

JavaScript no Browser

<DOM e Outros>

Linguagens Script @ LEI / LEI-PL / LEI-CE

Departamento de Engenharia Informática e de Sistemas

Cristiana Areias < cris@isec.pt >

2022/2023

JavaScript

JS no Browser

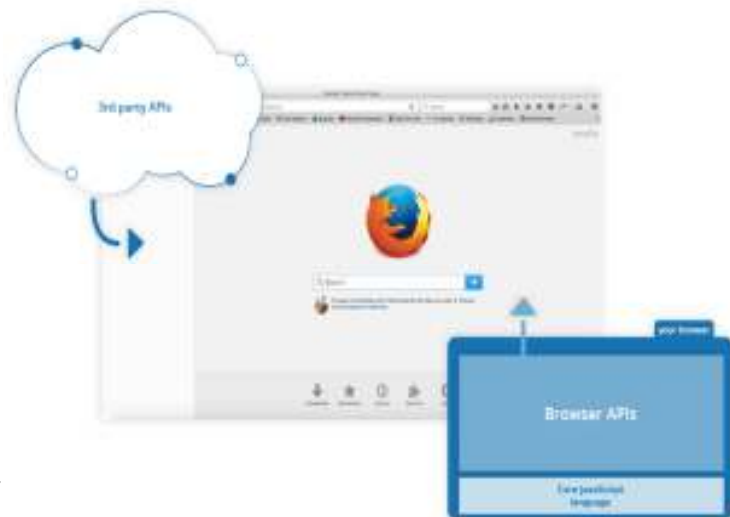
- › Inserção de Scripts
- › Document Object Model (DOM)
 - › Tipos de Objectos DOM
 - › Manipulação DOM
- › Eventos
 - › Declaração e Propagação (*bubbling* e *capturing*)
 - › Delegação de Eventos

> JavaScript no *Browser*

- Os browsers disponibilizam um conjunto de *Web Application Programming Interfaces* (APIs) que facilitam a implementação de páginas web, permitindo, por exemplo, exportar dados do ambiente, efetuar operações mais complexas, entre outras.

- Algumas Web APIs :

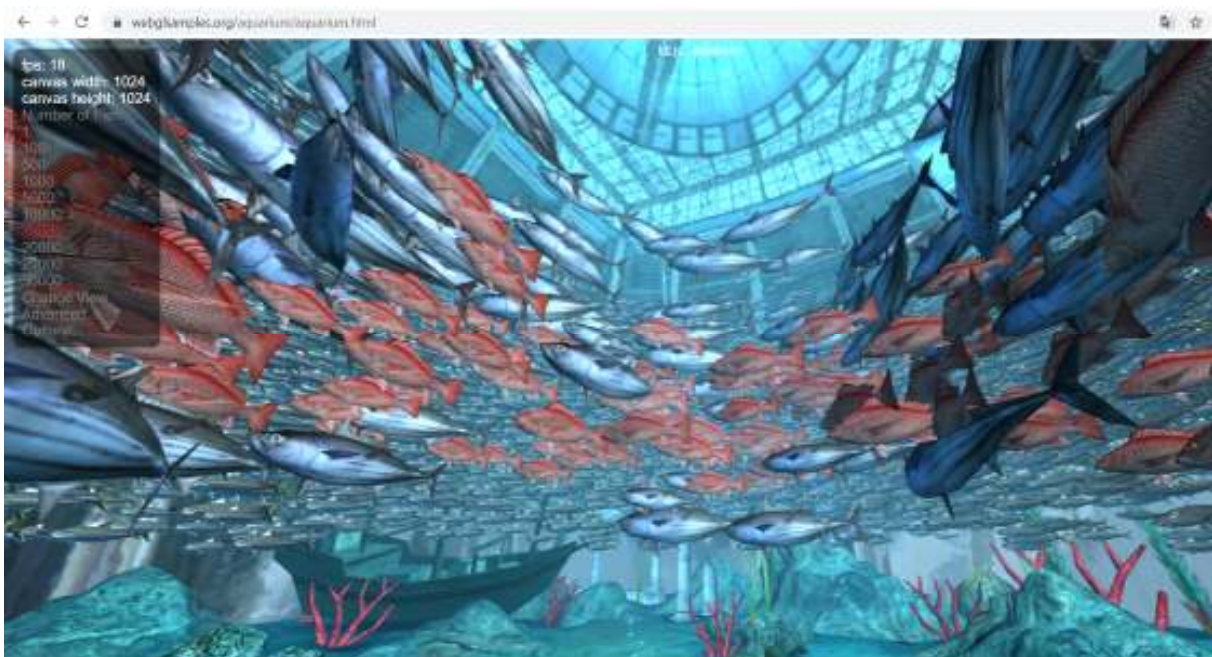
- API DOM
- API de Geolocalização
- API Canvas e WebGL
- APIs de áudio e vídeo
- APIs de “terceiros”:
 - API do *Google Maps*
 - Twitter*



<https://developer.mozilla.org/>

> JavaScript no Browser

- API Canvas e WebGL

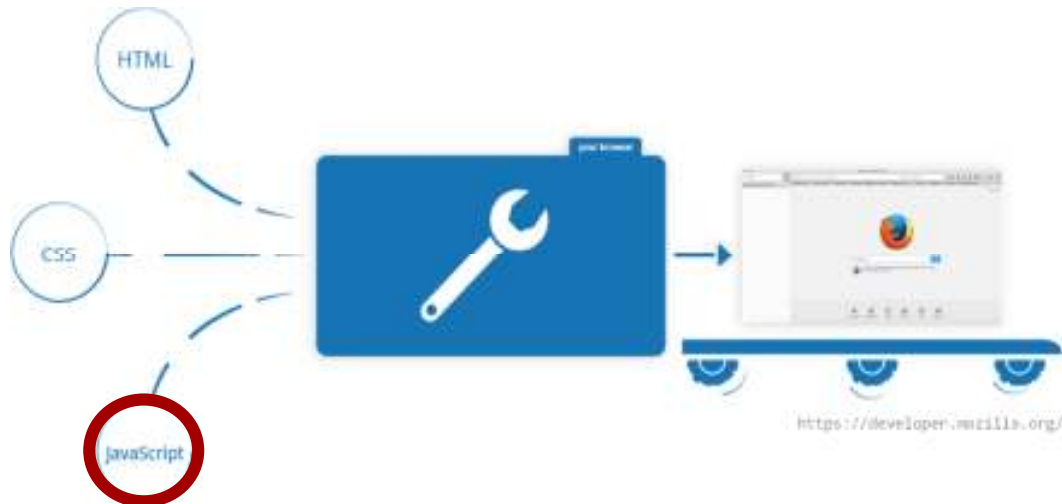


<https://webglsamples.org/aquarium/aquarium.html>

> Javascript no *Browser*

- O JavaScript é muitas vezes utilizado para modificar dinamicamente a estrutura e estilos de uma página web , alterando então o HTML e CSS, e desse modo atualizar uma interface do utilizador.

- **API do Document Object Model (DOM)**



> Inserção de *scripts*

- JavaScript Embebido

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Linguagens Script</title>
  <link rel="stylesheet" href="css/stylesLS.css" />
  <script>
    let strLS="Linguagens Script";
    let strJS="Javascript";

    if (strLS === "Linguagens Script") alert("Bem vindo a Linguagens Script");

    console.log(strJS);
  </script>
</head>
<body>
  <h1>Linguagens Script</h1>
</body>
</html>
```



> Inserção de *scripts*

■ JavaScript num ficheiro externo

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Linguagens Script</title>
  <link rel="stylesheet" href="css/stylesLS.css" />
</head>
<body>
  <h1>Linguagens Script</h1>
  <script src="js/script.js"></script>
</body>
</html>
```



script.js

```
let strLS="Linguagens Script";
let strJS="Javascript";

if (strLS === "Linguagens Script")
  alert("Bem vindo a Linguagens Script!");

console.log(strJS);
```



> Inserção de *scripts*

■ <script defer ...>

- Informa o *browser* para não esperar pelo carregamento total do script. Por sua vez, o browser continuará a processar o HTML, compilando o DOM.
- O script é carregado “em segundo plano” e, em seguida, é executado quando o DOM estiver totalmente construído;
- Não bloqueia a página;
- O scrip é sempre executado quando o DOM está pronto (mas antes do DOMContentLoaded event).



> Inserção de scripts

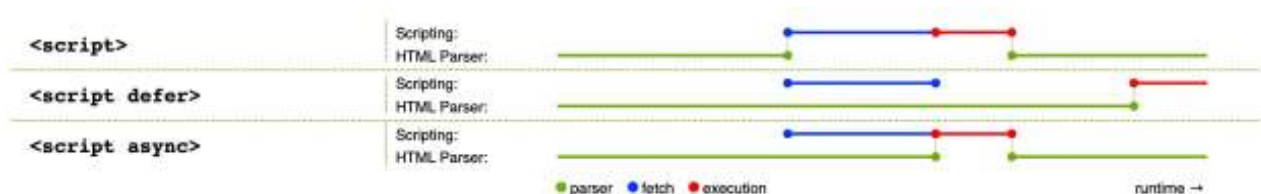
▪ <script async ...>

- O browser não bloqueia *scripts async* (como defer).
- Apenas para scripts externos;
- Os scripts são carregados em segundo plano e executados quando estiverem prontos.
 - O DOM e outros scripts não esperam por eles e não esperam por nada;
 - Significa que um script é completamente independente;
 - Outros scripts não esperam por scripts async, e scripts async não esperam por eles;
- Útil quando se pretende integrar um script de terceiros na página: contadores, anúncios...

```
<script async src="https://google-analytics.com/analytics.js"></script>
```

> Defer vs async

- **defer** é usado para scripts que precisam do DOM e/ou a sua ordem de execução é importante;
- **async** usado para scripts independentes, como anúncios, contadores, em que a ordem de execução não é importante.



<https://html.spec.whatwg.org/images/asyncdefer.svg>

> Document Object Model(DOM)

- A biblioteca DOM permite a interação com páginas web através do Javascript;
- Trata-se de uma representação estruturada de documentos HTML permitindo, usando JavaScript, o acesso e a manipulação de estilos e elementos HTML
 - Alteração de texto
 - Alteração dos atributos HTML
 - Alteração de estilos CSS
- Quando uma página é carregada o browser cria o Document Object Model (DOM) da página.
- API bem complexa que contém muitos métodos e propriedades para interagir com a árvore dom



<https://eloquentjavascript.net/>

> Document Object Model

- Representação estruturada de documentos HTML.
- Permite usar JavaScript para aceder e manipular elementos e estilos HTML, como exemplo:
 - Alteração de atributos HTML
 - Modificação do texto de um elemento HTML
 - Alteração de estilo



> Árvore DOM

- Criada automaticamente pelo browser quando a página é carregada
 - Cada elemento HTML é um objecto

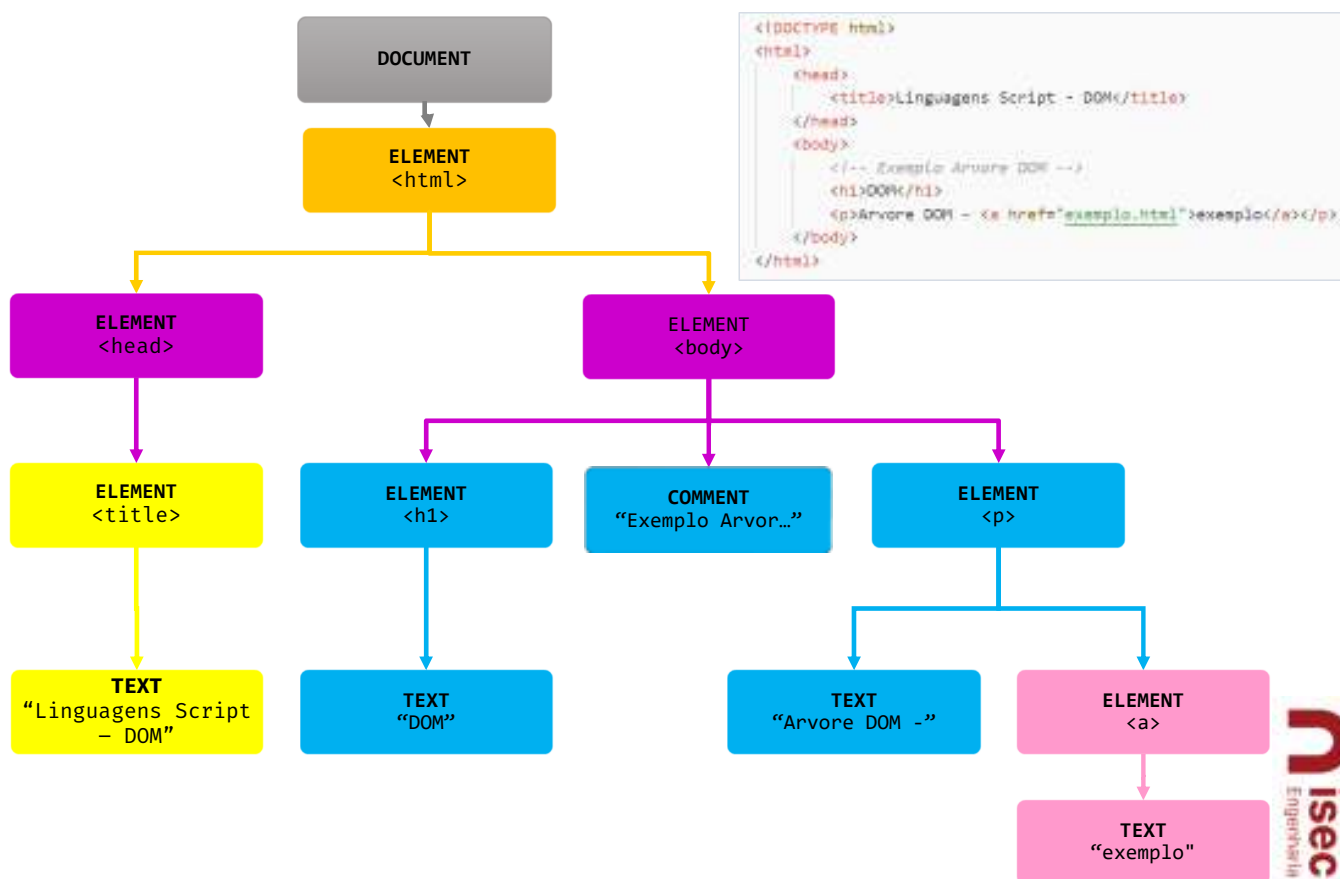
```
<!DOCTYPE html>
<html>
  <head>
    <title>Linguagens Script - DOM</title>
  </head>
  <body>
    <!-- Exemplo Arvore DOM -->
    <h1>DOM</h1>
    <p>Arvore DOM - <a href="exemplo.html">exemplo</a></p>
  </body>
</html>
```

```
DOCTYPE: html
HTML
  HEAD
    #text:
    TITLE
    #text: Linguagens Script - DOM
    #text:
  BODY
    #text:
    #comment: Exemplo Arvore DOM
    #text:
    H1
    #text: DOM
    #text:
    P
    #text: Arvore DOM -
    A href="exemplo.html"
    #text: exemplo
    #text:
```

Gerado em <https://software.hixie.ch/utilities/js/live-dom-viewer/>



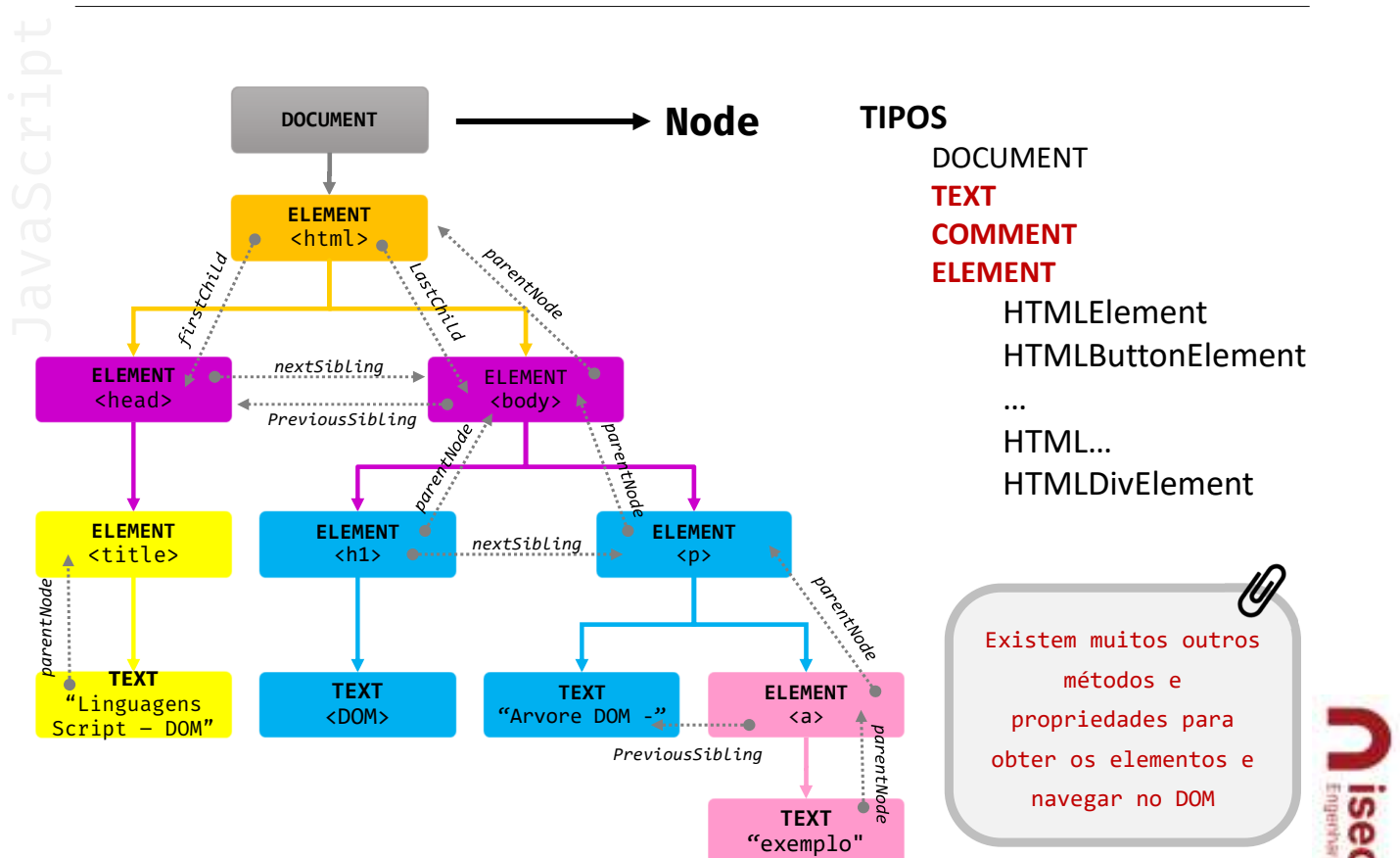
> Árvore DOM



> DOM > Tipos de Elementos

- Todos os elementos DOM são nós - **nodes**.
- Existem vários tipos de nós, entre eles, os mais utilizados:
 - **Element** <h1 >... </h1>
 - **Text**
 - **Comment**
- O documento é um nó, que é a raiz de todos os outros nós
 - **DOCUMENT_TYPE_NODE**
 - Ex: <!DOCTYPE HTML>
- Cada nó num documento é de um determinado um tipo, que é acedido através da propriedade **nodeType**.

> DOM > Tipos e Navegação



> DOM > Navegação

```
<html lang="en">
  <head>...</head>
  <body>
    <section id="container"> flex == $0
      
      <h1 data-toggle-id="lista-temas">
        <h2>Aulas Teóricas</h2>
        <ul id="lista-temas">...</ul>
        <script> </script>
      </section>
      <!-- Code injected by live-server -->
      <script type="text/javascript">...
    </body>
  </html>
```

```
> p=$0
< > <section id="container">...</section> flex
> p.parentNode
< > <body>...</body>
> p.parentNode.parentNode
< > <html lang="en">
  > <head>...</head>
  > <body>...</body>
  > </html>
> p.lastChild
< > #text
> p.ELEMENT_NODE
< > 1
> p.lastElementChild
< > <script> </script>
> ul = p.lastElementChild.previousElementSibling
< > <ul id="lista-temas">...</ul>
> |
```

> DOM > Aceder aos Elementos

```
document.getElement
```

```
getElementB... (method) Document.getElementById(elementId: string): HTMLElement
getElementsByClassName
getElementsByName
getElementsByTagName
getElementsByTagNameNS
```

```
(method) Document.getElementById(elementId: string): HTMLElement
Returns a reference to the first object with the specified value of the ID attribute.
@param elementId — String that specifies the ID value.
```

```
document.getElementById
```

```
(method) ParentNode.querySelector<K extends keyof HTMLElementTagNameMap>(selectors: K): HTMLElementTagNameMap[K] (+2 overloads)
```

Returns the first element that is a descendant of node that matches selectors.

```
document.querySelector
```

```
querySelector... (method) ParentNode.querySelector<K extends...>
querySelectorAll
```

> DOM >Estilos

```
<h1>Linguagens Script</h1>
<h2 id="msg">Aulas Teóricas</h2>
```

Linguagens Script
Aulas Teóricas

```
let mensagemTxt = "Teórica";
let mensagem=document.getElementById("msg");
mensagem.textContent=mensagemTxt;
mensagem.style.background="lightGrey";
mensagem.style.borderRadius="10px";
mensagem.style.width="90%"
mensagem.style.border="3px #440404 solid";
```



```
<h1>Linguagens Script</h1>
<h2 id="msg" style="background: lightgrey; border-radius: 10px;
width: 90%; border: 3px solid rgb(68, 4, 4);">Teórica</h2>
```

Linguagens Script
Teórica

> DOM >Estilos

```
const mensagemTxt = "Teórica";
const mensagem=document.getElementById("msg");
mensagem.textContent=mensagemTxt;
mensagem.style.setProperty('background',"lightGrey");
mensagem.style.setProperty('border-Radius',"10px");
mensagem.style.setProperty('width',"90%");
mensagem.style.setProperty('border',"3px #440404 solid");
```



```
console.log("Color = "+mensagem.style.color);
console.log("Border Radius = "+mensagem.style.borderRadius)
```

```
const cssObjMsg = window.getComputedStyle(mensagem,null);
console.log("Color = "+cssObjMsg.color);
console.log("Border Radius = "+cssObjMsg.borderRadius);
console.log("Border Radius = "+ cssObjMsg.getPropertyValue('border-Radius'));
```



```
<h1>Linguagens Script</h1>
<h2 id="msg" style="background: lightgrey; border-radius: 10px;
width: 90%; border: 3px solid rgb(68, 4, 4);">Teórica</h2>
```

> DOM >Estilos

```
<h1>Linguagens Script</h1>
<h2 id="msg" style="background: lightgrey; border-radius: 10px;
width: 90%; border: 3px solid rgb(68, 4, 4);">Teórica</h2>
```

```
console.log("Color = "+mensagem.style.color);
console.log("Border = "+mensagem.style.border);
```

```
Color = #ff0000  
Border = 3px solid rgb(68, 4, 4)
```



E os estilos das classes?

➤ DOM >Estilos e Atributos

```
const mensagemTxt = "Teórica";
const mensagem=document.getElementById("msg");
mensagem.textContent=mensagemTxt;
mensagem.style.setProperty('background',"lightGrey");
mensagem.style.setProperty('border-Radius',"10px");
mensagem.style.setProperty('width',"90%");
mensagem.style.setProperty('border',"3px #440404 solid");

console.log("Color = "+mensagem.style.color);
console.log("Border Radius = "+mensagem.style.borderRadius);
```

```
const cssObjMsg = window.getComputedStyle(mensagem, null);
console.log("Color = " + cssObjMsg.color);
console.log("Border Radius = " + cssObjMsg.borderRadius);
console.log("Border Radius = " + cssObjMsg.getPropertyValue('border-radius'));
```

```
Color = rgba(114, 111, 111, 0.69)
Border-Radius = 10px
Border-Radius = 10px
```

```
Color =
Border Radius = 10px
Color = rgba(114, 111, 111, 0.69)
Border Radius = 10px
Border Radius = 10px
```


> DOM > Métodos Manipulação Class

- `elemento.classList.add()`
 - `elemento.classList.add('c','p','d')`
- `elemento.classList.remove()`
- `elemento.classList.toggle()`
- `elemento.classList.contains()`
- `elemento.className="X"`
 - Remove todos os outros que possam existir!



> DOM > Atributos



```

<h1 id="uc">Linguagens Script</h1>
<h2 id="msg">Aulas Teóricas</h2>
```

```
const logoUC = document.querySelector("#logo");
console.log("Atributo id = "+logoUC.id);
console.log("Atributo src = "+logoUC.src);
console.log("Atributo title = "+logoUC.title);
console.log("Atributo alt = "+logoUC.alt);
```

Atributo id = logo

Atributo src = <http://127.0.0.1:5500/logoIsec.png>

Atributo title = Logotipo do ISEC

Atributo alt =



```
logoUC.src="js.png";
logoUC.title="Logotipo do Javascript"
logoUC.alt="Logotipo do Javascript"

Atributo id = logo
Atributo src = http://127.0.0.1:5500/js.png
Atributo title = Logotipo do Javascript
Atributo alt = Logotipo Javascript
```

> DOM > Atributos *não standard*

```

<h1 id="uc">Linguagens Script</h1>
<h2 id="msg">Aulas Teóricas</h2>
```

```
const logoUC = document.querySelector("#logo");
console.log("Atributo 'atributoExemplo'? "+logoUC.atributoExemplo);
console.log("Atributo 'atributoExemplo' ? "+logoUC.getAttribute("atributoExemplo"));
console.log("Atributo 'atributoNaoExistente' ? "+logoUC.atributoNaoExistente);
```

Atributo 'atributoExemplo'	undefined
Atributo 'atributoExemplo'	Teste
Atributo 'atributoNaoExistente'	undefined

> Atributos > *getAttribute*

```

<h1 id="uc">Linguagens Script</h1>
<h2 id="msg">Aulas Teóricas</h2>
```

```
logoUC.atributoExemplo="Demo";
logoUC.atributoNaoExistente="Demo2";
console.log("Atributo 'atributoExemplo' ? "+logoUC.atributoExemplo);
console.log("Atributo 'atributoNaoExistente' "+logoUC.atributoNaoExistente);

console.log("Atributo 'atributoNaoExistente' ? "+logoUC.getAttribute("atributoExemplo"));
console.log("Atributo 'atributoNaoExistente' ? "+logoUC.getAttribute("atributoNaoExistente"));
```

```

```

Atributo 'atributoExemplo' ?	Demo
Atributo 'atributoNaoExistente' ?	Demo2
Atributo 'atributoNaoExistente' ?	Teste
Atributo 'atributoNaoExistente' ?	null

> Atributos > *setAttribute*

```

<h1 id="uc">Linguagens Script</h1>
<h2 id="msg">Aulas Teóricas</h2>
```

```
logoUC.atributoExemplo="Demo";
logoUC.atributoNaoExistente="Demo2";
logoUC.setAttribute("atributoExemplo", "DemoFinal");
logoUC.setAttribute("atributoNaoExistente", "DemoFinal2");

console.log("Atributo 'atributoExemplo' ? "+logoUC.atributoExemplo);
console.log("Atributo 'atributoExemplo' ? "+logoUC.getAttribute("atributoExemplo"));
console.log("Atributo 'atributoNaoExistente' ? "+logoUC.atributoNaoExistente);
console.log("Atributo 'atributoNaoExistente' ? "+logoUC.getAttribute("atributoNaoExistente"));
```

```

```

```
Atributo 'atributoExemplo' ? Demo
Atributo 'atributoExemplo' ? DemoFinal
Atributo 'atributoNaoExistente' ? Demo2
Atributo 'atributoNaoExistente' ? DemoFinal2
```



> Atributos

- Caminhos relativos ou absolutos

```

<h1 id="uc">Linguagens Script</h1>
<h2 id="msg">Aulas Teóricas</h2>
```

```
const logoUC = document.querySelector("#logo");
console.log("Atributo src = "+logoUC.src);
console.log("Atributo src = "+logoUC.getAttribute("src"));
```

```
Atributo src = http://127.0.0.1:5500/logoIsec.png
Atributo src = logoIsec.png
```



> Atributos Data > data-*

- Atributo “especial”;
- Muito utilizados em manipulação da UI;
- Encontram-se armazenados no objeto *DataSet* que permite armazenamento de dados;
- Sintaxe:
 - **data-xxx-yyy**
 - Inicia com **data**, seguido do –
 - Para aceder:
 - **elemento.dataset.xxxYyy**
 - Usar camelCase e não – (como acontece com as *propriedades*)

> DOM > Eventos

- HTML DOM permite o JavaScript reagir a eventos HTML;
- **Ação que pode ser detetada pelo JavaScript, provocando uma determinada execução**
 - Quando termina de carregar o document, faz....;
 - Quando for detectado movimento do rato no link, muda a cor;
 - Quando clicar com o rato, expande texto;
 - Quando uma tecla é pressionada, faz.....
- Um evento HTML pode ser algo que o browser faz ou que o utilizador faz!
- O browser gera um evento sempre que algo acontece no **document**, no **browser** ou então num **element** ou **objecto** associado a ele.
- Os eventos podem ocorrer em qualquer elemento dentro de um documento HTML.

> DOM > Eventos

■ Manipulação DOM com Eventos

- **Quando** o carregamento da página acontecer,
 - executar o video.
- **Quando** o clique no botão proximo acontecer,
 - mostrar página nova
- **Quando** movimento do rato no painel surgir
 - mostrar foto
- ...



https://eloquentjavascript.net/15_event.html

UM EVENTO
QUANDO _____ ACONTECER,

EXECUTA _____
REAÇÃO A UM EVENTO

> Eventos > Formas de Declaração

HTML
*Event handler
Attributes*

Inline

- Recorre directamente a um **atributo**;
- **HTML Events** :
*onchange, onclick,
onmouseover,
onkeydown, onload,*
...

DOM
*Events
Handlers*

- Recorrendo a uma **propriedade**
*directamente no
elemento.*
- Ex:*
`element.onclick = func;`

DOM Events
EventListener

- Recorrendo ao método *EventListener*
- Permite adicionar vários ao mesmo ou remover;
- Mecanismo recomendado para adicionar *eventhandlers* em páginas web

> Evento > Declaração > *Inline* (1)

▪ *HTML Event handler Attributes / Inline*

```
<h1 onmouseenter="console.log('HTML Event: onmouseenter')">Linguagens Script</h1>
```



-----> HTML Event: onmouseenter

```
const mouseLog = function(e) {  
  console.log("mouseLog Function!");  
};
```

jscript.js

```
<h1 onmouseenter="mouseLog()">Linguagens Script</h1>
```

mouseLog Function!



> Evento > Declaração > (2)

▪ *DOM Event Handlers* - Recorrendo uma *propriedade diretamente no elemento*



```
h1.onmouseenter = function(e) {  
  console.log("onmouseenter: h1");  
};
```

-----> onmouseenter: h1

```
h1.onmouseenter = mouseLog;
```

-----> mouseLog Function!

```
h1.onmouseenter = mouseLog;  
h1.onmouseenter = function(e) {  
  console.log("onmouseenter: h1");  
};
```

}-----> onmouseenter: h1



> Evento > Declaração > (3)

■ DOM Event Listeners - addEventListener()

```
const h1 = document.querySelector('h1');  
h1.addEventListener('mouseenter', function(e) {  
  console.log("addEventListener mouseenter ");  
});
```

```
const mouseLog = function(e) {  
  console.log("mouseLog Function!");  
};
```

```
h1.addEventListener('mouseenter', mouseLog);
```

```
const img = document.querySelector('img');  
img.addEventListener('mouseenter', mouseLog);  
img.addEventListener('mouseenter', function(e) {  
  img.src="js.png";  
  console.log("Images Changed! ");  
});
```



addEventListener mouseenter

mouseLog Function!

mouseLog Function!

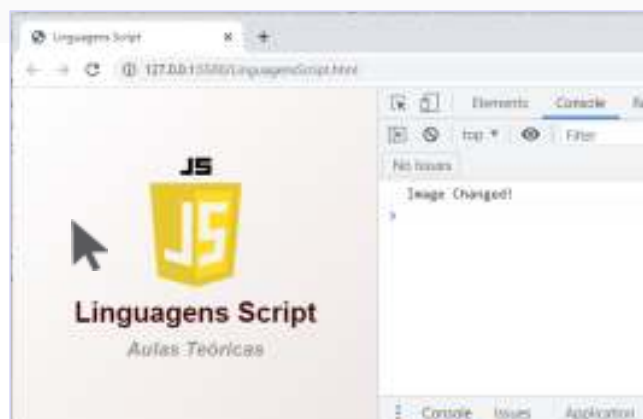
Images Changed!



> Eventos > Adicionar

■ DOM Event Listeners - addEventListener()

```
const img = document.querySelector('img');  
img.addEventListener('mouseenter', function(e) {  
  e.target.src="js.png";  
  console.log("Image Changed! ");  
});
```



> Eventos > Remove

- DOM Event Listeners - **removeEventListener()**

```
const toggleSrc = function(e) {  
  const src=e.target.getAttribute("src");  
  if (src=="js.png") {  
    e.target.src="logoIsec.png";  
    e.target.removeEventListener('mouseover',toggleSrc)  
  }  
  else  
    e.target.src="js.png";  
  console.log("Image Changed! "+src);  
}  
  
const img = document.querySelector('img');  
img.addEventListener('mouseover',toggleSrc );
```

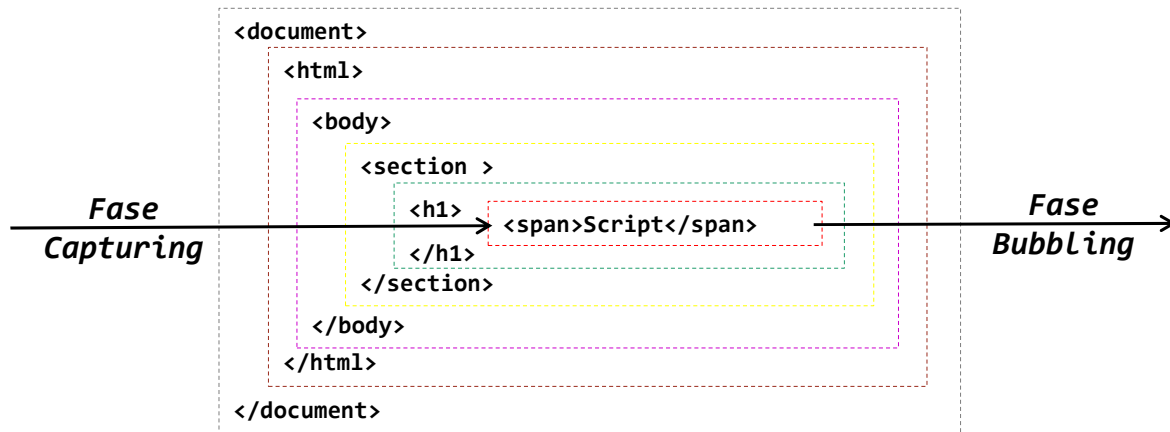


> Eventos > Propagação

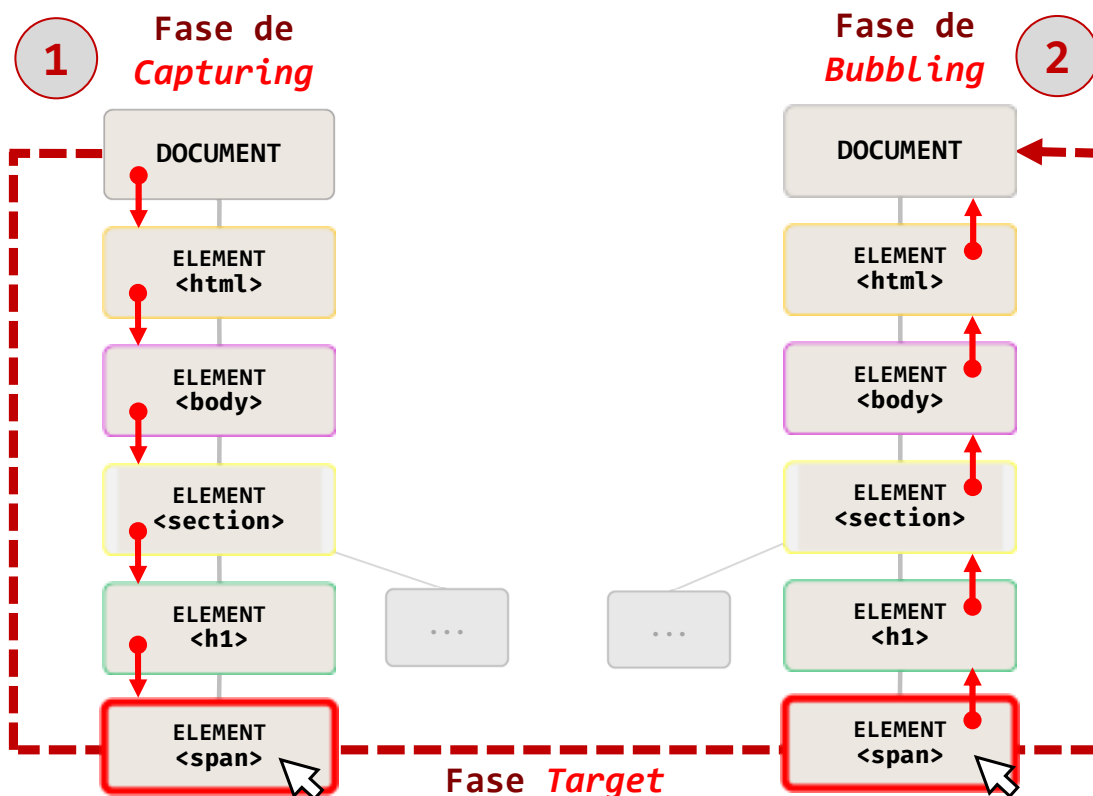
- Em JavaScript, existe **propagação de eventos** – a propagação do evento informa sobre o evento e a sua direcção de propagação.
- Existem dois tipos de abordagens na propagação de eventos:
 - Bubbling**
 - A direcção dos eventos circulam do **elemento filho para o seu elemento pai**
 - Usado como padrão quando se especifica o addEventListener()** sem o segundo parâmetro.
 - Capturing**
 - Raramente utilizada
 - Ordem de propagação é **oposta ao Bubbling**.
 - A captura dos eventos ocorre do **elemento pai para o elemento filho**
- Nem todos os eventos tem ambas as fases, alguns são criados directamente no elemento destino;

> Eventos > Capturing e Bubbling

```
<body>
  <section id="container">
    
    <h1>Linguagens <span>Script</span> </h1>
    <h2 id="msg">Aulas Teóricas</h2>
  </section>
</body>
```



> Eventos > Propagação



> Eventos > Propagação > Exemplo

```
const span=document.querySelector('span');
const h1=document.querySelector('h1');
const section=document.querySelector('section');
```

```
<body>
  <section id="container">
    
    <h1>Linguagens <span> Script </span> </h1>
    <h2 id="msg">Aulas Teóricas</h2>
  </section>
</body>
```

```
const randomColor = function(e){
  this.style.background=randomColorValue();
  this.style.cursor="pointer";
  console.log("Target:" +e.target.tagName+
    " - currentTarget:" + e.currentTarget.tagName+
    "\nMudou de cor para "+randomColorValue());
}
```

```
span.addEventListener("click",randomColor);
h1.addEventListener("click",randomColor);
section.addEventListener("click",randomColor);
```

> Eventos > Bubbling > Exemplo



Target:SPAN - currentTarget:SPAN
Mudou de cor para #532994 ●

Target:SPAN - currentTarget:H1
Mudou de cor para #ae33e ●

Target:SPAN - currentTarget:SECTION
Mudou de cor para #34929 ●



Target:H1 - currentTarget:H1
Mudou de cor para #c7102c ●

Target:H1 - currentTarget:SECTION
Mudou de cor para #4f838f ●



Target:SECTION - currentTarget:SECTION
Mudou de cor para #e1d22 ●

```
<body>
  <section id="container">
    
    <h1>Linguagens <span> Script </span> </h1>
    <h2 id="msg">Aulas Teóricas</h2>
  </section>
</body>
```

> Eventos > Capturing > Exemplo



Target:SPAN - currentTarget:SECTION
Mudou de cor para #aa6d74

Target:SPAN - currentTarget:H1
Mudou de cor para #99cb7

Target:SPAN - currentTarget:SPAN
Mudou de cor para #7fa49a



Target:H1 - currentTarget:SECTION
Mudou de cor para #570cac

Target:H1 - currentTarget:H1
Mudou de cor para #db40d0



Target:SECTION - currentTarget:SECTION
Mudou de cor para #816c1c

>

```
span.addEventListener("click",randomColor,true);  
h1.addEventListener("click",randomColor,true);  
section.addEventListener("click",randomColor,true);
```



> Eventos > Cancelar Propagação

- stopPropagation();

```
const randomColor = function(e){  
  this.style.background=randomColorValue();  
  this.style.cursor="pointer";  
  console.log("Target:"+e.target.tagName+  
    " - currentTarget:"+ e.currentTarget.tagName+  
    "\nMudou de cor para "+randomColorValue());  
  e.stopPropagation()  
}
```

```
span.addEventListener("click",randomColor);  
h1.addEventListener("click",randomColor);  
section.addEventListener("click",randomColor);
```



> Eventos > Propagação



> Eventos > Delegação

- A fase de **capturing** e de **bubbling** permite implementar a **delegação de eventos** (*event delegation*)
- **Padrão** que **permite efetuar a manipulação e eventos de forma eficiente**. Em vez de adicionar um *event listener* para cada um dos elementos similares, pode-se adicionar um event listener ao element pai e chamar o evento num element especiiço com recurso à propriedade **.target** do evento do objecto.



> Eventos > Delegação

- Como se pode obter este comportamento?

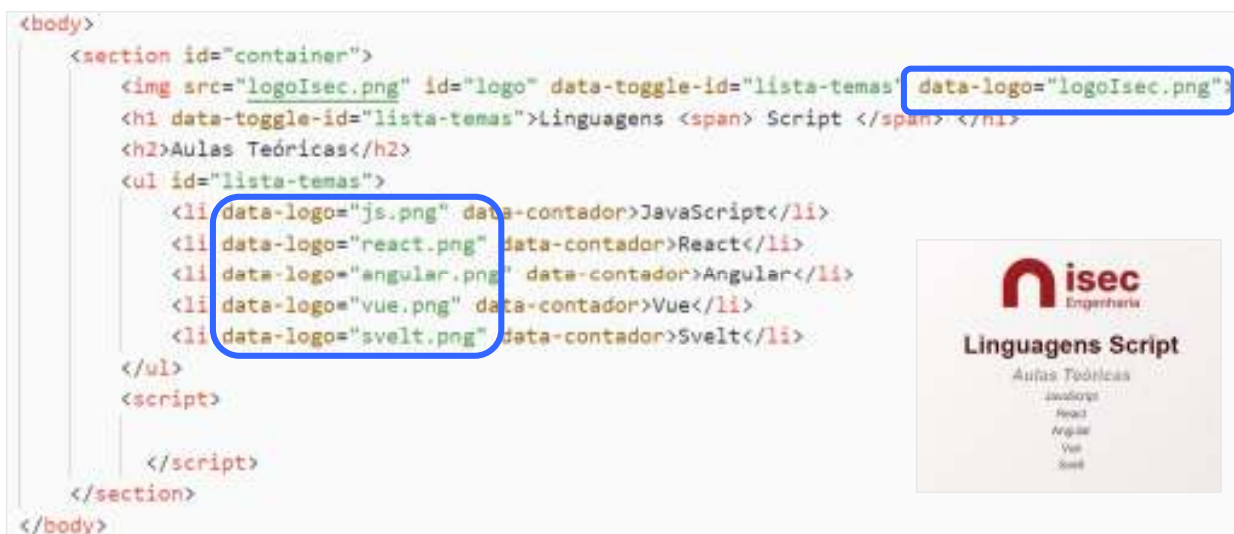


- Apresentar a imagem do logotipo quando se passa o rato por cima do tópico e voltar ao ISEC quando se tira...



> Eventos > Delegação > Exemplo

- Considerando o seguinte HTML



> Eventos > Delegação > Exemplo

▪ Solução 1)

- Algum problema nesta abordagem? Será isto delegação de eventos?

```
const logoElement=document.querySelector('#logo');
document.querySelectorAll('li').forEach(function(element) {
  element.addEventListener('mouseover', function (e) {
    logoElement.src=element.getAttribute("data-logo");
    element.style.background="grey";
  });
  element.addEventListener('mouseout', function (e) {
    element.style.background="none";
    logoElement.src=logoElement.dataset.logo;
  });
});
```



> Eventos > Delegação > Exemplo

```
const logoElement=document.querySelector('#logo');
document.querySelector('#lista-temas').addEventListener('mouseover', function (e) {
  const elemento=e.target;
  if (e.target.dataset.logo) {
    logoElement.src=elemento.dataset.logo;
    elemento.style.background="grey";
  }
});
document.querySelector('#lista-temas').addEventListener('mouseout', function (e) {
  e.target.style.background="none";
  logoElement.src =logoElement.dataset.logo;
});
```



</JavaScript no Browser>

