&&(e=0|e),void 0!==r&&
(r=0|r),d(e,r,i))return t;var o=m(e,i),u=g(r,i);if(o!==o||u!==u)return pe(t.toSe
q().cacheResult(),e,r,n);var s,a=u-o;a===a&&(s=0>a?0:a);var h=Ee(t);return h.siz
e=0===s?s:t.size&&s||void 0,!n&&L(t)&&s>=0&&(h.get=function(e,r){return e=l(this
,e),e>=0&&s>e?t.get(e+o,r):r}),h.__iterateUncached=function(e,r){var i=this;if(0
```

```
===s)return 0;if(r)return this.cacheResult().__iterate(e,r);var u=0,a=!0,h=0;ret
urn t.__iterate(function(t,r){return a&&(a=u++<o)?void 0:(h++,e(t,n?r:h-1,i)!==!
1&&h!==s)}),h},h.__iteratorUncached=function(e,r){if(0!==s&&r)return this.cacheR
esult().__iterator(e,r);var i=0!==s&&t.__iterator(e,r),u=0,a=0;return new S(func
tion(){for(;u++<o;)i.next();if(++a>s)return I();var t=i.next();return n||e===wr?
t:e===gr?z(e,a-1,void 0,t):z(e,a-1,t.value[1],t)})},h}function ve(t,e,r){var n=E
e(t);return n.__iterateUncached=function(n,i){var o=this;if(i)return this.cacheR
esult().__iterate(n,i);var u=0;return t.__iterate(function(t,i,s){return e.call(
r,t,i,s)&&++u&&n(t,i,o)}),u},n.__iteratorUncached=function(n,i){var o=this;if(i)
return this.cacheResult().__iterator(n,i);var u=t.__iterator(Sr,i),s=!0;return n
ew S(function(){if(!s)return I();var t=u.next();if(t.done)return t;var i=t.value
,a=i[0],h=i[1];return e.call(r,h,a,o)?n===Sr?t:z(n,a,h,t):(s=!1,I())})},n}functi
on le(t,e,r,n){var i=Ee(t);return i.__iterateUncached=function(i,o){var u=this;i
f(o)return this.cacheResult().__iterate(i,o);var s=!0,a=0;return t.__iterate(fun
ction(t,o,h){return s&&(s=e.call(r,t,o,h))?void 0:(a++,i(t,n?o:a-1,u))}),a},i.__
iteratorUncached=function(i,o){var u=this;if(o)return this.cacheResult().__itera
tor(i,o);var s=t.__iterator(Sr,o),a=!0,h=0;return new S(function(){var t,o,f;do{
if(t=s.next(),t.done)return n||i===wr?t:i===gr?z(i,h++,void 0,t):z(i,h++,t.value
[1],t);
var c=t.value;o=c[0],f=c[1],a&&(a=e.call(r,f,o,u))}while(a);return i===Sr?t:z(i,
o,f,t)})},i}function ye(t,e){var n=u(t),i=[t].concat(e).map(function(t){return o
(t)?n&&(t=r(t)):t=n?W(t):B(Array.isArray(t)?t:[t]),t}).filter(function(t){return
 0!==t.size});if(0===i.length)return t;if(1===i.length){var a=i[0];if(a===t||n&&
u(a)||s(t)&&s(a))return a}var h=new j(i);return n?h=h.toKeyedSeq():s(t)||(h=h.to
SetSeq()),h=h.flatten(!0),h.size=i.reduce(function(t,e){if(void 0!==t){var r=e.s
ize;if(void 0!==r)return t+r}},0),h}function de(t,e,r){var n=Ee(t);return n.__it
erateUncached=function(n,i){function u(t,h){var f=this;t.__iterate(function(t,i)
{return(!e||e>h)&&o(t)?u(t,h+1):n(t,r?i:s++,f)===!1&&(a=!0),!a},i)}var s=0,a=!1;
return u(t,0),s},n.__iteratorUncached=function(n,i){var u=t.__iterator(n,i),s=[]
,a=0;return new S(function(){for(;u;){var t=u.next();if(t.done===!1){var h=t.val
ue;if(n===Sr&&(h=h[1]),e&&!(e>s.length)||!o(h))return r?t:z(n,a++,h,t);s.push(u)
,u=h.__iterator(n,i)}else u=s.pop()}return I()})},n}function me(t,e,r){var n=Me(
t);return t.toSeq().map(function(i,o){return n(e.call(r,i,o,t))}).flatten(!0)}fu
nction ge(t,e){var r=Ee(t);return r.size=t.size&&2*t.size-1,r.__iterateUncached=
function(r,n){var i=this,o=0;return t.__iterate(function(t,n){return(!o||r(e,o++
,i)!==!1)&&r(t,o++,i)!==!1},n),o},r.__iteratorUncached=function(r,n){var i,o=t._
_iterator(wr,n),u=0;return new S(function(){return(!i||u%2)&&(i=o.next(),i.done)
?i:u%2?z(r,u++,e):z(r,u++,i.value,i)})},r}function we(t,e,r){e||(e=xe);var n=u(t
),i=0,o=t.toSeq().map(function(e,n){return[n,e,i++,r?r(e,n,t):e]}).toArray();ret
urn o.sort(function(t,r){return e(t[3],r[3])||t[2]-r[2]}).forEach(n?function(t,e
){o[e].length=2}:function(t,e){o[e]=t[1]}),n?x(o):s(t)?k(o):A(o)}function Se(t,e
,r){if(e||(e=xe),r){var n=t.toSeq().map(function(e,n){return[e,r(e,n,t)]}).reduc
e(function(t,r){return ze(e,t[1],r[1])?r:t});return n&&n[0]}return t.reduce(func
tion(t,r){return ze(e,t,r)?r:t})}function ze(t,e,r){var n=t(r,e);return 0===n&&r
!==e&&(void 0===r||null===r||r!==r)||n>0}function Ie(t,r,n){
var i=Ee(t);return i.size=new j(n).map(function(t){return t.size}).min(),i.__ite
rate=function(t,e){for(var r,n=this.__iterator(wr,e),i=0;!(r=n.next()).done&&t(r
.value,i++,this)!==!1;);return i},i.__iteratorUncached=function(t,i){var o=n.map
(function(t){return t=e(t),D(i?t.reverse():t)}),u=0,s=!1;return new S(function()
{var e;return s||(e=o.map(function(t){return t.next()}),s=e.some(function(t){ret
urn t.done})),s?I():z(t,u++,r.apply(null,e.map(function(t){return t.value})))})}
,i}function be(t,e){return L(t)?e:t.constructor(e)}function qe(t){if(t!==Object(
t))throw new TypeError(\"Expected [K, V] tuple: \"+t)}function De(t){return ft(t
```

```
.size),v(t)}function Me(t){return u(t)?r:s(t)?n:i}function Ee(t){return Object.c
reate((u(t)?x:s(t)?k:A).prototype)}function Oe(){return this._iter.cacheResult?(
this._iter.cacheResult(),this.size=this._iter.size,this):O.prototype.cacheResult
.call(this)}function xe(t,e){return t>e?1:e>t?-1:0}function ke(t){var r=D(t);if(
!r){if(!E(t))throw new TypeError(\"Expected iterable or array-like: \"+t);r=D(e(
t))}return r}function Ae(t,e){var r,n=function(o){if(o instanceof n)return o;if(
!(this instanceof n))return new n(o);if(!r){r=!0;var u=Object.keys(t);Re(i,u),i.
size=u.length,i._name=e,i._keys=u,i._defaultValues=t}this._map=ct(o)},i=n.protot
ype=Object.create(Gr);return i.constructor=n,n}function je(t,e,r){var n=Object.c
reate(Object.getPrototypeOf(t));return n._map=e,n.__ownerID=r,n}function Ke(t){r
eturn t._name||t.constructor.name||\"Record\"}function Re(t,e){try{e.forEach(Ue.
bind(void 0,t))}catch(r){}}function Ue(t,e){Object.defineProperty(t,e,{get:funct
ion(){return this.get(e)},set:function(t){Z(this.__ownerID,\"Cannot set on an im
mutable record.\"),this.set(e,t)}})}function Le(t){return null===t||void 0===t?C
e():Te(t)&&!h(t)?t:Ce().withMutations(function(e){var r=i(t);ft(r.size),r.forEac
h(function(t){return e.add(t)})})}function Te(t){return!(!t||!t[Zr])}function We
(t,e){return t.__ownerID?(t.size=e.size,t._map=e,t):e===t._map?t:0===e.size?t.__
empty():t.__make(e)}function Be(t,e){var r=Object.create($r);
return r.size=t?t.size:0,r._map=t,r.__ownerID=e,r}function Ce(){return tn||(tn=B
e(zt()))}function Je(t){return null===t||void 0===t?He():Ne(t)?t:He().withMutati
ons(function(e){var r=i(t);ft(r.size),r.forEach(function(t){return e.add(t)})})}
function Ne(t){return Te(t)&&h(t)}function Pe(t,e){var r=Object.create(en);retur
n r.size=t?t.size:0,r._map=t,r.__ownerID=e,r}function He(){return rn||(rn=Pe(ee(
)))}function Ve(t){return null===t||void 0===t?Xe():Ye(t)?t:Xe().unshiftAll(t)}f
unction Ye(t){return!(!t||!t[nn])}function Qe(t,e,r,n){var i=Object.create(on);r
eturn i.size=t,i._head=e,i.__ownerID=r,i.__hash=n,i.__altered=!1,i}function Xe()
{return un||(un=Qe(0))}function Fe(t,e){var r=function(r){t.prototype[r]=e[r]};r
eturn Object.keys(e).forEach(r),Object.getOwnPropertySymbols&&Object.getOwnPrope
rtySymbols(e).forEach(r),t}function Ge(t,e){return e}function Ze(t,e){return[e,t
]}function $e(t){return function(){return!t.apply(this,arguments)}}function tr(t
){return function(){return-t.apply(this,arguments)}}function er(t){return\"strin
g\"==typeof t?JSON.stringify(t):t}function rr(){return p(arguments)}function nr(
t,e){return e>t?1:t>e?-1:0}function ir(t){if(t.size===1/0)return 0;var e=h(t),r=
u(t),n=e?1:0,i=t.__iterate(r?e?function(t,e){n=31*n+ur(ot(t),ot(e))|0}:function(
t,e){n=n+ur(ot(t),ot(e))|0}:e?function(t){n=31*n+ot(t)|0}:function(t){n=n+ot(t)|
0});return or(i,n)}function or(t,e){return e=xr(e,3432918353),e=xr(e<<15|e>>>-15
,461845907),e=xr(e<<13|e>>>-13,5),e=(e+3864292196|0)^t,e=xr(e^e>>>16,2246822507)
,e=xr(e^e>>>13,3266489909),e=it(e^e>>>16)}function ur(t,e){return t^e+2654435769
+(t<<6)+(t>>2)|0}var sr=Array.prototype.slice;t(r,e),t(n,e),t(i,e),e.isIterable=
o,e.isKeyed=u,e.isIndexed=s,e.isAssociative=a,e.isOrdered=h,e.Keyed=r,e.Indexed=
n,e.Set=i;var ar=\"@@__IMMUTABLE_ITERABLE__@@\",hr=\"@@__IMMUTABLE_KEYED__@@\",f
r=\"@@__IMMUTABLE_INDEXED__@@\",cr=\"@@__IMMUTABLE_ORDERED__@@\",_r=\"delete\",p
r=5,vr=1<<pr,lr=vr-1,yr={},dr={value:!1},mr={value:!1},gr=0,wr=1,Sr=2,zr=\"funct
ion\"==typeof Symbol&&Symbol.iterator,Ir=\"@@iterator\",br=zr||Ir;
S.prototype.toString=function(){return\"[Iterator]\"},S.KEYS=gr,S.VALUES=wr,S.EN
TRIES=Sr,S.prototype.inspect=S.prototype.toSource=function(){return\"\"+this},S.
prototype[br]=function(){return this},t(O,e),O.of=function(){return O(arguments)
},O.prototype.toSeq=function(){return this},O.prototype.toString=function(){retu
rn this.__toString(\"Seq {\",\"}\")},O.prototype.cacheResult=function(){return!t
his._cache&&this.__iterateUncached&&(this._cache=this.entrySeq().toArray(),this.
size=this._cache.length),this},O.prototype.__iterate=function(t,e){return N(this
,t,e,!0)},O.prototype.__iterator=function(t,e){return P(this,t,e,!0)},t(x,O),x.p
rototype.toKeyedSeq=function(){return this},t(k,O),k.of=function(){return k(argu
```

```
ments)},k.prototype.toIndexedSeq=function(){return this},k.prototype.toString=fu
nction(){return this.__toString(\"Seq [\",\"]\")},k.prototype.__iterate=function
(t,e){return N(this,t,e,!1)},k.prototype.__iterator=function(t,e){return P(this,
t,e,!1)},t(A,O),A.of=function(){return A(arguments)},A.prototype.toSetSeq=functi
on(){return this},O.isSeq=L,O.Keyed=x,O.Set=A,O.Indexed=k;var qr=\"@@__IMMUTABLE
_SEQ__@@\";O.prototype[qr]=!0,t(j,k),j.prototype.get=function(t,e){return this.h
as(t)?this._array[l(this,t)]:e},j.prototype.__iterate=function(t,e){for(var r=th
is._array,n=r.length-1,i=0;n>=i;i++)if(t(r[e?n-i:i],i,this)===!1)return i+1;retu
rn i},j.prototype.__iterator=function(t,e){var r=this._array,n=r.length-1,i=0;re
turn new S(function(){return i>n?I():z(t,i,r[e?n-i++:i++])})},t(K,x),K.prototype
.get=function(t,e){return void 0===e||this.has(t)?this._object[t]:e},K.prototype
.has=function(t){return this._object.hasOwnProperty(t)},K.prototype.__iterate=fu
nction(t,e){for(var r=this._object,n=this._keys,i=n.length-1,o=0;i>=o;o++){var u
=n[e?i-o:o];if(t(r[u],u,this)===!1)return o+1}return o},K.prototype.__iterator=f
unction(t,e){var r=this._object,n=this._keys,i=n.length-1,o=0;return new S(funct
ion(){var u=n[e?i-o:o];return o++>i?I():z(t,u,r[u])})},K.prototype[cr]=!0,t(R,k)
,R.prototype.__iterateUncached=function(t,e){if(e)return this.cacheResult().__it
erate(t,e);
var r=this._iterable,n=D(r),i=0;if(q(n))for(var o;!(o=n.next()).done&&t(o.value,
i++,this)!==!1;);return i},R.prototype.__iteratorUncached=function(t,e){if(e)ret
urn this.cacheResult().__iterator(t,e);var r=this._iterable,n=D(r);if(!q(n))retu
rn new S(I);var i=0;return new S(function(){var e=n.next();return e.done?e:z(t,i
++,e.value)})},t(U,k),U.prototype.__iterateUncached=function(t,e){if(e)return th
is.cacheResult().__iterate(t,e);for(var r=this._iterator,n=this._iteratorCache,i
=0;n.length>i;)if(t(n[i],i++,this)===!1)return i;for(var o;!(o=r.next()).done;){
var u=o.value;if(n[i]=u,t(u,i++,this)===!1)break}return i},U.prototype.__iterato
rUncached=function(t,e){if(e)return this.cacheResult().__iterator(t,e);var r=thi
s._iterator,n=this._iteratorCache,i=0;return new S(function(){if(i>=n.length){va
r e=r.next();if(e.done)return e;n[i]=e.value}return z(t,i,n[i++])})};var Dr;t(G,
k),G.prototype.toString=function(){return 0===this.size?\"Repeat []\":\"Repeat [
 \"+this._value+\" \"+this.size+\" times ]\"},G.prototype.get=function(t,e){retu
rn this.has(t)?this._value:e},G.prototype.includes=function(t){return X(this._va
lue,t)},G.prototype.slice=function(t,e){var r=this.size;return d(t,e,r)?this:new
 G(this._value,g(e,r)-m(t,r))},G.prototype.reverse=function(){return this},G.pro
totype.indexOf=function(t){return X(this._value,t)?0:-1},G.prototype.lastIndexOf
=function(t){return X(this._value,t)?this.size:-1},G.prototype.__iterate=functio
n(t,e){for(var r=0;this.size>r;r++)if(t(this._value,r,this)===!1)return r+1;retu
rn r},G.prototype.__iterator=function(t,e){var r=this,n=0;return new S(function(
){return r.size>n?z(t,n++,r._value):I()})},G.prototype.equals=function(t){return
 t instanceof G?X(this._value,t._value):F(t)};var Mr;t($,k),$.prototype.toString
=function(){return 0===this.size?\"Range []\":\"Range [ \"+this._start+\"...\"+t
his._end+(1!==this._step?\" by \"+this._step:\"\")+\" ]\"},$.prototype.get=funct
ion(t,e){return this.has(t)?this._start+l(this,t)*this._step:e},$.prototype.incl
udes=function(t){var e=(t-this._start)/this._step;return e>=0&&this.size>e&&e===
Math.floor(e);
},$.prototype.slice=function(t,e){return d(t,e,this.size)?this:(t=m(t,this.size)
,e=g(e,this.size),t>=e?new $(0,0):new $(this.get(t,this._end),this.get(e,this._e
nd),this._step))},$.prototype.indexOf=function(t){var e=t-this._start;if(e%this.
_step===0){var r=e/this._step;if(r>=0&&this.size>r)return r}return-1},$.prototyp
e.lastIndexOf=function(t){return this.indexOf(t)},$.prototype.__iterate=function
(t,e){for(var r=this.size-1,n=this._step,i=e?this._start+r*n:this._start,o=0;r>=
o;o++){if(t(i,o,this)===!1)return o+1;i+=e?-n:n}return o},$.prototype.__iterator
=function(t,e){var r=this.size-1,n=this._step,i=e?this._start+r*n:this._start,o=
```

```
0;return new S(function(){var u=i;return i+=e?-n:n,o>r?I():z(t,o++,u)})},$.proto
type.equals=function(t){return t instanceof $?this._start===t._start&&this._end=
==t._end&&this._step===t._step:F(this,t)};var Er;t(tt,e),t(et,tt),t(rt,tt),t(nt,
tt),tt.Keyed=et,tt.Indexed=rt,tt.Set=nt;var Or,xr=\"function\"==typeof Math.imul
&&-2===Math.imul(4294967295,2)?Math.imul:function(t,e){t=0|t,e=0|e;var r=65535&t
,n=65535&e;return r*n+((t>>>16)*n+r*(e>>>16)<<16>>>0)|0},kr=Object.isExtensible,
Ar=function(){try{return Object.defineProperty({},\"@\",{}),!0}catch(t){return!1
}}(),jr=\"function\"==typeof WeakMap;jr&&(Or=new WeakMap);var Kr=0,Rr=\"__immuta
blehash__\";\"function\"==typeof Symbol&&(Rr=Symbol(Rr));var Ur=16,Lr=255,Tr=0,W
r={};t(ct,et),ct.of=function(){var t=sr.call(arguments,0);return zt().withMutati
ons(function(e){for(var r=0;t.length>r;r+=2){if(r+1>=t.length)throw Error(\"Miss
ing value for key: \"+t[r]);e.set(t[r],t[r+1])}})},ct.prototype.toString=functio
n(){return this.__toString(\"Map {\",\"}\")},ct.prototype.get=function(t,e){retu
rn this._root?this._root.get(0,void 0,t,e):e},ct.prototype.set=function(t,e){ret
urn It(this,t,e)},ct.prototype.setIn=function(t,e){return this.updateIn(t,yr,fun
ction(){return e})},ct.prototype.remove=function(t){return It(this,t,yr)},ct.pro
totype.deleteIn=function(t){return this.updateIn(t,function(){return yr})},ct.pr
ototype.update=function(t,e,r){return 1===arguments.length?t(this):this.updateIn
([t],e,r);
},ct.prototype.updateIn=function(t,e,r){r||(r=e,e=void 0);var n=Kt(this,ke(t),e,
r);return n===yr?void 0:n},ct.prototype.clear=function(){return 0===this.size?th
is:this.__ownerID?(this.size=0,this._root=null,this.__hash=void 0,this.__altered
=!0,this):zt()},ct.prototype.merge=function(){return xt(this,void 0,arguments)},
ct.prototype.mergeWith=function(t){var e=sr.call(arguments,1);return xt(this,t,e
)},ct.prototype.mergeIn=function(t){var e=sr.call(arguments,1);return this.updat
eIn(t,zt(),function(t){return\"function\"==typeof t.merge?t.merge.apply(t,e):e[e
.length-1]})},ct.prototype.mergeDeep=function(){return xt(this,kt,arguments)},ct
.prototype.mergeDeepWith=function(t){var e=sr.call(arguments,1);return xt(this,A
t(t),e)},ct.prototype.mergeDeepIn=function(t){var e=sr.call(arguments,1);return
this.updateIn(t,zt(),function(t){return\"function\"==typeof t.mergeDeep?t.mergeD
eep.apply(t,e):e[e.length-1]})},ct.prototype.sort=function(t){return Zt(we(this,
t))},ct.prototype.sortBy=function(t,e){return Zt(we(this,e,t))},ct.prototype.wit
hMutations=function(t){var e=this.asMutable();return t(e),e.wasAltered()?e.__ens
ureOwner(this.__ownerID):this},ct.prototype.asMutable=function(){return this.__o
wnerID?this:this.__ensureOwner(new _)},ct.prototype.asImmutable=function(){retur
n this.__ensureOwner()},ct.prototype.wasAltered=function(){return this.__altered
},ct.prototype.__iterator=function(t,e){return new mt(this,t,e)},ct.prototype.__
iterate=function(t,e){var r=this,n=0;return this._root&&this._root.iterate(funct
ion(e){return n++,t(e[1],e[0],r)},e),n},ct.prototype.__ensureOwner=function(t){r
eturn t===this.__ownerID?this:t?St(this.size,this._root,t,this.__hash):(this.__o
wnerID=t,this.__altered=!1,this)},ct.isMap=_t;var Br=\"@@__IMMUTABLE_MAP__@@\",C
r=ct.prototype;Cr[Br]=!0,Cr[_r]=Cr.remove,Cr.removeIn=Cr.deleteIn,pt.prototype.g
et=function(t,e,r,n){for(var i=this.entries,o=0,u=i.length;u>o;o++)if(X(r,i[o][0
]))return i[o][1];return n},pt.prototype.update=function(t,e,r,n,i,o,u){for(var
s=i===yr,a=this.entries,h=0,f=a.length;f>h&&!X(n,a[h][0]);h++);
var _=f>h;if(_?a[h][1]===i:s)return this;if(c(u),(s||!_)&&c(o),!s||1!==a.length)
{if(!_&&!s&&a.length>=Nr)return Mt(t,a,n,i);var v=t&&t===this.ownerID,l=v?a:p(a)
;return _?s?h===f-1?l.pop():l[h]=l.pop():l[h]=[n,i]:l.push([n,i]),v?(this.entrie
s=l,this):new pt(t,l)}},vt.prototype.get=function(t,e,r,n){void 0===e&&(e=ot(r))
;var i=1<<((0===t?e:e>>>t)&lr),o=this.bitmap;return 0===(o&i)?n:this.nodes[Rt(o&
i-1)].get(t+pr,e,r,n)},vt.prototype.update=function(t,e,r,n,i,o,u){void 0===r&&(
r=ot(n));var s=(0===e?r:r>>>e)&lr,a=1<<s,h=this.bitmap,f=0!==(h&a);if(!f&&i===yr
)return this;var c=Rt(h&a-1),_=this.nodes,p=f?_[c]:void 0,v=bt(p,t,e+pr,r,n,i,o,
```

```
u);if(v===p)return this;if(!f&&v&&_.length>=Pr)return Ot(t,_,h,s,v);if(f&&!v&&2=
==_.length&&qt(_[1^c]))return _[1^c];if(f&&v&&1===_.length&&qt(v))return v;var l
=t&&t===this.ownerID,y=f?v?h:h^a:h|a,d=f?v?Ut(_,c,v,l):Tt(_,c,l):Lt(_,c,v,l);ret
urn l?(this.bitmap=y,this.nodes=d,this):new vt(t,y,d)},lt.prototype.get=function
(t,e,r,n){void 0===e&&(e=ot(r));var i=(0===t?e:e>>>t)&lr,o=this.nodes[i];return
o?o.get(t+pr,e,r,n):n},lt.prototype.update=function(t,e,r,n,i,o,u){void 0===r&&(
r=ot(n));var s=(0===e?r:r>>>e)&lr,a=i===yr,h=this.nodes,f=h[s];if(a&&!f)return t
his;var c=bt(f,t,e+pr,r,n,i,o,u);if(c===f)return this;var _=this.count;if(f){if(
!c&&(_--,Hr>_))return Et(t,h,_,s)}else _++;var p=t&&t===this.ownerID,v=Ut(h,s,c,
p);return p?(this.count=_,this.nodes=v,this):new lt(t,_,v)},yt.prototype.get=fun
ction(t,e,r,n){for(var i=this.entries,o=0,u=i.length;u>o;o++)if(X(r,i[o][0]))ret
urn i[o][1];return n},yt.prototype.update=function(t,e,r,n,i,o,u){void 0===r&&(r
=ot(n));var s=i===yr;if(r!==this.keyHash)return s?this:(c(u),c(o),Dt(this,t,e,r,
[n,i]));for(var a=this.entries,h=0,f=a.length;f>h&&!X(n,a[h][0]);h++);var _=f>h;
if(_?a[h][1]===i:s)return this;if(c(u),(s||!_)&&c(o),s&&2===f)return new dt(t,th
is.keyHash,a[1^h]);var v=t&&t===this.ownerID,l=v?a:p(a);return _?s?h===f-1?l.pop
():l[h]=l.pop():l[h]=[n,i]:l.push([n,i]),v?(this.entries=l,this):new yt(t,this.k
eyHash,l)},dt.prototype.get=function(t,e,r,n){return X(r,this.entry[0])?this.ent
ry[1]:n;
},dt.prototype.update=function(t,e,r,n,i,o,u){var s=i===yr,a=X(n,this.entry[0]);
return(a?i===this.entry[1]:s)?this:(c(u),s?void c(o):a?t&&t===this.ownerID?(this
.entry[1]=i,this):new dt(t,this.keyHash,[n,i]):(c(o),Dt(this,t,e,ot(n),[n,i])))}
,pt.prototype.iterate=yt.prototype.iterate=function(t,e){for(var r=this.entries,
n=0,i=r.length-1;i>=n;n++)if(t(r[e?i-n:n])===!1)return!1},vt.prototype.iterate=l
t.prototype.iterate=function(t,e){for(var r=this.nodes,n=0,i=r.length-1;i>=n;n++
){var o=r[e?i-n:n];if(o&&o.iterate(t,e)===!1)return!1}},dt.prototype.iterate=fun
ction(t,e){return t(this.entry)},t(mt,S),mt.prototype.next=function(){for(var t=
this._type,e=this._stack;e;){var r,n=e.node,i=e.index++;if(n.entry){if(0===i)ret
urn gt(t,n.entry)}else if(n.entries){if(r=n.entries.length-1,r>=i)return gt(t,n.
entries[this._reverse?r-i:i])}else if(r=n.nodes.length-1,r>=i){var o=n.nodes[thi
s._reverse?r-i:i];if(o){if(o.entry)return gt(t,o.entry);e=this._stack=wt(o,e)}co
ntinue}e=this._stack=this._stack.__prev}return I()};var Jr,Nr=vr/4,Pr=vr/2,Hr=vr
/4;t(Wt,rt),Wt.of=function(){return this(arguments)},Wt.prototype.toString=funct
ion(){return this.__toString(\"List [\",\"]\")},Wt.prototype.get=function(t,e){i
f(t=l(this,t),t>=0&&this.size>t){t+=this._origin;var r=Qt(this,t);return r&&r.ar
ray[t&lr]}return e},Wt.prototype.set=function(t,e){return Ht(this,t,e)},Wt.proto
type.remove=function(t){return this.has(t)?0===t?this.shift():t===this.size-1?th
is.pop():this.splice(t,1):this},Wt.prototype.insert=function(t,e){return this.sp
lice(t,0,e)},Wt.prototype.clear=function(){return 0===this.size?this:this.__owne
rID?(this.size=this._origin=this._capacity=0,this._level=pr,this._root=this._tai
l=null,this.__hash=void 0,this.__altered=!0,this):Pt()},Wt.prototype.push=functi
on(){var t=arguments,e=this.size;return this.withMutations(function(r){Xt(r,0,e+
t.length);for(var n=0;t.length>n;n++)r.set(e+n,t[n])})},Wt.prototype.pop=functio
n(){return Xt(this,0,-1)},Wt.prototype.unshift=function(){var t=arguments;return
 this.withMutations(function(e){Xt(e,-t.length);for(var r=0;t.length>r;r++)e.set
(r,t[r]);
})},Wt.prototype.shift=function(){return Xt(this,1)},Wt.prototype.merge=function
(){return Ft(this,void 0,arguments)},Wt.prototype.mergeWith=function(t){var e=sr
.call(arguments,1);return Ft(this,t,e)},Wt.prototype.mergeDeep=function(){return
 Ft(this,kt,arguments)},Wt.prototype.mergeDeepWith=function(t){var e=sr.call(arg
uments,1);return Ft(this,At(t),e)},Wt.prototype.setSize=function(t){return Xt(th
is,0,t)},Wt.prototype.slice=function(t,e){var r=this.size;return d(t,e,r)?this:X
t(this,m(t,r),g(e,r))},Wt.prototype.__iterator=function(t,e){var r=0,n=Jt(this,e
```

```
);return new S(function(){var e=n();return e===Xr?I():z(t,r++,e)})},Wt.prototype
.__iterate=function(t,e){for(var r,n=0,i=Jt(this,e);(r=i())!==Xr&&t(r,n++,this)!
==!1;);return n},Wt.prototype.__ensureOwner=function(t){return t===this.__ownerI
D?this:t?Nt(this._origin,this._capacity,this._level,this._root,this._tail,t,this
.__hash):(this.__ownerID=t,this)},Wt.isList=Bt;var Vr=\"@@__IMMUTABLE_LIST__@@\"
,Yr=Wt.prototype;Yr[Vr]=!0,Yr[_r]=Yr.remove,Yr.setIn=Cr.setIn,Yr.deleteIn=Yr.rem
oveIn=Cr.removeIn,Yr.update=Cr.update,Yr.updateIn=Cr.updateIn,Yr.mergeIn=Cr.merg
eIn,Yr.mergeDeepIn=Cr.mergeDeepIn,Yr.withMutations=Cr.withMutations,Yr.asMutable
=Cr.asMutable,Yr.asImmutable=Cr.asImmutable,Yr.wasAltered=Cr.wasAltered,Ct.proto
type.removeBefore=function(t,e,r){if(r===e?1<<e:0===this.array.length)return thi
s;var n=r>>>e&lr;if(n>=this.array.length)return new Ct([],t);var i,o=0===n;if(e>
0){var u=this.array[n];if(i=u&&u.removeBefore(t,e-pr,r),i===u&&o)return this}if(
o&&!i)return this;var s=Yt(this,t);if(!o)for(var a=0;n>a;a++)s.array[a]=void 0;r
eturn i&&(s.array[n]=i),s},Ct.prototype.removeAfter=function(t,e,r){if(r===(e?1<
<e:0)||0===this.array.length)return this;var n=r-1>>>e&lr;if(n>=this.array.lengt
h)return this;var i;if(e>0){var o=this.array[n];if(i=o&&o.removeAfter(t,e-pr,r),
i===o&&n===this.array.length-1)return this}var u=Yt(this,t);return u.array.splic
e(n+1),i&&(u.array[n]=i),u};var Qr,Xr={};t(Zt,ct),Zt.of=function(){return this(a
rguments)},Zt.prototype.toString=function(){return this.__toString(\"OrderedMap
{\",\"}\");
},Zt.prototype.get=function(t,e){var r=this._map.get(t);return void 0!==r?this._
list.get(r)[1]:e},Zt.prototype.clear=function(){return 0===this.size?this:this._
_ownerID?(this.size=0,this._map.clear(),this._list.clear(),this):ee()},Zt.protot
ype.set=function(t,e){return re(this,t,e)},Zt.prototype.remove=function(t){retur
n re(this,t,yr)},Zt.prototype.wasAltered=function(){return this._map.wasAltered(
)||this._list.wasAltered()},Zt.prototype.__iterate=function(t,e){var r=this;retu
rn this._list.__iterate(function(e){return e&&t(e[1],e[0],r)},e)},Zt.prototype._
_iterator=function(t,e){return this._list.fromEntrySeq().__iterator(t,e)},Zt.pro
totype.__ensureOwner=function(t){if(t===this.__ownerID)return this;var e=this._m
ap.__ensureOwner(t),r=this._list.__ensureOwner(t);return t?te(e,r,t,this.__hash)
:(this.__ownerID=t,this._map=e,this._list=r,this)},Zt.isOrderedMap=$t,Zt.prototy
pe[cr]=!0,Zt.prototype[_r]=Zt.prototype.remove;var Fr;t(ne,x),ne.prototype.get=f
unction(t,e){return this._iter.get(t,e)},ne.prototype.has=function(t){return thi
s._iter.has(t)},ne.prototype.valueSeq=function(){return this._iter.valueSeq()},n
e.prototype.reverse=function(){var t=this,e=he(this,!0);return this._useKeys||(e
.valueSeq=function(){return t._iter.toSeq().reverse()}),e},ne.prototype.map=func
tion(t,e){var r=this,n=ae(this,t,e);return this._useKeys||(n.valueSeq=function()
{return r._iter.toSeq().map(t,e)}),n},ne.prototype.__iterate=function(t,e){var r
,n=this;return this._iter.__iterate(this._useKeys?function(e,r){return t(e,r,n)}
:(r=e?De(this):0,function(i){return t(i,e?--r:r++,n)}),e)},ne.prototype.__iterat
or=function(t,e){if(this._useKeys)return this._iter.__iterator(t,e);var r=this._
iter.__iterator(wr,e),n=e?De(this):0;return new S(function(){var i=r.next();retu
rn i.done?i:z(t,e?--n:n++,i.value,i)})},ne.prototype[cr]=!0,t(ie,k),ie.prototype
.includes=function(t){return this._iter.includes(t)},ie.prototype.__iterate=func
tion(t,e){var r=this,n=0;return this._iter.__iterate(function(e){return t(e,n++,
r)},e)},ie.prototype.__iterator=function(t,e){var r=this._iter.__iterator(wr,e),
n=0;
return new S(function(){var e=r.next();return e.done?e:z(t,n++,e.value,e)})},t(o
e,A),oe.prototype.has=function(t){return this._iter.includes(t)},oe.prototype.__
iterate=function(t,e){var r=this;return this._iter.__iterate(function(e){return
t(e,e,r)},e)},oe.prototype.__iterator=function(t,e){var r=this._iter.__iterator(
wr,e);return new S(function(){var e=r.next();return e.done?e:z(t,e.value,e.value
,e)})},t(ue,x),ue.prototype.entrySeq=function(){return this._iter.toSeq()},ue.pr
```

```
ototype.__iterate=function(t,e){var r=this;return this._iter.__iterate(function(
e){if(e){qe(e);var n=o(e);return t(n?e.get(1):e[1],n?e.get(0):e[0],r)}},e)},ue.p
rototype.__iterator=function(t,e){var r=this._iter.__iterator(wr,e);return new S
(function(){for(;;){var e=r.next();if(e.done)return e;var n=e.value;if(n){qe(n);
var i=o(n);return z(t,i?n.get(0):n[0],i?n.get(1):n[1],e)}}})},ie.prototype.cache
Result=ne.prototype.cacheResult=oe.prototype.cacheResult=ue.prototype.cacheResul
t=Oe,t(Ae,et),Ae.prototype.toString=function(){return this.__toString(Ke(this)+\
" {\",\"}\")},Ae.prototype.has=function(t){return this._defaultValues.hasOwnProp
erty(t)},Ae.prototype.get=function(t,e){if(!this.has(t))return e;var r=this._def
aultValues[t];return this._map?this._map.get(t,r):r},Ae.prototype.clear=function
(){if(this.__ownerID)return this._map&&this._map.clear(),this;var t=this.constru
ctor;return t._empty||(t._empty=je(this,zt()))},Ae.prototype.set=function(t,e){i
f(!this.has(t))throw Error('Cannot set unknown key \"'+t+'\" on '+Ke(this));if(t
his._map&&!this._map.has(t)){var r=this._defaultValues[t];if(e===r)return this}v
ar n=this._map&&this._map.set(t,e);return this.__ownerID||n===this._map?this:je(
this,n)},Ae.prototype.remove=function(t){if(!this.has(t))return this;var e=this.
_map&&this._map.remove(t);return this.__ownerID||e===this._map?this:je(this,e)},
Ae.prototype.wasAltered=function(){return this._map.wasAltered()},Ae.prototype._
_iterator=function(t,e){var n=this;return r(this._defaultValues).map(function(t,
e){return n.get(e)}).__iterator(t,e)},Ae.prototype.__iterate=function(t,e){
var n=this;return r(this._defaultValues).map(function(t,e){return n.get(e)}).__i
terate(t,e)},Ae.prototype.__ensureOwner=function(t){if(t===this.__ownerID)return
 this;var e=this._map&&this._map.__ensureOwner(t);return t?je(this,e,t):(this.__
ownerID=t,this._map=e,this)};var Gr=Ae.prototype;Gr[_r]=Gr.remove,Gr.deleteIn=Gr
.removeIn=Cr.removeIn,Gr.merge=Cr.merge,Gr.mergeWith=Cr.mergeWith,Gr.mergeIn=Cr.
mergeIn,Gr.mergeDeep=Cr.mergeDeep,Gr.mergeDeepWith=Cr.mergeDeepWith,Gr.mergeDeep
In=Cr.mergeDeepIn,Gr.setIn=Cr.setIn,Gr.update=Cr.update,Gr.updateIn=Cr.updateIn,
Gr.withMutations=Cr.withMutations,Gr.asMutable=Cr.asMutable,Gr.asImmutable=Cr.as
Immutable,t(Le,nt),Le.of=function(){return this(arguments)},Le.fromKeys=function
(t){return this(r(t).keySeq())},Le.prototype.toString=function(){return this.__t
oString(\"Set {\",\"}\")},Le.prototype.has=function(t){return this._map.has(t)},
Le.prototype.add=function(t){return We(this,this._map.set(t,!0))},Le.prototype.r
emove=function(t){return We(this,this._map.remove(t))},Le.prototype.clear=functi
on(){return We(this,this._map.clear())},Le.prototype.union=function(){var t=sr.c
all(arguments,0);return t=t.filter(function(t){return 0!==t.size}),0===t.length?
this:0!==this.size||this.__ownerID||1!==t.length?this.withMutations(function(e){
for(var r=0;t.length>r;r++)i(t[r]).forEach(function(t){return e.add(t)})}):this.
constructor(t[0])},Le.prototype.intersect=function(){var t=sr.call(arguments,0);
if(0===t.length)return this;t=t.map(function(t){return i(t)});var e=this;return
this.withMutations(function(r){e.forEach(function(e){t.every(function(t){return
t.includes(e)})||r.remove(e)})})},Le.prototype.subtract=function(){var t=sr.call
(arguments,0);if(0===t.length)return this;t=t.map(function(t){return i(t)});var
e=this;return this.withMutations(function(r){e.forEach(function(e){t.some(functi
on(t){return t.includes(e)})&&r.remove(e)})})},Le.prototype.merge=function(){ret
urn this.union.apply(this,arguments)},Le.prototype.mergeWith=function(t){var e=s
r.call(arguments,1);return this.union.apply(this,e)},
Le.prototype.sort=function(t){return Je(we(this,t))},Le.prototype.sortBy=functio
n(t,e){return Je(we(this,e,t))},Le.prototype.wasAltered=function(){return this._
map.wasAltered()},Le.prototype.__iterate=function(t,e){var r=this;return this._m
ap.__iterate(function(e,n){return t(n,n,r)},e)},Le.prototype.__iterator=function
(t,e){return this._map.map(function(t,e){return e}).__iterator(t,e)},Le.prototyp
e.__ensureOwner=function(t){if(t===this.__ownerID)return this;var e=this._map.__
ensureOwner(t);return t?this.__make(e,t):(this.__ownerID=t,this._map=e,this)},Le
```

```
.isSet=Te;var Zr=\"@@__IMMUTABLE_SET__@@\",$r=Le.prototype;$r[Zr]=!0,$r[_r]=$r.r
emove,$r.mergeDeep=$r.merge,$r.mergeDeepWith=$r.mergeWith,$r.withMutations=Cr.wi
thMutations,$r.asMutable=Cr.asMutable,$r.asImmutable=Cr.asImmutable,$r.__empty=C
e,$r.__make=Be;var tn;t(Je,Le),Je.of=function(){return this(arguments)},Je.fromK
eys=function(t){return this(r(t).keySeq())},Je.prototype.toString=function(){ret
urn this.__toString(\"OrderedSet {\",\"}\")},Je.isOrderedSet=Ne;var en=Je.protot
ype;en[cr]=!0,en.__empty=He,en.__make=Pe;var rn;t(Ve,rt),Ve.of=function(){return
 this(arguments)},Ve.prototype.toString=function(){return this.__toString(\"Stac
k [\",\"]\")},Ve.prototype.get=function(t,e){var r=this._head;for(t=l(this,t);r&
&t--;)r=r.next;return r?r.value:e},Ve.prototype.peek=function(){return this._hea
d&&this._head.value},Ve.prototype.push=function(){if(0===arguments.length)return
 this;for(var t=this.size+arguments.length,e=this._head,r=arguments.length-1;r>=
0;r--)e={value:arguments[r],next:e};return this.__ownerID?(this.size=t,this._hea
d=e,this.__hash=void 0,this.__altered=!0,this):Qe(t,e)},Ve.prototype.pushAll=fun
ction(t){if(t=n(t),0===t.size)return this;ft(t.size);var e=this.size,r=this._hea
d;return t.reverse().forEach(function(t){e++,r={value:t,next:r}}),this.__ownerID
?(this.size=e,this._head=r,this.__hash=void 0,this.__altered=!0,this):Qe(e,r)},V
e.prototype.pop=function(){return this.slice(1)},Ve.prototype.unshift=function()
{return this.push.apply(this,arguments)},Ve.prototype.unshiftAll=function(t){
return this.pushAll(t)},Ve.prototype.shift=function(){return this.pop.apply(this
,arguments)},Ve.prototype.clear=function(){return 0===this.size?this:this.__owne
rID?(this.size=0,this._head=void 0,this.__hash=void 0,this.__altered=!0,this):Xe
()},Ve.prototype.slice=function(t,e){if(d(t,e,this.size))return this;var r=m(t,t
his.size),n=g(e,this.size);if(n!==this.size)return rt.prototype.slice.call(this,
t,e);for(var i=this.size-r,o=this._head;r--;)o=o.next;return this.__ownerID?(thi
s.size=i,this._head=o,this.__hash=void 0,this.__altered=!0,this):Qe(i,o)},Ve.pro
totype.__ensureOwner=function(t){return t===this.__ownerID?this:t?Qe(this.size,t
his._head,t,this.__hash):(this.__ownerID=t,this.__altered=!1,this)},Ve.prototype
.__iterate=function(t,e){if(e)return this.reverse().__iterate(t);for(var r=0,n=t
his._head;n&&t(n.value,r++,this)!==!1;)n=n.next;return r},Ve.prototype.__iterato
r=function(t,e){if(e)return this.reverse().__iterator(t);var r=0,n=this._head;re
turn new S(function(){if(n){var e=n.value;return n=n.next,z(t,r++,e)}return I()}
)},Ve.isStack=Ye;var nn=\"@@__IMMUTABLE_STACK__@@\",on=Ve.prototype;on[nn]=!0,on
.withMutations=Cr.withMutations,on.asMutable=Cr.asMutable,on.asImmutable=Cr.asIm
mutable,on.wasAltered=Cr.wasAltered;var un;e.Iterator=S,Fe(e,{toArray:function()
{ft(this.size);var t=Array(this.size||0);return this.valueSeq().__iterate(functi
on(e,r){t[r]=e}),t},toIndexedSeq:function(){return new ie(this)},toJS:function()
{return this.toSeq().map(function(t){return t&&\"function\"==typeof t.toJS?t.toJ
S():t}).__toJS()},toJSON:function(){return this.toSeq().map(function(t){return t
&&\"function\"==typeof t.toJSON?t.toJSON():t}).__toJS()},toKeyedSeq:function(){r
eturn new ne(this,!0)},toMap:function(){return ct(this.toKeyedSeq())},toObject:f
unction(){ft(this.size);var t={};return this.__iterate(function(e,r){t[r]=e}),t}
,toOrderedMap:function(){return Zt(this.toKeyedSeq())},toOrderedSet:function(){r
eturn Je(u(this)?this.valueSeq():this)},toSet:function(){return Le(u(this)?this.
valueSeq():this)},toSetSeq:function(){return new oe(this);
},toSeq:function(){return s(this)?this.toIndexedSeq():u(this)?this.toKeyedSeq():
this.toSetSeq()},toStack:function(){return Ve(u(this)?this.valueSeq():this)},toL
ist:function(){return Wt(u(this)?this.valueSeq():this)},toString:function(){retu
rn\"[Iterable]\"},__toString:function(t,e){return 0===this.size?t+e:t+\" \"+this
.toSeq().map(this.__toStringMapper).join(\", \")+\" \"+e},concat:function(){var
 t=sr.call(arguments,0);return be(this,ye(this,t))},includes:function(t){return t
his.some(function(e){return X(e,t)})},entries:function(){return this.__iterator(
Sr)},every:function(t,e){ft(this.size);var r=!0;return this.__iterate(function(n
```

```
,i,o){return t.call(e,n,i,o)?void 0:(r=!1,!1)}),r},filter:function(t,e){return b
e(this,fe(this,t,e,!0))},find:function(t,e,r){var n=this.findEntry(t,e);return n
?n[1]:r},findEntry:function(t,e){var r;return this.__iterate(function(n,i,o){ret
urn t.call(e,n,i,o)?(r=[i,n],!1):void 0}),r},findLastEntry:function(t,e){return
this.toSeq().reverse().findEntry(t,e)},forEach:function(t,e){return ft(this.size
),this.__iterate(e?t.bind(e):t)},join:function(t){ft(this.size),t=void 0!==t?\"\
"+t:\",\";var e=\"\",r=!0;return this.__iterate(function(n){r?r=!1:e+=t,e+=null!
==n&&void 0!==n?\"\"+n:\"\"}),e},keys:function(){return this.__iterator(gr)},map
:function(t,e){return be(this,ae(this,t,e))},reduce:function(t,e,r){ft(this.size
);var n,i;return arguments.length<2?i=!0:n=e,this.__iterate(function(e,o,u){i?(i
=!1,n=e):n=t.call(r,n,e,o,u)}),n},reduceRight:function(t,e,r){var n=this.toKeyed
Seq().reverse();return n.reduce.apply(n,arguments)},reverse:function(){return be
(this,he(this,!0))},slice:function(t,e){return be(this,pe(this,t,e,!0))},some:fu
nction(t,e){return!this.every($e(t),e)},sort:function(t){return be(this,we(this,
t))},values:function(){return this.__iterator(wr)},butLast:function(){return thi
s.slice(0,-1)},isEmpty:function(){return void 0!==this.size?0===this.size:!this.
some(function(){return!0})},count:function(t,e){return v(t?this.toSeq().filter(t
,e):this)},countBy:function(t,e){return ce(this,t,e)},equals:function(t){
return F(this,t)},entrySeq:function(){var t=this;if(t._cache)return new j(t._cac
he);var e=t.toSeq().map(Ze).toIndexedSeq();return e.fromEntrySeq=function(){retu
rn t.toSeq()},e},filterNot:function(t,e){return this.filter($e(t),e)},findLast:f
unction(t,e,r){return this.toKeyedSeq().reverse().find(t,e,r)},first:function(){
return this.find(y)},flatMap:function(t,e){return be(this,me(this,t,e))},flatten
:function(t){return be(this,de(this,t,!0))},fromEntrySeq:function(){return new u
e(this)},get:function(t,e){return this.find(function(e,r){return X(r,t)},void 0,
e)},getIn:function(t,e){for(var r,n=this,i=ke(t);!(r=i.next()).done;){var o=r.va
lue;if(n=n&&n.get?n.get(o,yr):yr,n===yr)return e}return n},groupBy:function(t,e)
{return _e(this,t,e)},has:function(t){return this.get(t,yr)!==yr},hasIn:function
(t){return this.getIn(t,yr)!==yr},isSubset:function(t){return t=\"function\"==ty
peof t.includes?t:e(t),this.every(function(e){return t.includes(e)})},isSuperset
:function(t){return t=\"function\"==typeof t.isSubset?t:e(t),t.isSubset(this)},k
eySeq:function(){return this.toSeq().map(Ge).toIndexedSeq()},last:function(){ret
urn this.toSeq().reverse().first()},max:function(t){return Se(this,t)},maxBy:fun
ction(t,e){return Se(this,e,t)},min:function(t){return Se(this,t?tr(t):nr)},minB
y:function(t,e){return Se(this,e?tr(e):nr,t)},rest:function(){return this.slice(
1)},skip:function(t){return this.slice(Math.max(0,t))},skipLast:function(t){retu
rn be(this,this.toSeq().reverse().skip(t).reverse())},skipWhile:function(t,e){re
turn be(this,le(this,t,e,!0))},skipUntil:function(t,e){return this.skipWhile($e(
t),e)},sortBy:function(t,e){return be(this,we(this,e,t))},take:function(t){retur
n this.slice(0,Math.max(0,t))},takeLast:function(t){return be(this,this.toSeq().
reverse().take(t).reverse())},takeWhile:function(t,e){return be(this,ve(this,t,e
))},takeUntil:function(t,e){return this.takeWhile($e(t),e)},valueSeq:function(){
return this.toIndexedSeq()},hashCode:function(){return this.__hash||(this.__hash
=ir(this))}});var sn=e.prototype;sn[ar]=!0,sn[br]=sn.values,
sn.__toJS=sn.toArray,sn.__toStringMapper=er,sn.inspect=sn.toSource=function(){re
turn\"\"+this},sn.chain=sn.flatMap,sn.contains=sn.includes,function(){try{Object
.defineProperty(sn,\"length\",{get:function(){if(!e.noLengthWarning){var t;try{t
hrow Error()}catch(r){t=r.stack}if(-1===t.indexOf(\"_wrapObject\"))return consol
e&&console.warn&&console.warn(\"iterable.length has been deprecated, use iterabl
e.size or iterable.count(). This warning will become a silent error in a future
version. \"+t),this.size}}})}catch(t){}}(),Fe(r,{flip:function(){return be(this,
se(this))},findKey:function(t,e){var r=this.findEntry(t,e);return r&&r[0]},findL
astKey:function(t,e){return this.toSeq().reverse().findKey(t,e)},keyOf:function(
```

```
t){return this.findKey(function(e){return X(e,t)})},lastKeyOf:function(t){return
 this.findLastKey(function(e){return X(e,t)})},mapEntries:function(t,e){var r=th
is,n=0;return be(this,this.toSeq().map(function(i,o){return t.call(e,[o,i],n++,r
)}).fromEntrySeq())},mapKeys:function(t,e){var r=this;return be(this,this.toSeq(
).flip().map(function(n,i){return t.call(e,n,i,r)}).flip())}});var an=r.prototyp
e;an[hr]=!0,an[br]=sn.entries,an.__toJS=sn.toObject,an.__toStringMapper=function
(t,e){return JSON.stringify(e)+\": \"+er(t)},Fe(n,{toKeyedSeq:function(){return
new ne(this,!1)},filter:function(t,e){return be(this,fe(this,t,e,!1))},findIndex
:function(t,e){var r=this.findEntry(t,e);return r?r[0]:-1},indexOf:function(t){v
ar e=this.toKeyedSeq().keyOf(t);return void 0===e?-1:e},lastIndexOf:function(t){
var e=this.toKeyedSeq().reverse().keyOf(t);return void 0===e?-1:e},reverse:funct
ion(){return be(this,he(this,!1))},slice:function(t,e){return be(this,pe(this,t,
e,!1))},splice:function(t,e){var r=arguments.length;if(e=Math.max(0|e,0),0===r||
2===r&&!e)return this;t=m(t,0>t?this.count():this.size);var n=this.slice(0,t);re
turn be(this,1===r?n:n.concat(p(arguments,2),this.slice(t+e)))},findLastIndex:fu
nction(t,e){var r=this.toKeyedSeq().findLastKey(t,e);return void 0===r?-1:r},fir
st:function(){return this.get(0)},flatten:function(t){return be(this,de(this,t,!
1));
},get:function(t,e){return t=l(this,t),0>t||this.size===1/0||void 0!==this.size&
&t>this.size?e:this.find(function(e,r){return r===t},void 0,e)},has:function(t){
return t=l(this,t),t>=0&&(void 0!==this.size?this.size===1/0||this.size>t:-1!==t
his.indexOf(t))},interpose:function(t){return be(this,ge(this,t))},interleave:fu
nction(){var t=[this].concat(p(arguments)),e=Ie(this.toSeq(),k.of,t),r=e.flatten
(!0);return e.size&&(r.size=e.size*t.length),be(this,r)},last:function(){return
this.get(-1)},skipWhile:function(t,e){return be(this,le(this,t,e,!1))},zip:funct
ion(){var t=[this].concat(p(arguments));return be(this,Ie(this,rr,t))},zipWith:f
unction(t){var e=p(arguments);return e[0]=this,be(this,Ie(this,t,e))}}),n.protot
ype[fr]=!0,n.prototype[cr]=!0,Fe(i,{get:function(t,e){return this.has(t)?t:e},in
cludes:function(t){return this.has(t)},keySeq:function(){return this.valueSeq()}
}),i.prototype.has=sn.includes,i.prototype.contains=i.prototype.includes,Fe(x,r.
prototype),Fe(k,n.prototype),Fe(A,i.prototype),Fe(et,r.prototype),Fe(rt,n.protot
ype),Fe(nt,i.prototype);var hn={Iterable:e,Seq:O,Collection:tt,Map:ct,OrderedMap
:Zt,List:Wt,Stack:Ve,Set:Le,OrderedSet:Je,Record:Ae,Range:$,Repeat:G,is:X,fromJS
:H};return hn})();
" ;;

let top_level_definitions = [
  ("print", TFun([T("A")], TUnit));
  ("print_string", TFun([TString], TUnit));
  ("print_num", TFun([TNum], TUnit));
  ("print_bool", TFun([TBool], TUnit));
  ("num_to_string", TFun([TNum], TString));
  ("hd", TFun([TList(T("B"))], T("B")));
  ("empty?", TFun([TList(T("C"))], TBool));
  ("tl", TFun([TList(T("D"))], TList(T("D"))));
  ("get", TFun([TMap(T("E"), T("F")); T("E")], T("F")));
  ("set", TFun([TMap(T("G"), T("H")); T("G"); T("H")], TMap(T("G"), T("H"))));
  ("has?", TFun([TMap(T("I"), T("J")); T("I")], TBool));
  ("del", TFun([TMap(T("K"), T("L")); T("K");], TMap(T("K"), T("L"))));
  ("keys", TFun([TMap(T("M"), T("N"));], TList(T("M"))));
];;

let predefined = List.fold_left
```

```
    (fun acc (id, t) -> NameMap.add id t acc)
    NameMap.empty top_level_definitions
;;

let list_definitions = [
  ("length", TFun([TList(T("P"))], TNum));
  ("rev", TFun([TList(T("Q"))], TList(T("Q"))));
  ("nth", TFun([TList(T("R")); TNum], T("R")));
  ("filter", TFun([TFun([T("S")],TBool); TList(T("S"));], TList(T("S"))));
  ("map", TFun([TFun([T("T")],T("U")); TList(T("T"));], TList(T("U"))));
  ("iter", TFun([TFun([T("V")], TUnit); TList(T("V"));], TUnit));
  ("range", TFun([TNum; TNum], TList(TNum)));
  ("concat", TFun([TList(T("W")); TList(T("W"))], TList(T("W"))));
  ("fold_left", TFun([TFun([T("X"); T("Y")], T("X")); T("X"); TList(T("Y"))], T(
"X")));
  ("insert", TFun([TList(T("Z")); T("Z"); TNum], TList(T("Z"))));
  ("remove", TFun([TList(T("AA")); TNum], TList(T("AA"))));
  ("sort", TFun([TFun([T("AH"); T("AH")], TBool); TList(T("AH"))], TList(T("AH")
)));
];;

let map_definitions = [
  ("count", TFun([TMap(T("AB"), T("AC"))], TNum));
  ("values", TFun([TMap(T("AD"), T("AE"))], TList(T("AE"))));
  ("merge", TFun([TMap(T("AF"), T("AG")); TMap(T("AF"), T("AG"))], TMap(T("AF"),
 T("AG"))));
];;


type definition = string * Ast.primitiveType

(* generates a list of module definitions, given a file and a type-checker
not being used at the moment *)
let generate_module_defn (module_name: string) (type_checker: program -> aexpr l
ist) definition list  =
  let file_path = Printf.sprintf "lib/%s.jsjs" module_name in
  let lexbuf = Lexing.from_channel (open_in file_path) in
  let program = Parser.program Scanner.token lexbuf in
  (* infer the program and create the definitions by top level expressions *)
  let inferred_program = type_checker program in
  List.fold_left
    (fun defs ae ->
       let def = (match ae with
        | AAssign(id, t, _, _) -> (id, t)
        | _ -> raise (failwith "top level expressions in modules must be named")
) in
       def :: defs)
    [] inferred_program
;;

let modules =
  (* a function that takes a module map and adds definitions as value
     and module name as key *)
```

```
  let update_module_map map name definitions =
    let definitions = List.fold_left
        (fun acc (id, t) -> NameMap.add id t acc)
        NameMap.empty definitions in
    ModuleMap.add name definitions map in

  (* generate the map for all definitions *)
  let module_defs = [("List", list_definitions); ("Map", map_definitions)]
  in List.fold_left (fun acc (name, defs) -> update_module_map acc name defs)
    ModuleMap.empty module_defs
;;
```

Standard Library

lib/list.jsjs

```
// returns the length of a list
val length = /\(xs: list T): num => {
  val aux = /\(c, ys) => {
    if empty?(ys) then c
    else aux(c + 1, tl(ys));
  };
  aux(0, xs);
};

// returns the nth element of the list
val nth = /\(xs: list T, n: num): T => {
  if n > length(xs) then throw "IndexOutOfBounds" else {
    if n == 0 then hd(xs)
    else nth(tl(xs), n-1);
  };
};

// Selects all elements of a list which satisfy a predicate.
val filter = /\(pred: (T) -> bool, xs: list T) => {
  if empty?(xs) then xs
  else {
    if pred(hd(xs))
    then hd(xs) :: filter(pred, tl(xs))
    else filter(pred, tl(xs));
  };
};

// returns a list of numbers from start to end. Like python
// start is inclusive, end is exclusive.
val range = /\(start: num, end: num): list num => {
  if start >= end then []
  else start :: range(start+1, end);
};
```

```
// reverses a list
val rev = /\(xs: list T): list T => {
  val aux = /\(acc: list T, ys: list T): list T => {
    if empty?(ys) then acc
    else aux(hd(ys) :: acc, tl(ys));
  };
  aux([], xs);
};

// concatenates two lists
val concat = /\(xs: list T, ys: list T): list T => {
  val aux = /\(as: list T, bs: list T): list T => {
    if empty?(as) then bs
    else aux(tl(as), hd(as) :: bs);
  };
  aux(rev(xs), ys);
};

// execute a function with side-effect (returns unit)
// on every element of the list
val iter = /\(f: (T) -> unit, xs: list T): unit => {
  if empty?(xs) then (-)
  else {
    val dontcare = f(hd(xs));
    iter(f, tl(xs));
  };
};

val map = /\(fn: (T) -> U, xs: list T): list U => {
  if empty?(xs) then []
  else fn(hd(xs)) :: map(fn, tl(xs));
};

val fold_left = /\(fn: (T, U) -> T, acc: T, xs: list U): T => {
  val aux = /\(acc: T, xs: list U): T => {
    if empty?(xs) then acc
    else aux(fn(acc, hd(xs)), tl(xs));
  };
  aux(acc, xs);
};

val insert = /\(xs: list T, x: T, pos: num): list T => {
  // handle error better later, right now just return
  // the original list for invalid position
  if (pos > length(xs) || pos < 0) then throw "IndexOutOfBounds"
  else {
    val aux = /\(count : num, acc : list T, xs : list T): list T =>
        if count == pos
        then concat(rev(acc), x :: xs)
        else aux(count + 1, hd(xs) :: acc, tl(xs));
    aux(0, [], xs);
  };
};
```

```
val remove = /\(xs: list T, pos: num): list T => {
  // handle error better later, right now just return
  // the original list for invalid position
  if (pos > length(xs) || pos < 0) then throw "IndexOutOfBounds"
  else {
    val aux = /\(count : num, acc : list T, xs : list T): list T =>
        if count == pos
        then concat(rev(acc), tl(xs))
        else aux(count + 1, hd(xs) :: acc, tl(xs));
    aux(0, [], xs);
  };
};

// sort function, takes a comparator that gives a boolean output
val sort = /\(compare: (T,T) -> bool, xs: list T) : list T => {
    if xs == [] then []
    else {
        val l1 = filter(/\(x) => compare(x, hd(xs)), tl(xs));
        val l2 = filter(/\(x) => !compare(x, hd(xs)), tl(xs));
        concat(sort(compare, l1), hd(xs) :: (sort(compare, l2)));
    };
};
```

lib/map.jsjs

```
val count = /\(m: <T: U>): num => {
  List.length(keys(m));
};

val values = /\(m: <T: U>): list U => {
  val aux = /\(ks: list T): list U => {
    if empty?(ks) then []
    else get(m, hd(ks)) :: aux(tl(ks));
  };
  aux(keys(m));
};

// Returns a map that consists of the rest of the maps conj-ed onto
// the first.  If a key occurs in more than one map, the mapping from
// the latter (left-to-right) will be the mapping in the result.
val merge = /\(m1: <T: U>, m2: <T: U>): <T: U> => {
  val aux = /\(m: <T: U>, mapkeys: list T): <T: U> => {
    if empty?(mapkeys) then m
    else {
      val key = hd(mapkeys);
      val value = get(m2, key);
      val newm = set(m, key, value);
      aux(newm, tl(mapkeys));
    };
  };
  aux(m1, keys(m2));
```

```
};
```

Test Suite

test/run.ml

```
open Printf

let test_location = "test/compiler-tests/"

type test_kind = Pass | Fail

type color = Grey | Red | Green | White

(* reference: http://misc.flogisoft.com/bash/tip_colors_and_formatting
   returns a string that has been color coded *)
let colorize msg c =
  let pad = match c with
    | Grey -> "90"
    | Red -> "31"
    | Green -> "32"
    | White -> "37" in
  let template = format_of_string "
    \027[%sm%s" in
  printf template pad msg;
  flush stdout;
;;

(* runs a unix command and returns
   the output as a list and the status code *)
let run_cmd cmd =
  let chan = Unix.open_process_in cmd in
  let res = ref ([] : string list) in
  let rec aux () =
    let line = input_line chan in
    res := line :: !res;
    aux () in
  try aux ()
  with End_of_file ->
    let status = Unix.close_process_in chan in
    let cmd_result = match status with
      | Unix.WEXITED(c) -> if c == 0 then Pass else Fail
      | _ -> Fail in
    (List.rev !res, cmd_result)
;;

(* takes a diff between a list of lines and a filename
   by first dumping the lines to a file
   and then running the `diff` command in unix`
   Returns an option with the diff output *)
let diff_output lines filename =
```

```
  (* dumps a list of lines to a file *)
  let dump_to_file lines fname =
    let oc = open_out fname in
    List.iter
      (fun line -> fprintf oc "%s\n" line)
      lines;
    close_out oc in

  let _ = dump_to_file lines "temp.out" in
  let cmd = sprintf "diff -B temp.out %s" filename in
  let diff_output, status = run_cmd cmd in
  begin
    match status with
    | Pass -> None
    | Fail -> Some(String.concat "\n" diff_output)
  end
;;

let run_testcase fname =
  (* determine test type - pass or fail *)
  let test_type, test_name =
    match (Str.split (Str.regexp "-") fname) with
    | "fail" :: x :: [] -> Fail, x
    | "pass" :: x :: [] -> Pass, x
    | _ -> raise
        (let msg = sprintf "Invalid file format - %s. Must have only one '-'" fn
ame in
        failwith msg) in

  (* generate command to run *)
  let fpath = Filename.concat test_location fname in
  let cmd = sprintf "./jsjs.out %s" fpath in

  (* get output filename and path *)
  let output_filename = Str.replace_first (Str.regexp "jsjs") "out" fname in
  let output_path = Filename.concat test_location output_filename in

  (* run command and pattern match on result *)
  let cmd_output, status = run_cmd cmd in
  match test_type, status with
  (* expected and actual match on test type - both passing *)
  | Pass, Pass -> begin
      (* run the generated file with node and diff output *)
      let node_output, status = run_cmd "node out.js" in
      (match diff_output node_output output_path with
     | None -> colorize (sprintf "  %s" fname) Green; Pass
     | Some(op) -> begin
         colorize (sprintf "  %s" fname) Red;
         colorize (sprintf "%s\n\n" op) Red;
       end; Fail)
    end

  (* expected and actual match on test type - both failing *)
```

```
  | Fail, Fail ->
    (match diff_output cmd_output output_path with
     | None -> colorize (sprintf "  %s" fname) Green; Pass
     | Some(op) -> begin
         colorize (sprintf "  %s" fname) Red;
         colorize (sprintf "%s\n\n" op) Red;
       end; Fail)

  (* expected pass and got failure *)
  | Pass, Fail -> begin
      colorize (sprintf "  %s" fname) Red;
      colorize "Expected test case to pass, but it failed" Red;
    end; Fail

  (* expected failure but passed *)
  | Fail, Pass -> begin
      colorize (sprintf "  %s" fname) Red;
      colorize "Expected test case to fail, but it passed" Red;
    end; Fail
;;

let run testcases () : test_kind =
  let total = List.length testcases in
  let t_start = Sys.time() in
  let passing = List.fold_left
      (fun acc t ->
         acc + (match run_testcase t with Pass -> 1 | Fail -> 0))
      0 testcases
  in
  let template = format_of_string "\027[37m

    Test Summary
    ------------------------
    All testcases complete.
    Total Testcases : %d
    Total Passing   : %d
    Total Failed    : %d
    Execution time  : %fs
      \n" in
  let failures = total - passing in
  Printf.printf template total passing failures
    (Sys.time() -. t_start);
  if failures = 0 then Pass else Fail
;;

(* returns a list of file names in a directory *)
let get_files dirname =
    let d = Unix.opendir dirname in
    let files = ref ([] : string list) in
    let rec aux () =
      let fname = Unix.readdir d in
      files := fname :: !files;
      aux () in
```

```
    try aux () with End_of_file -> Unix.closedir d; files
;;

let init () =
  let files = !(get_files test_location) in
  (* testcases -> all files that end in .jsjs *)
  let testcases = List.filter
      (fun f ->
         try ignore (Str.search_forward (Str.regexp ".jsjs") f 0); true
         with Not_found -> false)
      files in
  match (run testcases ()) with
  | Pass -> exit 0
  | Fail -> exit 1
;;

init ();
```

Makefile

```
DOCS=docs/proposal.md
TESTDEPS=oUnit -linkpkg -g
filename=jsjs
ENTRYPOINT=driver
EXECUTABLE=jsjs.out

jsjs:
    ocamlbuild -j 0 -lib str -r -use-ocamlfind -pkgs core -use-menhir src/$(
ENTRYPOINT).native
    @mv $(ENTRYPOINT).native $(EXECUTABLE)
    @echo ---------------------------------
    @echo JSJS is ready to be served!
    @echo ---------------------------------


.PHONY: infer
infer:
    ocamlc -o infer.out src/type_infer.ml
    @rm src/*.cm*

.PHONY: test
test:
    ocamlc -o run-tests.out str.cma unix.cma test/run.ml
    @rm test/*.cm*

.PHONY: run-test
run-test:
    @#python test/parser-tests/menhir.py
    @#find examples -name "*.jsjs" -exec /bin/echo "Compiling {}" \; -exec .
/jsjs.out {} \;
    @./run-tests.out

.PHONY : clean
```

```
clean:
    ocamlbuild -clean
```

Project Log

```
cbc65f0 - cleaned up codegen for jsjs keywords, added Unix time for run tests
ca6d432 - catchign exceptions during codegen
450bcc5 - removed old requirements.txt
0ba6f85 - codegen errors for top level and lambdas, fixes #76
814d834 - no redefinition of JSJS keywords (repeating commit, please pull) and m
ore test cases. also fixed boolean operator precedence
c516501 - some moretest cases..
154db4f - Merge pull request #74 from prakhar1989/codegen
d03d74e - ensured no redefinition of JSJS top level vals
dd0ff76 - fixed all issues with codegen - pushing new branch, #72
d7ca683 - new codegen working for everything except modules, #72
de6ef78 - tests on if-then-else
413165e - more tests on maps
10a756b - map has and del tests
c2d0313 - map get and set tests
ed01569 - list filter tests
769797e - added functions on list range and length functions
bc06642 - test cases for boolean ops
c4964db - added tests for map merge and anonymous function application
99eaa6c - fixed tests on unit comparison
5eb70d1 - updated tests
bc2d622 - test cases for lists
c7c5954 - type checking in module fn calls, fixes #73
361b0cd - fixed bug with duplicate vals in function blocks
e9977e7 - unit equality test - need to fix
b75bcb1 - added failing test case for list sort, fix still reqd, #73
f6eb6b1 - fixed bug with map key types
efd4850 - Merge branch 'master' of github.com:prakhar1989/JSJS
9e6f2fb - more test cases..
d0e0bb2 - new logo
20ceb2b - added more test cases and take function in examples
d868fac - added list.sort() with passing test cases
871f3f3 - better error message on unreachable state
1a5a7ad - print_me function added for JS codegen
7aa20e2 - better error msg for parser errors
f7887db - Update README.md
be4657c - new img
adf6d29 - added the logo
abffb8e - added ability to take input from stdin. fixes #62
8e5d6ea - string_of_type now returns prettier type placeholders.
6483314 - done with maps. fixes #70
44ae763 - done with lists #70
fca5ef8 - list and map equality. added prilimary tests for list testing #70
4d30ac7 - added codegen based on annotated expr. fixes #71
9fe263b - more tests, #53
250b3de - fixed arg count check in unify_one
ff5e2fa - more tests, #53
720266f - fixes error with recursive generic types
```

```
3ddf9ca - testcase for identity fn #53
6d1ffee - test cases, #53
5e58e81 - added test cases for annotations, #53
e41958c - added \n at the end to allow easier test setup
97d248e - added the exception example from slides as test
6521aea - added _ as disposable var, alongwith tests. fixes #65
cdf8fd2 - added printing of missing ast expressions. fixes #69
1b92c87 - adding travis
d6766e0 - all test cases pass :boom: :boom: :boom:
ab4693f - all but 1 test cases pass
9715c56 - made generic type placeholders to be exclusive.
c9c09ef - fixed name scoping issues in modules
d30ff96 - minor fixes across stdlib. commented implementation of stdlib generati
on
500afb9 - fixed bug with annotating generic types.
6370bf7 - added explicit type annotations in libraries
dcecaa4 - generic annotations now resolve correctly like OCaml
0842620 - generic annotations are disregarded.
fbc62fb - generated print fns correctly. fixes #64
b30a09d - fixed name precedence resolution in modules.
c9e946e - changed type_placeholder start. fixes #63. fixes #66.
45aa481 - made a few tests to pass
3ff2e18 - updated the typechecker to raise correct exceptions
46be6cd - updated typechecker with exceptions
de457eb - Merge pull request #57 from prakhar1989/type-inference
51bb2ab - added generic print function
7f3c1d4 - fixed error in codegen
2b781d3 - type inference works correctly on whole program
9e001b2 - module definitions and top level definitions are used for inferrence c
orrectly
c9d3243 - fixed examples and added comments for type-checker.
b487f2c - replaced gen_new_type function to generate infinite placeholders
aed7dfb - changed stdlib functions acc to new syntax. all parser errors fixed #3
1
98d63da - driver updated to print type-inferred ast and do codegen.
c530963 - fixed codegen with for new AST. commented the code
5dcbcd0 - fixed codegen for new AST
1633c1a - modified associated module functions
144f536 - module type definition changes after hmt
1803838 - module definition changes. #31
7ff74c4 - added all types in HMTI, #31
caf27bc - type checker initial working #31
a7b8599 - pipeline works #31
36369c2 - added stringify of aexpr #31
6282119 - collect function constraints, #31
9b70ebb - working through collect. #31
c2e0461 - added type_of #31
fc8f795 - started work on annotating types #31
8113370 - annotated types in ast, parser changes for new funlits. #31
14ad406 - updated syntax changes for type-inference work
37a1a4b - adding better comments
88a3fa4 - added type inference for lambda expressions
c8f101b - commentzzzz
66ad853 - hmt prototype near completion #31
```

```
eed4e1e - prototyping HMT #31
c0f5685 - tests now print one-by-one
6b80ebd - added a test for hello world
8a7e42c - added tests for few map functions
e2e037f - added a test case to demonstrate function composition
cad6f0b - added List.iter stdlib function. #32
90960e8 - added some tests for map
150719b - fixed missing output for test case
555a79e - tests for list nth and insert exception
c87b4d3 - added exceptions in list std lib.
8dee437 - fixed codegen issues in nested exceptions. fixes #38
31cc361 - added codegen and typechecking for exceptions. #38
253c2df - syntax and AST changes for exceptions. #38
82c99f9 - disallowed JS keywords to be defined. fixes #33
45abab1 - renamed reduce to fold_left. added testcases #49 #32
1a2e897 - added type-checking for non-generic args in generic fn calls. fixes #5
0
bce637a - added test cases for list.insert and list.remove #49
d0c4d55 - added List.remove(), #32
b38a057 - added List.insert, #32
1deeb5e - reduce function added in list module, #32
49edb7e - test cases for type annotataions in definition
72500f8 - removing wrong implementation. correct impl added by @bahuljain
d178ced - Merge branch 'master' of github.com:prakhar1989/JSJS
7882c62 - sieve of eratosthenes in jsjs
459f903 - added a countprimes function.
2ca5ac7 - added a list filter_not stdlib function.
53242da - better error message for testcase file naming. #48
f9947ee - makefile changes for new test suite. #48
f4a838a - cleaned output with colorize. other minor changes. fixes #48
c270b06 - automatically finds files to run as testcases.#48
b1496cd - added ability to run passing testcases with node and compare output #4
8
93daaee - handling failure cases properly. #48
5bba1e7 - added colored output, diffing and testcases. #48
c104e2e - added a function for validating output #48
1b896f2 - added ability to dump to file #48
54537a2 - porting the test suite over to OCaml #48
5d89f9f - initial code for setting up test pipeline #48
3e975f5 - errors are now outputted to stderr. catching exceptions for files / di
r failure
bd68de9 - started work on test suite. #48
2952e75 - added separate error message for illegal generic fn decl. fixes #34
7b06cc8 - more comments #41
f0981cb - added comments for ast. #41
fd6ffb0 - adding comments
1ac9098 - added comments to ast.mli
b24345b - changed codegen to deal with string keys in Immutable. Fixes #44
748c122 - fixed validate_types for maps. #44 fixes #37
f301523 - added one-liner for make test
d0b0ef8 - updated all examples with latest syntax
4b699b3 - bug fix: re-added missing type-checking for fn args
0d5e50c - cleaned up examples folder
b0588d8 - fixed assigments as last expr in blocks. Fixes #45
```

```
7d2efa8 - added merge in map stdlib #37
834d804 - created modules map on top. this allows modules to call each other
6354dba - added more stdlib Map functions. #37
e1eca5b - added top-level fns for map. #37
459caf4 - started work on map stdlib. added getter #37
4b0b250 - completed codegen for modules. #42
4b2b2bf - tested codegen for list modules. #42
a1857d5 - modified js_of_expr to support modules in codegen. #42
c5ce7d6 - added type-checking for modules. #42
c8b27dc - started work on modules. added type definitions for List stdlib. #42
4d77fe2 - added validate_types function check for funlit assigns. #35
87b1a5a - fixed validate_types by making it recursive for TAny resolution. #35
1ef23c4 - added the map function for stdlib list #32
6ee6ffd - fixed stdlib concat list function. #32
2311c7b - added type-checking for TAny #35
4ed7db5 - fixed associativity for cons operator and handled empty list cons. #36
28772e7 - comments for TAny and fixed empty map literal types. #35 #41
1ec681c - renamed TSome to TAny
b916459 - added a few comments for the funLit typechecking part. #41
90b9fc9 - added two more stdlib list functions #32
b182593 - added the list stdlib function for printing a list. #32
f3cbb05 - added replacement for ? string in var names for codegen. fixes #22
1708dc5 - added a range function and a generic print function
100c55b - fixed example syntax in README
984fce0 - finished work on the cons operator. added filter in stdlib for testing
. #32 fixes #36
944706a - added cons operator in scanner, parser & ast. #36
12fcfe5 - added two stdlib functions for lists. #32
27eb97e - added isempty for lists. #32
26dcdb9 - added cons for lists. #32
a45c46c - added function to generate return types in generic fns. #29
cc5cf89 - added hd for list. #32
700a230 - added the correct immutable lib. list creations
e7d97e1 - added immutable library for lists. #32
31afb1a - generics for function calls completed for list and maps. fixes #29
50bcdd1 - disallowed unresolved types to be returned from generic functions. #29
2fd1536 - parser changes for supporting type annotations of generic functions. #
29
f7dc1b1 - typechecking for generic arguments as functions. #29
c8d4fe6 - few commentzz #29
488dfd7 - incorrect argument fix for generic function calls. #29
803ddb3 - initial version of generics for simple args. #29
0b27e0a - made changes in parser for new generic syntax. #29
ce13da4 - removed package json'
570c7f5 - cleaned up some codegen js
43e9621 - did codegen for hello world and 99bottles. #27
6868be1 - done with codegen for funlit and calls #27
4a1c0e7 - added print_num and print_str functions into globals
7a381b7 - codegen for if-then-else and blocks works. #27
9a5d444 - cleaned up code
c373514 - started work on codegen. version 2
70d8154 - added type checking for modules. fixes #24 :star: :star: :star:
4b5c222 - added the unit literal and associated tests #24
6df4683 - completed type checking for funlits and calls. #24
```

```
ad21a1e - better error messaging for mismatch operand types and arg counts
32d75d1 - implemented type checking for function literals. #24
0342907 - took care of re-defining values
1bca115 - added type checking for blocks. fixes #26
e8101ec - vals are now typechecked as globals #24
5e72074 - added type checking for vals #24
310f088 - extended type checking func to return type env. #24
d363952 - cleaned up driver exceptions
8cd3f3b - added the boilerplate for type environment. #24
d5f3df1 - removed trailing whitespace from the codebase
1b512e7 - fixed bug with !=. cleaned up testcases #24
635afc1 - added type checking for maps #24
8ecc064 - fixed bug in type checking for list literals #24
92bad7e - added a placeholder for assign type checking #24
bf30bf6 - added type checking for block and if
e1c62d9 - added type checking for more literals
c171d51 - added the typechecker in the pipeline
45e5f78 - fixed all examples
6a32fea - fixed all syntax issues. Fixes #25
c735c14 - new syntax changes for LRM. #25
72be9c7 - Update README.md
a7af889 - Update README.md
c46617d - Update README.md
c7e13de - added a Block type in AST. #25
9ac82a6 - fixed failing tests cases after syntax change
84727bc - start work on type checking. #24
c5ae09d - moved to std lib method
537ab43 - Cleaned up stringify module functions
957a7ac - replaced interpret with semantic
0ff22ba - added flags to the driver. fixes #18
e8c0354 - setup Core. and Command module for the driver
5d80d07 - removed = from func decls
56ef9b1 - moved to ocamlbuild. no more warnings :clap:. thanks @DavidWatkins. Fi
xes #23
78d3cf7 - if then else can now have single line exprs. Fixes #15
946630d - back to pretty map syntax #15
14f1cf0 - make now handles npm install
dd39087 - cleaned up example code for demo
3a7ffdd - fixed build steps for generating JS
4d49a08 - fixed bug with ^ op
a2f009e - hello world ready
905df47 - codegen for simple expressions complete
e98f79b - assigns escodegen working
73d93ed - started work on js codegen
aa306dd - removed auto-generated files from GH
8444b01 - Merge pull request #17 from prakhar1989/syntax-fix
773d8b2 - changed stringify to update to Map syntax
8874536 - removed ml file at root
f36f73e - map example in readm #15
8804d71 - resolving merge conflicts
3c0e07e - clean parser
2fee7eb - Merge pull request #16 from prakhar1989/print_func_decls
ace0341 - pretty printing function decls
6ef6c8d - added a driver. cleaned up some code
```

```
61852bf - added python driver for testing menhir
a7ee0ff - tested generic types
61b0b51 - fixed the stringify of expr
136f958 - done with generics. grammar complete
86c6b6b - maps complete.
0242cb4 - done with lists
d1464d6 - Merge pull request #14 from prakhar1989/grammar
9abbaf8 - no spaces
54dedbc - fixed examples for new lambda syntax
f2e17fa - sexy looking  s in da house
875a7d8 - wrote expr parsing for anon functions. fuck yeah :metal:
9cbd235 - finished function call.
fecebae - done with def grammar. added menhir tests
f3bae13 - fixed blocks and expr_lists
903e2a0 - solved exisential crisis. moved to menhir :neckbeard:
398ebf3 - Merge pull request #13 from prakhar1989/move-to-map
c5b01ab - removed parens from IF predicate
f328f8a - added ability to stringify expressions
726b224 - renamed main to interpret
277e56b - added nested if example
1ae697c - added if-then-else
84c519d - moved from hashtable to map. :metal:
d656bd5 - got rid of first part of errors
40eed1f - added a program main type in parser
881ae69 - Update README.md
57c8264 - added a separate stringify module for hygiene. :dash:
2813777 - rephrased error msgs for mismatched types
843cb53 - better type mismatch errors. fixes #10
e7de5bf - raise exception on redefining existing value. fixes #9
debb41d - fancier error reporting. yeah baby :dancer:
e2fdf83 - line number reporting for parser errors.
7ad37fc - separated primitive type in parser
ec796d5 - structured the parser
4d5fe5e - fixed the executable
7d3a83f - the beginnings of a testsuite
0de51a2 - added parenthesis in expressions. fixes 8
e86857d - supports the - as unary operator. fixes #7
3a09ca3 - support for all relational operators added
77cbaf1 - added type checks errors for all ops. additional errors file checked-i
n
6e3af6c - booleans now support the unary operator :neckbeard:
c9d4d22 - added binary operations on booleans
777cb39 - added the literals for bools
720d3f3 - added exceptions for operations on types
321ef0c - added the missing associativity rules for caret.
b7a05aa - added string concat operation for strings
0d51b36 - expr can now be strings
9123368 - eval now returns primitivetypes
dd8ff9e - added line no and cursor location to scanner errors. fixes #3
75acfbe - cleaned up makefile further
a924ce4 - added custom exception in scanner
8cf507d - added flags in makefile
14b4a37 - cleaned up the makefile
54f6f9f - added ability to read input from files. fixes #4
```

```
da3a1bb - added variables in calculator
8804abe - wrote an AST for arithmetic expr
34da79a - removed extra makefile
8a62e79 - added tokens in the parser :dancer:
c2f83e1 - from printfs to tokenzzz
57a154d - removed compiled junk
8f78842 - added all tokens for scanner
4a91176 - started work on scanner
2256e1e - added examples
91acf5a - added comparison table
2d5687b - fixed mergesort line
38a4e12 - to markdown
0970ab5 - added proposal
8db1a90 - Initial commit
```