JSJS - Report

A general purpose, strongly typed programming language for the web

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

Description

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	Exists in JS?	In scope	Out of scope
Type Safety	No	Strong types	Type inference
Immutable values	Yes - ES6 only	No variables	NA
Immutable Data Structures	No	Lists and Maps	Arrays and Sets
Polymorphic Types	No	For built-ins	User-defined types
First-class Functions	Yes	Syntactic sugar	NA
Compiles down to JS	NA	Node & browser	NA

Lexical Conventions

Name	Function
Keywords	`if`, `then`, `else`, `num`, `string`, `bool`, `val`, `def`, `list`, `true`, `false`
Comments	`// this is a comment`
Multi-line Blocks	`{ ... ; }`
Conditionals	`if (x == 10) then { 10 * 10; } else { 100; }`

Primitive Data Types

Data Types	Example	Declaration
num	42, -42, 4.2, -0.00042	`val a : num = 42;`
bool	true, false	`val isAwesome? : bool = true;`
string	“jsjs”, “is”, “the”, “best”, “!!!”	`val name : string = "jsjs";`
unit	`unit`	`val _ : unit = println("Hello");`

Type Declaration

Values	Type Declaration	Description
Any	`a: T`, `a : U`	where T and U can be any of the types given below
Number	`a: num`	where num represents the type for Numbers
Boolean	`a: bool`	where bool represents the type for Logical values like true and false
String	`a: string`	where string represents the type for string literals
Unit	`a: unit`	used for dealing with side-effects
List	`a: list T`	where T is type of elements of the list
Map	`a: <T : U>`	where T is type of the key and U is type of the value
Function	`a: (T1, T2, ..) -> U`	where T1, T2,… are the types of the arguments taken by the function and U is the return type of the function

Operators

Operator Name	Syntax	Applicable Data Types
Assignment	`a = b`	num, bool, string
Integer Arithmetic	`a + b, a - b, a * b, a / b`	num
Modulo	`a % b`	num
Equal to	`a == b`	num, bool, string

Operator Name	Syntax	Applicable Data Types
Not Equal to	`a != b`	num, bool, string
Comparison	`a <= b, a < b, a >= b, a > b`	num, bool, string
Logical AND	`a && b`	bool
Logical OR	`a || b`	bool
Logical NOT	`!a`	bool
Concat	`a ^ b`	string

Composite Data Types

List

List Type Declaration

- The **list** keyword is used to declare the type of a list
- Type declaration for a list: `a : list T`

List Declaration

```
// List of strings
val names : list string = ["foo", "bar", "baz"];

// List of nums
val nums : list num = [1, 2, 3, 4];

// List of lists of string
val friends_list : list list string = [["foo"], ["bar"], ["baz"]];
```

List Functions

```
// Returns the first element of the list
val head : num = List.hd([1, 2, 3, 4, 5, 6])

// Returns the list without the head of the list
val tail : list num = List.tl([1, 2, 3, 4, 5, 6])

// Returns the length of the list
val length : num = List.length([1, 2, 3, 4, 5, 6])
```

Map

Map Type Declaration

- The <> construct is used to declare a map
- Map Type Declaration: `a : <T, U>` where T is type of the key and U is the type of the value.

Map Declaration

```
// creates a map with key of string type and value of num type.
val passwords : <string: num> = {
 "foo" : 1245,
 "bar" : 5567,
 "baz" : 5533
};

// creates a map of key string type and value being another map
// that has key of string type and value of num type.
val students : <string: <string: num>> = {
  "foo": { "marks": 97 },
  "bar": { "marks": 23 },
  "baz": { "marks": 47 },
};
```

Map Functions

```
// Returns value associated with key = "foo" in map named passwords
val foo = Map.get(passwords, "foo");

// Sets the value of field "foo" as 12345 in map named passwords
val foo = Map.set(passwords, "foo", 12345);
```

Functions

Function Type Declaration

- JSJS has a unique way of declaring the type or signature of a function.
- This is particularly needed when declaring anonymous functions.
- Function type is declared as: `f : (T, T, ...) -> U` where T are types of the arguments and U is the return type of the function

Function Expressions

```
// defines a function square that takes
// a `num` argument and returns a `num`
def square (x: num) : num = x * x;
```

Multi-line functions

```
// multi-line statements (i.e blocks) are separated by
// a semicolon. The last line in a block is returned automatically.
// Hence, there is no explicit `return` keyword
def cube (x: num): num = {
  val sq = square(x);
  x * sq;
}
```

Functions as values

```
// Assigns the name `sq` to an anonymous function.
// Functionally equivalent to def sq (x: num) : num = x * x;
val sq : (num) -> num = (x: num) : num => x * x;
```

Anonymous Functions

```
// passing an anonymous function using the arrow syntax
val squares : list num = map((x: num) : num => x * x, [1, 2, 3, 4]);
```

Polymorphic Functions

```
// defining a polymorphic function that takes two arguments
// The first argument is a function, the second is a list.
// Finally, the return type of this function is another list
// of (possibly) a different type than the input list.
def map (f: (T) -> U, l: list T): list U {
   if (List.empty?(l))
   then { []; }
   else {
     val res = f(List.hd(l));
     List.cons(res, map(f, List.tl(l)));
   }
}
```

Example Programs

99 bottles of beer

```
def ninetyNineBottles (): unit = {
   def toStr(x:num) : string = {
      if (num == 0)
      then { "no more"; }
      else { String.toString(num); }
   }

   def ninetyNine (x:num) : unit = {
      if(x==0)
      then {
         println("No more bottles of beer on the wall, no more bottles of beer.");
         println("Go to the store and buy some more, 99 bottles of beer on the wall.");
      }
      else {
         println(toStr(num) ^ "bottle of beer on the wall, " ^
                 toStr(num) ^  " bottle of beer.");

         if(x==2)
         then {
            println("Take one down and pass it around, " ^
                    toStr(num - 1) ^ " bottle of beer on the wall.");
            nintetyNine(x-1);
         }
         else{
            println("Take one down and pass it around, " ^
                    toStr(num - 1) ^ " bottles of beer on the wall.");
            nintetyNine(x-1);
         }
      }
   }
   ninetyNine(99);
}
```

Euclid's algorithm for GCD

```
def gcd (a : num, b : num) : bool = {
   if (a == b)
   then { a; }
   else {
      if (a > b)
      then { gcd((a - b), b); }
      else { gcd((b - a), a); }
   }
}
```

Merge Sort

```
def merge (a : list num, b : list num) : list num = {
   if(List.hd(a) < List.hd(b))
   then { List.cons(List.hd(a), merge(List.tl(a), b); }
   else { List.cons(List.hd(b), merge(a, List.tl(b)); }
}

def split (a : list num, start : num, end : num) : list num = {
   if (start != 0)
   then { split(List.tl(num), start - 1, num); }
   else {
      if (end == -1)
      then { a; }
      else { List.cons(List.hd(a), split(List.tl(a), start, end - 1)); }
   }
}

def mergeSort (a : list num) : list num = {
   val size = List.length(a);
   if (size == 0 || size == 1)
   then { a; }
   else { merge(mergeSort(split(a, 0, size/2)),
                mergeSort(split(a, size/2 + 1, size - 1))); }
}
```

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

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.
`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  : 0.123073s
```

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  : 0.123073s
```

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.