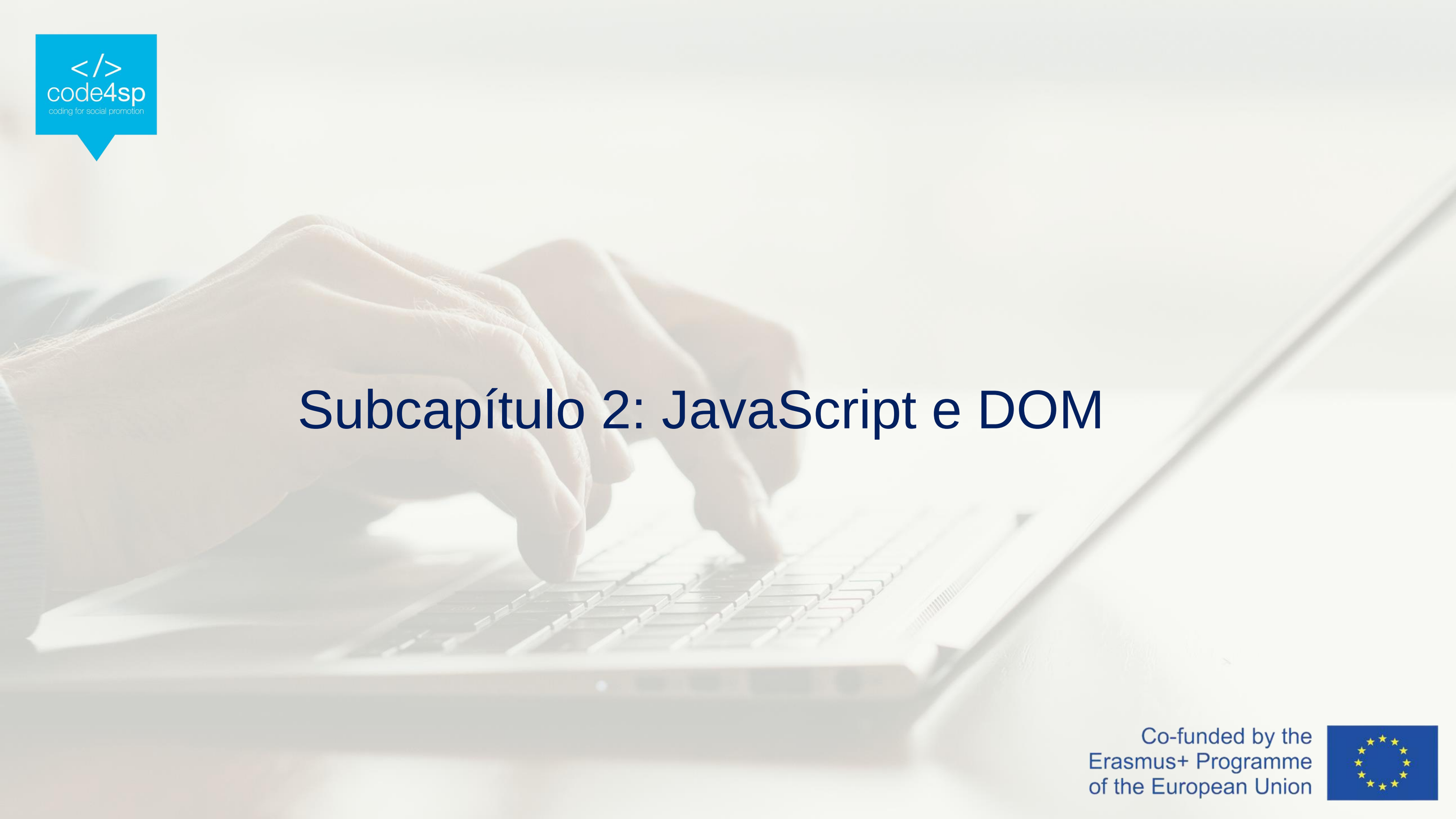
Co-funded by the
Erasmus+ Programme
of the European Union

code4sp
coding for social promotion

Materiais de Formação de JavaScript
Subcapítulo 2 – JavaScript e DOM

WP3: Materiais de Formação Code4SP

Preparado por: CITIZENS IN POWER



Subcapítulo 2: JavaScript e DOM

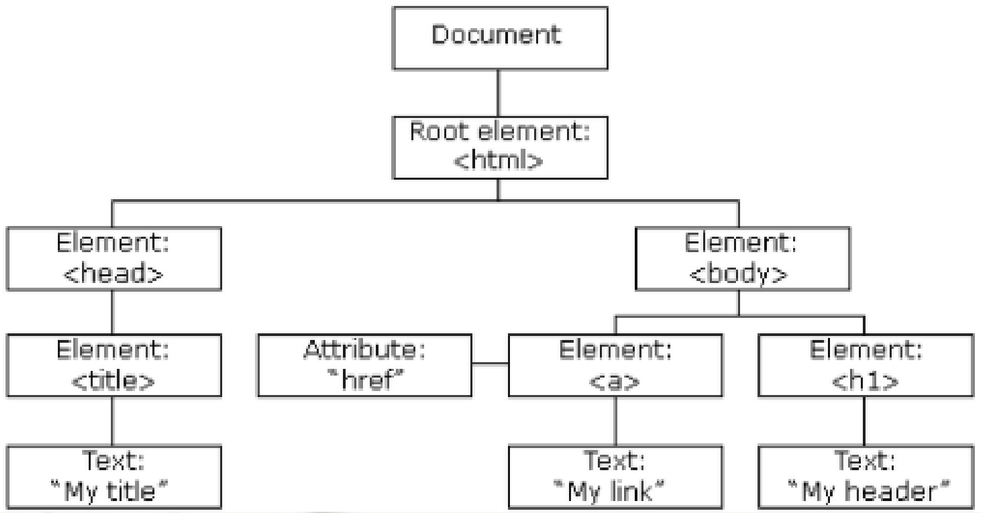
Co-funded by the
Erasmus+ Programme
of the European Union

O que é o Documento de Modelo do Objeto? (DOM)?

DOM é criado pelo browser quando uma página web é carregada em documentos HTML ou XML. É usada para definir as estruturas lógicas deste documento e para aceder e alterar os seus elementos.

Neste subcapítulo, iremos focar-nos no DOM HTML, que pode ser usado para aceder e manipular documentos HTML em JavaScript.

O DOM é construído numa árvore hierárquica de objetos, que incluem todas as partes de um documento HTML como elementos, atributos, texto, etc..



Co-funded by the
Erasmus+ Programme
of the European Union

O que pode o JavaScript fazer no DOM HTML?

Dentro do DOM HTML, o JavaScript pode fazer os seguintes:

- mudar todos os elementos HTML na página
- mudar todos os atributos HTML na página
- mudar todos os estilos CSS na página
- remove atributos e elementos HTML existentes
- remover todos os elementos e atributos HTML
- adicionar todos os elementos e atributos HTML
- reagir a todos os eventos HTML existentes na página
- criar novos eventos HTML existentes na página

Co-funded by the
Erasmus+ Programme
of the European Union

Selecionar Elementos DOM em JavaScript

O JavaScript é usado para **criar ou modificar conteúdos ou valores de elementos numa página web HTML**, assim como para aplicar alguns efeitos visuais como animações. .

Para ser capaz de realizar qualquer ação, é necessário encontrar ou selecionar um elemento alvo em HTML.

Iremos analisar algumas das formas mais comuns de selecionar elementos numa página e de os manipular com JavaScript.

Co-funded by the
Erasmus+ Programme
of the European Union

Selecionar os Elementos *Topmost*

Os elementos *topmost* podem ser acedidos diretamente como propriedades de um documento.

Por exemplo, para aceder ao elemento <html>, deve usar-se a propriedade document.documentElement. Para o elemento <head>, podemos usar a propriedade document.head e, para o elemento <body>, usamos a propriedade document.body.

Co-funded by the
Erasmus+ Programme
of the European Union

Selecionar os Elementos *Topmost*

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Topmost Elements</title>
</head>
<body>
    <script>
    // Display lang attribute value of html element
    alert(document.documentElement.getAttribute("lang")); // Outputs: en

    // Set background color of body element
    document.body.style.background = "yellow";

    // Display tag name of the head element's first child
    alert(document.head.firstElementChild.nodeName); // Outputs: meta
    </script>
</body>
</html>
```

É importante considerar que o document.body não deve ser usado antes do elemento <body>, pois, se tal acontecer, irá retornar como nulo. Este programa necessita de ir, primeiramente, pelo elemento <body> para que se possa aceder à propriedade document.body.

Co-funded by the
Erasmus+ Programme
of the European Union

Selecionar os Elementos *Topmost*

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Document.body Demo</title>
    <script>
    alert("From HEAD: " + document.body); // Outputs: null (since <body> is not parsed yet)
    </script>
</head>
<body>
    <script>
    alert("From BODY: " + document.body); // Outputs: HTMLBodyElement
    </script>
</body>
</html>
```

Este exemplo demonstra o que vimos no início sobre as relações hierárquicas que existem entre os nós. É necessário ser calculoso no sentido em que, de forma a aceder à propriedade document.body, irá ser necessário começar pelo elemento <body> para evitar valores nulos.

Co-funded by the
Erasmus+ Programme
of the European Union

Selecionar Elementos pelo ID

Se quiser encontrar o selecionar um elemento HTML, a forma mais simples seria selecioná-lo com base no seu ID único. É possível fazer isto através do método getElementById()..

O método getElementById(). é usado para devolver um elemento como objeto se for encontrado um objeto correspondente. De outra maneira, irá regressar nulo.

* É preciso relembrar que qualquer elemento HTML pode ter um atributo, que deve ser um valor único na página. Isto significa que apenas um elemento pode ter o mesmo id.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Element by ID</title>
</head>
<body>
    <p id="mark">This is a paragraph of text.</p>
    <p>This is another paragraph of text.</p>

    <script>
    // Selecting element with id mark
    var match = document.getElementById("mark");

    // Highlighting element's background
    match.style.background = "yellow";
    </script>
</body>
</html>
```

Co-funded by the Erasmus+ Programme of the European Union

Selecionar Elementos pelo Nome da Classe

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Elements by Class Name</title>
</head>
<body>
    <p class="test">This is a paragraph of text.</p>
    <div class="block test">This is another paragraph of text.</div>
    <p>This is one more paragraph of text.</p>

    <script>
    // Selecting elements with class test
    var matches = document.getElementsByClassName("test");

    // Displaying the selected elements count
    document.write("Number of selected elements: " + matches.length);

    // Applying bold style to first element in selection
    matches[0].style.fontWeight = "bold";

    // Applying italic style to last element in selection
    matches[matches.length - 1].style.fontStyle = "italic";

    // Highlighting each element's background through loop
    for(var elem in matches) {
        matches[elem].style.background = "yellow";
    }
    </script>
</body>
</html>
```

Se quiser selecionar todos os elementos com nomes de classes específicos, use o método getElementsByClassName(). Irá devolver um objeto semelhante a uma matriz com todos os seus elementos criança, que têm todos o nome de classe atribuído.

Co-funded by the Erasmus+ Programme of the European Union

Selecionar Elementos pelo Nome da *Tag*

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Elements by Tag Name</title>
</head>
<body>
    <p>This is a paragraph of text.</p>
    <div class="test">This is another paragraph of text.</div>
    <p>This is one more paragraph of text.</p>

    <script>
    // Selecting all paragraph elements
    var matches = document.getElementsByTagName("p");

    // Printing the number of selected paragraphs
    document.write("Number of selected elements: " + matches.length);

    // Highlighting each paragraph's background through loop
    for(var elem in matches) {
        matches[elem].style.background = "yellow";
    }
    </script>
</body>
</html>
```

Se quiser selecionar elementos pelo seu nome da *tag*, use o método getElementsByTagName(). Este método irá também devolver um objeto semelhante a uma matriz com todos os seus elementos criança, que têm todos o nome de classe atribuído.

Co-funded by the
Erasmus+ Programme
of the European Union

Selecionar Elementos com Seletores CSS

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Select Elements with CSS Selectors</title>
</head>
<body>
    <ul>
        <li>Bread</li>
        <li class="tick">Coffee</li>
        <li>Pineapple Cake</li>
    </ul>

    <script>
    // Selecting all li elements
    var matches = document.querySelectorAll("ul li");

    // Printing the number of selected li elements
    document.write("Number of selected elements: " + matches.length + "<hr>")

    // Printing the content of selected li elements
    for(var elem of matches) {
        document.write(elem.innerHTML + "<br>");
    }

    // Applying line through style to first li element with class tick
    matches = document.querySelectorAll("ul li.tick");
    matches[0].style.textDecoration = "line-through";
    </script>
</body>
</html>
```

Os seletores CSS oferecem uma forma potente e eficiente de selecionar elementos HTML num documento.

Para selecionar elementos que combinem com um seletor de CSS específico, podemos usar o método querySelectorAll().

Este método irá devolver uma lista de todos os elementos que combinam com os seletores específicos.

Co-funded by the Erasmus+ Programme of the European Union

Estilo dos Elementos DOM em JavaScript

É também possível mudar a apresentação visual de um documento HTML de uma forma dinâmica ao usar o JavaScript para aplicar diferentes estilos aos elementos HTML. Quase todos os estilos de elementos podem ser adicionados, como fontes, cores, margens, bordas, imagens de fundo, alinhamento de texto, largura e altura, posição, entre outros.

Aqui iremos ver vários métodos que podem ser usados para definir estilos em JavaScript.

Co-funded by the
Erasmus+ Programme
of the European Union

Definir Estilos Inline nos Elementos

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Set Inline Styles Demo</title>
</head>
<body>
    <p id="intro">This is a paragraph.</p>
    <p>This is another paragraph.</p>

    <script>
    // Selecting element
    var elem = document.getElementById("intro");

    // Appling styles on element
    elem.style.color = "blue";
    elem.style.fontSize = "18px";
    elem.style.fontWeight = "bold";
    </script>
</body>
</html>
```

Este atributo de estilo é usado para aplicar estilos inline diretamente no elemento HTML específico. A propriedade style é usada no JavaScript para ter ou definir o estilo inline num documento.

No seguinte exemplo, as propriedades das cores e fontes serão definidas para um elemento com id="intro"

Co-funded by the
Erasmus+ Programme
of the European Union

Denominar Convenções de Propriedades CSS em JavaScript

É importante mencionar que muitas das propriedades de CSS contêm hífens (-) nos seus nomes como font-size, background-image, text-decoration, etc. No entanto, em JavaScript, o hífen é um operador reservado que significa o sinal menos. Posto isto, não é possível escrever uma expressão da seguinte maneira: elem.style.font-size.

De forma a resolver este problema, o nome das propriedades CSS em JavaScript que contêm um ou mais hífens são convertidos em estilos com palavras subcapitalizadas. Isto, em resumo, significa que os hífens são removidos e a primeira letra após o hifen é maiúscula. Por exemplo, a propriedade CSS font-size, torna-se em propriedade DOM fontSize.

Co-funded by the
Erasmus+ Programme
of the European Union

Obter a Informação do Estilo dos Elementos

A propriedade de estilo é também usada para aplicar os estilos aos elementos HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JS Get Element's Style Demo</title>
</head>
<body>
    <p id="intro" style="color:red; font-size:20px;">This is a paragraph.</p>
    <p>This is another paragraph.</p>

    <script>
    // Selecting element
    var elem = document.getElementById("intro");

    // Getting style information from element
    alert(elem.style.color);  // Outputs: red
    alert(elem.style.fontSize);  // Outputs: 20px
    alert(elem.style.fontStyle);  // Outputs nothing
    </script>
</body>
</html>
```

A propriedade de estilo não é a mais útil na questão de recolher informação dos elementos, pois apenas devolve as regras de estilo que foram aplicadas no atributo de estilo do elemento e não naqueles que vieram de outros locais como regras de estilo das folhas de estilo implementadas ou em folhas de estilo externas.

Co-funded by the
Erasmus+ Programme
of the European Union

Obter a Informação do Estilo dos Elementos

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>JS Get Computed Style Demo</title>
<style type="text/css">
    #intro {
        font-weight: bold;
        font-style: italic;
    }
</style>
</head>
<body>
    <p id="intro" style="color:red; font-size:20px;">This is a paragraph.</p>
    <p>This is another paragraph.</p>

    <script>
    // Selecting element
    var elem = document.getElementById("intro");

    // Getting computed style information
    var styles = window.getComputedStyle(elem);

    alert(styles.getPropertyValue("color"));  // Outputs: rgb(255, 0, 0)
    alert(styles.getPropertyValue("font-size"));  // Outputs: 20px
    alert(styles.getPropertyValue("font-weight"));  // Outputs: 700
    alert(styles.getPropertyValue("font-style"));  // Outputs: italic
    </script>
</body>
</html>
```

Se quiser obter os valores de todas as propriedades CSS que são usadas para renderizar um objeto, pode utilizar o método window.getComputedStyle(), como mostrado no seguinte exemplo:

* É preciso lembrar que o valor 700 para a propriedade font-weight do CSS é o mesmo que o negrito que usamos normalmente. A cor vermelha usada normalmente corresponde ao rgb(255,0,0), o que corresponde à notação rgb de uma cor.

Co-funded by the Erasmus+ Programme of the European Union

Adicionar Classes CSS a Elementos

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>JS Add or Replace CSS Classes Demo</title>
<style>
    .highlight {
        background: yellow;
    }
</style>
</head>
<body>
    <div id="info" class="disabled">Something very important!</div>

    <script>
    // Selecting element
    var elem = document.getElementById("info");

    elem.className = "note";  // Add or replace all classes with note class
    elem.className += " highlight";  // Add a new class highlight
    </script>
</body>
</html>
```

Outra forma de obter classes CSS em elementos HTML é usar a propriedade className. Uma classe é uma palavra reservada no JavaScript; posto isto, o JavaScript usa a propriedade className para se referir ao valor do atributo de classe HTML.

Co-funded by the Erasmus+ Programme of the European Union

Adicionar Classes CSS a Elementos

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>JS classList Demo</title>
<style>
    .highlight {
        background: yellow;
    }
</style>
</head>
<body>
    <div id="info" class="disabled">Something very important!</div>

    <script>
    // Selecting element
    var elem = document.getElementById("info");

    elem.classList.add("hide");  // Add a new class
    elem.classList.add("note", "highlight");  // Add multiple classes
    elem.classList.remove("hide"); // Remove a class
    elem.classList.remove("disabled", "note"); // Remove multiple classes
    elem.classList.toggle("visible"); // If class exists remove it, if not add it

    // Determine if class exist
    if(elem.classList.contains("highlight")) {
        alert("The specified class exists on the element.");
    }
    </script>
</body>
</html>
```

Uma forma ainda melhor de trabalhar com classes CSS é usando a propriedade classList para obter, estabelecer ou remover simplesmente as classes CSS de um elemento. Esta propriedade é suportada em todos os maiores browsers exceto as edições anteriores ao Internet Explorer 10.

Co-funded by the Erasmus+ Programme of the European Union

Trabalhar com atributos

Os atributos são uma palavra especial usada, muitas vezes, dentro do inicio de uma *tag* de um elemento HTML de forma a controlar o comportamento da *tag* ou para obter mais informações sobre a *tag*.

Nesta secção, iremos analisar vários métodos de adição, remoção ou alteração dos atributos de um elemento HTML.

Co-funded by the
Erasmus+ Programme
of the European Union

Obter o Valor do Atributo do Elemento

Para obter o valor corrente do atributo de um elemento, temos que usar o método getAttribute().

Se este atributo em particular não for encontrado no elemento, a resposta vai ser nula.

```
<a href="https://www.google.com/" target="_blank" id="myLink">Google</a>

<script>
    // Selecting the element by ID attribute
    var link = document.getElementById("myLink");

    // Getting the attributes values
    var href = link.getAttribute("href");
    alert(href); // Outputs: https://www.google.com/

    var target = link.getAttribute("target");
    alert(target); // Outputs: _blank
</script>
```

Co-funded by the
Erasmus+ Programme
of the European Union

Definir Atributos nos Elementos

Se quiser definir um atributo num elemento especifico, poderá usar o método setAttribute(). Se o atributo já existir nesse elemento, o valor será atualizado. Se não, será adicionar um novo atributo com o nome e o valor específico.

```
<button type="button" id="myBtn">Click Me</button>

<script>
    // Selecting the element
    var btn = document.getElementById("myBtn");

    // Setting new attributes
    btn.setAttribute("class", "click-btn");
    btn.setAttribute("disabled", "");
</script>
```

Co-funded by the
Erasmus+ Programme
of the European Union

Definir Atributos nos Elementos

Se quiser atualizar ou alterar o valor de um atributo existente de um elemento, pode usar o método setAttribute().

Vejamos um exemplo que irá atualizar o valor de um atributo href existente de uma âncora (<a>) do elemento:

```
<a href="#" id="myLink">Tutorial Republic</a>

<script>
    // Selecting the element
    var link = document.getElementById("myLink");

    // Changing the href attribute value
    link.setAttribute("href", "https://www.tutorialrepublic.com");
</script>
```

Co-funded by the
Erasmus+ Programme
of the European Union

Remover Atributos dos Elementos

Para remover um atributo de um elemento específico, podemos usar o método removeAttribute()..

Tenha em mente o atributo href que alteramos do elemento âncora; iremos agora removê-lo no seguinte exemplo:

```
<a href="https://www.google.com/" id="myLink">Google</a>

<script>
    // Selecting the element
    var link = document.getElementById("myLink");

    // Removing the href attribute
    link.removeAttribute("href");
</script>
```

Co-funded by the
Erasmus+ Programme
of the European Union

Manipular Elementos DOM em JavaScript

Até agora, aprendemos como selecionar e alterar o estilo dos elementos DOM HTML.

Agora, irmos aprender como adicionar ou remover elementos DOM de uma forma dinâmica, como obter os seus conteúdos e muito mais.

Co-funded by the
Erasmus+ Programme
of the European Union

Adicionar Novos Elementos ao DOM

```
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>

<script>
// Creating a new div element
var newDiv = document.createElement("div");

// Creating a text node
var newContent = document.createTextNode("Hi, how are you doing?");

// Adding the text node to the newly created div
newDiv.appendChild(newContent);

// Adding the newly created element and its content into the DOM
var currentDiv = document.getElementById("main");
document.body.appendChild(newDiv, currentDiv);
</script>
```

O método document.createElement() é usado para criar um novo elemento num documento HTML. Este cria um novo elemento, no entanto, não o adiciona ao DOM.

Para adicioná-lo ao DOM é necessário um passo em separado. No exemplo da esquerda, o appendChild() é usado para adicionar um novo elemento ao fim de cada outro nó filho, dentro do nó de parente específico.

Co-funded by the Erasmus+ Programme of the European Union

Adicionar Novos Elementos ao DOM

```
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>

<script>
// Creating a new div element
var newDiv = document.createElement("div");

// Creating a text node
var newContent = document.createTextNode("Hi, how are you doing?");

// Adding the text node to the newly created div
newDiv.appendChild(newContent);

// Adding the newly created element and its content into the DOM
var currentDiv = document.getElementById("main");
document.body.insertBefore(newDiv, currentDiv);
</script>
```

Tem também a opção de adicionar novos elementos antes de qualquer outro filho, como se pode ver no exemplo da esquerda.

Co-funded by the
Erasmus+ Programme
of the European Union

Obter ou Estabelecer Conteúdos HTML para DOM

Se quiser obter ou estabelecer os conteúdos de elementos HTML, pode usar a propriedade innerHTML. Esta propriedade é usada obter ou estabelecer um markup HTML dentro do elemento, que contem conteúdo entre as *tags* de abertura e de fecho.

```
1  <div id="main">
2      <h1 id="title">Hello World!</h1>
3      <p id="hint">This is a simple paragraph.</p>
4  </div>
5
6  <script>
7  // Getting inner HTML conents
8  var contents = document.getElementById("main").innerHTML;
9  alert(contents); // Outputs inner html contents
10
11 // Setting inner HTML contents
12 var mainDiv = document.getElementById("main");
13 mainDiv.innerHTML = "<p>This is <em>newly inserted</em> paragraph.</p>";
14 </script>
```

Como é possível observar no exemplo, os novos elementos são inseridos com facilidade no DOM através da propriedade innerHTML. Mas esta propriedade substitui todo o conteúdo existente de um elemento.

Co-funded by the
Erasmus+ Programme
of the European Union

Obter ou Estabelecer Conteúdos HTML para DOM

```
<!-- beforebegin -->
<div id="main">
    <!-- afterbegin -->
    <h1 id="title">Hello World!</h1>
    <!-- beforeend -->
</div>
<!-- afterend -->

<script>
// Selecting target element
var mainDiv = document.getElementById("main");

// Inserting HTML just before the element itself, as a previous sibling
mainDiv.insertAdjacentHTML('beforebegin', '<p>This is paragraph one.</p>');

// Inserting HTML just inside the element, before its first child
mainDiv.insertAdjacentHTML('afterbegin', '<p>This is paragraph two.</p>');

// Inserting HTML just inside the element, after its last child
mainDiv.insertAdjacentHTML('beforeend', '<p>This is paragraph three.</p>');

// Inserting HTML just after the element itself, as a next sibling
mainDiv.insertAdjacentHTML('afterend', '<p>This is paragraph four.</p>');
</script>
```

Se não quiser alterar o conteúdo existente de um elemento, deve usar o método insertAdjacentHTML().

Este método requer dois fatores: a inserção do HTML e a sua posição.

Este método requer dois fatores: a inserção do HTML e a sua posição. A posição deve ser uma das seguintes: “beforebegin", "afterbegin", "beforeend", e "afterend". É importante notar que este método é suportado por todos os navegadores de grande dimensão.

Co-funded by the Erasmus+ Programme of the European Union

Remover Elementos Existentes do DOM

Para remover um nó filho do DOM, pode-se usar o método removeChild(). Este método irá, também, devolver o nó removido.

```
1  <div id="main">
2      <h1 id="title">Hello World!</h1>
3      <p id="hint">This is a simple paragraph.</p>
4  </div>
5
6  <script>
7  var parentElem = document.getElementById("main");
8  var childElem = document.getElementById("hint");
9  parentElem.removeChild(childElem);
10 </script>
```

Co-funded by the
Erasmus+ Programme
of the European Union

Remover Elementos Existentes do DOM

É também possível remover o elemento filho sem saber o elemento parente. Pode encontrar o elemento filho e usar a propriedade parentNode para encontrar o seu parente. Irá devolver o parente desse nó à árvore DOM.

```
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>

<script>
var childElem = document.getElementById("hint");
childElem.parentNode.removeChild(childElem);
</script>
```

Co-funded by the
Erasmus+ Programme
of the European Union

Substituir Elementos Existentes em DOM

```
<div id="main">
    <h1 id="title">Hello World!</h1>
    <p id="hint">This is a simple paragraph.</p>
</div>

<script>
var parentElem = document.getElementById("main");
var oldPara = document.getElementById("hint");

// Creating new elememt
var newPara = document.createElement("p");
var newContent = document.createTextNode("This is a new paragraph.");
newPara.appendChild(newContent);

// Replacing old paragraph with newly created paragraph
parentElem.replaceChild(newPara, oldPara);
</script>
```

Pode também ter a opção de substituir um elemento em HTML DOM por outro ao usar o método replaceChild().

Este método requer dois parâmetros: Este método requer dois parâmetros: que o nó esteja inserido e que o nó seja substituído.

A sintaxe usada é a seguinte: parentNode.replaceChild(newChild, oldChild);

Co-funded by the Erasmus+ Programme of the European Union

Navegar Entre Nós do DOM

Neste momento já deve ter uma melhor ideia sobre como selecionar elementos individuais numa página web. Existem várias ocasiões onde é preciso aceder a elementos filhos, parentes ou *antepassados*. Fomos falando em nós desde o início deste subcapítulo e vamos agora ver como podemos aceder aos diferentes tipos de nós.

Os nós DOM têm múltiplas propriedades e métodos que permitem navegar ou percorrer a estrutura das árvores DOM e fazer as alterações necessárias de uma forma simples.

Co-funded by the
Erasmus+ Programme
of the European Union

Aceder a Nós Filhos

As propriedades firstChild e lastChild cedem o acesso direto ao primeiro e ao último nó respetivamente. Se um nó não tiver nenhum elemento filho, será devolvido como nulo.

```
<div id="main">
    <h1 id="title">My Heading</h1>
    <p id="hint"><span>This is some text.</span></p>
</div>

<script>
var main = document.getElementById("main");
console.log(main.firstChild.nodeName); // Prints: #text

var hint = document.getElementById("hint");
console.log(hint.firstChild.nodeName); // Prints: SPAN
</script>
```

* É necessário notar que nodeName é uma propriedade apenas de leitura, o que devolve o nome do nó atual como linha. Por exemplo, irá devolver o nome da *tag* de um elemento de um nó, #text para nó text, #comment para nó comment, #document para nó document, e muitos outros.

Co-funded by the Erasmus+ Programme of the European Union

Aceder a Nós Filhos

```
1  <div id="main">
2      <h1 id="title">My Heading</h1>
3      <p id="hint"><span>This is some text.</span></p>
4  </div>
5
6  <script>
7  var main = document.getElementById("main");
8  console.log(main.firstChild.nodeName); // Prints: #text
9
10 var hint = document.getElementById("hint");
11 console.log(hint.firstChild.nodeName); // Prints: SPAN
12 </script>
```

Neste exemplo, o nodeName do nó do primeiro filho do principal elemento DIV foi devolvido como #text em vez de H1.

Isto deveu-se à existência de um **espaço branco**, p.e.: espaços, linhas novas, tags, entre outros, são considerados **caracteres válidos e tornam-se parte da árvore DOM na forma de nós #text. Depois, a tag , que conrem uma nova linha antes do <h1>, irá criar o nó #text.**

Assim, a tag , que conrem uma nova linha antes do <h1>, irá criar o nó #text.

Co-funded by the Erasmus+ Programme of the European Union

Aceder a Nós Filhos

```
<div id="main">
    <h1 id="title">My Heading</h1>
    <p id="hint"><span>This is some text.</span></p>
</div>

<script>
var main = document.getElementById("main");
alert(main.firstElementChild.nodeName); // Outputs: H1
main.firstElementChild.style.color = "red";

var hint = document.getElementById("hint");
alert(hint.firstElementChild.nodeName); // Outputs: SPAN
hint.firstElementChild.style.color = "blue";
</script>
```

De forma a prevenir este problema com o firstChild e lastChild a devolverem nós #text e #comment, podemos usar as propriedades firstElementChild ou lastElementChild como alternativa.

Estas propriedades devolverão apenas o primeiro e último elemento do nó, respetivamente. No entanto, isto não irá funcionar nas versões anteriores do Internet Explorer 9.

Co-funded by the
Erasmus+ Programme
of the European Union

Aceder a Nós Filhos

```
<div id="main">
    <h1 id="title">My Heading</h1>
    <p id="hint"><span>This is some text.</span></p>
</div>

<script>
var main = document.getElementById("main");

// First check that the element has child nodes
if(main.hasChildNodes()) {
    var nodes = main.childNodes;

    // Loop through node list and display node name
    for(var i = 0; i < nodes.length; i++) {
        alert(nodes[i].nodeName);
    }
}
</script>
```

Para aceder a todos os nós filhos de um elemento dado, podemos usar a propriedade childNodes.

É necessário ter em conta que ao nó do primeiro filho é-lhe atribuído o índex de 1.

Aqui, os childNodes devolvem todos os nós filhos, incluindo nós não-elementos como nós de texto e comentários.

Co-funded by the Erasmus+ Programme of the European Union

Aceder a Nós Filhos

```
1  <div id="main">
2      <h1 id="title">My Heading</h1>
3      <p id="hint"><span>This is some text.</span></p>
4  </div>
5
6  <script>
7  var main = document.getElementById("main");
8
9  // First check that the element has child nodes
10 if(main.hasChildNodes()) {
11     var nodes = main.children;
12
13     // Loop through node list and display node name
14     for(var i = 0; i < nodes.length; i++) {
15         alert(nodes[i].nodeName);
16     }
17 }
18 </script>
```

Se quiser ter uma *coleção de elementos apenas*, terá de usar a propriedade children em substituição

Co-funded by the
Erasmus+ Programme
of the European Union

Aceder a Nós Parentes

Para aceder ao nó parente de um nó específico da árvore DOM, podemos usar a propriedade parentNode.

* É de salientar que a propriedade parentNode irá sempre retomar valores nulos para nós em documentos, pois estes não têm parentes.

```
<div id="main">
    <h1 id="title">My Heading</h1>
    <p id="hint"><span>This is some text.</span></p>
</div>

<script>
var hint = document.getElementById("hint");
alert(hint.parentNode.nodeName); // Outputs: DIV
alert(document.documentElement.parentNode.nodeName); // Outputs: #document
alert(document.parentNode); // Outputs: null
</script>
```

É bom saber que os nós *topmost* das árvores DOM pode ser acedidos diretamente como propriedades dos documentos. Vimos alguns exemplos sobre os nós *topmost* das árvores DOM em seleções anteriores como o elemento <html>, que pode ser acedido com a propriedade document.documentElement.

Co-funded by the Erasmus+ Programme of the European Union

Aceder a Nós Parentes

Existe também uma opção para obter apenas um nó de elemento com o parentElement, tal como está exemplificado no seguinte exemplo:

```
<div id="main">
    <h1 id="title">My Heading</h1>
    <p id="hint"><span>This is some text.</span></p>
</div>

<script>
var hint = document.getElementById("hint");
alert(hint.parentNode.nodeName); // Outputs: DIV
hint.parentNode.style.backgroundColor = "yellow";
</script>
```

Co-funded by the
Erasmus+ Programme
of the European Union

Aceder a Nós Irmãos

Para aceder aos anteriores e aos próximos nós numa árvore DOM, podemos usar, respetivamente, as propriedades previousSibling e nextSibling

```
1  <div id="main">
2      <h1 id="title">My Heading</h1>
3      <p id="hint"><span>This is some text.</span></p><hr>
4  </div>
5
6  <script>
7  var title = document.getElementById("title");
8  alert(title.previousSibling.nodeName); // Outputs: #text
9
10 var hint = document.getElementById("hint");
11 alert(hint.nextSibling.nodeName); // Outputs: HR
12 </script>
```

Co-funded by the
Erasmus+ Programme
of the European Union

Aceder a Nós Irmãos

Para avançar qualquer nó de texto com espaços em branco, podemos usar previousElementSibling e nextElementSibling como alternativas para obter os elementos irmãos anteriores e posteriores. Se não for encontrado nenhum irmão, estas propriedades irão ser devolvidas como valores nulos.

```
<div id="main">
    <h1 id="title">My Heading</h1>
    <p id="hint"><span>This is some text.</span></p>
</div>

<script>
var hint = document.getElementById("hint");
alert(hint.previousElementSibling.nodeName); // Outputs: H1
alert(hint.previousElementSibling.textContent); // Outputs: My Heading

var title = document.getElementById("title");
alert(title.nextElementSibling.nodeName); // Outputs: P
alert(title.nextElementSibling.textContent); // Outputs: This is some text.
</script>
```

A propriedade textContent que é usada aqui **significa o conteúdo de um nó e todos os seus descendentes.**

Co-funded by the
Erasmus+ Programme
of the European Union

Tipos de Nós DOM

Uma árvore DOM é composta por diferentes tipos de nós que incluem elementos, texto, comentários e muito mais. Todos os nós têm a propriedade nodeType que consegue ajudar a entender como pode aceder e manipular o nó em questão.

Constant	Value	Description
`ELEMENT_NODE`	1	An element node such as `<p>` or `<img>`.
`TEXT_NODE`	3	The actual text of element.
`COMMENT_NODE`	8	A comment node i.e. `<!-- some comment -->`
`DOCUMENT_NODE`	9	A document node i.e. the parent of `<html>` element.
`DOCUMENT_TYPE_NODE`	10	A document type node e.g. `<!DOCTYPE html>` for HTML5 documents.

Co-funded by the
Erasmus+ Programme
of the European Union

VAMOS PRATICAR!

É hora de colocar aquilo que aprendemos em prática!

Para isso, sigamos cada um dos links:

- https://www.w3resource.com/javascript-exercises/javascript-dom-exercises.php
- https://www.tutorialrepublic.com/javascript-examples.php

Co-funded by the
Erasmus+ Programme
of the European Union

OBRIGADO!

Subcapítulo seguinte: JavaScript & BOM

Co-funded by the
Erasmus+ Programme
of the European Union