

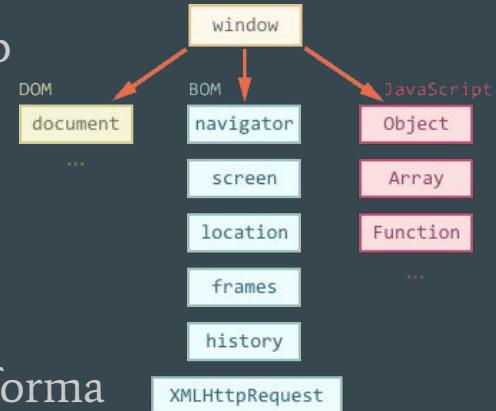
Programação Orientada a Objetos

...

M03 - Document Object Model (DOM)

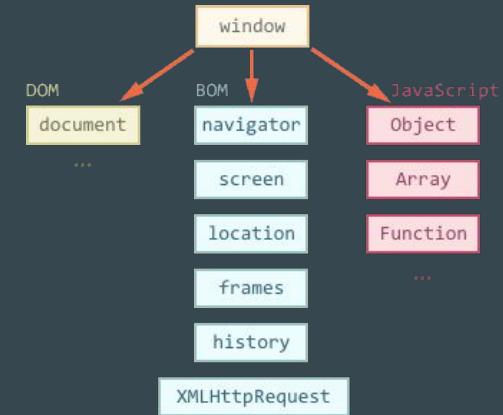
M03 - Document Object Model (DOM)

- O JavaScript foi inicialmente criado para navegadores da web
- Desde então, evoluiu para muitas plataformas:
 - Web (navegadores e servidores da web)
 - Desktop
 - Mobile
 - Outras plataformas: máquinas de lavar roupa, ...
- Cada uma delas fornece funcionalidades específicas da plataforma
- Quando o JS corre num browser há um objeto "raiz" chamado **window**:
 - é um objeto global para código JS
 - representa a “janela do navegador” e fornece métodos para controlá-la



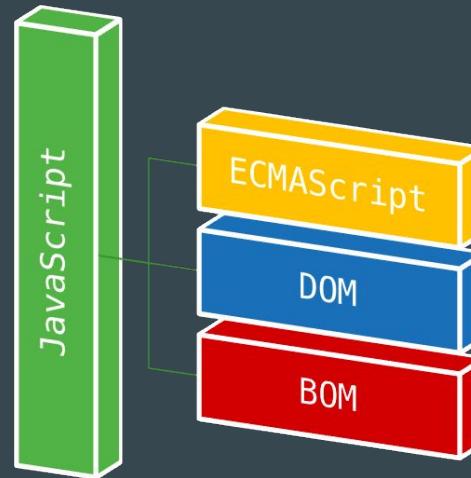
M03 - Document Object Model (DOM)

- O objeto global **window** dá acesso a três modelos de objetos:
 - **Document Object Model (DOM)**
 - tem um objeto **document** que dá acesso ao conteúdo da página
 - pode-se criar/alterar qualquer coisa na página HTML
 - **Browser Object Model (BOM)**
 - objetos adicionais fornecidos pelo navegador para trabalhar com tudo, exceto o documento
 - **JavaScript**
 - objetos nativos da linguagem JavaScript



M03 - Document Object Model (DOM)

1. Document Object Model (DOM)
2. Browser Object Model (BOM)



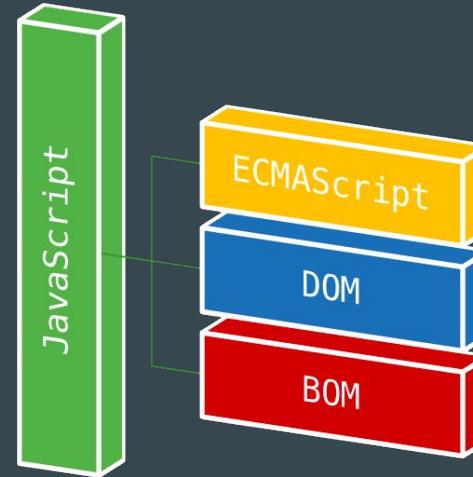
M03 - Document Object Model (DOM)

1. Document Object Model

M03 - Document Object Model (DOM)

1. Document Object Model

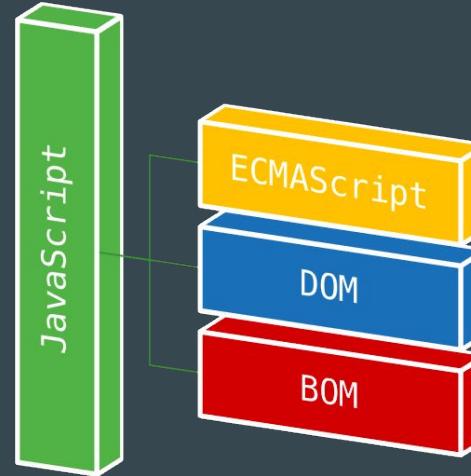
- Árvore DOM
- Procura
- Alteração
- Navegação
- Gestão de nodos
- Eventos
- Formulários



M03 - Document Object Model (DOM)

1. Document Object Model

- Árvore DOM
- Procura
- Alteração
- Navegação
- Gestão de nodos
- Eventos
- Formulários

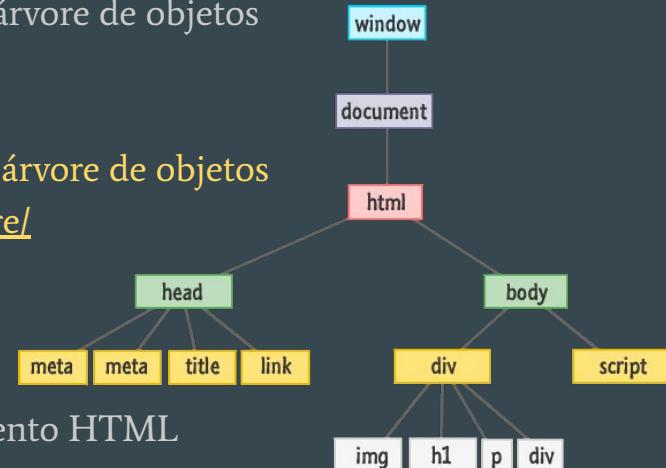


M03 - Document Object Model (DOM)

1. Document Object Model

Árvore DOM

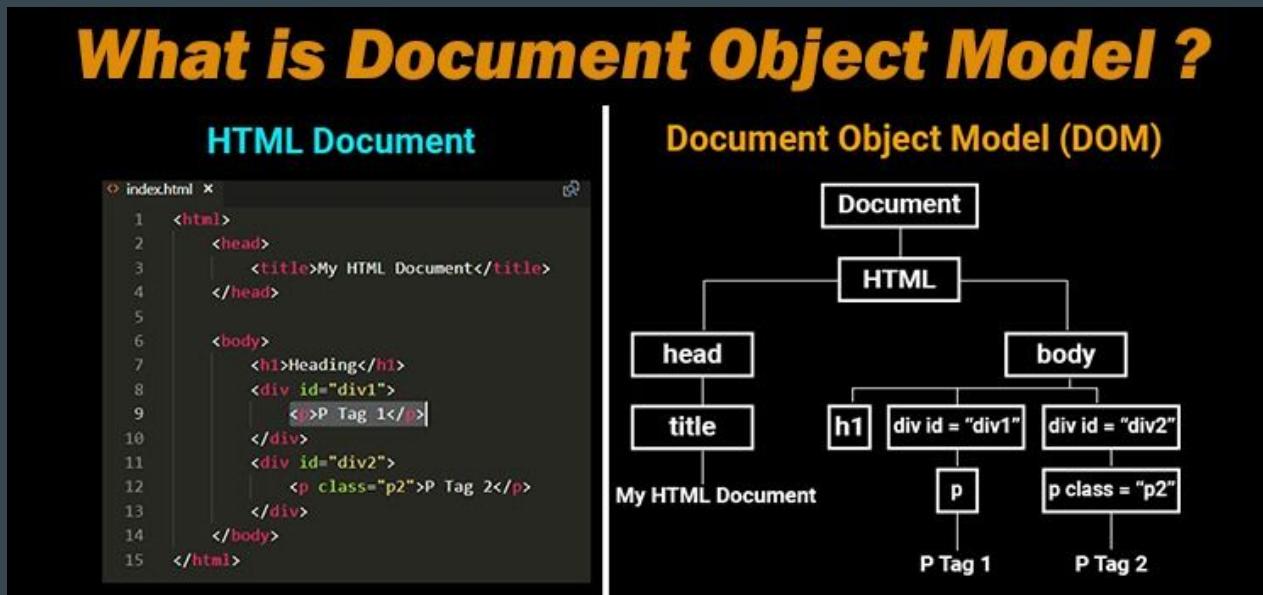
- Quando uma página HTML é carregada, o browser cria uma árvore de objetos em memória representando todo o documento HTML
- O HTML DOM é
 - uma interface que vai permitir o acesso standard a essa árvore de objetos
 - é um standard da W3C: <https://www.w3.org/TR/domcore/>
- Com essa interface, o JavaScript pode criar HTML dinâmico:
 - adicionar/alterar/remover elementos e atributos HTML
 - criar/reagir a eventos HTML existentes na página
- O objeto **document** é o ponto de partida de acesso ao documento HTML



M03 - Document Object Model (DOM)

1. Document Object Model

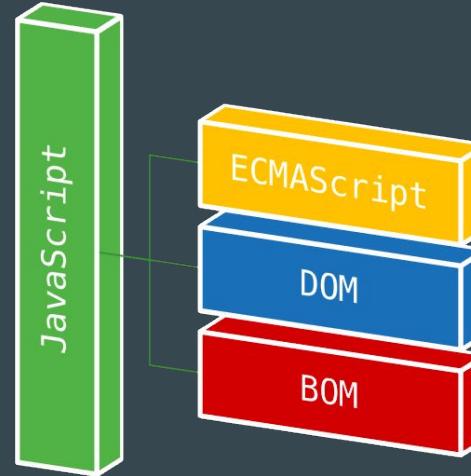
Árvore DOM



M03 - Document Object Model (DOM)

1. Document Object Model

- Árvore DOM
- Procura
- Alteração
- Navegação
- Gestão de nodos
- Eventos
- Formulários



M03 - Document Object Model (DOM)

1. Document Object Model

Procurar elementos

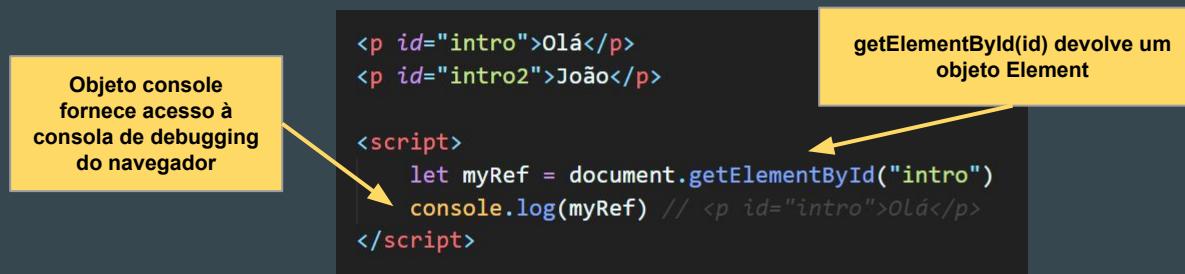
- Muitas vezes, com JavaScript, é necessário manipular elementos HTML
- Para fazer isso, é necessário encontrar primeiro os elementos respetivos
- O DOM permite procurar elementos por
 - id
 - nome de tag
 - nome de classe
 - seletores CSS

M03 - Document Object Model (DOM)

1. Document Object Model

Procurar elementos por id

- Forma mais simples de procurar um elemento HTML
- Uso do método `getElementById(identifier)`
- Procura por um elemento HTML que contenha um atributo `id` com um valor igual ao `identifier`



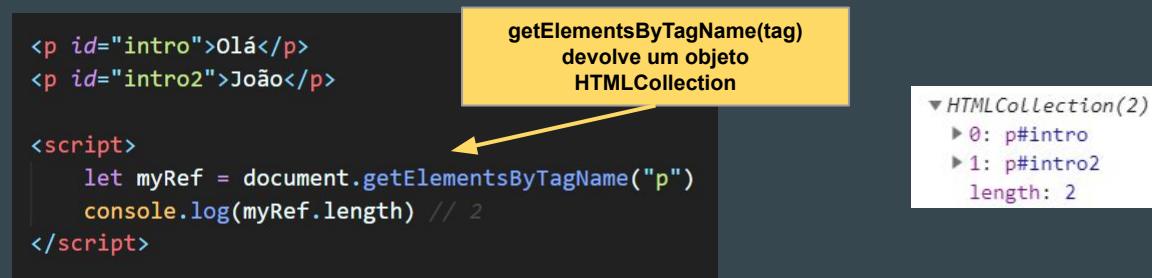
- Se o elemento a procurar
 - for encontrado, o método devolve o correspondente objeto `Element`
 - não for encontrado, devolve valor `null`

M03 - Document Object Model (DOM)

1. Document Object Model

Procurar elementos por nome de tag

- Uso do método `getElementsByName(tag)`
- Procura por elementos HTML que sejam definidos por uma etiqueta igual a `tag`



- O método devolve sempre um objeto `HTMLCollection`
- Um objeto `HTMLCollection` é um array (coleção) de elementos HTML

M03 - Document Object Model (DOM)

1. Document Object Model

Procurar elementos por nome de classe

- Uso do método `getElementsByClassName(class)`
- Procura por elementos HTML que contenham um atributo `class` com um valor igual ao `class`

```
<p id="intro" class="red">Olá</p>
<p id="intro2" class="blue">João</p>

<script>
  let myRef = document.getElementsByClassName("red")
  console.log(myRef.length) // 1
</script>
```

getElementsByClassName(class)
devolve um objeto
HTMLCollection

```
▼ HTMLCollection []
  ▷ 0: p#intro.red
      length: 1
```

- O método devolve também um objeto `HTMLCollection`

M03 - Document Object Model (DOM)

1. Document Object Model

Procurar elementos por selectores CSS

- Se quiser encontrar todos os elementos HTML que correspondem a um seletor CSS especificado (id, nomes de classe, tipos, atributos, valores de atributos, etc.)
- Uso do método `querySelectorAll(selector)`

```
<p id="intro">Olá</p>
<p id="intro">Pedro</p>
<p id="intro2">João</p>

<script>
  let myRef = document.querySelectorAll("p#intro")
  console.log(myRef) // 2
</script>
```

querySelectorAll(selector) devolve
um objeto NodeList

```
▼ NodeList(2) [p#intro, p#intro]
  ► 0: p#intro
  ► 1: p#intro
  length: 2
```

- O método devolve também um objeto `NodeList`
- Um objeto `NodeList` pode ser visto como um conjunto de nodos
- Pode usar o método `querySelector` se quiser encontrar apenas um elemento

M03 - Document Object Model (DOM)

1. Document Object Model

Procurar elementos

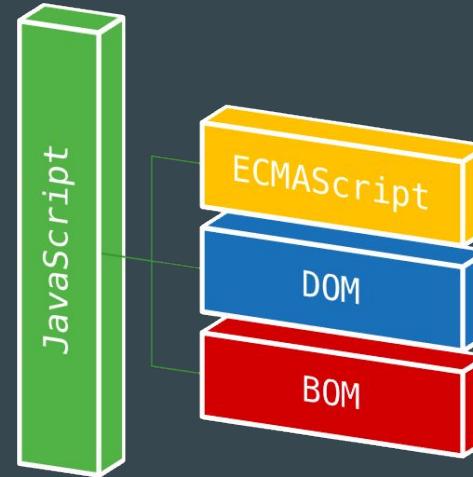
- Quadro resumo

Method	Searches by...	Can call on an element?	Live?
<code>getElementById</code>	<code>id</code>	-	-
<code>getElementsByName</code>	<code>name</code>	-	✓
<code>getElementsByTagName</code>	<code>tag or '*'</code>	✓	✓
<code>getElementsByClassName</code>	<code>class</code>	✓	✓
<code>querySelector</code>	CSS-selector	✓	-
<code>querySelectorAll</code>	CSS-selector	✓	-

M03 - Document Object Model (DOM)

1. Document Object Model

- Árvore DOM
- Procura
- Alteração
- Navegação
- Gestão de nodos
- Eventos
- Formulários



M03 - Document Object Model (DOM)

1. Document Object Model

Alterar elementos

- O DOM permite alterar elementos, mais concretamente:
 - alterar conteúdo de elementos
 - alterar valores de atributos
 - alterar estilos de elementos

M03 - Document Object Model (DOM)

1. Document Object Model

Alterar conteúdo de elementos

- Forma mais fácil de modificar conteúdo de um elemento HTML é usando a propriedade `innerHTML`
- Sintaxe:

```
document.getElementById(id).innerHTML = new HTML
```

- Exemplo:

```
<p id="p1">Olá Mundo!</p>
<p id="p2">Outro texto</p>

<script>
|   document.getElementById("p1").innerHTML = "Novo texto!"
</script>
```

Novo texto!

Outro texto

M03 - Document Object Model (DOM)

1. Document Object Model

Alterar valores de atributos

- Para alterar o valor de um atributo HTML, use esta sintaxe:

```
document.getElementById(id).attribute = new value
```

- Exemplo:

```


<script>
|   document.getElementById("myImage").src = "landscape.jpg";
</script>
```

M03 - Document Object Model (DOM)

1. Document Object Model

Alterar estilos de elementos

- Para alterar o estilo de um elemento HTML, use esta sintaxe:

```
document.getElementById(id).style.property = new style
```

- Exemplo:

```
<p id="p1">Olá Mundo!</p>

<script>
  document.getElementById("p1").style.color = "blue"
</script>
```

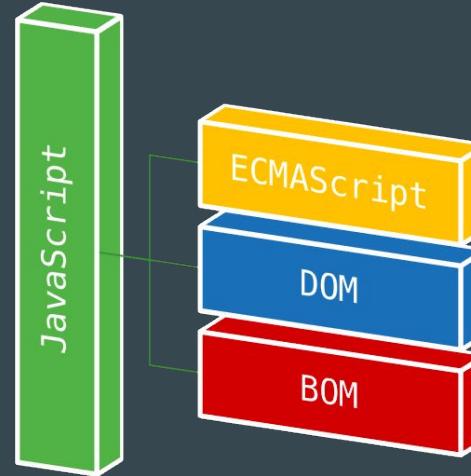


Olá Mundo!

M03 - Document Object Model (DOM)

1. Document Object Model

- Árvore DOM
- Procura
- Alteração
- Navegação
- Gestão de nodos
- Eventos
- Formulários

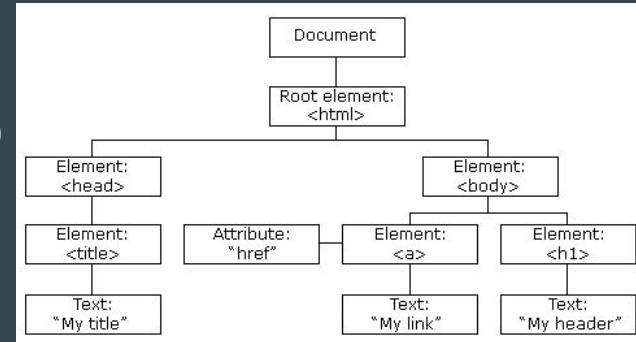


M03 - Document Object Model (DOM)

1. Document Object Model

Navegação

- Com o HTML DOM, pode navegar na árvore de nodos usando relacionamentos de nodos
- O padrão HTML DOM do W3C define que tudo num documento HTML é um nodo:
 - O documento inteiro é um nodo de documento
 - Todo elemento HTML é um nodo de elemento
 - Texto dentro de elementos HTML são nodos de texto
 - Atributo HTML é um nodo de atributo (descontinuado)
 - Todos os comentários são nodos de comentários

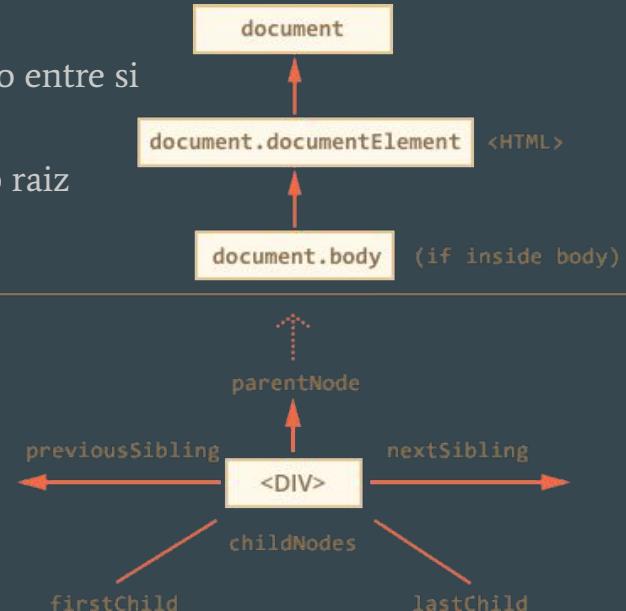


M03 - Document Object Model (DOM)

1. Document Object Model

Navegação

- Os nodos na árvore HTML têm um relacionamento hierárquico entre si
- Os termos pai, filho e irmão descrevem os relacionamentos
 - Na árvore de nodos, o nodo superior é chamado de nodo raiz
 - Cada nodo tem um pai, exceto a raiz (que não tem pai)
 - Um nodo pode ter um número de filhos
 - Irmãos (siblings) são nodos com o mesmo pai

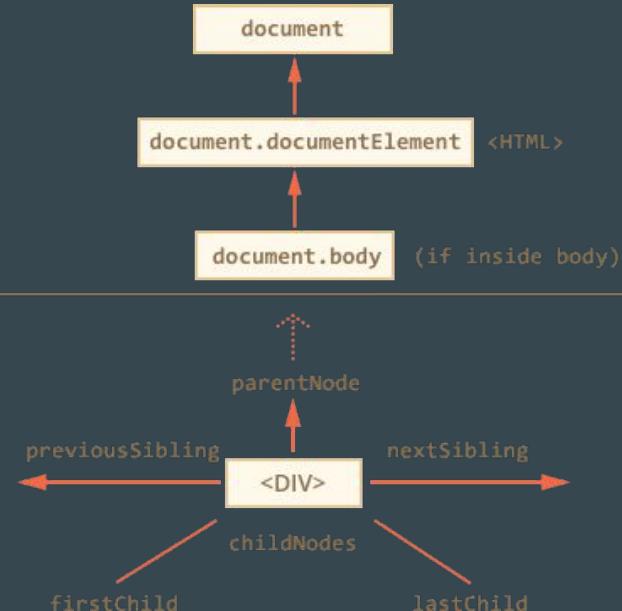


M03 - Document Object Model (DOM)

1. Document Object Model

Navegação

- Os nodos da árvore superior estão disponíveis diretamente como propriedades do documento:
- `document.documentElement`
 - O 1º nodo do documento. É o nodo DOM da tag `<html>`
- `document.body`
 - Nodo que referencia o elemento `<body>`
- `document.head`
 - Nodo que referencia o elemento `<head>`



M03 - Document Object Model (DOM)

1. Document Object Model

Navegação

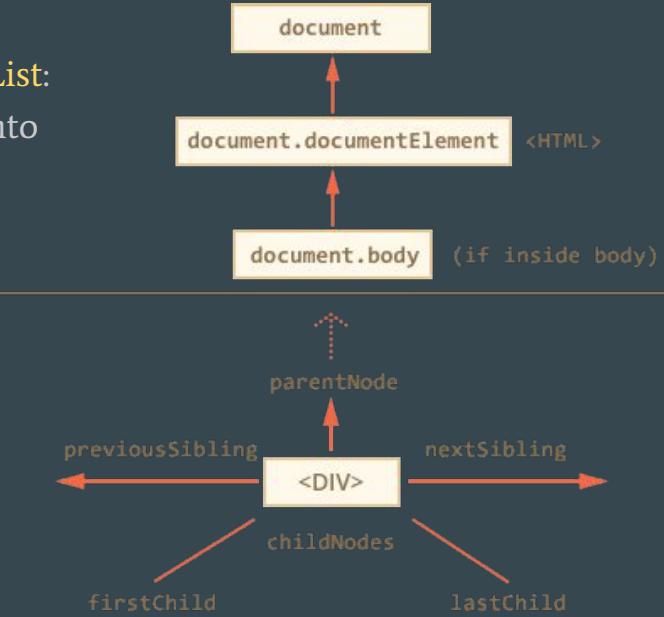
- Coleções são normalmente representadas por objetos `NodeList`:
 - é uma lista (coleção) de nós extraídos de um documento
 - é quase o mesmo que um objeto `HTMLCollection`
- Exemplo: pintar todos os `<p>` de vermelho

```
<h2>Sou um header!</h2>

<p>Sou um par!</p>

<p>Sou um segundo par!</p>

<script>
    const myNodelist = document.querySelectorAll("p");
    for (let node of myNodelist) {
        node.style.color = "red";
    }
</script>
```



M03 - Document Object Model (DOM)

1. Document Object Model

Navegação

- Propriedades importantes de um nodo
 - nodetype
 - nodename
 - nodevalue

M03 - Document Object Model (DOM)

1. Document Object Model

Navegação

- Propriedades importantes de um nodo
 - nodetype - especifica o tipo de nodo (só de leitura)
 - nodename
 - nodevalue

```
<title id="demo">Introdução ao DOM</title>
<script>
    console.log(document.getElementById("demo").nodeType) // 1
</script>
```

Node	Type	Example
ELEMENT_NODE	1	<h1 class="heading">W3Schools</h1>
ATTRIBUTE_NODE	2	class = "heading" (deprecated)
TEXT_NODE	3	W3Schools
COMMENT_NODE	8	<!-- This is a comment -->
DOCUMENT_NODE	9	The HTML document itself (the parent of <html>)
DOCUMENT_TYPE_NODE	10	<!Doctype html>

M03 - Document Object Model (DOM)

1. Document Object Model

Navegação

- Propriedades importantes de um nodo
 - nodetype
 - **nodename** - especifica o nome de um nodo (somente de leitura)
 - Obtenção de diferentes valores num
 - nodo de elemento é o mesmo que o nome da tag
 - nodo de atributo é o nome do atributo
 - nodo de texto é sempre **#text**
 - nodo de documento é sempre **#document**
 - nodevalue

```
<title id="demo">Introdução ao DOM</title>
<script>
|   console.log(document.getElementById("demo").nodeName) // TITLE
</script>
```

M03 - Document Object Model (DOM)

1. Document Object Model

Navegação

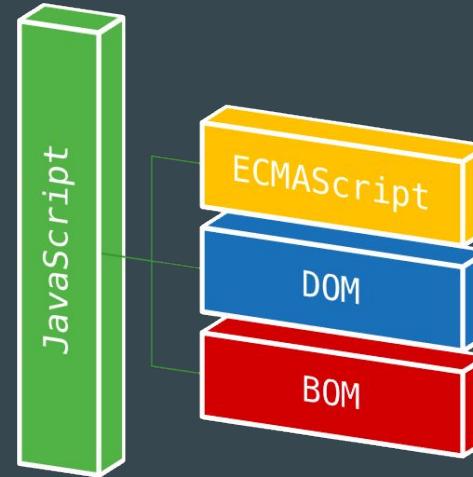
- Propriedades importantes de um nodo
 - nodetype
 - nodename
 - **nodevalue** - especifica o valor de um nodo
 - Obtenção de diferentes valores num nodo de elemento é **null**
 - nodo de texto é o próprio texto
 - nodo de atributo é o valor do atributo

```
<title id="demo">Introdução ao DOM</title>
<script>
    console.log(document.getElementById("demo").nodeValue) // null
</script>
```

M03 - Document Object Model (DOM)

1. Document Object Model

- Árvore DOM
- Procura
- Alteração
- Navegação
- Gestão de nodos
- Eventos
- Formulários



M03 - Document Object Model (DOM)

1. Document Object Model

Gestão de nodos

- A gestão de nodos passa pela
 - Criação de novos nodos
 - Substituição de nodos existentes
 - Remoção de nodos existentes

M03 - Document Object Model (DOM)

1. Document Object Model

Gestão de nodos

- Criação de novos nodos
 - Para adicionar um novo elemento ao HTML DOM:
 - crie o elemento (nodo de elemento) com os métodos createElement/CreateTextNode
 - anexe-o a um elemento existente com o método appendChild

```
<div id="div1">
    <p id="p1">Um parágrafo</p>
    <p id="p2">Outro parágrafo</p>
</div>

<script>
    const para = document.createElement("p")
    const node = document.createTextNode("Um novo parágrafo")
    para.appendChild(node)

    const element = document.getElementById("div1")
    element.appendChild(para)
</script>
```

Um parágrafo
Outro parágrafo
Um novo parágrafo

M03 - Document Object Model (DOM)

1. Document Object Model

Gestão de nodos

- Substituição de nodos
 - Para substituir um elemento no HTML DOM, use o método `replaceChild`

```
<div id="div1">
    <p id="p1">Um parágrafo</p>
    <p id="p2">Outro parágrafo</p>
</div>

<script>
    const para = document.createElement("p")
    const node = document.createTextNode("Um novo parágrafo")
    para.appendChild(node)

    const parent = document.getElementById("div1")
    const child = document.getElementById("p1")
    parent.replaceChild(para, child)
</script>
```

Um novo parágrafo

Outro parágrafo

M03 - Document Object Model (DOM)

1. Document Object Model

Gestão de nodos

- Remoção de nodos
 - Para remover um elemento HTML, deve
 - conhecer o pai do elemento
 - usar o método `removeChild`

```
<div id="div1">
  <p id="p1">Um parágrafo</p>
  <p id="p2">Outro parágrafo</p>
</div>

<script>
  const parent = document.getElementById("div1")
  const child = document.getElementById("p1")
  parent.removeChild(child)
</script>
```

Outro parágrafo

M03 - Document Object Model (DOM)

1. Document Object Model

Gestão de nodos

- A gestão de nodos deve ser usada com parcimônia dada a verbosidade
- Por exemplo, na adição de elementos é preferível usar **template strings** em combinação com a propriedade **innerHTML**

```
<table id="table1">
  <tr>
    <th>Nome</th>
    <th>Idade</th>
  </tr>
  <tr>
    <td>Ricardo</td>
    <td>43</td>
  </tr>
</table>

<script>
  const myTable = document.getElementById("table1")
  const otherName = "Gabriela"
  const otherAge = 8

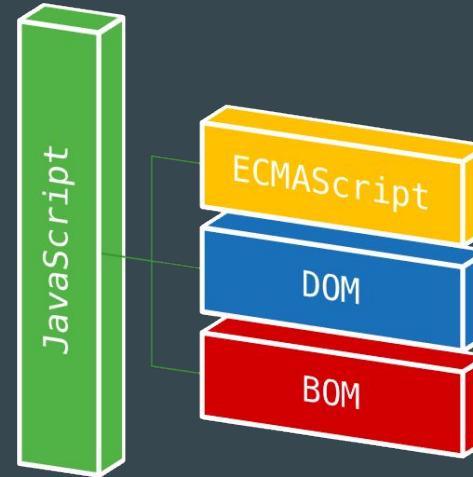
  myTable.innerHTML += `<tr><td>${otherName}</td><td>${otherAge}</td></tr>`
</script>
```

Nome	Idade
Ricardo	43
Gabriela	8

M03 - Document Object Model (DOM)

1. Document Object Model

- Árvore DOM
- Procura
- Alteração
- Navegação
- Gestão de nodos
- Eventos
- Formulários



M03 - Document Object Model (DOM)

1. Document Object Model

Introdução aos eventos

- Podemos associar código JavaScript quando ocorre um evento
- Tipos de eventos:
 - do sistema (ex.: carregamento de uma página HTML)
 - do utilizador (ex.: clique num botão)
- Para reagir a eventos, atribui-se um **manipulador** - uma função que é executada no caso de um evento
- Formas de associar de manipuladores:
 - Atributo HTML
 - Propriedade DOM
 - Método addEventListener

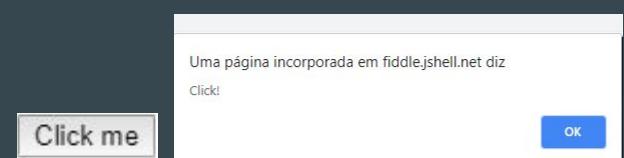
M03 - Document Object Model (DOM)

1. Document Object Model

Introdução aos eventos

- Formas de associar de manipuladores:
 - Atributo HTML
 - Propriedade DOM
 - Método addEventListener
- Um manipulador pode ser definido em HTML com um atributo denominado `on<event>`
- Por exemplo, para atribuir um manipulador de cliques a um botão, podemos usar o `onclick`:

```
<input value="Click me" onclick="alert('click!')" type="button"></input>
```



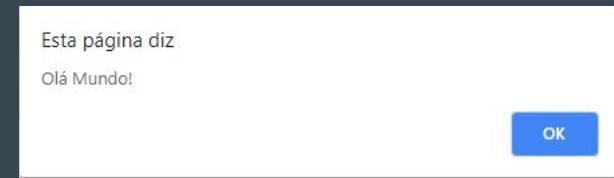
M03 - Document Object Model (DOM)

1. Document Object Model

Introdução aos eventos

- Formas de associar de manipuladores:
 - Atributo HTML
 - Propriedade DOM
 - Método addEventListener
- Uso da palavra reservada this
 - cria uma referência para o próprio elemento
 - no próximo exemplo, this cria uma referência para o elemento <p>

```
<p onclick="hello(this)">Olá Mundo!</p>
<script>
    function hello(obj) {
        alert(obj.innerHTML);
    };
</script>
```



M03 - Document Object Model (DOM)

1. Document Object Model

Introdução aos eventos

- Formas de associar de manipuladores:
 - Atributo HTML
 - Propriedade DOM
 - Método addEventListener
- Um manipulador pode ser definido como uma propriedade DOM denominada `on<event>`
- Técnica similar à anterior, mas preferível pois separa o HTML do JS

```
<input id="elem" type="button" value="Click me">
<script>
    elem.onclick = function () {
        alert('Obrigado!');
    };
</script>
```

M03 - Document Object Model (DOM)

1. Document Object Model

Introdução aos eventos

- Método `addEventListener` anexa um manipulador de eventos a um elemento sem sobrescrever manipuladores de eventos existentes
- Sintaxe: `element.addEventListener(event, handler[, options])`
- Pode adicionar manipuladores de eventos
 - a um elemento
 - do mesmo tipo a um elemento, ou seja, dois eventos de "clique"
 - a qualquer objeto DOM, não apenas elementos HTML (ex.: objeto `window`)
- O JavaScript é separado da marcação HTML
 - melhor legibilidade
 - permite adicionar manipuladores de eventos mesmo não controlando o HTML

M03 - Document Object Model (DOM)

1. Document Object Model

Introdução aos eventos

- Sintaxe: `element.addEventListener(event, handler[, options])`
- **event**
 - Nome do evento, por ex. "click"
- **handler**
 - A função de manipulador
- **options**
 - Um objeto opcional adicional com as seguintes propriedades:
 - **once**: se for verdade, o ouvinte é automaticamente removido depois de ser acionado
 - **capture**: permite controlar a propagação de eventos (**bubbling** - `false` e **capturing** - `true`)
 - **passive**: se verdadeiro, o manipulador não fará o `preventDefault()`, abordaremos isso posteriormente

M03 - Document Object Model (DOM)

1. Document Object Model

Introdução aos eventos

- Sintaxe: `element.addEventListener(event, handler[, options])`

```
<p id="p1">Olá Mundo!</p>
<script>
    document.getElementById("p1").addEventListener("click", myFunction);

    function myFunction() {
        alert("Olá Mundo!");
    }
</script>
```

Com função nomeada

```
<p id="p1">Olá Mundo!</p>
<script>
    document.getElementById("p1").addEventListener("click", function () {
        alert("Olá Mundo!")
    })
</script>
```

Com função anónima

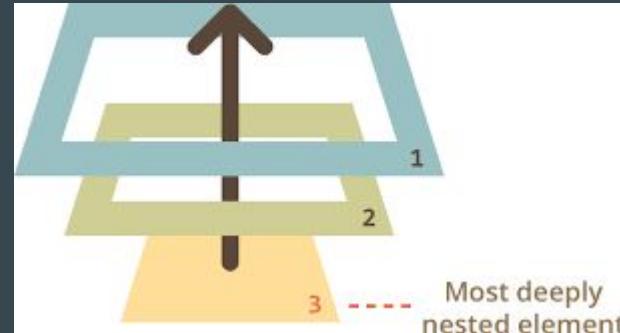
M03 - Document Object Model (DOM)

1. Document Object Model

Introdução aos eventos

- Propagação de eventos:
 - Bubbling (usecapture a falso – por omissão) - O evento do elemento mais interno é tratado primeiro e depois o externo
 - Capturing (usecapture a verdadeiro) - O evento do elemento mais externos é tratado primeiro e depois o interno

```
<p id="p1" onclick="alert('dois')">
  <button id="b1" onclick="alert('um')">click me</button>
</p>
```



M03 - Document Object Model (DOM)

1. Document Object Model

Introdução aos eventos

- Remoção de manipuladores
 - Sintaxe: `element.removeEventListener(event, handler)`
 - Exemplo:

```
<p id="p1">Olá Mundo!</p>

<script>
    function handler() {
        alert('Thanks!');
    }

    const myP = document.getElementById("p1")
    myP.addEventListener("click", handler)
    myP.removeEventListener("click", handler)

</script>
```

M03 - Document Object Model (DOM)

1. Document Object Model

Introdução aos eventos

- Criação de manipulador executado após a criação da árvore DOM em memória

```
<body>
  <script>
    window.addEventListener("load", function () {
      alert(document.getElementById("p1").innerHTML)
    })
  </script>
  <p id="p1">Olá Mundo!</p>
</body>
```

M03 - Document Object Model (DOM)

1. Document Object Model

Tipos de eventos

- Janela
- Rato
- Teclado
- Widgets

M03 - Document Object Model (DOM)

1. Document Object Model

Tipos de eventos

- Janela
 - **load** - janela totalmente disponível
 - **unload** - fim de disponibilidade de janela
 - **resize** - alteração das dimensões
 - **scroll** - navegação por scroll da janela

M03 - Document Object Model (DOM)

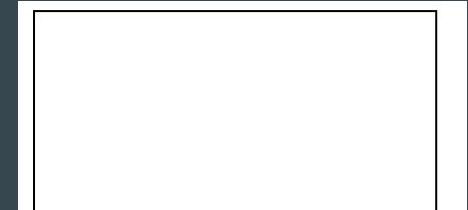
1. Document Object Model

Tipos de eventos

- Rato
 - click - clique (início e fim de pressão) num botão
 - doubleclick - dois cliques em sequência rápida
 - mousedown - início de pressão sobre o botão
 - mouseup - fim de pressão sobre o botão
 - mousemove - movimento do rato

```
<div id="div"></div>
<p id="result"></p>
<script>
    const myDiv = document.getElementById("div")
    myDiv.addEventListener("mousemove", function (event) {
        const coor = `Coordenadas: (${event.clientX},${event.clientY})`
        document.getElementById("result").innerHTML = coor;
    })
</script>
```

objeto event



Coordenadas: (142,60)

M03 - Document Object Model (DOM)

1. Document Object Model

Tipos de eventos

- Teclado
 - keypress - tecla pressionada
 - keydown - início de pressão de tecla
 - keyup - fim de pressão de tecla

```
<input type="text" id="demo">
<script>
    const myInput = document.getElementById("demo")
    myInput.addEventListener("keypress", function (event) {
        console.log(`Pressionou a letra ${event.key}`)
    })
</script>
```



M03 - Document Object Model (DOM)

1. Document Object Model

Tipos de eventos

- Widgets
 - change - alteração de valores (input, select e textarea)
 - focus - receber foco (input e textarea)
 - select - seleção de texto (input e textarea)
 - submit - submissão de formulário (form)

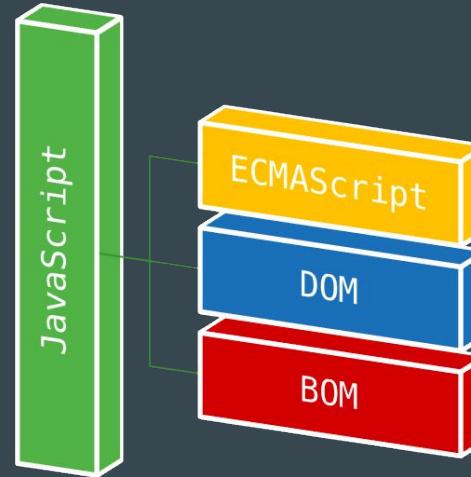
```
<input type="text" id="demo">
<script>
    const myInput = document.getElementById("demo")
    myInput.addEventListener("keypress", function (event) {
        console.log(`Pressionou a letra ${event.key}`)
    })
</script>
```



M03 - Document Object Model (DOM)

1. Document Object Model

- Árvore DOM
- Procura
- Alteração
- Navegação
- Gestão de nodos
- Eventos
- Formulários



M03 - Document Object Model (DOM)

1. Document Object Model

Formulários

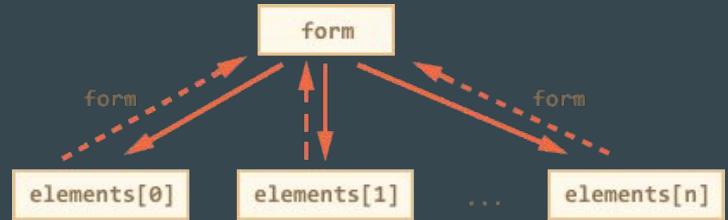
- Formulários e os seus elementos possuem muitas propriedades e eventos especiais
- Os formulários de um documento HTML são membros da coleção especial `document.forms`
- Para aceder aos elementos podemos usar, no contexto do form, a coleção `elements`
- O acesso pode ser feito pelo nome ou pelo índice

```
<form name="my">
  <input name="one" value="1">
  <input name="two" value="2">
</form>

<script>
  // obter o form
  let form = document.forms.my; // elemento <form name="my">

  // obter o elemento
  let elem = form.elements.one; // elemento <input name="one">

  alert(elem.value); // 1
</script>
```



M03 - Document Object Model (DOM)

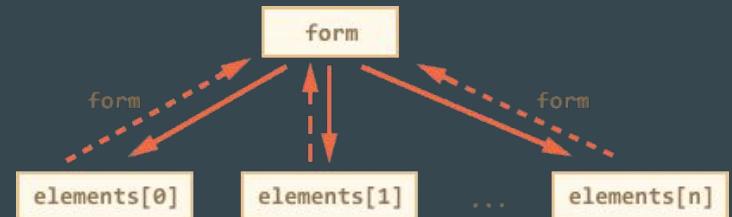
1. Document Object Model

Formulários

- Tome nota que pode haver vários elementos com o mesmo nome (ex.: botões de opção)
- Nesse caso, `form.elements[name]` é uma coleção, por exemplo:

```
<form>
  <input type="radio" name="age" value="10">
  <input type="radio" name="age" value="20">
</form>

<script>
  let form = document.forms[0];
  let ageElems = form.elements.age;
  alert(ageElems[0].value); // 10, o primeiro valor de input
</script>
```



M03 - Document Object Model (DOM)

1. Document Object Model

Formulários

- Principais elementos de formulário:
 - Input
 - Textarea
 - Select e option

M03 - Document Object Model (DOM)

1. Document Object Model

Formulários

- Input:

- Uso da propriedade **value** para definir/obter o valor da caixa de texto

```
<form>
  <input id="input1" value="12">
</form>

<script>
  const myInput = document.getElementById("input1")
  console.log(myInput.value) // obter valor atual (12)
  myInput.value = 15 // definir novo valor para a caixa de texto
  console.log(myInput.value) // obter valor atual (15)
</script>
```

propriedade value

```
<form>
  <input type="checkbox" id="input1">
</form>

<script>
  const myCheckbox = document.getElementById("input1")
  console.log(myCheckbox.checked) // obter valor atual (false)
  myCheckbox.checked = true // ativar a caixa de verificação
  console.log(myCheckbox.checked) // obter valor atual (true)
</script>
```

propriedade checked

- Uso da propriedade **checked** para ativar/desativar caixa de verificação (checkbox)

M03 - Document Object Model (DOM)

1. Document Object Model

Formulários

- Input

- Principais eventos:

- change - despoletado após perda de foco da caixa de texto
 - input - despoletado a cada entrada de dados

```
<input type="text" id="input1">
oninput: <span id="result"></span>
<script>
    const myInput = document.getElementById("input1")
    const myResult = document.getElementById("result")
    myInput.addEventListener("input", function () {
        myResult.innerHTML = myInput.value
    })
</script>
```



M03 - Document Object Model (DOM)

1. Document Object Model

Formulários

- Select e option
 - Um elemento <select> possui 3 propriedades importantes:
 - select.options - a coleção de elementos <option>
 - select.value - o valor da opção escolhida
 - select.selectedIndex - o número (índice) da opção selecionada

M03 - Document Object Model (DOM)

1. Document Object Model

Formulários

- Select e option
 - Exemplo de definição de valor para uma opção do elemento <select>
- Três formas de definir o valor de um <select>:
 - Encontre o <option> e defina option.selected como true
 - Configure select.value para o valor
 - Defina select.selectedIndex para o nº da opção

```
<select id="select">
  <option value="apple">Apple</option>
  <option value="pear">Pear</option>
  <option value="banana">Banana</option>
</select>

<script>
  const mySelect = document.getElementById("select")
  // as próximas 3 linhas fazem a mesma coisa
  mySelect.options[2].selected = true;
  mySelect.selectedIndex = 2;
  mySelect.value = 'banana';
</script>
```

M03 - Document Object Model (DOM)

1. Document Object Model

Formulários

- Select e option
 - Criação de uma nova opção
 - Sintaxe: `option = new Option(text, value, defaultSelected, selected)`
 - Parâmetros:
 - `text` - o texto dentro da opção
 - `value` - o valor da opção
 - `defaultSelected` - se verdadeiro, então o atributo selecionado é criado
 - `selected` - se verdadeiro, a opção é selecionada

```
<script>
  let option = new Option("Text", "value");
    // cria <option value="value">Text</option>
</script>
```

M03 - Document Object Model (DOM)

1. Document Object Model

Formulários

- Select e option
 - Exemplo: preencher um elemento <select> com dados de um array

```
<select id="select"></select>

<script>
    const names = ["Maria", "Pedro", "Rui"],

    const mySelect = document.getElementById("select")
    let result = ""
    for (const name of names) {
        result += `<option value='${name}'>${name}</option>`
    }
    mySelect.innerHTML = result;
</script>
```

Ciclo for...of
para iterar sobre array de strings

Uso de template strings para
composição dos elementos <option>

Propriedade
innerHTML

M03 - Document Object Model (DOM)

1. Document Object Model

Formulários

- Submissão

- O evento **submit** é acionado quando o formulário é submetido
- Usado para validar o formulário antes de enviá-lo ao servidor ou para abortar o envio e processá-lo em JavaScript

```
<form id="form1">
    Nome: <input type="text" required><br>
    Idade: <input type="number" min="0" max="120" required><br>
    Carta de condução: <input type="checkbox"><br>
    <input type="submit" value="registar">
</form>
<script>
    const myForm = document.getElementById("form1")
    myForm.addEventListener("submit", function () {
        // Validar
    })
</script>
```

validações do JS

validações do HTML5

evento submit

M03 - Document Object Model (DOM)

1. Document Object Model

Formulários

- Submissão
 - Uso de `event.preventDefault()` para prevenir a ação predefinida do evento
 - Neste caso a sua chamada faz com que o form não seja submetido ao servidor
 - Passagem no manipulador do objeto `event` como parâmetro

```
<form id="form1">
  Nome: <input id="name" type="text" required><br>
  Idade: <input id="age" type="number" min="0" max="120" required><br>
  Carta de condução: <input id="drivingLicense" type="checkbox"><br>
  <input type="submit" value="registar">
</form>
```

```
const myForm = document.getElementById("form1")           | objeto event
                                                       ↑
myForm.addEventListener("submit", function (event) {      |
  // Validar
  if (myForm.age.value < 18 && myForm.drivingLicense.checked === true) {
    alert("Não pode ter carta com essa idade!")
    event.preventDefault()                                | método preventDefault()
  } else {
    alert("Tudo OK, o form será submetido ao servidor!")
  }
})
```

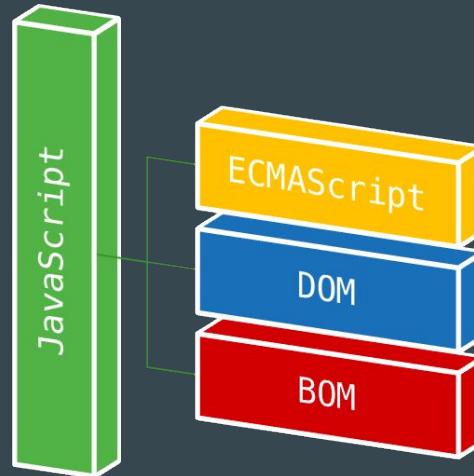
M03 - Document Object Model (DOM)

2. Browser Object Model

M03 - Document Object Model (DOM)

2. Browser Object Model

- O Browser Object Model (BOM) permite que o JavaScript "fale com" o navegador
- Principais objetos:
 - Window
 - Screen
 - Location
 - History
 - Navigator
 - Modals
 - Timing



M03 - Document Object Model (DOM)

1. Browser Object Model

- Objeto `window`

- representa a janela do navegador
- suportado por todos os navegadores
- todos os objetos, funções e variáveis globais do JS tornam-se automaticamente membros
 - Variáveis globais são propriedades do objeto da janela.
 - Funções globais são métodos do objeto de janela.
 - Mesmo o objeto `document` (do HTML DOM) é uma propriedade do objeto de janela
- Principais propriedades e métodos:
 - `window.innerHeight` - a altura interna da janela do navegador (em pixels)
 - `window.innerWidth` - a largura interna da janela do navegador (em pixels)
 - `window.open()` - abre uma nova janela
 - `window.close()` - fecha a janela atual
 - `window.moveTo()` - move a janela atual
 - `window.resizeTo()` - redimensiona a janela atual

M03 - Document Object Model (DOM)

1. Browser Object Model

- Objeto screen

- contém informações sobre o ecrã do utilizador
- pode ser escrito sem o prefixo window
- Principais propriedades e métodos:
 - `screen.width/screen.height` - retorna a largura/altura do ecrã do utilizador em pixels
 - `screen.availWidth/screen.availHeight` - retorna a largura/altura do ecrã do utilizador , em pixels, menos os recursos da interface, como a Barra de Tarefas do Windows
 - `screen.colorDepth` - retorna o número de bits usados para exibir uma cor.
 - Todos os computadores modernos usam hardware de 24 ou 32 bits para resolução de cor:
 - 24 bits = 16.777.216 diferentes "cores verdadeiras"
 - 32 bits = 4.294.967.296 diferentes "cores profundas"

M03 - Document Object Model (DOM)

1. Browser Object Model

- Objeto location

- pode ser usado para obter endereço da página atual (URL) e redirecionar o browser para nova página
- principais propriedades:
 - `window.location.href` - retorna o href (URL) da página atual
 - `window.location.hostname` - retorna o nome do domínio do host
 - `window.location.pathname` - retorna o caminho e o nome do arquivo da página atual
 - `window.location.protocol` - retorna o protocolo da web usado (http: ou https :)
 - `window.location.assign` - carrega um novo documento

```
alert(`A URL desta página é ${window.location.href}`);
```



M03 - Document Object Model (DOM)

1. Browser Object Model

- Objeto history

- contém o histórico do navegador
- principais métodos:
 - `history.back()` - o mesmo que clicar **back** no navegador
 - `history.forward()` - o mesmo que clicar **forward** no navegador



```
<input type="button" value="Back">

<script>
    const myButton = document.querySelector("input")
    myButton.addEventListener("click", function () {
        window.history.back()
    })
</script>
```

M03 - Document Object Model (DOM)

1. Browser Object Model

- Objeto **navigator**
 - contém informações sobre o navegador do visitante
 - Alguns exemplos:
 - `navigator.appName`
 - `navigator.appCodeName`
 - `navigator.platform`
 - As informações do objeto **navigator** podem muitas vezes ser enganosas e não devem ser usadas para detectar versões do navegador

M03 - Document Object Model (DOM)

1. Browser Object Model

- Eventos de sincronização
 - JavaScript pode ser executado em intervalos de tempo
 - Isso é chamado de eventos de sincronização
 - Os dois principais métodos para usar com JavaScript são:
 - `setTimeout(função, milissegundos)`
 - Executa uma função, após aguardar um número especificado de milissegundos
 - `setInterval(função, milissegundos)`
 - O mesmo que `setTimeout()`, mas repete a execução da função continuamente

```
<button onclick="setTimeout(myFunction, 3000)">Saudação após 3 segundos</button>
<button onclick="setInterval(myFunction, 3000)">Saudação de 3 em 3 segundos</button>

<script>
    function myFunction() {
        console.log('Olá Mundo!');
    }
</script>
```