
# HTML5, CSS3 E JAVASCRIPT

JEMERSON FERNANDO MAIA

**PÓS-GRADUAÇÃO EM DESENVOLVIMENTO DE APLICAÇÕES WEB E MOBILE**

UNIPAR – UNIVERSIDADE PARANAENSE

# EMENTA

HTML5 e CSS3; Fundamentos de HTML5 e CSS3; Responsividade com Media Queries; Seletores; SASS; UX; Javascript; ES6;

# AGENDA

**27/03:** HTML5, Modernizr, HTML5 Boilerplate, Formulários e Validações, CSS3, Normalizer.css, Media Query, Framework Fron-End Bootstrap

**10/04:** SASS, JS, ES6, 2016, 2017, 2018, JQuery

**24/04:** API HTML5, UX, Acessibilidade, Testes de Responsividade, Performance, Admin Templates

# Ferramentas

- Sass
  - https://sass-lang.com/

# Referências

https://www.w3schools.com
https://sass-lang.com/documentation
https://www.w3schools.com/js/js_es6.asp

# Sass

Syntactically Awesome StyleSheets

# Sass

É um pré-processador CSS.

É uma extensão do CSS que adiciona potência e elegância ao CSS nativo.

Less e Stylus

SCSS

```scss
section {
    height: 100px;
    width: 100px;

    .class-one {
        height: 50px;
        width: 50px;

        .button {
            color: #074e68;
        }
    }
}
```

CSS

```css
section {
    height: 100px;
    width: 100px;
}

section .class-one {
    height: 50px;
    width: 50px;
}

section .class-one .butto
    color: #074e68;
}
```

Sass - Syntactically Awesome StyleSheets

```sass
$font-stack:    Helvetica, sans-serif
$primary-color: #333

body
  font: 100% $font-stack
  color: $primary-color
```

sass

```scss
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

scss

# Arquivo .sass vs .scss

# Instalação

- https://github.com/sass/dart-sass/releases, (add PATH)
- npm install -g sass (node.js)
- choco install sass (chocolatey – Windows)
- brew install sass/sass/sass (mac)
- https://scout-app.io/ (App)

https://sass-lang.com/install

sass --version

- Live Sass Compiler (VSCode)
- Ruby (gem install sass)

# Execução

```
sass source/stylesheets/index.scss build/stylesheets/index.css
```

```
sass --watch source/stylesheets/index.scss: build/stylesheets/index.css
```

```
sass --watch app/sass:public/stylesheets
```

```
$ sass --style=nested
h1 {
  font-size: 40px; }
  h1 code {
    font-face: Roboto Mono; }

$ sass --style=expanded style.scss
h1 {
  font-size: 40px;
}
h1 code {
  font-face: Roboto Mono;
}

$ sass --style=compact style.scss
h1 { font-size: 40px; }
h1 code { font-face: Roboto Mono; }

$ sass --style=compressed style.scss
h1{font-size:40px}h1 code{font-face:Roboto Mono}
```

# Saídas

# Variáveis

Permite definir valores de propriedades de forma dinâmica.

# Variáveis

```
$myFont: Helvetica, sans-serif;
$myColor: red;
$myFontSize: 18px;
$myWidth: 680px;

body {
  font-family: $myFont;
  font-size: $myFontSize;
  color: $myColor;
}

#container {
  width: $myWidth;
}
```

```
body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: red;
}

#container {
  width: 680px;
}
```

# Variáveis - Escopos

```
$myColor: red;

h1 {
  $myColor: green;
  color: $myColor;
}

p {
  color: $myColor;
}
```

```
h1 {
  color: green;
}

p {
  color: red;
}
```

# Aninhamentos

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
  li {
    display: inline-block;
  }
  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

# Aninhamentos em propriedades

```
font: {
  family: Helvetica, sans-serif;
  size: 18px;
  weight: bold;
}

text: {
  align: center;
  transform: lowercase;
  overflow: hidden;
}
```

```
font-family: Helvetica, sans-serif;
font-size: 18px;
font-weight: bold;

text-align: center;
text-transform: lowercase;
text-overflow: hidden;
```

# Aninhamentos em propriedades

```
span{
    text:{
        align:center;
        indent: 20px;
    }

    font:{
        family: 'Helvetica';
        size: 30px;
    }

    border:{
        top:{
            style:dashed;
            left:{
                radius: 10px;
            }
        }
    }
}
```

```
span {
  text-align: center;
  text-indent: 20px;
  font-family: 'Helvetica';
  font-size: 30px;
  border-top-style: dashed;
  border-top-left-radius: 10px; }
```

# @Mixin

Permitem a construção de blocos de código independentes, permitindo misturá-los para criar uma grande folha de estilo

# @Mixin

```
@mixin important-text {
  color: red;
  font-size: 25px;
  font-weight: bold;
  border: 1px solid blue;
}

      .danger {
        @include important-text;
        background-color: green;
      }
```

```
.danger {
  color: red;
  font-size: 25px;
  font-weight: bold;
  border: 1px solid blue;
  background-color: green;
}
```

# @Mixin – Passando Parâmetros

```
@mixin bordered($color, $width) {
  border: $width solid $color;
}

.myArticle {
  @include bordered(blue, 1px);  // Call mixin with two values
}

.myNotes {
  @include bordered(red, 2px); // Call mixin with two values
}
```

```
.myArticle {
  border: 1px solid blue;
}

.myNotes {
  border: 2px solid red;
}
```

# @Mixin – Uso para Prefixos

```
@mixin transform($property) {
  -webkit-transform: $property;
  -ms-transform: $property;
  transform: $property;
}

.myBox {
  @include transform(rotate(20deg));
}
```

```
.myBox {
  -webkit-transform: rotate(20deg);
  -ms-transform: rotate(20deg);
  transform: rotate(20deg);
}
```

# @extend

Permite que um seletor herde propriedades e valores declarados em outro

# @extend

```
.button-basic  {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.button-report  {
  @extend .button-basic;
  background-color: red;
}

.button-submit  {
  @extend .button-basic;
  background-color: green;
  color: white;
}
```

```
.button-basic, .button-report, .button-submit {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.button-report  {
  background-color: red;
}

.button-submit  {
  background-color: green;
  color: white;
}
```

# @import

Permite particionar a formatação CSS em arquivos para reaproveita-los em vários arquivos.

# @import

_reset.scss

```
html,
body,
ul,
ol {
  margin: 0;
  padding: 0;
}
```

```
@import "reset";

body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: red;
}
```

```
html, body, ul, ol {
  margin: 0;
  padding: 0;
}

body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: red;
}
```

# @import

Reset/reboot

variáveis

colores

fontes

Componentes

Módulos

Vendors

Páginas

# Exemplos

# Atividade

# Sass - Módulos Integrados

Sass provê muitos módulos integrados que contém funções úteis.

https://sass-lang.com/documentation/modules

# Sass- Regras de controle de Fluxo

Com Sass é possível controlar se estilos são gerados ou gera-los várias vezes com variações. Permite criar pequenos algoritmos para gerar o estilo css.

https://sass-lang.com/documentation/at-rules/control

# Sass- Compass

O Compass fornece funções extras de cores e uma série de mixins como reset, clear-fix, truncate e montagem de gradientes.

http://compass-style.org/

# JavaScript

Permite interatividade com conteúdo.

# Como usar

- Interna
  - Tag script dentro do documento
    - Geralmente dentro da tag head ou como último elemento dentro da tag body

# Como usar

```html
<!DOCTYPE html>
<html>

<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>

<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<body>

<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

# Como usar

- Externa
  - Arquivo externo ao html
  - Arquivo com extensão js

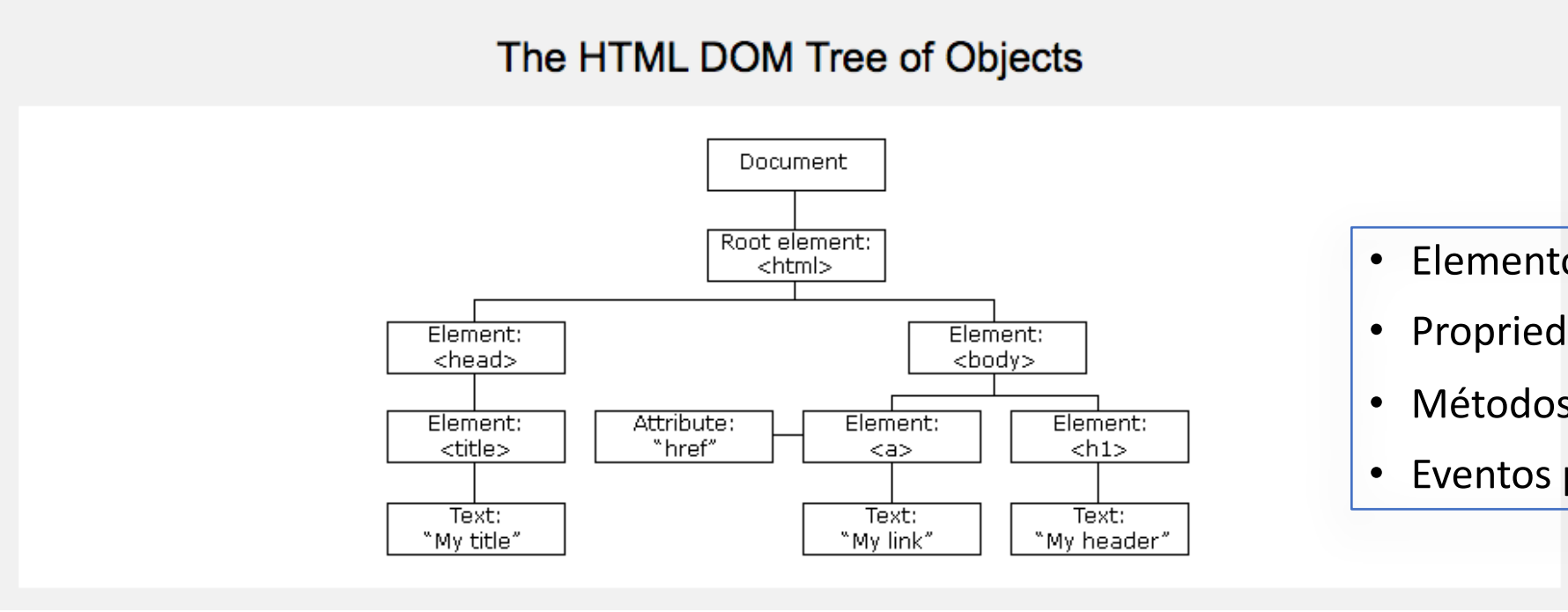
```
<script src="myScript1.js"></script>
<script src="myScript2.js"></script>
```

```
<script src="/js/myScript1.js"></script>
```

```
<script src="https://www.w3schools.com/js/myScript1.js"></script>
```

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

# Manipulação de HTML




- Elementos HTML como objetos
- Propriedades de todos os elementos
- Métodos para acessar os elementos
- Eventos para todos os elementos

# Manipulação de HTML

## Finding HTML Elements

| Method | Description |
| --- | --- |
| document.getElementById(*id*) | Find an element by element id |
| document.getElementsByTagName(*name*) | Find elements by tag name |
| document.getElementsByClassName(*name*) | Find elements by class name |

# Manipulação de HTML

```
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
```

```
var x = document.getElementsByClassName("intro");
```

es5

```
var x = document.querySelectorAll("p.intro");
```

# Manipulação de HTML

## Changing HTML Elements

| **Property** | **Description** |
|---|---|
| *element*.innerHTML = *new html content* | Change the inner HTML of an element |
| *element.attribute = new value* | Change the attribute value of an HTML element |
| *element*.style.*property = new style* | Change the style of an HTML element |
| **Method** | **Description** |
| *element*.setAttribute*(attribute, value)* | Change the attribute value of an HTML element |

# Manipulação de HTML

## Adding and Deleting Elements

| Method | Description |
| --- | --- |
| document.createElement(*element*) | Create an HTML element |
| document.removeChild(*element*) | Remove an HTML element |
| document.appendChild(*element*) | Add an HTML element |
| document.replaceChild(*new, old*) | Replace an HTML element |
| document.write(*text*) | Write into the HTML output stream |

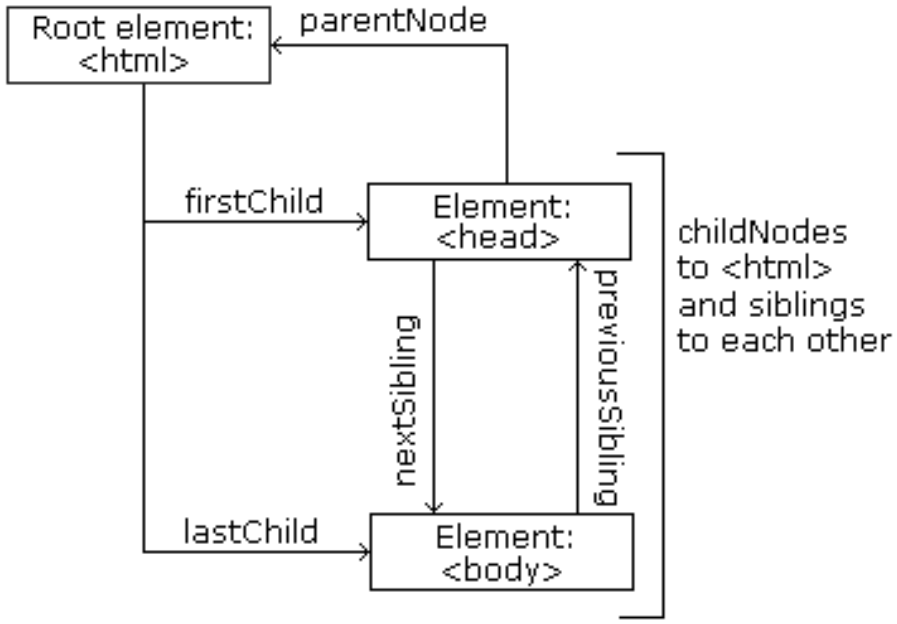
# Manipulação de HTML

```
<html>

  <head>
    <title>DOM Tutorial</title>
  </head>

  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>

</html>
```



# Manipulação de HTML

## Navigating Between Nodes

You can use the following node properties to navigate between nodes with JavaScript:

- `parentNode`
- `childNodes[nodenumber]`
- `firstChild`
- `lastChild`
- `nextSibling`
- `previousSibling`

# Manipulação de HTML- Eventos

- JavaScript podem ser executados quando eventos acontecem, exemplos:
  - Quando usuário clica com mouse
  - Quando uma página é carregada
  - Quando uma imagem é carregada
  - Quando mouse é movido sobre elementos
  - Quando campos de entrada em formulários são alterados
  - Quando formulários tem seus dados submetidos
  - Quando usuário pressiona uma tecla

https://www.w3schools.com/jsref/dom_obj_event.asp

# Variáveis e Saídas

```
var x = 5;
var y = 6;
var z = x + y;
```

```
<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>
```

```
<script>
document.write(5 + 6);
</script>
```

```
<script>
window.alert(5 + 6);
</script>
```

```
<script>
console.log(5 + 6);
</script>
```

# Exemplos

# JQuery

É uma biblioteca que simplifica a programação em JS.

**“Escrever menos, fazer mais”**

# Como usar

https://jquery.com/

```
<head>
<script src="jquery-3.5.1.min.js"></script>
</head>
```

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
```

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="my_jquery_functions.js"></script>
</head>
```

# Sintaxe

**$(*selector*).*action*()**

```
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
```

# Sintaxe - Seletores

```
$("p")

$("#test")

$(".test")
```

https://www.w3schools.com/jquery/jquery_selectors.asp

# Eventos

```
$("p").click(function(){
  $(this).hide();
});
```

| Mouse Events | Keyboard Events | Form Events | Document/Window Events |
|---|---|---|---|
| click | keypress | submit | load |
| dblclick | keydown | change | resize |
| mouseenter | keyup | focus | scroll |
| mouseleave | | blur | unload |

# text(), html(), val() e attr()

- text() – define ou retorna os texto contido nos elementos selecionados
- html() – define ou retorna o conteúdo, incluindo marcações html, dos elementos selecionados
- val() – define ou retorna valor de campos selecionados de formulários
- attr() – define ou retorna valor de atributos dos elementos selecionados

```
alert("Text: " + $("#test").text());

alert("HTML: " + $("#test").html());

alert("Value: " + $("#test").val());

alert($("#w3s").attr("href"));
```

# Manipulação HTML- Adicionar

- append() - Insere conteúdo no fim do elemento selecionado
- prepend() - Insere conteúdo no começo do elemento selecionado
- after() - Insere conteúdo depois do elemento selecionado
- before() - Insere conteúdo antes do elemento selecionado

```
function appendText() {
  var txt1 = "<p>Text.</p>";                // Create element with HTML
  var txt2 = $("<p></p>").text("Text.");    // Create with jQuery
  var txt3 = document.createElement("p");   // Create with DOM
  txt3.innerHTML = "Text.";
  $("body").append(txt1, txt2, txt3);       // Append the new elements
}
```

# Manipulação HTML - Remover

- remove() - Remove o elemento selecionado (e seus filhos)
- empty() - Remove os filhos do elemento selecionado

```
$("p").remove(".test, .demo");
```

# Manipulação HTML- CSS

- addClass() - Adiciona uma ou mais classes no elemento selecionado
- removeClass() - Remove uma ou mais classes do elemento selecionado
- toggleClass() - Alterna entre adicionar e remover classes no elemento selecionado
- css() - Adiciona ou recupera atributos de estilo

```
$("p").css("background-color");

  $("p").css("background-color", "yellow");
```

```
$("button").click(function(){
  $("h1, h2, p").addClass("blue");
  $("div").addClass("important");
});
```

# Manipulação HTML- Ancestrais

- parent() – Retorna o elemento pai do elemento selecionado
- parents() – Retorna todos os elementos ancestrais do elemento selecionado
- parentsUntil() – Retorna todos os elementos ancestrais entre dois elementos (selecionado e outro)

```
$(document).ready(function(){
  $("span").parentsUntil("div");
});
```

# Manipulação HTML- Descendentes

- children() – Retorna todos os filhos diretos do elemento selecionado
- find() – Retorna todos os elementos descendentes do elemento selecionado

```
$(document).ready(function(){
  $("div").children("p.first");
});
```

```
$(document).ready(function(){
  $("div").find("span");
});
```

# Manipulação HTML - Irmãos

- siblings() – Retorna todos os irmãos do elemento selecionado
- next() – Retorna o próximo elemento irmão do elemento selecionado
- nextAll() – Retorna todos os próximos elementos irmãos do elemento selecionado
- nextUntil() – Retorna todos os próximos elementos irmãos entre dois elementos
- prev() – Retorna o elemento anterior irmão do elemento selecionado
- prevAll() – Retorna todos os elementos anteriores irmãos do elemento selecionado
- prevUntil() – Retorna todos os elementos anteriores irmãos entre dois elementos

# Manipulação HTML- Filtrar

- first() – Filtra retornando o primeiro elemento informado
- last() – Filtra retornando o último elemento informado
- eq() – Filtra retornando o elemento do índice informado
- filter() – Filtra retornando os elementos encontrados de acordo com um critério de busca
- not() – Filtra retornando os elementos que não satisfazem o critério de busca

```
$(document).ready(function(){
  $("p").filter(".intro");
});
```

# Exemplos

# Atividade

- Adicionar coluna na tabela contendo link em cada linha que ao clicar remove a linha toda da tabela;

# ES6 – ECMAScript 6 ou ES2015

O ECMAScript (ES) é a especificação da linguagem de script que o JavaScript implementa.

https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

# Versões

- As versões foram abreviadas para ES1, ES2, ES3, ES5, and ES6.
- Desde 2016 as novas versões são nomeadas pelo ano (ECMAScript 2016 / 2017 / 2018 / 2019 /2020).
- ES1 foi lançada em 1997
- ES2 em 1998
- ES3 em 1999
- ES4 nunca foi lançado
- **ES5 em 2009**
- **ES6 em 2015**

# ES5- "use strict"

- O modo “strict” transforma a "sintaxe incorreta" anteriormente aceita em erros reais.

```
"use strict";
x = 3.14;       // This will cause an error because x is not declared
```

# ES5- Array.forEach()

- Percorre o Array chamando uma função para cada elemento do Array
- Passa os parâmetros:
  - Valor
  - Índice
  - Array

```
var txt = "";
var numbers = [45, 4, 9, 16, 25];
numbers.forEach(myFunction);

function myFunction(value, index, array) {
  txt = txt + value + "<br>";
}
```

# ES5- Array.map()

- Cria novo Array executando uma função para cada elemento do Array
- Passa os parâmetros :
  - Valor
  - Índice
  - Array

```
var numbers1 = [45, 4, 9, 16, 25];
var numbers2 = numbers1.map(myFunction);

function myFunction(value, index, array) {
  return value * 2;
}
```

# ES5- Array.filter()

- Cria novo Array executando uma função para cada elemento do Array
- Novo Array contém somente valores que passarem no “teste”
- Passa os parâmetros:
  - Valor
  - Índice
  - Array

```
var numbers = [45, 4, 9, 16, 25];
var over18 = numbers.filter(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

# ES5- Array.reduce() e Array.reduceRight()

- Reduce
  - Reduz o Array a um único valor
  - Percorre o Array da esquerda para direita
  - Passa os parâmetros:
    - Total (será retornado)
    - Valor
    - Índice
    - Array
  - Permite passar valor inicial
- ReduceRight
  - Percorre o Array da direita para esquerda

```
var numbers1 = [45, 4, 9, 16, 25];
var sum = numbers1.reduce(myFunction);

function myFunction(total, value, index, array) {
  return total + value;
}
```

```
var numbers1 = [45, 4, 9, 16, 25];
var sum = numbers1.reduce(myFunction, 100);

function myFunction(total, value) {
  return total + value;
}
```

# ES5- Array. every()

- Verifica se todos os elementos do Array passam em um “teste”
- Passa os parâmetros:
  - Valor
  - Índice
  - Array

```
var numbers = [45, 4, 9, 16, 25];
var allOver18 = numbers.every(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

# ES5- Array. some()

- Verifica se algum elemento do Array passam em um “teste”
- Passa os parâmetros:
  - Valor
  - Índice
  - Array

```
var numbers = [45, 4, 9, 16, 25];
var someOver18 = numbers.some(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

# JSON.parse() e JSON.stringify()

- Parse
  - Converte texto contento JSON em objeto JSON

```
var obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');
```

- Stringify
  - Converte um objeto JSON em texto;

```
var myJSON = JSON.stringify(obj);
```

# ES6 – let e const

- Para declarar escopo de bloco utiliza-se **let**

```
{
  var x = 2;
}
// x CAN be used here
```

```
{
  let x = 2;
}
// x can NOT be used here
```

```
var x = 10;
// Here x is 10
{
  var x = 2;
  // Here x is 2
}
// Here x is 2
```

```
var x = 10;
// Here x is 10
{
  let x = 2;
  // Here x is 2
}
// Here x is 10
```

# ES6 – let e const

- Para declarar constantes utiliza-se **const**
- Obedece mesma regra de escopo de let

```
const PI = 3.141592653589793;
PI = 3.14;      // This will give an error
PI = PI + 10;   // This will also give an error
```

```
var x = 10;
// Here x is 10
{
  const x = 2;
  // Here x is 2
}
// Here x is 10
```

# ES6 – Arrow Functions

- Permite escrever functions com uma sintaxe mais curta.

```
// ES5
var x = function(x, y) {
    return x * y;
}

// ES6
const x = (x, y) => x * y;
```

# ES6 – For/of

- Permite fazer a iteração de elementos.
- Diferente do *for/in* que permite acessar o elemento através de uma chave.

```
var cars = ["BMW", "Volvo", "Mini"];
var x;

for (x of cars) {
  document.write(x + "<br >");
}
```

```
var txt = "JavaScript";
var x;

for (x of txt) {
  document.write(x + "<br >");
}
```

# ES6 – Classes

- Permite criação de classes (templates de objetos)

```
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
}

let myCar1 = new Car("Ford", 2014);
let myCar2 = new Car("Audi", 2019);
```

```
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
  age() {
    let date = new Date();
    return date.getFullYear() - this.year;
  }
}

let myCar = new Car("Ford", 2014);
document.getElementById("demo").innerHTML =
"My car is " + myCar.age() + " years old.";
```

# ES6 – Promises

- É um objeto JavaScript que conecta “Código de Produção” e “Código de Consumo”
- “Código de produção” pode levar alguma tempo para finalizar e o “código de consumo” aguarda esse resultado;
- Recebe duas funções, uma quando “Código de Produção” for executado com sucesso e outra quando for rejeitada.

```
let myPromise = new Promise(function(myResolve, myReject) {
  setTimeout(function() { myResolve("I love You !!"); }, 3000);
});

myPromise.then(function(value) {
  document.getElementById("demo").innerHTML = value;
});
```

# ES6 – Promises

```
const show = (data) => document.getElementById("demo").innerHTML = data;

const showError = (data) => show("Erro: " + data);

let divide = (a,b) => new Promise((resolve, reject) => {
  setTimeout(() => {
    if (b == 0) reject('Divisor não pode ser zero!');

    resolve(a / b);
  }, 2000);
});

divide(10,2).then(show, showError);
```

# ES6 – Parâmetro Default

- Permite que funções tenham parâmetro default

```
function myFunction(x, y = 10) {
  // y is 10 if not passed or undefined
  return x + y;
}
myFunction(5); // will return 15
```

# ES6 – Parâmetro Rest (...)

- Permite passar vários parâmetros como argumento de forma indefinida.
- Parâmetros são tratados como Array

```
function sum(...args) {
  let sum = 0;
  for (let arg of args) sum += arg;
  return sum;
}

let x = sum(4, 9, 16, 25, 29, 100, 66, 77);
```

# ES6 – Array.find() e Array.findIndex()

- Array.find()
  - Retorna primeiro elemento do Array que passe em um “teste”
  - Passa os parâmetros:
    - Valor
    - Índice
    - Array
- Array.findIdex
  - Retorna índice do primeiro elemento do Array que passe em um “teste”

```
var numbers = [4, 9, 16, 25, 29];
var first = numbers.find(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

# ES2016

- JavaScript Exponentiation (**)
- JavaScript Exponentiation assignment (**=)
- JavaScript Array.prototype.includes

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.includes("Mango"); // is true
```

# ES2017

- JavaScript String padding
- JavaScript Object.entries
- JavaScript Object.values
- JavaScript async functions

```
const success = (data) => document.getElementById("demo").innerHTML = data;

const error = (data) => document.getElementById("demo").innerHTML = data;

async function foo () {
 if (Math.random() > 0.5) return 'yeah'
 throw new Error('ops')
}

foo().then(success).catch(error);
```

# ES2018 – Interação Assíncrona

- Asynchronous Iteration
- Promise Finally
- Object Rest Properties
- New RegExp Features

# Exemplos

# Mais Exemplos

https://www.javascripture.com/