tag: É uma marcação que define com um elemento deve ser mostrado na página. É composta por um nome entre "<" ">". As tags servem para marcar onde começam e terminam elementos.

Elemento: É uma combinação de uma tag de abertura, o conteúdo do elemento e, quando necessário, uma tag de fechamento.

Atributos: São informações que passamos na tag para que ela se comporte de determinada maneira.

```
<p>Escrever um paragrafo dentro de uma div com um link para o site da UEvora</p>

<div>
<h1>Sou estudande da <a href="www.uevora.pt">UE</a> desde 2019!</h1>
</div>
```

WYSIWYG : "What You See Is What You Get", Trata-se de um tipo de editor que permite aos utilizadores criar e editar conteúdo de forma visual, de modo a que a aparência no ecrã seja muito semelhante à versão final que será apresentada numa página da web. Não requer conhecimentos em HTML ou outras linguagens de marcação.

Elemento img: <img src="/images/icons/next.svg" alt="next"> O uso do atributo alt é obrigatório. O atributo alt tem o texto a mostrar se a imagem não puder apresentar-se.

3 tipos de selectores: Selectores são formas de referir elementos na estrutura dos documentos.
Referir por tipo de elemento: todos os elementos deste tipo (p) no documento. p {property: value;}
Referir por ID selector (prefixo #): para selecionar o elemento com determinado identificador (no caso, com id="intro"). #intro {property: value}
Referir por Class selector(uso de "." Antes do nome da classe): escolhe elementos associados aquela classe (no caso, "class"). .class {property: value}

Seleção de um tipo de elemento contido noutro elemento; por contexto (**descendente**). Exemplo para blocos com ênfase dentro de elementos do tipo h1:

```
h1 em {color: red;}
    /* (only emphasized text in h1s would be red) */
```

Selector **universal** (*) abrange qualquer elemento.

```
#intro * {border: 1px solid gray;}
```

(no caso: todos os elementos dentro do elemento com ID **intro**)

Selector	Exemplo	Descrição
element element *descendente (direto ou não)*	`div p`	*Selects all <p> elements inside elements*
element>element *todos os filhos diretos de determinado tipo*	`div > p`	*Selects all <p> elements where the parent is a element*
element+element *primeiro adjacente daquele tipo (não é dentro!)*	`div + p`	*Selects the first <p> element immediately after elements*
element1~element2 *todos os posteriores daquele tipo, ao mesmo nível, e não necessariamente consecutivos*	`p ~ ul`	*Selects every element that are preceded by a <p> element*

```
<p>Representar uma regra de CSS em que todos os cabçalhos de nivel dentro da classe tw devem ter a cor vermelha</p>

.tw h1(2,3,4,5,6) {
   color: red;
}
```

Floating: Afastar o elemento para uma direção, colocando-o à deriva, afastado, e com eventual conteúdo à sua volta.

Clearing Floated Elements: Evita adjacência a floated element.

Web Design Responsivo: Abordagem na conceção de conteúdos que permite que uma página se adapte ao ecrã e ás condições em que é apresentada.

Componentes base de WDResponsivo

- De acordo com Ethan Marcotte, WDR depende de

1)Grelha flexível
- Elementos podem ser redimensionados e movidos para espaços disponíveis na área mostrada pelo browser

2)Imagens Flexíveis
- Capacidade de apresentar imagens e outros elementos multimédia com diferentes tamanhos, para se ajustarem ao espaço em que se inserem (nos respetivos *container elements*)

3)CSS media queries
- Mecanismo de teste/interrogação às condições do browser, para uso do estilo de formatação mais adequado

Verificação: propriedades do dispositivo (*Media feature*)

- Desde CSS3, é possível testar alguns atributos do viewport
- Exemplo:
  cabeçalhos de nível 1 **h1** terão fonte Lobster apenas se a largura do viewport >=*40em*

```
h1 {
      font-family: Georgia, serif;
   }
@media screen and (min-width: 40em) {
    h1 {
          font-family: 'Lobster', cursive;
       }
}
```

Viewport: Área retangular de apresentação do conteúdo na janela do browser.

CSS *Media Queries* - exemplos

```
@media screen and (orientation: landscape) {
  body {
     background: skyblue;
     }
}
@media screen and (orientation: portrait) {
  body {
    background: coral;
    }
}
```

CSS: aplicação das regras de verificação (*queries*)

- Dentro do código CSS
  - Inserir regra com **@media** com um critério e as regras de estilo para essas condições
  - Primeiro deve ter a formatação de estilo por omissão, só depois a regra **@media** para a formatação condicionada
- Com CSS externa (link, @import)
  - A ativação da formatação no ficheiro dependerá da verificação da regra

```
<link rel="stylesheet" href="base.css">
<link rel="stylesheet" href="2column-styles.css"
                media="screen and (min-width:1024px)">
```

Imagens de largura variável (*w-descriptor*)

```
<img src="strawberries-640.jpg"
     alt="baskets of ripe strawberries"
     srcset="strawberries-480.jpg 480w,
             strawberries-960.jpg 960w,
             strawberries-1280.jpg 1280w,
             strawberries-2400.jpg 2400w"
     sizes="100vw">
```

Imagens de largura variável e proporção variável

```
<img src="strawberries-640.jpg" alt="baskets of ripe strawberries"
  srcset="strawberries-240.jpg 240w,
          strawberries-480.jpg 480w,
          strawberries-672.jpg 672w"
  sizes="(max-width: 480px) 100vw,
         (max-width: 960px) 70vw,
         240px">
```

O atributo srcset tem as alternativas, separadas por vírgula. – É escolhida a primeira opção compatível (cuidado com a ordem). w-descriptors indicam a largura da imagem em px. O atributo sizes indica o tamanho da imagem quando apresentada no viewport. No caso é 100% da largura da janela.
Se a largura do viewport é 480px ou menos, a imagem é colocada a 100% da largura do elemento • Caso contrário, se é maior mas não ultrapassa 960px, e imagem é colocada em 70%.

Imagens cortadas: elemento picture

O elemento img deve aparecer no fim.

```
<picture>
  <source media="(min-width: 1024px)" srcset="icecream-large.jpg">
  <source media="(min-width: 760px)" srcset="icecream-medium.jpg">
  <img src="icecream-small.jpg" alt="hand holding ice cream cone and
text that reads Savor the Summer">
</picture>
```

Form : Mecanismo para preenchimento de dados em documentos web.
<form action="URL" method="POST or GET"> <!-- Form content and inputs here --> </form>
O atributo action indica o destino para onde os dados do formulário serão submetidos.
O atributo method determina como os dados codificados serão enviados. GET: dados acrescentados ao URL de destino. POST: Os dados não são expostos no endereço.

Incorporar Scripts numa página HTML

Modo Embebido (*embedded script*)
inserido no conteúdo HTML e delimitado pelas tags do elemento **`script`**:

```
<script>
  … JavaScript code goes here
</script>
```

Modo externo (*external script*)
recurso externo ao documento HTML, referido no atributo **`src`** do elemento **`script`**, como apontador para um ficheiro autónomo *.js*:

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

Âmbito de Variáveis

Uma variável usada apenas dentro do código de uma função, ou bloco, diz-se de **âmbito local**.

Para definir a variável numa função, usamos a palavra reservada **`var`** para marcar o âmbito local:

```
var foo = "value";
```

Uma variável acessível desde qualquer script ou zona de código na página, diz-se de **âmbito global**.

- Qualquer variável definida fora de uma função é automaticamente **global**:

```
var foo = "value";
```

- Para transformar uma variável criada dentro de uma função numa variável global (raro, desaconselhado), omitir a palavra reservada **`var`**:

```
foo = "value";
```

Um evento é uma ação que pode ser detata com JS e desencadear a execução de scripts.

Tratamento de Eventos

Os controladores de eventos podem ser aplicados de três maneiras:

1) atributo de um elemento HTML:

```
<body onclick="myFunction();">
/* myFunction runs when the user clicks anything
within 'body' */
```

2) propriedade `onclick` do objeto (janela, botão, link...):

```
window.onclick = myFunction;
/* myFunction will run when the user clicks anything
within the browser window */
```

3) Função **`addEventListener()`**:

```
window.addEventListener("click", myFunction);
```

Note-se a ausência do prefixo "on" na sintaxe desta abordagem.

DOM (Document Object Model) é uma interface para consultar e manipular o conteúdo do documento Web.

Referir nós pelos objetos que os representam:
var foo = document.getElementById("beginner").innerHTML;

innerHTML representa o conteúdo HTML dentro daquele elemento.