Introcution to JavaScript

Nils Twelker

March 2023

What learned we last Week?

- Functions `function add(a, b) { return a + b; }`
  - (Default) Arguments, Return Values, Scope, Hoisting
- Arrow Functions `let add = (a, b) => a + b;`
- Function Expressions

```
let add = function(a, b) { return a + b; }
```

- Basic Arrays `let myList = [1, 2, 3, 4]`
- Accessing Arrays `myList[0] // 1`

Goals of this week

- Objects
  - Accessing Properties
  - Adding Properties
  - Removing Properties
  - “this” Keyword
  - “in” Operator
  - “For in” loop
  - Object references
- Garbage Collection

- Methods on Primitives
- More about Arrays
  - Arrays are Objects
  - Array Methods

Objects

```
const player = {
    health: 30,
    color: "red",
    running: false,
    sayHello: function() { console.log("Hello!") }
}
```

Accessing Properties

```
const player = {
    health: 30,
    "player color": "red",
    running: false,
    sayHello: function() { console.log("Hello!") }
}
player.health // 30
player["player color"] // "red"
player.running // false
player.sayHello() // "Hello!"
```

Adding Properties

```
const player = {
    health: 30,
    color: "red",
    running: false
}

console.log(player.name) // undefined
player.name = "Paul"
console.log(player.name) // "Paul"
```

Removing Properties

```
const player = {
    health: 30,
    color: "red",
    running: false
}

console.log(player.color) // "red"
delete player.color
console.log(player.color) // undefined
```

“this” Keyword

```
const player = {
    health: 30,
    color: "red",
    sayColor: function() {
        console.log(this.color)
    }
    takeDamage() {
        this.health -= 10
    }
}
```

```
player.sayColor() // "red"
player.takeDamage()

console.log(player.health)
// 20
```

The “in” Operator

```
const player = {
    health: 30,
    color: "red",
    running: false
}

console.log("color" in player) // true
console.log("name" in player) // false
```

The “For in” loop

```
const player = {
    health: 30,
    color: "red",
    running: false
}

for (let key in player) {
    console.log(key, player[key])
}
```

Console output:

```
// health 30
// color red
// running false
```

Object references

```
const player = {
    health: 30,
    color: "red",
    running: false
}

const player2 = player
player2.health = 20

console.log(player.health) // 20
```

```
const a = 1
const b = a

b = 2
console.log(a) // 1
console.log(b) // 2
```

Garbage Collection

```
let player = {
    health: 30,
    color: "red",
    running: false
}

player = null
```

- `player` object is not accessible anymore
- Garbage Collector will remove it from memory

Methods on Primitives

```
const myString = "Hello World"
myString.length // 11
myString.toUpperCase() // "HELLO WORLD"
myString.toLowerCase() // "hello world"

const myNumber = 123
myNumber.toString() // "123"
myNumber.toFixed(2) // "123.00"

true.toString() // "true"
```

Arrays are Objects

```
const myArray = ["a", "b", "c", "d"]

const myFakeArray = {
    0: "a",
    1: "b",
    2: "c",
    3: "d",
    length: 4
}
console.log(typeof myArray) // "object"
```

Arrays are Objects

```
const myArray = ["a", "b", "c", "d"]

const notEmpty = '0' in myArray // true
myArray[0] // "a"
myArray[10] = "k"

for(let index in myArray) {
    console.log(index, myArray[index])
}
// 0 "a", 1 "b", 2 "c", 3 "d", 10 "k"
```

Methods on Arrays (Stack)

```
const myArray = ["a", "b", "c", "d"]

myArray.push("e") // 5

console.log(myArray) // ["a", "b", "c", "d", "e"]

const last = myArray.pop() // last = "e"

console.log(myArray) // ["a", "b", "c", "d"]
```

Methods on Arrays (Queue)

```
const myArray = ["a", "b", "c", "d"]

myArray.push("e") // 5

console.log(myArray) // ["a", "b", "c", "d", "e"]

const first = myArray.shift() // first = "a"

console.log(myArray) // ["b", "c", "d", "e"]
```

Methods on Arrays (at)

```
const myArray = ["a", "b", "c", "d"]

myArray.at(0) // "a"
myArray.at(1) // "b"

myArray[myArray.length - 1] // "d"

myArray.at(-1) // "d"
myArray.at(-2) // "c"
```

Tasks and Points

Goal is to get 100 Points.

- `basic-arrays` (25 Points)
- `basic-functions` (25 Points)
- `default-arguments` (25 Points)
- `expressions-arrows` (25 Points)
- `return-values` (25 Points)
- `shop` (50 Points)
- `tic-tac-toe` (50 Points)