Roteiro Didático de Aula
Aula 03 – JavaScript
23/09/2020

Assuntos que serão abordados:

ECMA 6

ECMAScript x JavaScript x ES

var x let

Referências

ECMA 6

ES6, ECMAScript 6 ou ES2015, é simplesmente a mais nova versão do JavaScript.
Na verdade, o nome mais usado atualmente é ES2015. A ideia do comitê responsável (conhecido como TC39) pelas atualizações da linguagem é justamente fazer um release anual. Então nesse ano teremos o ES2016 (ou ES7). E assim sucessivamente.

O TC39 focou em alguns objetivos no desenvolvimento do ES6:

- Ser uma linguagem melhor para construir aplicações complexas
- Resolver problemas antigos do JavaScript
- Facilidade no desenvolvimento de libraries

ECMAScript x JavaScript x ES

Uma dúvida bem comum é o porquê dessa mudança do nome.
Na verdade não houve nenhuma mudança: JavaScript é como nós chamamos a linguagem, só que esse nome é um trademark da Oracle (que veio após a compra da Sun). O nome oficial da linguagem é ECMAScript. E ES é simplesmente uma abreviação do mesmo.

var x let

A diferença principal entre o var e o let é que enquanto o primeiro tem escopo de função, o segundo possui escopo de bloco:

```
// escopo de função com var
function doSomething() {
  var a = 1;
  if (true) {
    var b = 2; // b é declarado dentro do if mas é visível fora
  }
  var c = a + b; // 3
}
```


Prof. Caique Zaneti Kirilo & Prof. Priscila Medeiro
ESCOLA SENAI DE INFORMÁTICA - Al. Barão de Limeira, 539 - Santa Cecília - São Paulo/SP - CEP 01202-001 Telefone: (11) 3273-5000 |E-mail:informatica@sp.senai.br

```
//escopo de bloco com let
function doSomethingElse() {
  let a = 1
  if (true) {
    let b = 2 // b é declarado dentro do if e não é visível fora
  }
  let c = a + b // Uncaught ReferenceError: b is not defined
}
```

let x const

const funciona de forma semelhante. A única diferença é que as variáveis criadas não podem ser reatribuídas:

```
let a = 1
a = 2

const b = 1
b = 2 // Uncaught SyntaxError "b" is read-only
```

É comum achar que const deixa a variável imutável, assim como algumas libs como ImmutableJS. Isso não é verdade. As propriedades de um objeto, por exemplo, podem ser alteradas:

```
const object = {
  property: 1
}

object.property = 2
console.log(object.property) // 2
```

De acordo com as boas praticas devemos declarar todas as variáveis com const e quando a variável precisa ser reatribuída, e somente nesse caso, devemos usar o let. Não devemos usar o var (praticamente) nunca.

Exercício 1 - Faça um Programa que peça o valor da gasolina e do álcool de um posto e diga qual é o combustível mais vantajoso abastecer, sabendo que somente é vantagem abastecer álcool se o preço do mesmo é menor ou igual a 70% do valor da gasolina.

Parâmetro de funções

Algumas pequenas alterações foram adicionadas em relação a parametrização de funções. Apesar dessas mudanças serem pequenas, elas trazem enormes benefícios.

default parameters

Os parâmetros de funções têm undefined como valor default. Porém, em alguns casos, pode ser necessário utilizar um outro valor. O ES6 introduziu uma nova forma, bem mais simples, de se fazer isso. Basta adicionar o valor default na definição do parâmetro desejado:

Prof. Caique Zaneti Kirilo & Prof. Priscila Medeiro
ESCOLA SENAI DE INFORMÁTICA - Al. Barão de Limeira, 539 - Santa Cecília - São Paulo/SP - CEP 01202-001 Telefone: (11) 3273-5000 |E-mail:informatica@sp.senai.br

```
const multiply = (x, y = 1) => {
  return x * y
}

multiply(3, 2) // 6
multiply(3) // 3
```

Ou então, com apenas uma linha:

```
const multiply = (x, y = 1) => x * y

multiply(3, 2) // 6
multiply(3) // 3
```

rest parameters
Na versão ES5 do JavaScript podemos utilizar o objeto arguments para pegar todos os parâmetros de uma função:

```
var sum = function() {
    var result = 0;
    for (var i=0; i < arguments.length; i++) {
        result += arguments[i];
    }
    return result;
}

sum(1, 2, 3, 4, 5); // 15
```

Fluxo da função
result = result + argumentos
0 = 0 + 1
1 = 1 + 2
3 = 3 + 3
6 = 6 + 4
10 = 10 + 5
15<<<<

O arguments porém, apresenta alguns problemas:

1. O objeto parece com um array, mas não é exatamente um
2. Todos os parâmetros da função são automaticamente atribuídos ao arguments. Não temos uma forma clara de diferenciar os parâmetros.

Prof. Caique Zaneti Kirilo & Prof. Priscila Medeiro
ESCOLA SENAI DE INFORMÁTICA - Al. Barão de Limeira, 539 - Santa Cecília - São Paulo/SP - CEP 01202-001 Telefone: (11) 3273-5000 |E-mail:informatica@sp.senai.br

Com esses problemas em mente, os Rest Parameters foram adicionados no ES6. O mesmo exemplo da soma poderia ser reescrito dessa forma:

```
function sum(...numbers) {
  let result = 0
  numbers.forEach((number) => {
    result += number
  })
  return result
}

sum(1, 2, 3, 4, 5) // 15
```

Ou dessa forma mais funcional:

```
const sum = (...numbers) =>
    numbers.reduce((acc, current) => acc + current, 0)

sum(1, 2, 3, 4, 5) // 15
```

Programação Funcional

arrow functions

Os arrow functions são um excelente syntax sugar (atalho visualmente atraente) na criação de funções. Uma função que seria escrita dessa forma em ES5:

```
var sum = function(x, y) {
  return x + y;
};

sum(1, 2); // 3
```

Pode ser escrita dessa forma em ES6 com o uso das arrow functions

```
const sum = (x, y) => {
  return x + y
}

sum(1, 2) // 3
```

Dessa forma já conseguimos ver uma maior expressividade e um menor número total de caracteres. Mas podemos melhorá-la ainda mais:

```
const sum = (x, y) => x + y
```


Prof. Caique Zaneti Kirilo & Prof. Priscila Medeiro
ESCOLA SENAI DE INFORMÁTICA - Al. Barão de Limeira, 539 - Santa Cecília - São Paulo/SP - CEP 01202-001 Telefone: (11) 3273-5000 |E-mail:informatica@sp.senai.br

```
sum(1, 2) // 3
```

Com funções de apenas uma linha, podemos simplesmente omitir o return e as chaves.
Mas o verdadeiro benefício das arrows functions não está na expressividade, ele se encontra na resolução de um antigo problema da linguagem: o this.

No exemplo abaixo em ES5, podemos observar o this sendo utilizado de forma errada:

```
function Widget() {
    var button = document.getElementById('button');
    button.addEventListener('click', function() {
        this.doSomething(); // o 'this' não aponta para Widget como esperado e provocará um erro.
    });
}
```

Uma das formas mais comuns de resolver esse problema, é usando o bind() ou então com self = this. Com as arrow functions isso não é necessário. O this funcionará exatamente da forma esperada:

```
function Widget() {
    const button = document.getElementById('button')
    button.addEventListener('click', () => {
        this.doSomething() // o 'this' aponta para Widget e não provocará nenhum erro.
    })
}
```

destructuring

Uma nova forma de declarar variáveis extraindo valores de objetos e arrays é através do destructuring. Ela funciona dessa forma:

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

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

E com rest parameters:

```
const [a, b, ...rest] = [1, 2, 3, 4, 5]

console.log(a) // 1
console.log(b) // 2
console.log(rest) // 3, 4, 5
```


Prof. Caique Zaneti Kirilo & Prof. Priscila Medeiro
ESCOLA SENAI DE INFORMÁTICA - Al. Barão de Limeira, 539 - Santa Cecília - São Paulo/SP - CEP 01202-001 Telefone: (11) 3273-5000 |E-mail:informatica@sp.senai.br

Exercicio 2 - Faça um Programa que peça as 4 notas bimestrais e mostre a média.

Referências

Lima, M. (s.d.). *Guia do ES6*. Fonte: Medium: https://medium.com/@matheusml/o-guia-do-es6-tudo-que-voc%C3%AA-precisa-saber-8c287876325f

MDN. (s.d.). *https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Fun%C3%A7%C3%B5es*. Fonte: MDN web docs.

Prof. Caique Zaneti Kirilo & Prof. Priscila Medeiro
ESCOLA SENAI DE INFORMÁTICA - Al. Barão de Limeira, 539 - Santa Cecília - São Paulo/SP - CEP 01202-001 Telefone: (11) 3273-5000 |E-mail:informatica@sp.senai.br