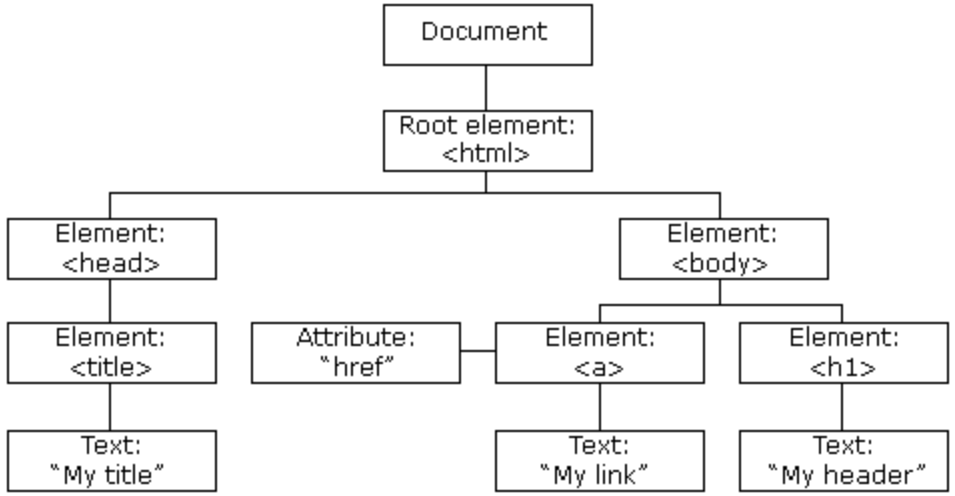
Front-end Essencial JavaScript

Prof. Yuri Weilemann

DOM: a página no mundo Javascript

Quando uma página da web é carregada, o navegador cria um modelo de objeto de documento (do inglês Document Object Model) da página.

O modelo HTML DOM é construído como uma árvore de objetos:



DOM: a página no mundo Javascript

Com o modelo de objeto, o JavaScript obtém todo o poder de que precisa para criar HTML dinâmico, sendo capaz de:

- Alterar todos os elementos HTML na página
- Alterar todos os atributos HTML na página
- Alterar todos os estilos CSS na página
- Remover elementos e atributos HTML existentes
- Adicionar novos elementos e atributos HTML
- Reagir a todos os eventos HTML existentes na página
- Criar novos eventos HTML na página

JavaScript HTML DOM Document

O objeto HTML DOM **document** é o proprietário de todos os outros objetos em sua página da web.
Se você deseja acessar qualquer elemento em uma página HTML, você sempre começa acessando o objeto **document**.
Veremos alguns exemplos, nos próximos slides, de como podemos usar o objeto **document** para acessar e manipular HTML.

Funções e Eventos da DOM

Os métodos/funções HTML DOM são ações que você pode executar (em elementos HTML).
Propriedades HTML DOM são valores (de elementos HTML) que você pode definir ou alterar.
Veremos alguns exemplos à seguir.

Método	Descrição
document.getElementById(id)	Encontra um elemento por id de elemento
element.setAttribute(attribute, value)	Altera o valor do atributo de um elemento HTML
document.createElement(element)	Cria um elemento HTML
document.getElementById(id).onclick = function(){code}	Adicionando código de manipulador de eventos a um evento onclick

Funções e Eventos da DOM

Alguns exemplos de propriedades :

Propriedade	Descrição
element.innerHTML = new html content	Alterar o HTML interno de um elemento
element.attribute = new value	Altere o valor do atributo de um elemento HTML
element.style.property = new style	Altere o estilo de um elemento HTML

O que é JavaScript?

Enquanto o HTML é usado para definir o conteúdo das páginas da web e o CSS para especificar o layout das páginas da web, o JavaScript é a linguagem responsável pela programação do comportamento das páginas da web.

Toda vez que uma página da web faz mais do que simplesmente mostrar a você informação estática — mostrando conteúdo que se atualiza em um intervalo de tempo, mapas interativos ou gráficos 2D/3D animados, etc. — você pode apostar que o JavaScript provavelmente está envolvido.

Os usos comuns do JavaScript são manipulação de imagens, validação de formulários e mudanças dinâmicas de conteúdo.

Características da linguagem

O JavaScript, como o próprio nome sugere, é uma linguagem de scripting. Uma linguagem de scripting é comumente definida como uma linguagem de programação que permite ao programador controlar uma ou mais aplicações de terceiros. No caso do JavaScript, podemos controlar alguns comportamentos dos navegadores através de trechos de código que são enviados na página HTML.

Características da linguagem

Outra característica comum nas linguagens de scripting é que normalmente elas são linguagens interpretadas, ou seja, não dependem de compilação para serem executadas.

Essa característica é presente no JavaScript: o código é interpretado e executado conforme é lido pelo navegador, linha a linha, assim como o HTML.

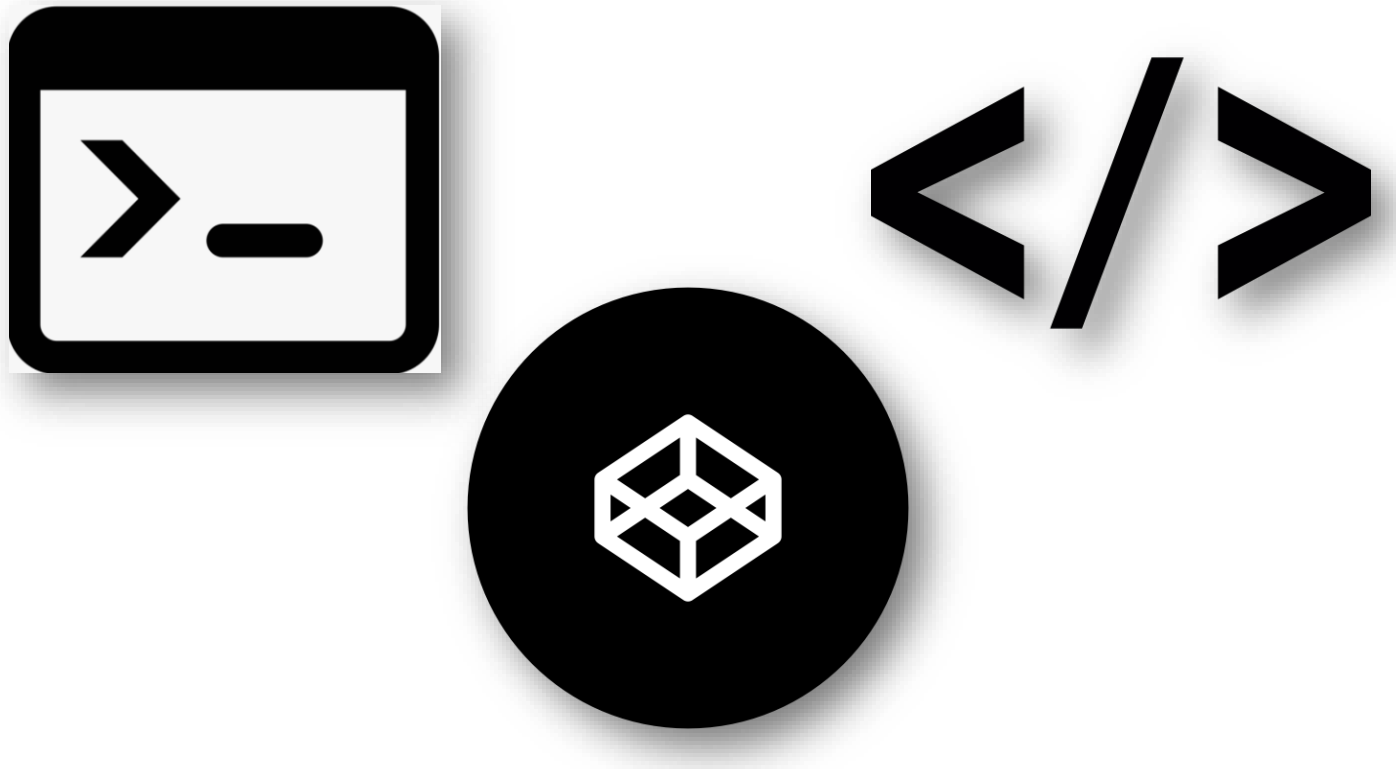
```
<body>
  <h1>JavaScript</h1>

  <button onclick="document.getElementById('exemplo').innerHTML = Date()">
    Clique aqui!
  </button>
  <div style="margin-top: 20px;" id="exemplo"></div>
</body>
```

Formas de Executar JavaScript

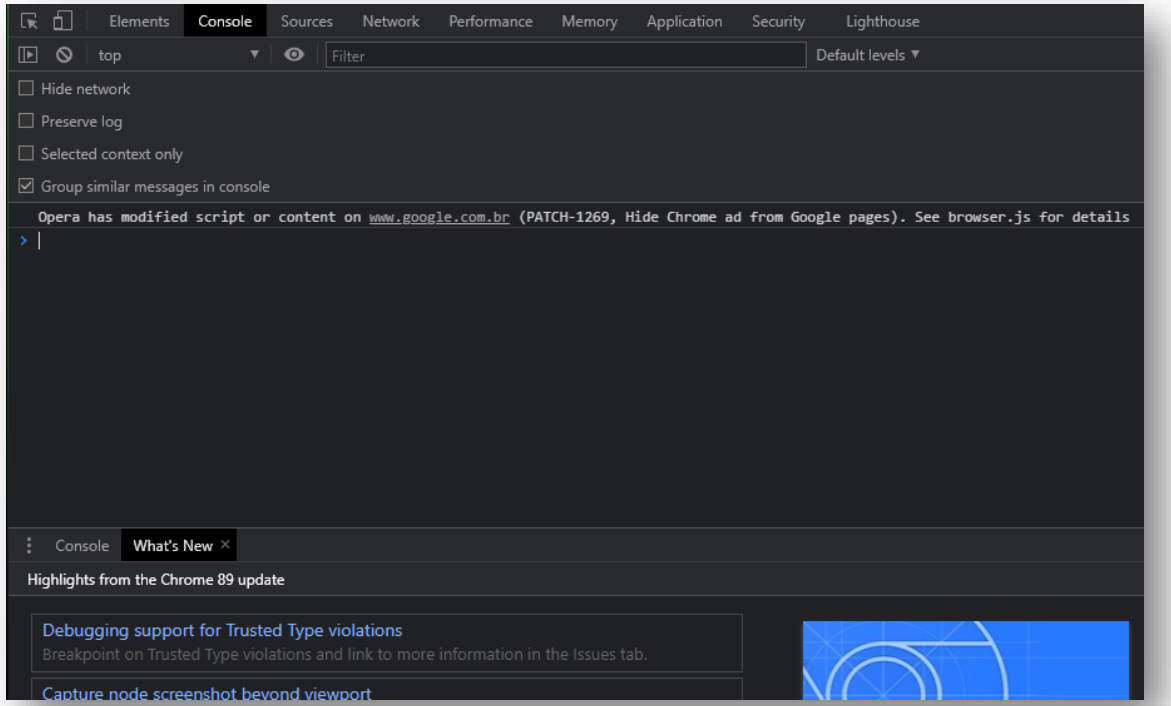
Assim como temos diversas formas importar o CSS para dentro de um documento HTML, temos algumas formas para executar o JavaScript:

- Console do Navegador
- CodePen
- Tag Script

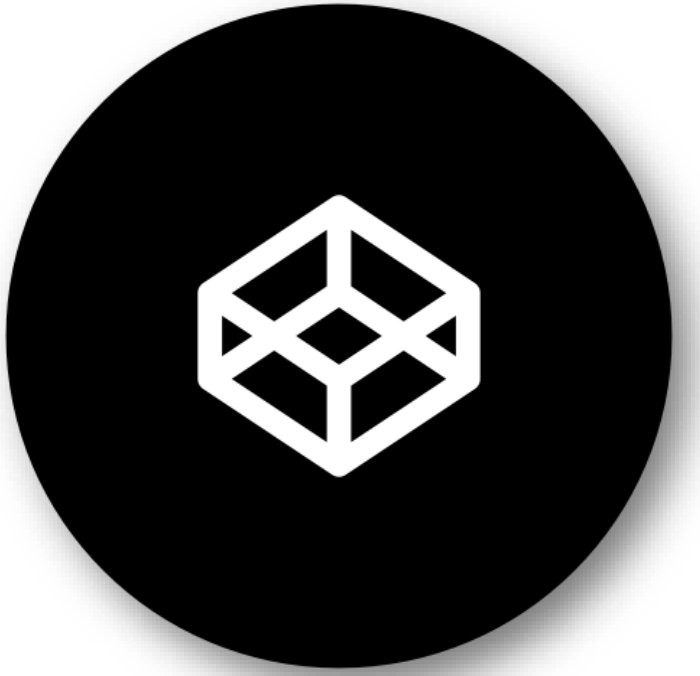


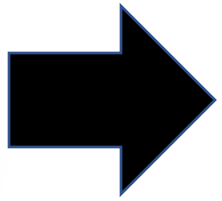
Console do Navegador

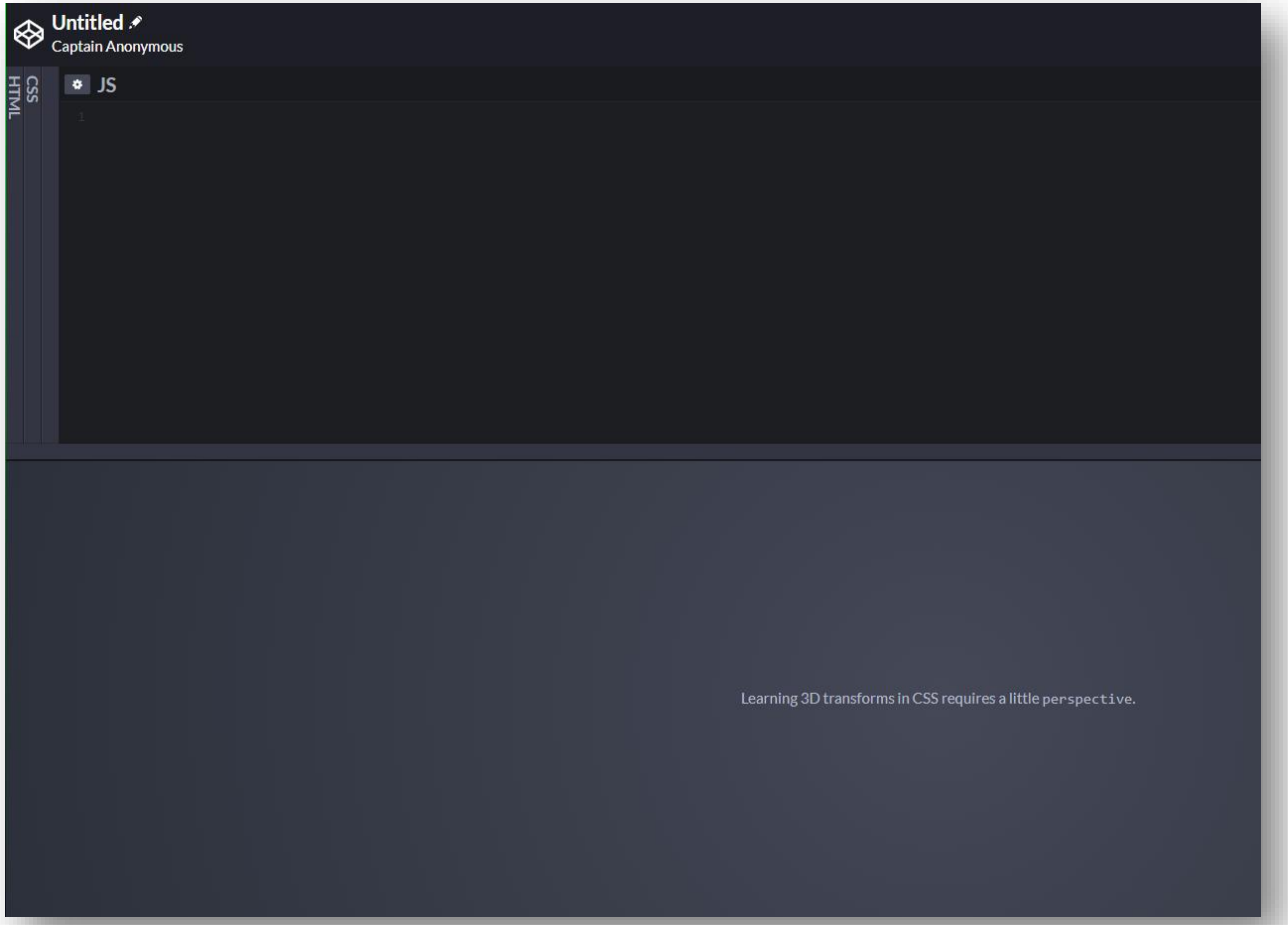
A maioria dos navegadores desktop já vem com essa ferramenta instalada. No Chrome, é possível chegar ao Console apertando F12 e em seguida acessar a aba "Console" ou por meio do atalho de teclado Control + Shift + C; no Firefox, pelo atalho Control + Shift + K.



CodePen.io






Untitled
Captain Anonymous
HTML
CSS
JS
Learning 3D transforms in CSS requires a little perspective.

A Tag Script

A tag <script> é usada para incorporar no código HTML um script do lado do cliente (JavaScript).

O elemento <script> contém instruções de script ou aponta para um arquivo de script externo por meio do atributo src.

```
<script>
  document.getElementById("exemplo")
</script>
```

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

Comentários

Comentários também estão presente no JavaScript e são muito úteis para organização do nosso código. Entretanto, devemos sempre ter cuidado com as informações que inserimos neles.

Para se criar comentários no JavaScript temos basicamente duas formas:

- // - Comentário em linha
- /* */ - Comentário em bloco (multi-linha)

Sintaxe Básica

Vamos olhar pelos exemplos da W3Schools a sintaxe básica do Javascript, que compreende os seus valores (literais e variáveis), operadores e tipos de dados.

Tipos de Dados

Antes de começarmos, efetivamente, a programar em JavaScript, é importante aprendermos alguns dos diferentes tipos de dados que existem.

- String
- Number
- Boolean
- Undefined e Null
- Object
- Array

String

String é uma cadeia de caracteres. Podem ser representados através das aspas simples (' '), aspas duplas (" ") ou template strings (```).

As Template Strings permitem que trabalhemos com strings multi-linhas e utilizemos expressões de linguagem utilizando a formatação ${}.

```
var nome = "Yuri Weilemann"

var frase = `${nome} é um cara muito legal.`
// Yuri Weilemann é um cara muito legal
```

Number

São tipos de dados os quais conseguimos manipular de forma numérica.

```
// Numbers
  33 // Int (Inteiro)
  12.5 // Float (Real)
  NaN // Not a Number
  Infinity // Infinito
```

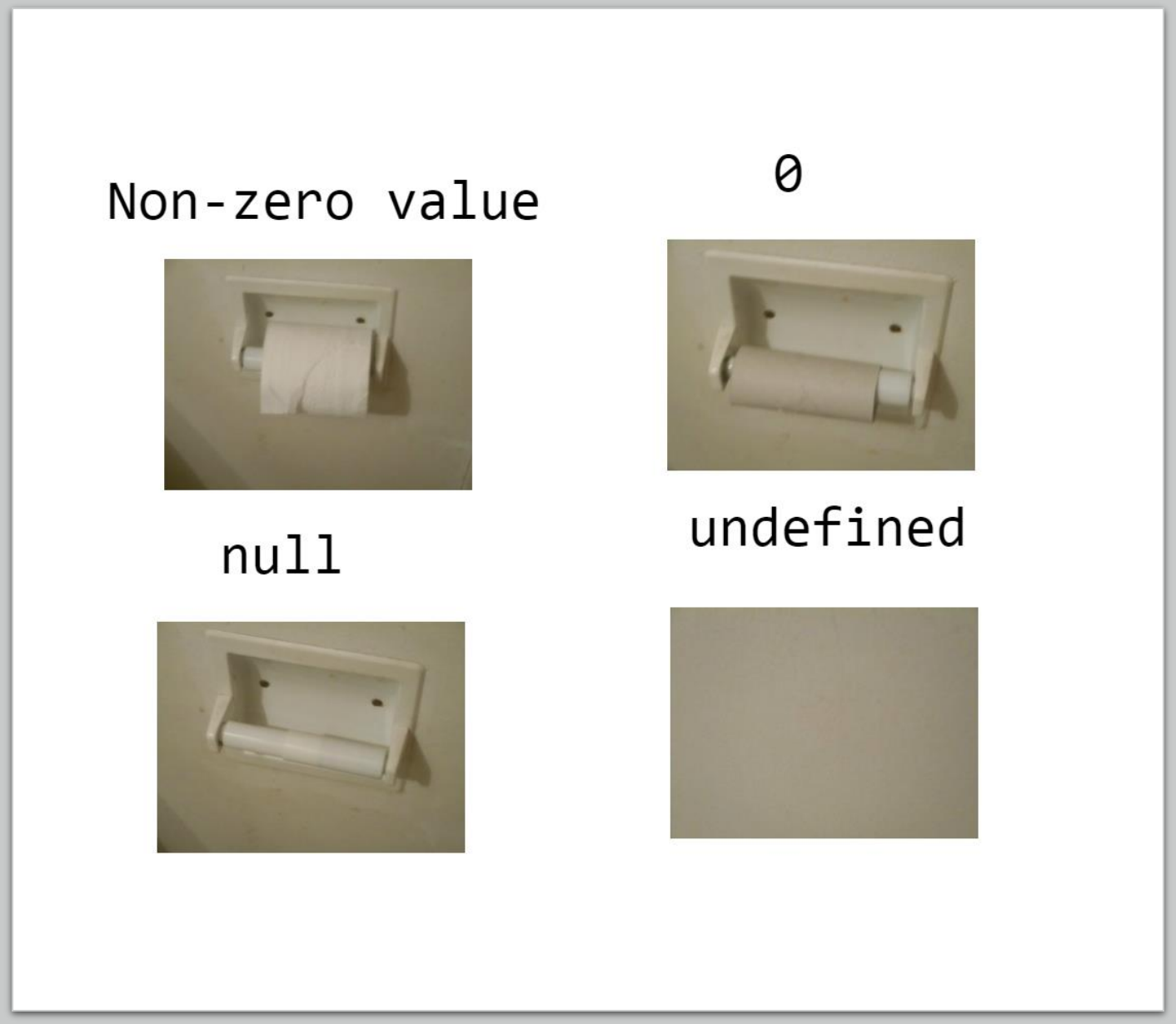
Boolean

True ou False. Muito utilizado para funções condicionais, entre outros.

Undefined e Null

Undefined e Null são dois tipos de caracteres muito confundidos no JavaScript, porém, entender a diferença entre eles é crucial durante o desenvolvimento das nossas aplicações.

- Undefined = Valor indefinido – Algo que não existe;
- Null = Valor nulo – Objecto que não tem nada dentro dele.



```
// {propriedade: "valor"}
console.log({
  name: "Yuri",
  idade: 27,
  endereço: {
    cidade: "Petrópolis",
    estado: "Rio de Janeiro"
  },
  gritar: function() {
    console.log("AAAAAAAAAAAAAAA")
  }
})
```

Object

Um objeto é composto de Propriedades/Atributos e Funcionalidades/Métodos. É necessária uma atenção especial neste tipo de dado pois ele estará presente frequentemente em nosso cotidiano.

Array (Vetores)

```
["Yuri Weilemann", "Gustavo Lima", "E você"]
```

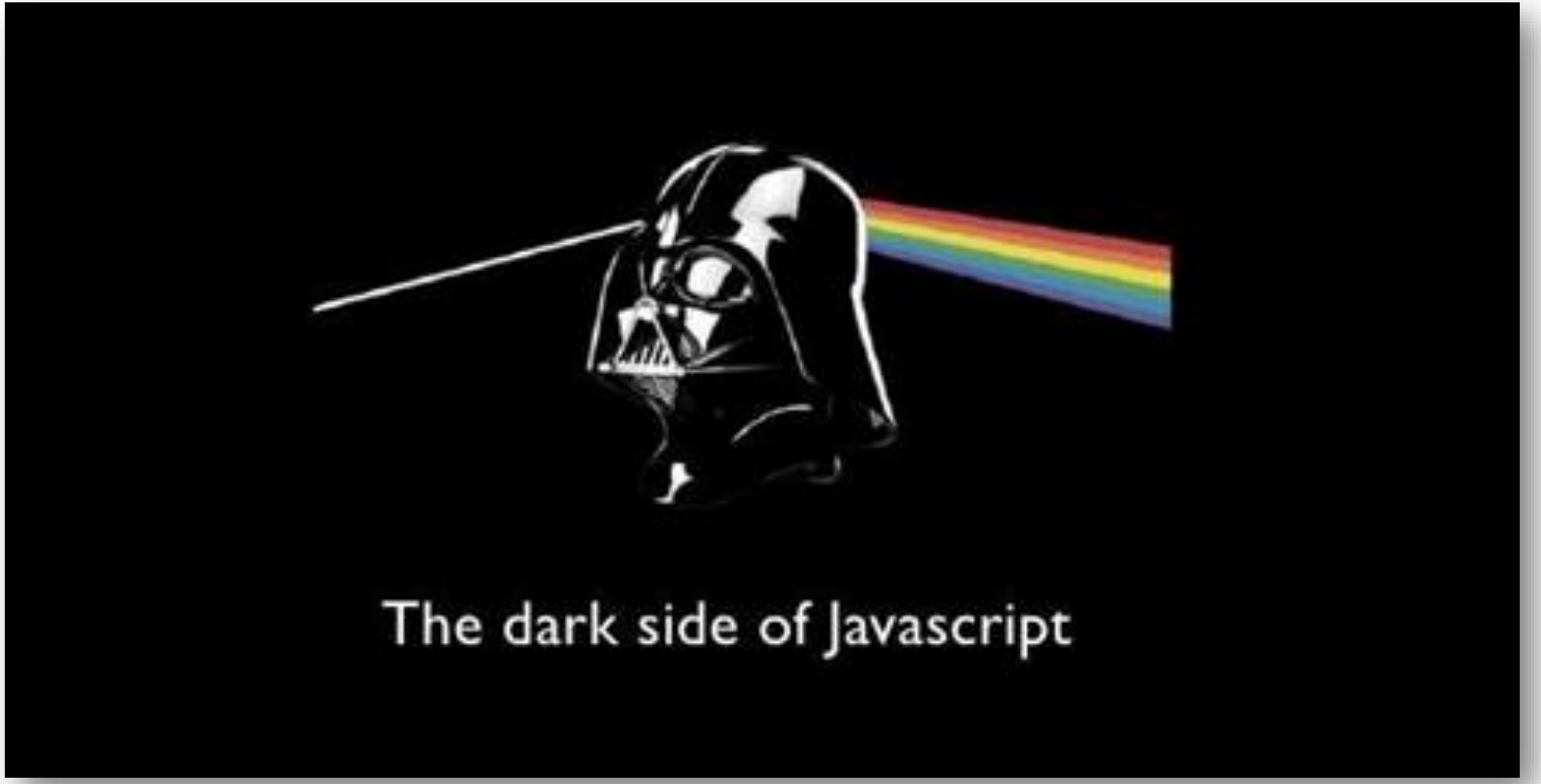
É uma lista, ou seja, um agrupamento de dados. Podem receber qualquer tipo de dados dentro de si.

Tipos de Dados

- Primitive / Primitive Value

  São valores que não são objetos e que possuem valores imutáveis (não sofrem alterações).
  - String
  - Number
  - Boolean
  - Undefined
  - Symbol
  - BigInt
- Structural
  - Object
    - Array
    - Map
    - Set
    - Date
    - ...
  - Function
- Structural Primitive
  - null

O Lado Obscuro do JavaScript



Funções Anônimas

Em casos onde não há um outra parte do código onde queremos referenciar uma função e ela será apenas referenciada ao invés de chamada, podemos usar o conceito de função anônima, já criando a função no lugar onde antes apenas indicamos seu nome. Por exemplo:

```
inputTamanho.oninput = function() {
  outputTamanho.value = inputTamanho.value
}
```

Manipulando Strings

Uma variável que armazena um string faz muito mais que isso! Ela permite, por exemplo, consultar o seu tamanho e realizar transformações em seu valor. Por exemplo :

```
var palavraTosca = "Catapora";

palavraTosca.length; // tamanho da string

palavraTosca.replace("pora","pimba");
```

Funcionou?

Manipulando Strings

Assim como em Java, podemos converter uma string para inteiro ou ponto flutuante usando o método parseInt e parseFloat:

```
var textoInteiro = "10";

var inteiro = parseInt(textoInteiro);

var textoFloat = "10.22";

var float = parseFloat(textoFloat);
```

Manipulando Números

Números, assim como strings, também são imutáveis. O exemplo abaixo altera o número de casas decimais com a função toFixed. Esta função retorna uma string, mas, para ela funcionar corretamente, seu retorno precisa ser capturado:

```
var milNumber = 1000;
var milString = milNumber.toFixed(2); // recebe o retorno da função
console.log(milString); // imprime a string "1000.00"
```

Arrays em Javascript

A utilização de arrays em javascript não é muito difeferente do que foi visto em Java. Os pontos interessantes são que, por javascript não ser tipado, podemos armazenar valores de tipos diferentes em vetores no Javascript.

```
var variosTipos = ["Mamona", 10, [1,2]];
```

Para recuperar um valor, basta referenciar o vetor e sua respectiva posição. Ex: variosTipos[0] // Mamona

Arrays em Javascript

Para adicionar elementos ao vetor, podemos utilizar a função **push** que adiciona um elemento na última posição do array ou adicionar direto em um índice selecionado:

```
var palavras = ["RSW", "Ensino"];
palavras.push("Inovação"); // adiciona a string "Inovação" por último no array
palavras[9] = "Criatividade";
```

Laços de Repetição e Condicionais

Todas as estruturas de laços de repetição e condicionais que vimos em disciplinas passadas funcionam em javascript. Apenas alguns exemplos de sintaxe:

```
while (contador ⩽ 10) {
  // código a ser repetido
}
```

```
if (condicao) {
  //  código a ser executado se condiçao for verdadeira
}
```

```
for (/* variável de controle */; /* condição */; /* pós execução */) {
  // código a ser repetido
}
```

Funções Temporais

Em JavaScript, podemos criar um timer para executar um trecho de código após um certo tempo, ou ainda executar algo de tempos em tempos.

A função **setTimeout** permite que agendemos alguma função para execução no futuro e recebe o nome da função a ser executada e o número de milissegundos a esperar:

```
// executa minhaFuncao daqui um segundo

setTimeout(minhaFuncao, 1000);
```

Funções Temporais

Se for um código recorrente, podemos usar o setInterval que recebe os mesmos argumentos mas executa a função indefinidamente de tempos em tempos:

```
// executa minhaFuncao de um em um segundo
setInterval(minhaFuncao, 1000);
```

É uma função útil para, por exemplo, implementar um banner rotativo, apresentado no exercício à seguir.