



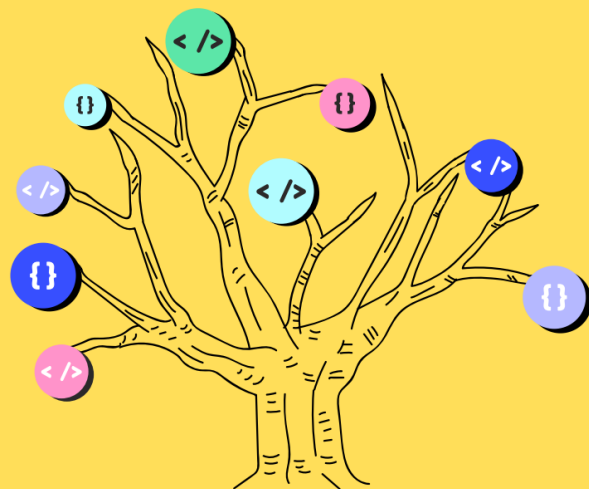
Desenvolvimento Web&Mobile

Desenvolvimento Web para Server Side

Sara Monteiro

sara.monteiro.prt@msft.cesae.pt

THE DOM



Centro para o Desenvolvimento
de Competências Digitais



O DOM

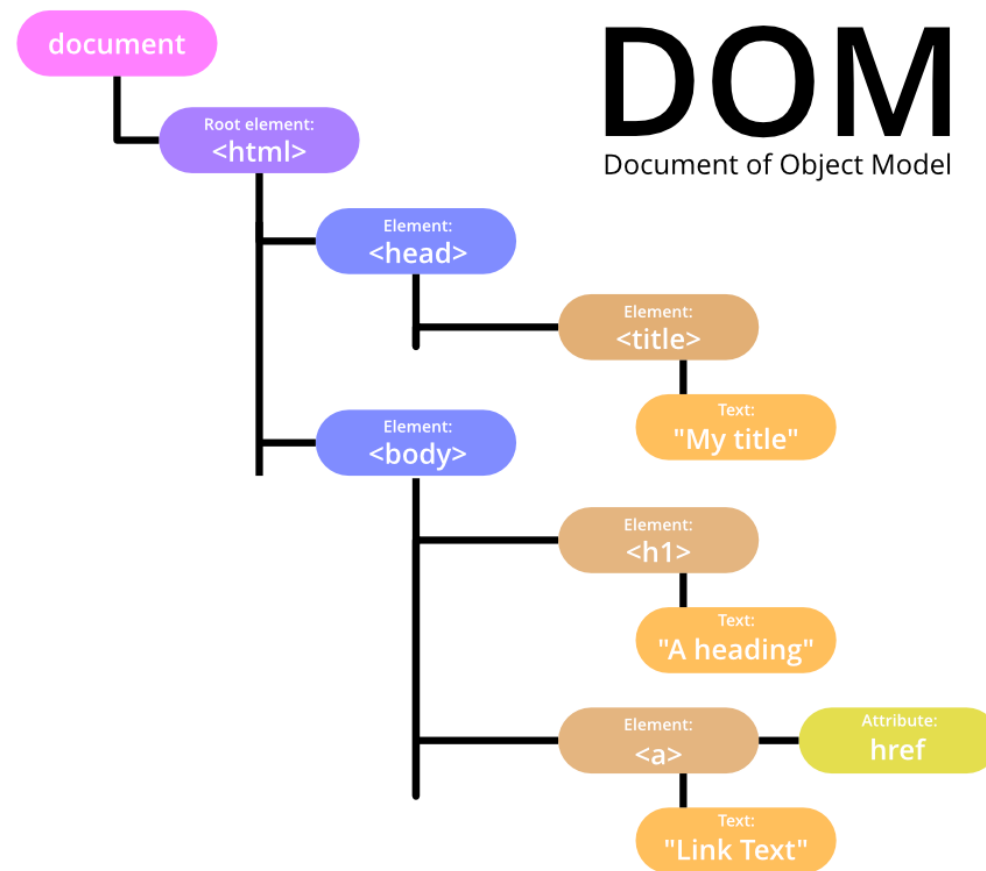
- Permite-nos combinar
HTML + CSS + JS



O DOM

- DOM significa Document Object Model.
- Objecto que representa a nossa página HTML em JS, a “janela” que o JS usa para espreitar a página.
- Conjunto de objectos com os quais podemos interagir via JS.

[Documentação](#)



O DOM

```
<body>  
  <h1>Hello!</h1>  
  <ul>  
    <li>Water Plants</li>  
    <li>Get Some Sleep</li>  
  </ul>  
</body>
```

HTML+CSS Go In...



JS Objects Come Out

O DOM

console.dir(document);

```
> console.dir(document);
#document
  location: Location {ancestorOrigins: DOMStringList, href: 'http://127.0.0.1:5500/index.html', origin: 'http://127.0.0.1:5500', protocol: 'http:', host: '127.0.0.1:5500', ...}
  URL: "http://127.0.0.1:5500/index.html"
  activeElement: body
  adoptedStyleSheets: Proxy(Array) {}
  alinkColor: ""
  all: HTMLAllCollection(77) [html, head, meta, meta, title, link, script, body, h1, img#banner, p, b, b, a, a, a, a, a, a, div#toc, input#toggles, div.toctitle, h2#mw-toc-heading, span.toggles, label.toggles, ul, li.toclevel-1.tocsection]
  anchors: HTMLCollection []
  applets: HTMLCollection []
  baseURI: "http://127.0.0.1:5500/index.html"
  bgColor: ""
  body: body
    characterSet: "UTF-8"
    charset: "UTF-8"
    childElementCount: 1
  childNodes: NodeList(2) [<!DOCTYPE html>, html]
  children: HTMLCollection [html]
  compatMode: "CSS1Compat"
  contentType: "text/html"
  cookie: ""
  currentScript: null
  defaultView: Window {window: Window, self: Window, document: document, name: '', location: Location, ...}
  designMode: "off"
  dir: ""
  doctype: <!DOCTYPE html>
  documentElement: html
  documentURI: "http://127.0.0.1:5500/index.html"
  domain: "127.0.0.1"
  embeds: HTMLCollection []
  featurePolicy: FeaturePolicy {}
  fgColor: ""
  firstChild: <!DOCTYPE html>
  firstElementChild: html
  fonts: FontFaceSet {onloading: null, onloadingdone: null, onloadingerror: null, ready: Promise, status: 'loaded', ...}
  forms: HTMLCollection []
  fragmentDirective: FragmentDirective {}
  fullscreen: false
  fullscreenElement: null
  fullscreenEnabled: true
  head: head
    hidden: false
  images: HTMLCollection(4) [img#banner, img.square, img.square, img.square, banner: img#banner]
  implementation: DOMImplementation {}
  inputEncoding: "UTF-8"
  isConnected: true
  lastChild: html
  lastElementChild: html
```

DOM - seleccionar elementos

Para conseguirmos dar movimento aos elementos do DOM precisamos de:

1. Seleccionar
2. Manipular

Selectores:

- [getElementById](#)
- [getElementsByTagName](#)
- [getElementsByClassName](#)

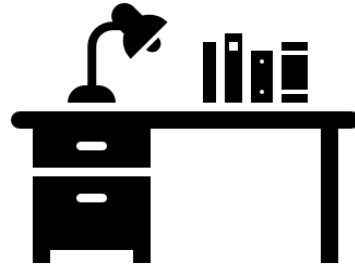
Seleccionar elementos: getElementById

```
const myBanner = document.getElementById('banner');  
console.log(myBanner);
```

```
app.js:21  

```


Exercício



1. Abre o ficheiro idExercise.html e encontras um html com conteúdo.
2. Adiciona um ficheiro JS para o exercício e linka o mesmo ao ficheiro.
3. Selecciona a imagem pelo seu id e guarda-a numa variável chamada image.
4. Selecciona o h1 pelo seu id e guarda-o numa variável chamada heading.
5. Faz console log das duas variáveis para verificar se guardou o pretendido.

Seleccionar elementos: TagName e ClassName

- Os métodos `getElementsByTagName` e `getElementsByClassName` retornam um coleção de elementos.

```
const myImages = document.getElementsByTagName('img');  
const squareImages = document.getElementsByClassName('square');  
  
console.log(myImages);  
console.log(squareImages);
```

```
▼ HTMLCollection(4) ⓘ  
  ▶ 0: img#banner  
  ▶ 1: img.square  
  ▶ 2: img.square  
  ▶ 3: img.square  
  ▶ banner: img#banner  
  length: 4  
  ▶ [[Prototype]]: HTMLCollection  
  
▼ HTMLCollection(3) ⓘ  
  ▶ 0: img.square  
  ▶ 1: img.square  
  ▶ 2: img.square  
  length: 3  
  ▶ [[Prototype]]: HTMLCollection
```

Outros selectores

- [querySelector](#)
- [querySelectorAll](#): retorna uma coleção e não só um
- Nos query selectors podemos usar uma combinação de selectors como faríamos em CSS

```
//Finds first h1 element:  
document.querySelector('h1');  
  
//Finds first element with ID of red:  
document.querySelector('#red');  
  
//Finds first element with class of  
document.querySelector('.big');
```

```
> document.querySelectorAll('a,p')  
< ▶ NodeList(26) [p, a, a, a, a, a, a, a, a, a, a, a, a, a, p, a, a, a, a, a, a, a, a.mw-red  
a]  
  
> document.querySelector("input[role='button']")  
< <input type="checkbox" role="button" id="toctogglecheckbox" class="toctogglecheckbox"  
y: none">
```

Exercício



1. Abre o ficheiro todos.html e encontras um html já predefinido.
2. Usando os selectores, cria uma variável chamada doneTodos e atribui-lhe todos os que têm a class done.
3. Seleciona uma checkbox e guarda-a numa variável chamada checkbox. Vais precisar de usar o atributo type, procura na internet :)
4. Faz um console.log() de ambas as variáveis para confirmar que está certo.

DOM: Manipular Elementos

- `classList`
- `getAttribute()`
- `setAttribute()`
- `appendChild()`
- `append()`
- `prepend()`
- `removeChild()`
- `remove()`
- `createElement`
- `innerText`
- `value`
- `Style`
-

...

Propriedades e Métodos

Element

- ▶ Instance properties
- ▶ Instance methods

Manipular Elementos: innerText



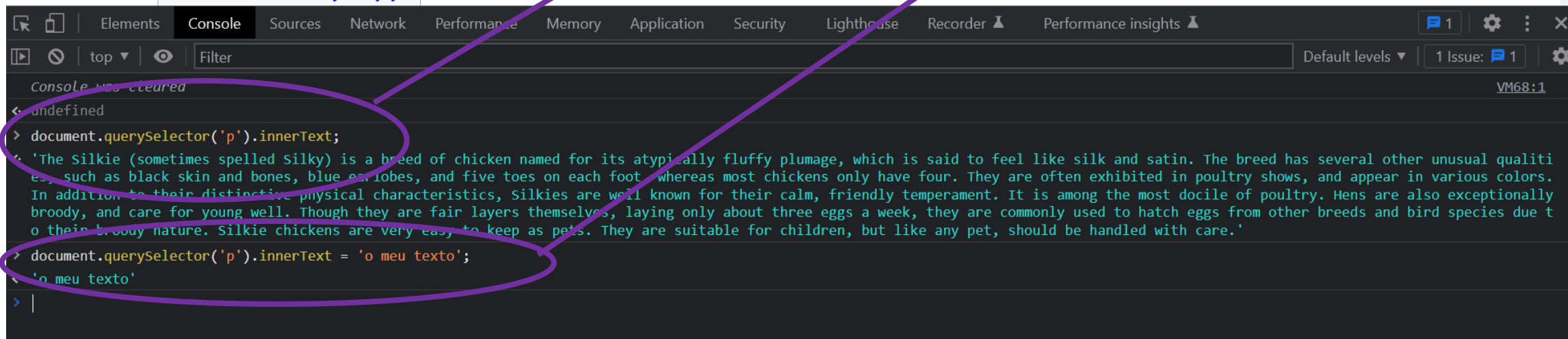
o meu texto

Contents

- 1 [History](#)
- 2 [Characteristics](#)
 - 2.1 [Bantams](#)
 - 2.2 [Polydactyly](#)

Seleção

Manipulação



```
Console was cleared
VM68:1
< undefined
> document.querySelector('p').innerText;
'The Silkie (sometimes spelled Silky) is a breed of chicken named for its atypically fluffy plumage, which is said to feel like silk and satin. The breed has several other unusual qualities, such as black skin and bones, blue earlobes, and five toes on each foot, whereas most chickens only have four. They are often exhibited in poultry shows, and appear in various colors. In addition to their distinctive physical characteristics, Silkies are well known for their calm, friendly temperament. It is among the most docile of poultry. Hens are also exceptionally broody, and care for young well. Though they are fair layers themselves, laying only about three eggs a week, they are commonly used to hatch eggs from other breeds and bird species due to their broody nature. Silkie chickens are very easy to keep as pets. They are suitable for children, but like any pet, should be handled with care.'
> document.querySelector('p').innerText = 'o meu texto';
'o meu texto'
> |
```

Manipular Elementos: O innerHtml

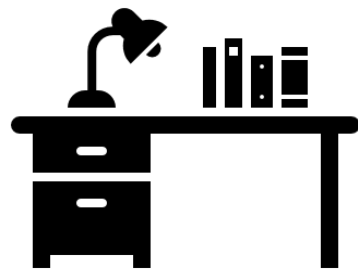
O innerHtml troca o elemento com o texto e não só o texto.

```
let myText = document.querySelector('h1');  
myText.innerHTML = '<button>Sou um botão que substitui o h1 </button>';  
myText.innerHTML += '<button>Sou um botão adicionado </button>';
```

Sou um botão que substitui o h1 Sou um botão adicionado



Exercício



1. Abre o ficheiro pickles.html e encontras um html já predefinido.
2. Usando JS, muda o conteúdo do Yammi para Yack.

Seleccionar e Manipular Atributos



Centro para o Desenvolvimento
de Competências Digitais

Além de conseguirmos manipular elementos, podemos também aceder e manipular os atributos desses elementos.

Esta manipulação pode ser feita de duas formas:

1. Acedendo à propriedade do elemento;
2. Usando os métodos `getAttribute` e `setAttribute`

Seleccionar e Manipular Atributos

1)

```
You, 1 second ago | 1 author (You)  
const myImage = document.querySelector('img');  
console.log(myImage );  
  
//const myHeading = document.querySelectorAll('#ma
```



```
app.js:2  

```

```
myImage.id = 'myNewID';  
console.log(myImage );  
  
//const myHeading = document
```



```

```

Seleccionar e Manipular Atributos

2)

```
console.log(myImage.getAttribute('src'));
```

Acedemos ao atributo utilizando o método
getAttribute('atributo a que queremos aceder');

```
https://images.unsplash.com/photo-1563281577-a7be47e20db9?ixlib=rb-1.2.1&ixid=eyJhchHBfaWQiOjEyMDd9&auto=format&fit=crop&w=2550&q=80
```

Atribuimos o novo conteúdo ao atributo com o setAttribute. Neste caso estamos a dizer que mudamos aquela src para outra imagem.

```
myImage.setAttribute('src', 'https://media.istockphoto.com/id/1223565803/pt/foto/portrait-of-a-man-with-an-exploding-mind.jpg?s=2048x2048&w=is&k=20&c=9qLgon17lnxfsyeN2nb6q-bt8zHAtbnayTsBepIca3g=')
```

Seleccionar e Manipular Atributos



The **Silkie** (sometimes spelled **Silky**) is a **breed** of **chicken** named for its atypically fluffy **plumage**, which is said to feel like **silk** and satin. The breed has several other unusual qualities, such as black skin and bones, blue earlobes, and five toes on each foot, whereas most chickens only have four. They are often exhibited in **poultry** shows, and appear in various colors. In addition to their distinctive physical characteristics, Silkies are well known for their calm, friendly temperament. It is among the most docile of poultry. Hens are also exceptionally **broody**, and care for young well. Though they are fair layers themselves, laying only about three eggs a week, they are commonly used to hatch eggs from

Exercício



1. Abre o ficheiro chicken.html e encontras um html já predefinido.
2. Usando JS, muda a src da imagem para
`'https://devsprouthosting.com/images/chicken.jpg'`.
3. Adiciona um botão que diga 'Ovo ou Galinha' e muda a imagem ao clicar no botão (através de funções);
4. Através de decisões if/else muda a foto tanto para a galinha como para o ovo.(Se estiver a galinha, ao clicar muda para o ovo; se estiver o ovo muda para a galinha);

DOM - Seleccionar e Manipular o estilo



Centro para o Desenvolvimento de Competências Digitais

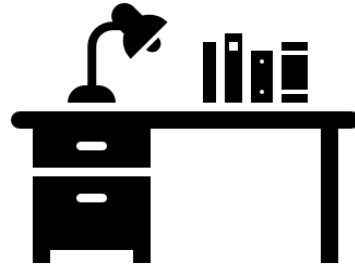
Iremos agora aprender como se muda o estilo de um elemento através de JS usando a propriedade style.

Atenção: ele muda a propriedade no CSS inline, logo não é aconselhável se forem muitas alterações. Ter também atenção aos conflitos, uma vez que o CSS é cascade e podemos estar a anular uma definição antiga.

```
const myText = document.querySelector('h1');  
myText.style.color = 'red';
```

```
> myText.style  
CSSStyleDeclaration {0: 'color', accentColor: '',  
  alignSelf: '', ...} i  
  0: "color"  
  accentColor: ""  
  additiveSymbols: ""  
  alignContent: ""  
  alignItems: ""  
  alignSelf: ""  
  alignmentBaseline: ""  
  all: ""  
  animation: ""  
  animationComposition: ""  
  animationDelay: ""  
  animationDirection: ""  
  animationDuration: ""  
  animationFillMode: ""  
  animationIterationCount: ""  
  animationName: ""
```


Exercício



1. No repositório do Git, na pasta exs, encontras um ficheiro chamado magicalForest.html. Copia para o teu projecto.
2. Adiciona ou usa o ficheiro JS anterior para o exercício.
3. Usando JS:
 - Selecciona a div com o id container e coloca o texto alinhado ao centro;
 - Selecciona a imagem e dá-lhe uma largura de 150px e um border radius de 50%;

DOM - Seleccionar e Manipular vários elementos

Quando estamos perante uma lista de elementos e queremos manipular de forma a todos mudarem algo, temos que usar um ciclo para acedermos elemento a elemento.

Ex: mudar todos os elementos li para um background verde.

```
const allLinks = document.querySelectorAll('a');  
for (let link of allLinks) {  
  link.style.backgroundColor = 'green';  
}
```

The Silkie (sometimes spelled Silky) is a **breed** of **chickens** named for its atypically fluffy **plumage**, which is said to feel like **fur**. It has five toes on each foot, whereas most chickens only have four. They are often exhibited in **poultry** shows, and appear in various temperaments. It is among the most docile of poultry. Hens are also exceptionally **broody**, and care for young well. Though they differ from other breeds and bird species due to their broody nature. Silkie chickens are very easy to keep as pets. They are suitable for chil

Contents

- **1 History**
- **2 Characteristics**
 - **2.1 Bantams**
 - **2.2 Polydactyls**
- **3 In cuisine**
- **4 References**
- **5 Further reading**

Exercício



1. Abre o ficheiro rainbow.html e encontras um html já predefinido.
2. Adiciona ou usa o ficheiro JS anterior para o exercício.
3. No Js declara o seguinte array: `const colors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];`
4. Selecciona os `` e itera de modo a que a cada um seja assignada uma das cores do array.

DOM - Seleccionar e Manipular o estilo

Uma vez que usar a propriedade style não é uma grande ideia quando estamos a mudar o estilo a vários elementos, iremos aprender a mudar o estilo adicionando classes. Para isso, no nosso ficheiro CSS iremos criar duas classes padrão que depois iremos usar em JS.

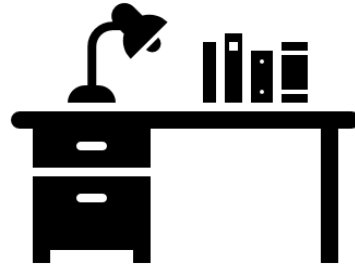
```
app.css M X
css > app.css > ...
28 }
29
30 ✓ .purple{
31   color: #561e8f;
32 }
33
34 ✓ .border{
35   border: 3px solid greenyellow;
36 }
37
38
```

```
const h2 = document.querySelector('h2');
h2.setAttribute('class', 'purple')
```

A melhor opção é usar a classList!

```
h2.classList.add('purple');
h2.classList.add('border');
h2.classList.remove('purple');
h2.classList.contains('border');
```

Exercício



1. Abre o ficheiro classList.html e encontras um html já predefinido.
2. Usando JS, inverte os elementos que têm a class highlight (basicamente as que têm devem deixar de ter e as que não têm passam a ter);

DOM - Seleccionar e Manipular elementos filho / pai

```
const firstBold = document.querySelector('b');  
console.log(firstBold);
```

```
<b>Silkie</b>
```

```
const firstBold = document.querySelector('b');  
console.log(firstBold.parentElement);  
console.log(firstBold.childElement);
```



```
> <p> ... </p>
```

```
undefined
```

Podemos também usar a propriedade `.children` que nos retorna uma coleção de todos os filhos, caso existam.

DOM - Seleccionar e Manipular Elementos irmão

```
const myA = document.querySelector('a');  
console.log(myA);
```

```
<a href="/wiki/List_of_chicken_breeds" title="List of chicken breeds">breed</a>
```

```
console.log(myA.nextElementSibling);
```



```
<a href="/wiki/Chicken" title="Chicken">chicken</a>
```

```
console.log(myA.previousElementSibling);
```



null

Adicionar ou remover Elementos

Por fim, podemos ainda adicionar ou remover elementos HTML através dos métodos:

- createElement()
- appendChild()
- removeChild()

Contents

- 1 [History](#)
- 2 [Characteristics](#)
 - 2.1 [Bantams](#)
 - 2.2 [Polydactyly](#)
- 3 [In cuisine](#)
- 4 [References](#)
- 5 [Further reading](#)
- [Sou uma nova Li](#)

Para adicionar um elemento:

1. criá-lo;
2. indicar onde vai ser colocado.

```
const myLi = document.createElement('li');  
myLi.innerText = 'Sou uma nova Li';  
  
const myUl = document.querySelector('ul');  
myUl.appendChild(myLi);
```

Adicionar Conteúdo a Elemento

```
const firstBold = document.querySelector('b');  
firstBold.append('adicionei coisas aqui no meu b');
```

The Silkie**adicionei coisas aqui no meu b** (sometimes spelled Silky) is a [breed](#) of [chicken](#) and bones, blue earlobes, and five toes on each foot, whereas most chickens only have four. known for their calm, friendly temperament. It is among the most docile of poultry. Hens are commonly used to hatch eggs from other breeds and bird species due to their broody nature.

O **append()** adiciona ao fim e o **prepend()** ao início;

Exercício



1. Abre o ficheiro buttons.html e encontras um html já predefinido.
2. Sem mexer no html, cria 100 botões com um texto à tua escolha dentro.
3. Os botões devem ser anexados à div container.

Remover Elementos

Podemos remover elementos HTML através dos métodos:

- `removeChild(elemento)`
- `remove()`, não suportado pelo IE.

```
const liToRemove = document.querySelector('li');  
myUl.removeChild(liToRemove);
```

```
liToRemove.remove();
```

Contents

- 2 Characteristics
 - 2.1 Bantams
 - 2.2 Polydactyly
- 3 In cuisine
- 4 References
- 5 Further reading
- Sou uma nova Li

DOM: Eventos

Respostas aos inputs e acções do utilizador.

Ex: quando o utilizador clica, quando faz scroll, etc.

[Documentação](#)

```
<button id="btnClick">Sou um botão que produz um evento</button>
```

```
const btn = document.querySelector('#btnClick');

btn.onclick = function () {
  alert('este botão foi activado!');
}
```

DOM: Eventos

[Documentação](#)

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form

Eventos Inline

- As tags html têm aberta a possibilidade de chamar eventos inline. Por exemplo, temos usado o `onclick="função()"` dos botões.
- No entanto, caso tenhamos vários botões que façam a mesma coisa teremos que fazer copy paste do evento para todos, o que não se torna muito funcional.
- A recomendação é para que cada coisa esteja no ficheiro correspondente e, como tal, todo o nosso código JS deverá ser colocado nos ficheiros JS.

DOM: addEventListener

O `addEventListener()` é o método JS mais recomendado para trabalhar com Eventos.

```
<button id="btnEvent">Sou um botão que produz um evento</button>
```

```
const btnEvent = document.querySelector('#btnEvent');  
btnEvent.addEventListener('click', function(){  
  alert('botão de evento');  
})
```

```
function hello(){  
  alert('estou a dizer olá');  
}  
  
function goodbye(){  
  alert('estou a dizer Adeus');  
}  
  
btnEvent.addEventListener('click', hello);  
btnEvent.addEventListener('click', goodbye);
```

DOM: addEventListener

É usado da seguinte forma:

1. Seleccionamos o elemento que queremos colocar com o evento
2. Colocamos nesse elemento um evento que está 'à escuta' à espera que algo o active.
3. Definimos qual o tipo de evento que activa a função e definimos a acção.

```
<button id="btnEvent">Sou um botão que produz um evento</button>
```

```
const btnEvent = document.querySelector('#btnEvent');  
btnEvent.addEventListener('click', function(){  
  alert('botão de evento');  
})
```

Vantagens addEventListener

- A um botão podemos adicionar várias funções com várias acções.

```
✓ function twist(){  
  console.log("TWIIISSTTT");  
}  
✓ function shout(){  
  console.log("SHHOOOUTTTT");  
}  
btnEvent.addEventListener('click', twist);  
btnEvent.addEventListener('click', shout);
```

TWIIISSTTT	app.js:14
SHHOOOUTTTT	app.js:17

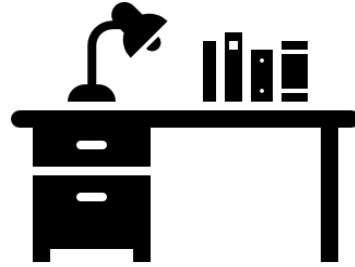
- Podemos correr só uma vez uma das acções

```
btnEvent.addEventListener('click', twist, {once:true});  
btnEvent.addEventListener('click', shout);
```

TWIIISSTTT	app.js:14
4 SHHOOOUTTTT	app.js:17

No geral é mais flexível e dá-nos mais controlo sobre como queremos que as funções corram!

Exercício



1. Abre o ficheiro hello.html e encontras um html já predefinido.
2. Cria os seguintes eventos em JS:
 - No botão Hello deverá fazer um alert que diga 'olá'
 - No botão Bye deverá fazer um alert que diga 'Adeus'

Exercício



1. Cria um Botão que crie uma cor aleatória para o background e mostre qual é no HTML.
Ajuda em [documentação](#).

rgb(165, 146, 186)

Click Me

rgb(55, 59, 49)

Click Me

rgb(153, 54, 56)

Click Me

Exercício



1. Usando tudo o que aprendemos até agora (HTML, CSS, JS) cria uma calculadora que receba dois números e faça as seguintes operações: soma, subtracção, divisão e multiplicação e apresente o resultado.

2. Dicas:

- os números são colocados em dois inputs diferentes, cada um com seu id; para o cálculo vamos buscar o seu value;
- a função click é no botão =, onde se calcula de acordo com o colocado nos inputs e na operação
- As operações são calculadas com um switch

Calculadora

Total:

Keyword this aplicada aos Eventos



Centro para o Desenvolvimento
de Competências Digitais

Já aprendemos a despoletar funções para um elemento, ou ir buscar vários elementos e a todos aplicar a mesma regra.

E se a vários elementos do nosso HTML quisermos aplicar a mesma função?

Click Me

Click Me3

Nova cor

Keyword this aplicada aos Eventos

```
<button>Click Me</button>  
<button id="btn2">Click Me3</button>  
<h2>Nova cor</h2>  
  
</div>
```

Click Me

Click Me3

Nova cor

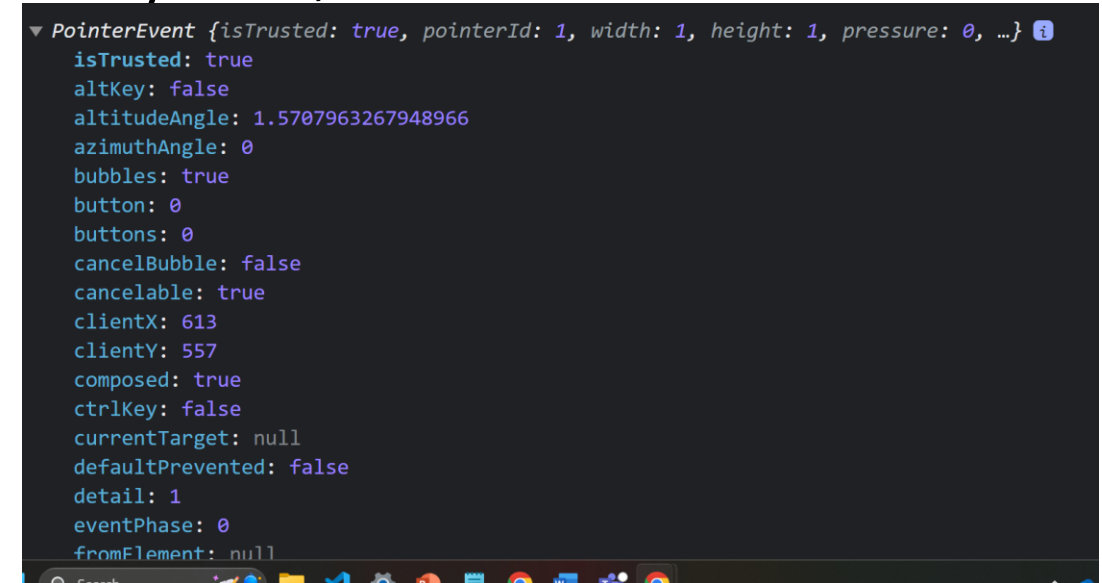
```
const button = document.querySelector('button');  
const h2 = document.querySelector('h2');  
const button2 = document.querySelector('#btn2');  
  
button.addEventListener('click', newColor);  
h2.addEventListener('click', newColor);  
button2.addEventListener('click', newColor);  
  
function newColor () {  
  const newColor = makeRandColor();  
  this.style.backgroundColor = newColor;  
}  
  
const makeRandColor = () => {  
  const r = Math.floor(Math.random() * 255);  
  const g = Math.floor(Math.random() * 255);  
  const b = Math.floor(Math.random() * 255);  
  return `rgb(${r}, ${g}, ${b})`;  
}
```


Event Object

Sempre que usamos o EventListener temos automaticamente acesso a um objecto do Evento accionado.

A informação do evento depende do tipo de elemento e evento seleccionado: um p é diferente de um button, um click diferente de keydown, etc..

```
const myPar = document.querySelector('p');  
✓ myPar.addEventListener('click', function(event){  
  console.log(event)  
});
```



Event Object: Exemplos

Informações sobre keydown

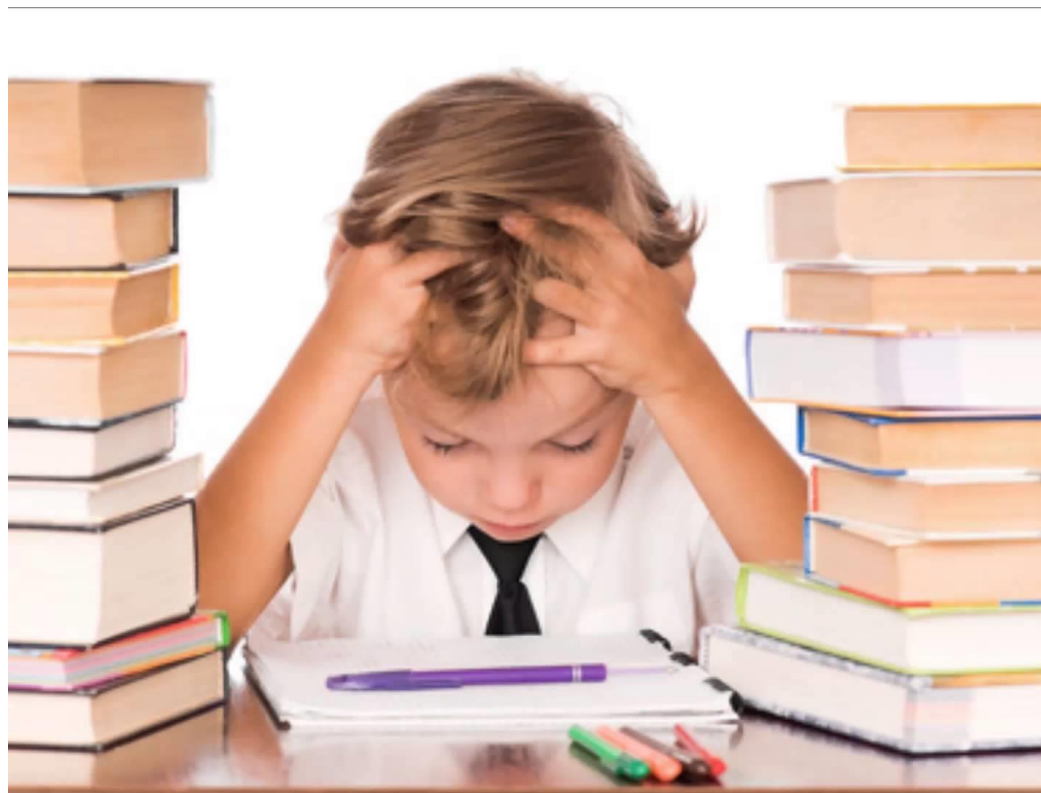
```
window.addEventListener('keydown', function (e){  
  console.log(e.code);  
})
```

ArrowUp
2 ArrowDown
ArrowLeft
2 KeyS
KeyD
KeyS
KeyD
ArrowDown
KeyF
ControlLeft
ShiftLeft
MetaLeft
ShiftLeft

Exercício



Ficha de Trabalho 2



Formulários e PreventDefault

Quando preenchemos um formulário, etc, estamos a submeter alguma informação e ao clicar no botão ele direcciona para a página onde vamos tratar os dados.

No entanto, o utilizador não precisa / não deve visualizar essa página.

Como prevenir que o utilizador vá para a página onde estamos a tratar os dados?

```
<form action="/myform">  
  <input type="text" placeholder="Nome"></br>  
  <input type="text" placeholder="Morada"></br>  
  <button type="submit">Submeter</button>  
</form>
```

Formulários e o PreventDefault

```
const form = document.querySelector('form');  
✓ form.addEventListener("submit", function(e){  
  e.preventDefault();  
  console.log('submetido!')  
})
```

Para prevenir que seja aberta uma nova página usamos o `.preventDefault()` do objecto evento.

O código irá continuar a correr e poderemos depois trabalhar os dados internamente, mas a informação que estamos a dar ao browser é para parar a execução ali.

```
submetido!
```

```
>
```

Recursos

- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>