Note Web Application

Brendon Mendicino

February 28, 2023

Contents

1 Javascript Introduction

Javascript is backward compapatible, to be able to use the previous features is use the directive:

```
"use strict";
```

JS has primitive types and non-primitive types, JS is also and strongly typed language, the primitive types are: string, number, boolean, null, undefined. The non-primitive are the objects, which can be: array, function, user-defined.

The all possible false values in JS: `0, -0, NaN, undefined, null,''`, in JS there are two main comparison operators:

```
a == b    // equal, convert types and compare
a === b   // strict equal, inhibits automatic type conversion
```

In JS you can create variable with:

```
// modern
let a = 10;    // can be changed
const b = 'a'; // constant

// old
var k = 9;
j = 30;
```

The difference between null and undefined, is that variable with null they old a value which is null, on the other way if a variable is declared and nothing is associated with it the value olds by default undefined.

A scope is defined by a **block**, which is created with ...

There two kinds of `foreach` in JS, using `in` allows iterating over objects, while `of` allows iterating over iterable objects:

```
for (let a in object) {
  ...
}

for (let b of iterable) {
  ...
}
```

Using arrays:

```
let a = [1, 2, 'ok', false];
let b = Array.of(1, 2, true);
a.push(5);      // append an element
b.unshift(2);   // insert at the beginning

let copy = Array.from(a);  // shallow copy, it does not deep copy
```

The **destructuring assignment** can be done, it extracts the values from the mast left-hand side:

```
let [x, y] = [1, 2];
[x, y] = [y, x]     // swap
```

The **spread operator** (...) expands on iterable object into it's values:

```
let [x, ...y] = [1, 2, 3, 4];       // y == [2, 3, 4]

const a = [1, 2];
const b = [0, ...a, 3]; // [0, 1, 2, 3]
```

Spreding can be from the left or from the right, usually the spread operator is used for copying array:

```
const a = [1, 2];
const b = [...a];
```

A **string** is JS is an immutable type (like python) encoded in Unicode. The **template literals** can be done with the **tick** operator ‘‘ (expression like Kotlin):

```
let name = 'Bre';
let sur = 'Mend';
// Template literal
let fullName = `${name} ${sur}`;
```