Symbol Type

Symbols

In JavaScript there is two primitive types that can be used as object property key

1. String
2. Symbol type

Don't confuse the symbol type in Ruby with JavaScript they are different albeit similar in that they are used to create unique identifier.

A symbol can be created using `Symbol()`

```
let id = Symbol();
```

You can give a symbol a description as it's parameter, mostly useful for debugging purposes.

```
// id is a symbol with description "this is id"
let id = Symbol("this is id");
```

Symbols are guaranteed to be unique, even if they have the same description, they are considered to be different values.

```
let id1 = Symbol("id");
let id2 = Symbol("id");

id1 == id2 // False
```

Property of symbols

- Symbols don't support implicit conversion to a String.

Hidden properties

Symbols allows you to create hidden properties on an object that no other part of code can accidentally access or overwrite

```
let user = {
	name: "John"
};


let id = Symbol("id");


user[id] = 1;


console.log(user[id]);
```

Here we are creating a hidden property using the id symbol that we have created. We can access it using the same symbol as the key.

However, if we tried to access it using String for example `user["id"]` we would get `undefined`.

Using symbol is to avoid conflict between say a third party code that wants to inject some of their own property into the object. If they are just using String key then it will likely overwrite some of the existing property for the object. However, if they use Symbols then there will be no conflicts between the identifiers.

Symbols... more

- You can use symbols in object literal, just use the square bracket when you are using symbols as key, like how you would use an variable as key
- Symbols are skipped in `for ... in` loops

Global symbols

If you want different named symbols to be referring to the same entity you can do that using global symbol registry. Create symbol in it and access them later, repeated access by the same name will return the same symbol.

The function `Symbol.for(key)` will look in the global registry for a key named `key` and return the Symbol that the `key` is mapped to. If it doesn't exist it will create one. Subsequent access to the same `key` will guaranteed to return the same Symbol.

```
let id = Symbol.for("id"); // read from global registry, create if it doesn't exist


let idAgain = Symbol.for("id"); // read it again in another part of code


id == idAgain // true
```

This symbol is the same symbol in Ruby.

Symbol.keyFor

If you want to find the `key` for a particular symbol using the Symbol object, you can do that with this function.

```
let globalSym1 = Symbol.for("name");
let globalSym2 = Symbol.for("name");


Symbol.keyFor(globalSym1); // "name"
Symbol.keyFor(globalSym2); // "name"
```

Revision #1
Created 20 December 2022 19:45:00 by Tamarine
Updated 20 December 2022 21:01:32 by Tamarine