Types and Literals

Overview

- Primitive types
- Literals
- Constructors
- Built-in objects

Primitive Types

- Strings
- Booleans
- Numbers
- Null
- Undefined

"typeof" operator

```
1 typeof <expression>
2
3 var result = typeof "hello";
4 // result == "string"
```

Strings

```
var str = "I am a string";
assertEquals(typeof str, "string");
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

Strings

```
var str = "I'm a string";
assertEquals(typeof str, "string");
```

```
var str = 'I\'m a string';
assertEquals(typeof str, 'string');
```

Works with both single and double quotes

Converting to String

```
var str = String(32);
assertEquals(str, "32");

str = "" + 1024;
assertEquals(str, "1024");
```

String methods

```
"hello world".toUpperCase();
"hello world".toLowerCase();
"   hello world   ".trim();
"hello world".charAt(3);
"hello world".split(' ');
"hello world".slice(1, 3);
```

Mini Quiz

```
var result = " hello world"
  .slice(6)
  .trim()
  .toUpperCase()

console.log(result);
```

```
>> WORLD
```

Booleans

```
var bool = true;
assertEquals(typeof bool, "boolean");
assert(true == !false);
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Boolean

Converting to Boolean

```
var bool = Boolean("String");
assertEquals(bool, true);
```

We'll get to the inner workings
of truthy and falsy-ness

Numbers

```
var num = 3;

assertEquals(typeof num, "number");
```

- Only one number type
- IEEE-754 Doubles
- (No ints)

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number

Precision

```
var num = 0.1 + 0.2;
assertEquals(num, ??);

assertNotEquals(num, 0.3); // ??

// 0.30000000000000004
```

Be aware!

Min/Max values

```
assertEquals(Number.MAX_VALUE, 1.79769313486231570e+308);

assertEquals(Number.MIN_VALUE, 5e-324);
```

Scientific notation

```
2e10 //=> 20000000000
```

Infinity

```
assertEquals(Number.POSITIVE_INFINITY, Infinity);

assertEquals(Number.NEGATIVE_INFINITY, -Infinity);
```

Not a Number

```
var num = 1 / {};
//=> NaN
num == NaN //=> false
NaN == NaN //=> false
```

Not equal to anything

IsNaN

```
var num = 1 / {};
assert(isNaN(num));
```

The Type of NaN?

```
assertEquals(typeof NaN, "number");

NaN * 10;
// => NaN
```

Can continue number operations,

but will always be NaN

Convert to Number

```
var num = Number("32");

assertEquals(num, 32);
```

Better conversion

```
var num = parseInt("32", 10);
assertEquals(num, 32);

var num = parseFloat("32.23");
assertEquals(num, 32.23);
```

Null

```
var obj = null;
assertEquals(obj, null);
assertEquals(typeof obj, "object");
assertEquals(typeof null, "object");
```

Unfortunate design flaw..?

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/null

Undefined

```
var obj;

assertEquals(obj, undefined);

assertEquals(typeof obj, "undefined");
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined

Check for undefined

```
var a;

a == undefined                  //=> true
a == null                       //=> true
typeof a == "undefined" //=> true
```

Undefined vs. ReferenceError

```
var undef;

jQuery == undef                  //=> ReferenceError
jQuery == null                   //=> ReferenceError
typeof jQuery == "undefined" //=> true
```

typeof is a "safe" operator in this case

Primitive Literals

- Strings
- Numbers
- Booleans
- Null
- Undefined

Primitive Literals

```
var myNumber = 10;
var myString = "hello!";
var myBool = true;
var myNull = null;
var myUndefined = undefined;
var myUndefined;
```

Literals

You use literals to represent values in JavaScript. These are fixed values, not variables, that you *literally* provide in your script.

Constructors

Alternative to literals

```
// Construct new instances
// from a parent class

var str = new String('hello');
var num = new Number(6);
var arr = new Array(1, 2, 3);
var obj = new Object();
```

Constructing an instance from a parent class

Object Literals

- Arrays
- Functions
- Objects
- Regular expressions

Object Literals

```
var object = {
  name: "Eirik",
  age: 29,
  isHuman: true,
  sayHello: function() {
    console.log('Hello from ' + this.name);
  }
};

console.log(object.name); // "Eirik"

console.log(object.age); // 29

object.sayHello(); // "Hello from Eirik"
```

Object Literals

More in depth later

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects

Array Literals

```
var arr = [1, 2, 3];

var arr2 = ["Mixed", { "Content": true }, null];
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Array Literals

- length is last set index + 1
- Bracket notation

Adding items

```
var arr = [];
assertEquals(arr.length, 0);

arr[0] = 1;
assertEquals(arr.length, 1); // [1]

arr = arr.concat([2, 3, 4, 5]);
assertEquals(arr.length, 5); // [1, 2, 3, 4, 5]

[1, 2, 3, 4, 5] == arr // False
```

Item access

```
var arr = [1, 2, 3, 4, 5];
assertEquals(arr[1], 2);
```

Avoid this

```
var arr = new Array();
var arr2 = new Array(1, 2, 3, 4, 5);
var arr3 = new Array(5);
var arr4 = Array(6, 7, 8);
```

It will not always give you the results you expect

Dates

```
var today = new Date();
// => 2019-10-25T11:00:10.916Z

today.toString();
// => 'Fri Oct 25 2019 13:00:28 GMT+0200
//     (Central European Summer Time)'
```

Has no literal - must use constructor

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

Equality Dates

```
var date = new Date(2015, 8, 23);

assert(date.getTime() == date);
```

Regular Expression Literals

```
var regex = /ab+c/;
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

Built-in objects

- Global object
- Object
- Function
- Array
- String
- Boolean
- Number
- Math
- Date
- RegExp
- Error

Tasks

Canvas: exercises-types-and-literals.zip