



# Web Vs. Internet

Internet: Sistema global de rede de computadores interligados usando o standard "Internet protocol suite" para ligar dispositivos por todo o mundo — redondante rede de redes

Web: Sistema de documentos de hipertexto interligados acedidos via Internet usando um Browser

World Wide Web = HTML + URL + HTTP

URL: notação para localizar recursos em servidores

HTTP: protocolo de alto-nível para transferência de ficheiros

HTML: linguagem SGML para hipertexto

Router: dispositivo que conecta computadores

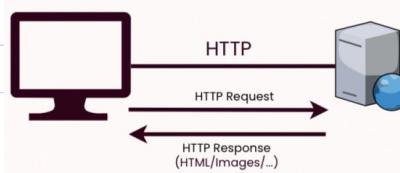
Endereço IP: identifica unicamente cada dispositivo ligado

ISP: organização que liga utilizadores à Internet

DNS: sistema de nomeação hierárquico e distribuído para computadores ligados à Internet

- O endereço IP externo de cada router é atribuído pelo ISP usando DHCP e consultado com uma DNS query

URL: esquema://servidor:porta/nome:@servidor:porto/path?query#id



# HTML

## Hyper Text Markup Language

- linguagem de marcação usada para criar páginas Web, mas não para design ou apresentação, só estrutura e semântica

CSS

HTML

### Estrutura:

- Um arquivo HTML tem uma estrutura em árvore em que cada nó é um elemento
- Elementos podem conter outros elementos e/ou texto
- Elementos são definidos usando tags e podem ter atributos
- Os browsers mostram cada tag com um estilo predefinido que pode ser mudado usando CSS

HTML diz ao browser como o documento é estruturado  
≠

CSS diz ao browser como deve ser mostrado

Tag: <nome> ... </nome>  
ABERTURA                    FECHO

Ex: 
<html>
  <head>
    <title> NÃO VAZIO </title>
  </head>
  <body>
    </body>
  </html>

```

<head> - metadados sobre o documento

<body> - estrutura e conteúdo

- Normalmente, espacos em branco juntos colapsam num só

|                            |                                  |
|----------------------------|----------------------------------|
| <p> parágrafo </p>         | <code> código </code>            |
| <br> mudança de linha      | <kbd> teclado </kbd>             |
| <br> infase </br>          | <small> amostra </small>         |
| <small> pequeno </small>   | <var> variável </var>            |
| <strong> forte </strong>   | <abbr> abreviação </abbr>        |
| <sub> subscrito </sub>     | <address> contacto </address>    |
| <sup> superscrito </sup>   | <time> tempo </time>             |
| <ins> inserido </ins>      | <progress> progresso </progress> |
| <del> apagado </del>       | <q> citação </q>                 |
| <mark> destacado </mark>   | <cite> título </cite>            |
| <pre> pré-formatado </pre> | <dfn> definição </dfn>           |

`<span>` útil para marcar texto, mas sem significado remântico `</span>`

`<a href = "...>` âncora (hiperligação) `</a>`

`<img alt = "... width = ... height = ...>` imagem

`<figure> <img ...>` figura `<figcaption>` legenda `</figcaption>` `</figure>`

Títulos/Cabeçalhos: `<h1> ... <h6>`

- só um `<h1>` por documento

`<article>` artigo `</article>`

`<nav>` links de navegação `</nav>`

`<section>` seção `</section>`

`<aside>` seção separada `</aside>`

`<header>` introdução de seção `</header>`

`<footer>` fecho de seção `</footer>`

`<main>` conteúdo dominante do documento `</main>`

`<div>` útil para agrupar conteúdo, mas sem significado remântico

`<ol type = "... start = ... [reversed]>` lista ordenada `<li value = "...>` item `</li>` `</ol>`

`<ul>` lista desordenada `<li>` item `</li>` `</ul>`

`<dl>` lista de definições `<dt>` termo `</dt>` `<dd>` definição `</dd>` `</dl>`

`<table>` tabela `</table>`

`<tr>` linha `</tr>`

`<td>` célula `</td>`

`<caption>` legenda `</caption>`

`<th scope = "row/col/...>` cabeçalho `</th>`

- É possível unir células horizontal ou verticalmente com rowspan/colspan

- É possível dividir tabelas em 3 seções: thead, tfoot e tbody

- É possível agrupar colunas com <colgroup> </colgroup> e <col span = "...>

`<form action = "... method = ...>` formulário (utilizador fornece dados ao servidor `</form>`)

action: URL do servidor que vai processar os dados

method:

get: valores enviados no URL

post: valores enviados no cabeçalho do HTTP

`<input type = ... name = ... value = ... placeholder = ... autocomplete = "on/off" [readonly] [required] [boolean]>`

type:

text/password/tel/search/url/email: texto normal

number/range: número [value = ... min = ... max = ... step = ...]

date/time/datetime-local/month/week: data [value min max step]

color: cor [value = "# ..."]

checkbox: permite selecionar várias opções [checked]

radio: permite selecionar uma opção [checked]

file: upload de ficheiros [accept = ... multiple]

method = "post" enctype = "multipart/form-data"

hidden: ocultas e inalteráveis pelo utilizador

submit: submete o formulário [value = ...]

`<button formaction = ... formmethod = ... type = ...> botão </button>`

`<select name = ... [multiple]>` seleção a partir de lista

`<option value = ... [selected]> opção </option>`

`<optgroup label = ...> grupo de opções </optgroup>`

`</select>`

`<datalist>` exibe e seleção a partir de lista `</datalist>`  
`<textarea name="..." rows="..." cols="...">` texto `</textarea>`  
`<label>` etiqueta para input `</label>`  
`<fieldset>` conjunto de campos `<legend>` legenda `</legend>` `</fieldset>`

Character Entities: & ...;

decimal: &#...;  
hexadecimal: &#x...;  
<: &lt;;  
>: &gt;;  
&: &amp;;  
": &quot;;  
\_: &nbsp;;

`<canvas width="..." height="...">` retângulo vazio para desenho `</canvas>`  
`<svg>` Scalable Vector Graphics `</svg>`  
`<audio>` som `</audio>`  
`<video>` vídeo `</video>`  
`<source>` mídia `</source>`  
`<track>` texto para vídeo e áudio `</track>`

`<head>` `<meta name="..." content="..." charset="utf-8">` `</head>`

# CSS

## Cascading Style Sheets

- Linguagem de estilo de folhas usada para descrever a aparência e formatar um documento escrito em linguagem de marcação (como HTML), baseada em seletores e propriedades

Seletores: permitem selecionar elementos HTML aos quais aplicar estilos

Propriedades: definem que aspectos do elemento selecionado vão ser estilizados

Ligações ao HTML:

1. Inline
2. Internal Style Sheet
3. External Style Sheet: permite reparamento e reutilização

Seletores:

1. Universal: \*
2. Tipo
3. Id: #
4. Classe: .
5. Atributo: [ ]

Combinar Seletores:

1. Descendente: ↴
2. Filho: >
3. Próximo Irmão: +
4. Única Irmãos Subsequentes: ~

Agrupar Seletores: ,

Espec:

1. de Texto
  2. de Fundo
- a. por Nome
  - b. por Hexadecimal
  - c. por Decimal
- i. Opacidade

## Fonte:

1. Família Générica: sans-serif; serif; monospace
2. Família Específica

- É possível especificar uma ordem/prioridade de fontes pretendidas
- É possível usar fontes remotas

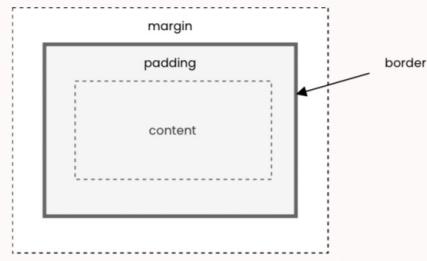
- a. Peso: normal, bold, bolder, lighter, [100, 900]
- b. Estilo: normal, italic, oblique
- c. Tamanho

## Texto:

1. Decoração
2. Alinhamento: left, right, center, justify
3. Case: uppercase, lowercase, capitalize
4. Identações

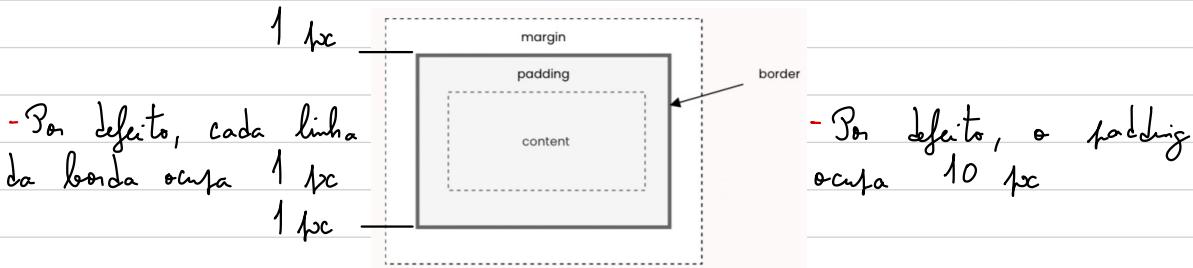
## Unidades de Comprimento:

1. Absolute: mm, cm, in, pt, pc, px (pixel)
2. Font-Relative: rem (root), em (parent)
3. Viewport-Percentage: vh, vhi, vmin, vmax
4. Percentagem: relativamente ao pai



Modulo da Caixa: todos os elementos da página são retangulares, todos terão uma borda e um espaço entre o elemento e o seu (padding), bem como um espaço entre o próprio elemento e o próximo (margin)

## Modelo da Caixa



Width/Height: length/percentage/auto - alteram o tamanho da "content area"  
POR DESENHO

Box-Sizing: border-box / content-box

↓  
inclui o "padding" e a borda

- Podem ser definidos valores extremos min/max-width/height para limitar quando a altura/largura é calculada com base noutro elemento

Padding/Margin:

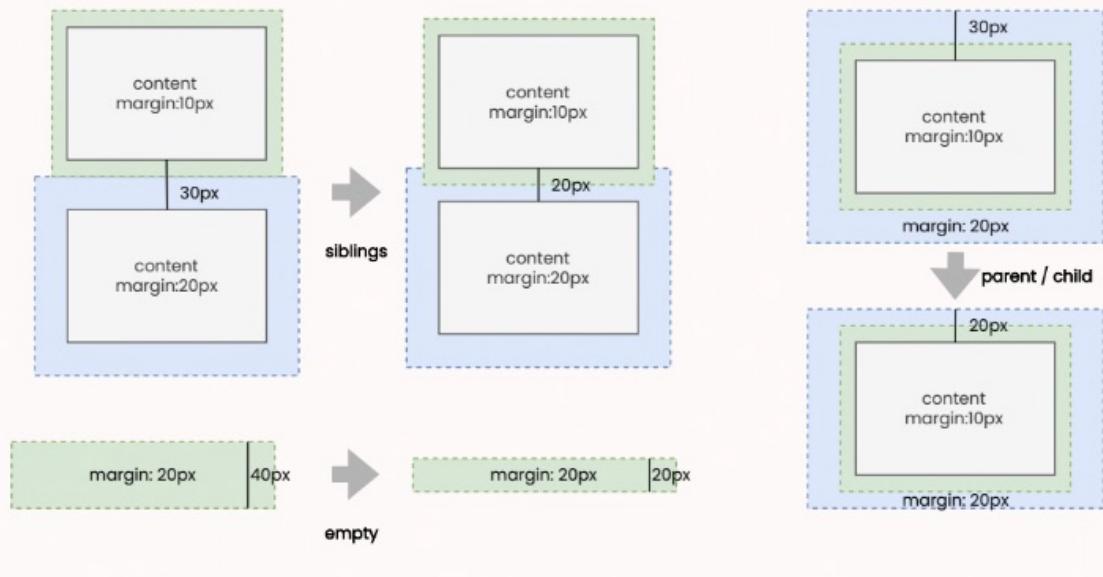
1. altera todos os valores
2. top/bottom, left/right
3. top, left/right, bottom
4. top, right, bottom, left

Borders:

1. width: length, thin, medium, thick
2. style: none, hidden, dotted, dashed, solid, double, groove, ridge, inset/outset
3. color
4. radius:  $\frac{1}{1} \frac{1}{2} \frac{1}{3} \frac{1}{4}$   $\frac{2}{1} \frac{2}{3} \frac{2}{4}$

Colapso de Margens: margens podem colapsar numa margem única com tamanho igual ao máximo das duas margens individuais se/ quando:

1. As margens de irmãos adjacentes estão colapsadas
2. Se a margem do pai e dos filhos não tem conteúdo a separá-las
3. As margens de topo e fundo de elementos vazios



## Pseudo-Seletores

Pseudo - Clas (:): seleciona elementos existentes, baseando-se no seu estado como se forem uma classe

a: link | visited | active | hover

form: focus | valid | invalid | required | optional | checked | -

:target | :first-child | :last-child | :first-of-type | :last-of-type ...

:only-child | :empty

:not(...)

:target: seleciona o elemento único cujo id corresponde ao identificador do fragmento no URL

Pseudo - Element (::): permite que rejam definições de elementos lógicos que não são nem elementos

::first-line | ::first-letter

::before | ::after

índice comeca em 0       $n \in \{0, 1, 2, \dots\}$

::nth-child( $a, n+b$ ): seleciona os elementos que não são os filhos índice  $b$  de um elemento depois de os seus filhos terem sido divididos em grupos de  $a$  elementos

content:

- none
- string
- url
- counter
- open-quote / close-quote

background-image: url(...)

background-position: left/right/center/length top/bottom/center/length

background-attachment: fixed/scroll/local

background-repeat: no-repeat/repeat-x/repeat-y/repeat

background-clip: border-box/padding-box/content-box

background: color position type repeat origin clip attachment image

list-style-type:

UL: none, disc, circle, square

OL: none, decimal, lower-alpha/greek/roman, upper-alpha/roman

list-style-image: url(...)

border:

border-collapse: separate/collapse

Transform: propriedade que modifica o espaço contido do modelo visual de formatação do CSS

Ex: rotate, skew, translate, scale

Transition: forma de controlar a velocidade de animação quando se mudam propriedades de CSS

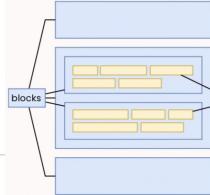
transition-property: opacity, left, top, height

transition-duration: ...

transition-delay: ...

transition-timing-function: ease-[in/out/in-out], linear, step-start/end

# The Flow



Fluxo Normal: como são colocados os elementos numa página antes de alterações no layout

Display: propriedade que controla se o elemento é block/inline/inline-block/non/...

Block: começam no topo e movem-se para baixo na página

- ocupam sempre uma nova linha
- se width = auto, ocupam o espaço horizontal interno do seu pai
- se height = auto, não tão altos quanto necessário para conta os seus filhos
- respeitam margens e padding
- podem conter outros elementos block-level e inline-level

Inline: começam à esquerda e movem-se para direita na página

- dispostos horizontalmente um depois do outro
- ignoram valores de altura e largura - ocupam o espaço necessário
- margem e padding de topo e fundo não afetam outros elementos
- podem ser alinhados verticalmente nos seus tops, fundos, linhas de base e outros
- podem partilhar de uma linha para a próxima se não houver mais espaço

Inline-Block: comportam-se como bloco que empilham horizontalmente

None: remove por completo o elemento da página, fazendo-o desaparecer e não ocupar espaço na página (mas diferente de fazê-lo invisível)

Position: static; relative; fixed; absolute

Static: o elemento mantém o seu lugar no fluxo do documento

Relative: o elemento mantém a sua posição no fluxo, mas pode ser movido de forma relativa à sua posição estática com top, left, right, bottom

Fixed: o elemento deixa de ser parte do fluxo e pode ser posicionado de forma relativa à janela do browser sem que o scrolling altere a sua posição

Absolute: o elemento deixa de ser parte do fluxo mas faz scroll com a página, podendo ser posicionado relativamente ao seu princípio faria não-estático

Float: propriedade que remove um elemento do fluxo do documento e move-o para a esquerda ou direita até tocar na costa da sua caixa contínua ou noutra elemento float

· Texto flui sempre à volta de elementos flutuantes (float)

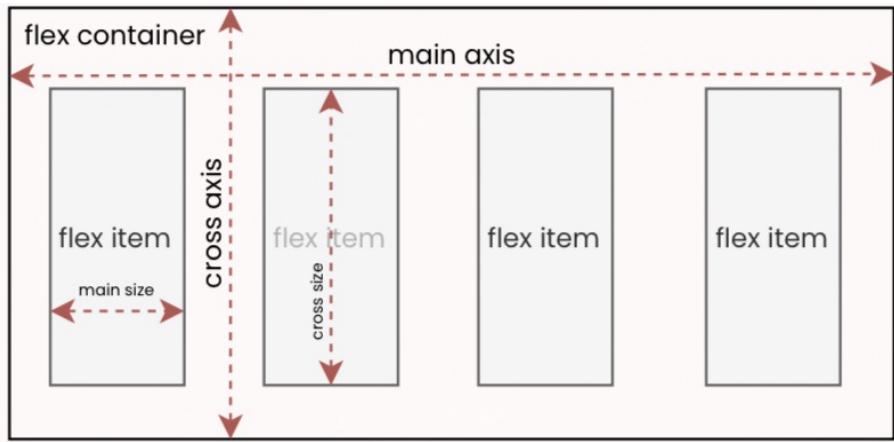
Clear: propriedade que indica se um elemento pode estar junto aos elementos flutuantes que precede ou deve ser movido para baixo (left/right/both)

z-index: propriedade que só pode ser aplicada a elementos posicionados (non-static) e que especifica a ordem em pilha de um elemento e dos seus descendentes, estando um elemento de maior ordem sempre à frente de um de menor ordem, útil para evitar que elementos fora do fluxo normal se sobreponham a outros

overflow: propriedade que especifica o comportamento de um elemento quando os seus conteúdos não cabem no tamanho especificado (visible/hidden/scroll/auto)

# Flebox

- Arraia os níveis dos blocos dos elementos pai e filho que não flexíveis ao elas
- Itens flebox não podem ser flutuantes
- As margens dos conteúdos flec não separam com as margens das suas caixas content boxes



display: flex

flebox-direction: row; row-reverse; column; column-reverse

flebox-wrap: nowrap; wrap; wrap-reverse

justify-content: flex-start; flex-end; center; space-around; space-between; space-evenly

align-items: flex-start; flex-end; center; baseline; stretch

align-self: ...

order / flebox-grow / flebox-shrink / gap

# Grid

- Permite alinhar elementos em colunas e linhas de tamanhos diferentes
- Elementos dispostos em grade podem ocupar a mesma celula que outros elementos, causando sobreposição e criando camadas
- Por defeito é atribuído um número a cada linha da grade começando em 1, mas pode ser atribuído um nome

display: grid

grid-template-columns/rows: auto / length / percentage / fraction  
repeat (n, size)

grid-column/row - start/end: atribui uma localização a um item  
grid-area: row-start/column-start / row-end/column-end

grid-template-areas: forma visual de definir a grade, atribuindo nome a item

[column-/row-] gap / justify-items / align-items / justify-content / align-content / justify-self / align-self

grid-auto-rows/columns: especifica o tamanho de grades implícitas

grid-auto-flow: row / column / dense

altera o comportamento do algoritmo de auto-colocação usado para colocar automaticamente itens não atribuídos a nenhuma localização

# Cascading

**Default:** cada browser tem o seu próprio conjunto de valores por defeito para as propriedades de cada elemento HTML

**Inherit:** valor especial que pode ser usado em quase todas as propriedades, significando herança do valor da propriedade de pai do elemento

**Especificidade:** a especificidade de cada regra é definida como três valores, (a, b, c), sendo cada um incrementado quando um certo tipo de seletor é usado

a: !!

b: Clas, Pseudo Clas, Attributo

c: Elemento, Pseudo Elemento

1. Regras com maior valor a são mais específicas

2. Se o valor a é igual para duas regras, usar-se b

3. Se necessário, usar-se c

A regra a ser aplicada é selecionada usando as seguintes regras, em ordem:

1. Origem (autor, utilizador, pré-definição)

2. Especificidade (maior é melhor)

3. Posição (último é melhor)

**Origem:**

autor: regras CSS definidas pelo criador da página

utilizador: preferências, links, etc., pelo utilizador

pré-definição: pré-definições do browser

Consider the following HTML code:

```
<section id="foo">
  <ul class="bar">
    <li class="first"><a href="#">A</a></li>
    <li class="second"><a href="#">B</a></li>
    <li><a href="#">C</a></li>
    <li><a href="#">D</a></li>
  </ul>
</section>
```

And the following CSS rules:

(ID, C|PC|A, E|PE)

section ul li	{ color : green }	/*R1*/	(0, 0, 3)
.bar .second	{ color : red }	/*R2*/	(0, 2, 0)
li a	{ color : cyan }	/*R3*/	(0, 0, 2)
section li:first-child ~ li	{ color : yellow }	/*R4*/	(0, 1, 3)
#foo .bar li :first-child	{ color : inherit }	/*R5*/	(1, 2, 1)
.bar li	{ color : magenta }	/*R6*/	(0, 1 1)

What rules apply to each of the elements: What is the rule with the highest specificity for each element:

1. section with id foo:  $\emptyset$
2. ul with class bar:  $\emptyset$
3. first list item:  $1, 6$
4. second list item:  $1, 2, 4, 6$
5. third list item:  $1, 4, 6$
6. fourth list item:  $1, 4, 6$

1. section with id foo:  $\emptyset$
2. ul with class bar:  $\emptyset$
3. first list item:  $6$
4. second list item:  $2$
5. third list item:  $4$
6. fourth list item:  $4$
7. first link:  $5$
8. second link:  $5$
9. third link:  $5$
10. fourth link:  $5$

What is the color value (including inherit) assigned to each element:

1. section with id foo:  $inherit$
2. ul with class bar:  $inherit$
3. first list item:  $magenta$
4. second list item:  $red$
5. third list item:  $yellow$
6. fourth list item:  $yellow$
7. first link:  $inherit$
8. second link:  $inherit$
9. third link:  $inherit$
10. fourth link:  $inherit$

What is the **final color value** presented in the browser:

1. first link:  $magenta$
2. second link:  $red$
3. third link:  $yellow$
4. fourth link:  $yellow$

# CSS Avançado

Variáveis:  
-- var-name: var-value;  
attribute: var(-- var-name);

Se nenhum valor é definido para uma variável num dado elemento, o valor do seu pai é usado — as variáveis também são herdados

Design Adaptativo: múltiplos layouts de largura fixa  
Design Flexível: múltiplos layouts em grade fluida

Design Hibrido: múltiplos layouts de largura fixa para ecrãs maiores, múltiplos layouts fluidos para ecrãs mais pequenos

Viewport: <meta name="viewport" content="width=device-width, initial-scale=1.0">

Media: <link rel="stylesheet" media="..." href="\_\_\_.css"/>

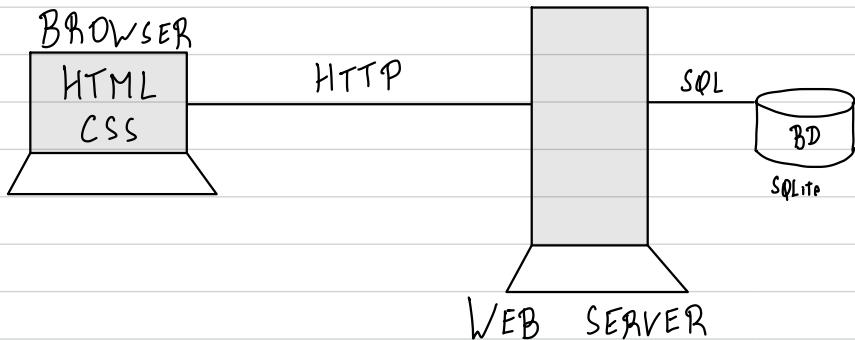
all; print; screen; speech; min/max-width/height; orientation = portrait/landscape

Operadores Lógicos:

- and
- , (or)
- not

Prefixo do Fornecedor: -webkit- / -moz- / -o- / -ms- / ...

# PHP



http://example.com/employee/view.php?id=1  
URL

PHP: PHP: Hypertext Preprocessor

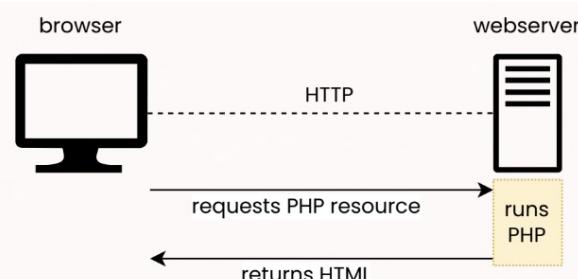
· Linguagem dinamicamente tipada

· <?php echo 'Hello World'; ?>  
· <?= 'Hello World'?>

Delimitadores: o interpretador de PHP não executa código PHP dentro dos seus delimitadores - tudo o que está fora dos delimitadores não é processado

Depois do processamento, os blocos de PHP não substituem pelo seu output

<?php echo 'Hello World'?>  
↓  
Hello World



1. O browser pede ao servidor por um recurso que corresponde a um script
2. O servidor corre o script PHP
3. O servidor retorna o resultado (normalmente um documento HTML) para o browser

Echo: output a strings <?php echo 'Hello World'?> @ <?= 'Hello World';?>

Comentário: // ... OU #... OU /\* ... \*/

Variável: \$var = ...; Tipos de Dado: bool; int; float; string  
gettype(\$var);

Atribuição por Valor: =

Atribuição por Referência: = &var

Type casting: conversão automática de tipos sempre que necessário

Null: variável sem valor

- 1. atribuída a constante null
- 2. não definido nenhum valor
- 3. unset()

var\_dump (...): função que mostra informação estruturada sobre uma ou mais expressões, incluindo o seu tipo e valor ≈ print\_r

## Estruturas de Controle

while: while(...) ...;      while(...) { ... };      while(...): ... endwhile;

Do While: do { ... } while(...);

For: for(...; ...; ...) ...;      for(...; ...; ...) { ... }

If: if(...) ...;      if(...) { ... }

Else: if(...) ... else ...      if(...) else { ... }

Break: termina a execução da estrutura de controle atual

Continue: salta o resto da iteração atual do ciclo e continua a execução na avaliação da condição

Switch: switch(...) { case: ... break; default: ... }

Die: termina a execução e recebe uma string de estado que impõem antes de sair

Exit: termina a execução e recebe um inteo como estado de saída

==: tipo comum antes da comparação - "loose"

===: tipos devem ser o mesmo - "strict"

## Estruturas de Dados

String: Série de caracteres

- '...': Sem sequências de escape (exceto '\ e \\') - para literais
- "...": Com sequências de escape e expansão de variáveis

Concatenação:

Array: mais ordenado organizado como uma coleção ordenada de pares chave - valor.

- Não tem de ser definido
- Não tem tamanho fixo
- Chaves podem ser inteiros ou strings
- Valores podem ser de qualquer tipo

Função: array (...) OU simplesmente usar uma variável como um array

- Quando uma chave não é fornecida, o PHP vai incrementar a maior chave anteriormente usada
- Pode existir arrays de arrays (multi-dimensionais)

Traversão: foreach (\$array as \$var) | foreach (\$array as \$key => \$value)

## Funções

function function name (...) { ... }

- As funções não têm de ser definidas antes de serem referenciadas
- Os nomes das funções não insensíveis a maiúsculas e minúsculas
- Por defeito, os parâmetros não passam por valor, mas podem ser passados por referência usando &
- Usar parâmetros com valores por defeito têm de ser os últimos
- O valor de retorno pode ser de qualquer tipo, incluindo array  
list(...) = f(...)

global \$variável: para usar uma variável global dentro de uma função

"Covice Typing": com type hints, os tipos não convertidos (não possível)  
↓

bool; int; float; string; array; object; class/interface; name; callable; iterable; self; parent

"Strict Typing": só aceita os valores correspondentes aos tipos declarados

→ declare(strict\_types=1), por default

? tipo: permite valores nulos

Exceção: evento que rompe com o fluxo normal da instrução

1. throw ...
2. try { ... } catch (...) { ... }

# Bases de Dados

PDO: extensão que define uma interface leve e consistente para aceder bases de dados em PHP

• \$dbh = new PDO('sqlite: database.db');

Exemplos:

1. \$query = \$dbh → prepare('INSERT INTO person (name) VALUES : (name)');
2. \$query → bindParam(': name', \$name);
3. \$query → execute

OU

1. \$query = \$dbh → prepare('INSERT INTO person (name) VALUES (?)');
2. \$query → execute(array(\$name));

Resultados: while (\$row = \$query → fetch())      OU      \$result = \$query → fetchAll()

\$dbh → setAttribute(PDO::ATTR\_DEFAULT\_FETCH\_MODE, PDO::FETCH\_ASSOC/NUM/BOTH);

Erros: PDO::ERRMODE\_SILENT/WARNING/EXCEPTION ↗ PDO Exception

Como configura? \$dbh → setAttribute(PDO::ATTR\_ERRMODE, ...);

Protocolo      Host      Port      Path      Query      Fragment  
Exemplo: http://www.example.com:80/path?key1=value1#fragment-id

\$-GET [...]      } variáveis superglobais  
\$-POST [...]

# Funções Utis

## STRINGS

1. int strlen (string \$string)
2. mixed strpos (string \$haystack, mixed \$needle [, int \$offset = 0])
3. string substr (string \$string, int \$start, [, int \$length])
4. mixed str\_replace (mixed \$search, mixed \$replace, mixed \$subject [, int & \$count])
5. array explode (string \$delimiter, string \$string [, int \$limit])
6. string implode (string \$glue, array \$pieces)

## ARRAYS

1. bool in\_array (mixed \$needle, array \$haystack [, bool \$strict = false])
2. mixed array\_search (mixed \$needle, array \$haystack [, bool \$strict = false])
3. bool array\_key\_exists (mixed \$key, array \$array) **COMPARA COMO STRING**
4. bool asort (array & \$array [, int \$sort\_flags = SORT\_REGULAR]) **VALORES**
5. bool krsort (array & \$array [, int \$sort\_flags = SORT\_REGULAR]) **CHAVES**
6. bool shuffle (array & \$array)
7. mixed array\_rand (array \$array [, int \$num = 1])
8. array list (mixed \$var\_1 [, mixed \$...])

# Classes

classe nome { definição }

**Propriedades:** public/protected/private e declaradas com ou sem inicialização (constante)

**Métodos:** devem aceder as propriedades da classe com \$this

**new:** cria uma instância de uma classe --construct(..) --destruct(..)

**extensão:** herança de métodos e propriedades **static:** partilha entre todas as instâncias

**self::** classe atual **parent::** classe pai **static::** classe do objeto atual

**abstract:** impede instanciação **interface:** classe puramente abstrata (... implement)

**final:** previne "overide" de métodos/extensão de classes

## Sessões

- O protocolo HTTP é stateless
- Nenhuma informação de estado é armazenada no servidor

Cookies: mecanismo para armazenar informação no browser  
enviada para o servidor em cada pedido ↗

```
bool setcookie ( string $name, string $value, int $expire = 0, string $path,
    string $domain, bool $secure = false, bool $httponly = false)
```

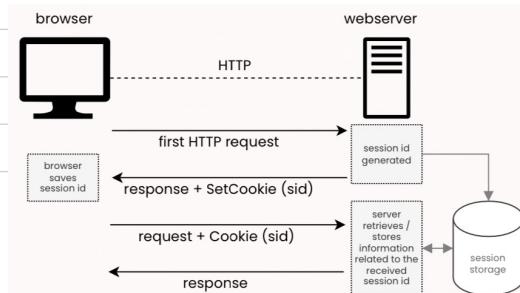
- Todos os parâmetros são opcionais exceto "name"
- Os cookies devem ser enviados antes de qualquer output do script, isto é, chamadas a esta função só antes de qualquer output

\$COOKIE: array especial para achar os cookies enviados pelo browser

- Como os cookies não são armazenados no browser, não podem ser usados com um mecanismo seguro para armazenar informação sensível
- Sessões são um mecanismo que pode ser usado para guardar informação de estado entre pedidos de página no servidor

Cliente: armazena um identificador único da sessão usando um cookie

Servidor: mantém informação da sessão associada com esse id da sessão



• PHP gera sessão automaticamente

Quando uma sessão é iniciada:

1. Recuperar informação da sessão

a. se um id de sessão é recebido (normalmente de um cookie), recuperar informação toda a informação de estado para esse id

b. se nenhum id de sessão é recebido, gerar um novo id de sessão e enviá-lo para o cliente (normalmente para um cookie)

2. Popular o array superglobal ***\$SESSION*** com informação da sessão associada com o id da sessão

3. Quando o script termina, serializar os conteúdos de ***\$SESSION*** e armazená-los

bool session\_start(): inicia uma sessão

• As sessões devem ser iniciadas antes de qualquer output

• Normalmente, chamada em todas as páginas para arreganhar que as variáveis da sessão estão sempre acessíveis

bool session\_destroy(): destrói toda a informação associada com a sessão atual

session\_set\_cookie\_params (int \$lifetime, string \$path, string \$domain, bool \$secure = false, bool \$httponly = false): altera os parâmetros do cookie usado para a sessão

• Todos os parâmetros são opcionais exceto *lifetime*

lifetime: tempo de vida do cookie da sessão, em segundos (0 = até fechar)

path: caminho no domínio onde o cookie deve funcionar (/ = todos os caminhos)

domain: domínio do cookie (... = todos os subdomínios)

## Cabeçalhos HTTP

- A função "header" envia um cabeçalho HTTP para o browser.
- Ex: `header("Location: page.php")`
- Os cabeçalhos devem ser enviados antes de qualquer output do script, mas não fazem a sua execução.

## Includes

- include (...): inclui e avalia o arquivo especificado
- require (...): se falha, produz um erro de compilação
- ... once (...): verifica se o arquivo já foi incluído

- "Includes" não são relativos ao ficheiro pedido pelo browser, não ao ficheiro que contém o "include"

\_\_FILE\_\_: caminho completo e nome do ficheiro do ficheiro atual  
\_\_DIR\_\_: pasta do ficheiro atual

string dirname (string \$path [, int \$levels = 1]): retorna a pasta do ficheiro

## JSON

JSON: formato leve de troca de informação (Java<sup>Script</sup> Object Notation)

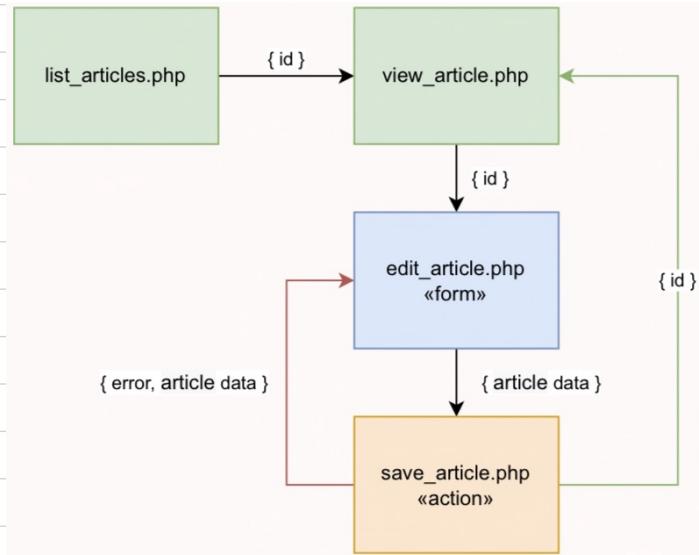
json\_encode (...)

json\_decode (...)

header ('Content-Type: application/json; charset=utf-8').

## Melhores Práticas

1. Validar o input: nunca confiar no utilizador, verificar sempre se a informação recebida está no formato esperado
2. Separar código PHP e HTML: começam sempre por calcular/consultar os dados e só depois dão impressão HTML
3. Não Repetir: usar includes e/ou funções para evitá-las repetições de código
4. Templates: criar e reutilizar funções template que imprimem código HTML, obtendo, no final, código PHP limpo
5. Separar Atão da Vista: nunca misturar scripts que retornam dados com scripts que alteram dados



## Java Script

- Linguagem dinâmica, imperativa e funcional → as funções não são cidadãos de primeira
- Linguagem orientada a objetos, mas baseada em protótipos (não em classes)
- Normalmente usada do lado do cliente, mas também pode ser de projeto geral

Consola: console.log(msg) imprime uma mensagem para a consola

'use strict' {  
· não há variáveis globais não declaradas  
· não há declaração de variáveis com "var" X  
· alguns warnings geram o re: erros

· Instruções não separadas por ;, mas pode ser omitida se forem separadas por uma mudança de linha

Variáveis: declaradas com "let"

· Java Script é uma linguagem larga/fraca e dinamicamente tipada

Constantes: declaradas com "const"

! Não declarar variáveis com "var"!  
! Declarar sempre variáveis!

Tipos de Dados Primitivos:

número	being Int
string	null
booleano	undefined (valor não atribuído)

Ystrings: definidas com ` `` ou `` (e não avaliadas as expressões dentro de \${}...{})  
+ retorna números, mas concatena strings (com strings ou com números)

Converção de Tipos: Boolean(); Ystring(); parseInt(..., base); parseFloat(..., base)

Comparação: ao comparar valores de tipos diferentes, não convertidos para números

• Primitivos não comparados por valor, mas objetos são comparados por referência

Falso: false; undefined; null; 0; NaN;  
tudo o resto é verdade (incluindo objetos)



typeof(...): retorna o tipo de uma variável

let a = b || c → atribui c se b avaliar para falso

let a = b ?? c → atribui c se b for null ou undefined

Estutura de Controle

- if ... else
- switch
- for / while
- for ... of
- break / continue

## Funções

function ... (...) { ... return ... }  $\Leftrightarrow$  let ... = function(...) { ... ... ... }

- Parâmetros primitivos são passados por valor
- Parâmetros não-primitivos (objetos) são passados por referência
- Uma função com "return" vazio ou sem "return" retorna undefined
- Se um parâmetro esperado por uma função não é passado, torna-se undefined e não sei que tipo de valor foi definido
- É possível passar funções como parâmetros para outras funções

Função leta: var f = function(...)  $\Rightarrow$  ...

- Não devem ser usadas como métodos
- Não podem ser usadas como constantes
- Não é ideal usar com call, apply e bind
- Não podem usar yield
- Se forem multi-linhas, têm de ter um retorno

## Objetos

Java Script é desenvolvida num paradigma simples baseado em objetos

Objeto: coleção de propriedades

Propriedade: associação nome - valor

Método: propriedade cujo valor é uma função

• Objetos constantes não podem ser reatribuídos, mas o que eles contêm pode mudar

for...in: itera sobre todas as propriedades de um objeto

[...]: accede às propriedades de um objeto

• Tudo é um objeto!

This: no contexto de execução global, refere-se ao objeto global

• chamada a função → undefined

• função seta → retém o contexto

• apply/call → primeiro argumento

• método → objeto

• evento do browser → objeto que dispara o evento

f.call(this, ..., ..., ..., ...)

f.apply(this, [..., ..., ..., ...])

bind: permite fixar "this" → recebe um contexto e cria uma função com ele

• As funções-seta não têm this, herdam-no do seu âmbito

## Protótipos

• Cada função tem um protótipo interno inicializado como um objeto vazio

new: cria um novo objeto derivado do protótipo

• A função é executada, tendo o novo objeto como contexto e retornando-o

• Cada objeto tem um protótipo da função que o criou → Object.getPrototypeOf(...)

• Quando se lê uma propriedade em falta de um objeto, JS vai ao protótipo

• Herança pode ser simulada com protótipos mudando a cadeia de protótipos

## Classes

classe: "syntactic sugar" para objetos baseados em protótipos

- Classes podem ter campos, métodos e um só construtor
- "this" refere-se ao objeto que chama o método
- "new" cria uma instância da classe

extender: herança/extensão de classes — o protótipo é a classe herdada

super: permite chamar o construtor e métodos da super-classe

static: permite declarar campos e métodos como parte da classe ( $\neq$  objeto)

- propriedade: protected

\* propriedade: private

instanceof (...)

## Arrays

array: objeto "tipo-lista" cujo protótipo tem métodos para travessia e mutação

• O [ ... ] for for ... of

Métodos: length; pop; push; reverse; sort; contains; join; slice; nome; every

• É possível adicionar métodos e propriedades mudando o protótipo

forEach(): executa uma função uma vez para cada elemento

filter(): retorna um novo array em que todos os elementos passam um teste

map(): cria um novo array aplicando uma função a todos os elementos originais

reduce(): aplica uma função a cada elemento sucessivamente

... : permite expandir um iterável em argumentos

**Dereferenciamento:** separa um iterável em variáveis ... : atribui o resto  
• Também funciona com objetos e em funções

**Map:** coleção de pares chave-valor que permite chaves de qualquer tipo  
→ get; set; delete; has; clear  
→ .keys(); .values(); .entries()

**Set:** coleção de valores de qualquer tipo que não contém repetidos  
→ add; delete; has; clear  
→ for ... of

## Erros

**try:** código a tentar

**catch:** código para lidar com exceções

**finally:** executa sempre

**throws:** lança exceções

## Ambito

• Uma variável definida dentro de um bloco de código só é visível dentro  
• Em blocos aninhados, cada um tem um ambiente lexical onde as variáveis são armazenadas com um apontador para o ambiente em que foi criado

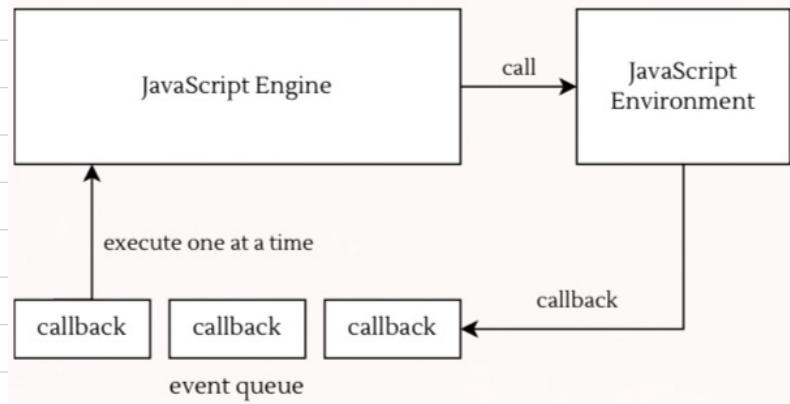
**Ambito:** ambiente atual → ambiente exterior → ... → global

• Quando uma função é criada, retém o ambiente lexical em que foi criada  
• Um novo fecho (combinacão função-ambiente) é criado sempre que uma função é criada

## Código assíncrono

- O código JavaScript é executado por uma JavaScript Engine que fornece uma heap, uma fila de chamadas e uma forma de correr código JavaScript é "single-threaded"
- Não há forma de fazer I/O

Ambientes JavaScript: fornecem as APIs necessárias para fazer I/O e permitem escalonar as asynchronous que correm em processos separados e independentes que quando terminam entram numa fila de eventos, à espera da sua vez de executar



- O ciclo de eventos é um ciclo infinito em que JSE espera por tarefas, executa-as e depois espera por mais tarefas
- Como JS é "single-threaded",
- Se as operações demoram muito tempo, o código vai esperar todo esse tempo

Promessa: representa o eventual resultado de uma operação assíncrona  
↳ objeto que tem uma função com dois parâmetros: funções "resolve" e "reject"

3 estados: 1. fulfilled      2. rejected      3. pending

- Quando uma promessa se resolve ou é rejeitada, .then e .catch consomem-na
- Em vez de usar "callbacks" para transformar código síncrono em assíncrono, funções assíncronas devem retornar promessas
- .then e .catch retornam sempre promessas
- Se se lança uma exceção dentro de .then, o código salta para .catch

Promise.all(...): recebe um array de promessas, corre-as em paralelo e espera por elas todas — .then é chamado quando elas todas se resolvem, mas se uma delas lança um erro, é chamado .catch

async: função retorna sempre uma promessa

await: JS espera por uma promessa e retorna o seu resultado

- Não funciona dentro de funções async
- usa o mecanismo do ciclo de eventos
- se um erro é lançado, a promessa é rejeitada

JSON  
↓

JSON.stringify (...)  
JSON.parse (...)

# DOM

Document Object Model

- É uma representação completamente orientada a objetos de uma página web como uma árvore lógica de noé
- Permite acesso programático à árvore, permitindo que programas leiam e alterem a estrutura, o estilo e o conteúdo do documento
- Pode ser manipulado pelo browser usando JavaScript

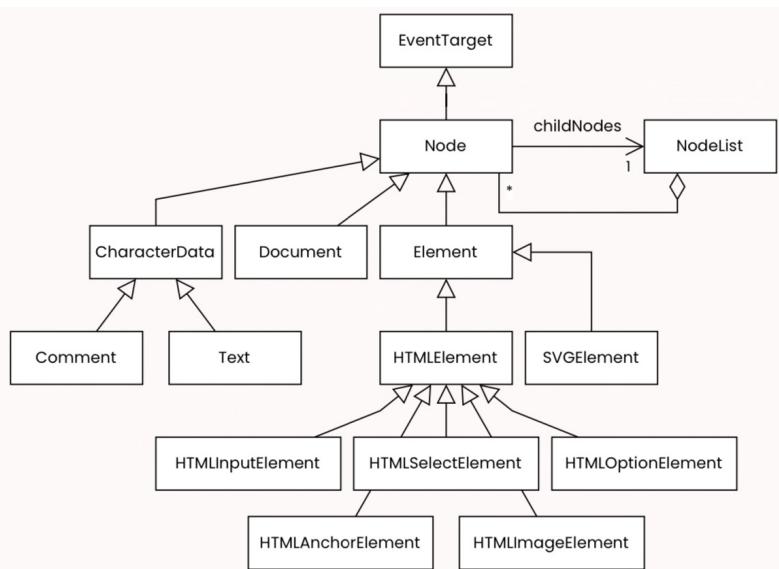
Document: representa o documento HTML → variável global "document"

## PROPRIEDADES

- URL: localização no de leitura do documento
- title: título do documento
- location: pode se atribuir um objeto Location para navegar para outro documento

Window: variável global que representa o browser

Location: função responsável por um URL nos seus vários componentes



`<script>` associa código JavaScript (embutido ou referenciado) com uma página HTML  
`</script>`

Por defeito, quando o browser encontra um elemento `<script>`, pausa a renderização de HTML, busca o script e executa o código

`defer`: só executa o código quando a página termina de carregar

`async`: não para o carregamento de HTML

## Elementos

Métodos de Document e Element:

1. get Element By ID (id) → Element
2. get Elements By ClassName (class) → NodeList
3. get Elements By TagName (tag) → NodeList
4. query Selector (selector) → Element (o primeiro)
5. query SelectorAll (selector) → NodeList

`Element`: representa um elemento num Document

### PROPRIEDADES

- id
- tagName
- innerHTML: código de marcação do conteúdo do elemento
- outerHTML: código de marcação que descreve o elemento, incluindo os descendentes
- textContent: conteúdo de texto

Adicionar/Modificar Atributos:

1. get Attribute (name)
2. set Attribute (name, value)
3. remove Attribute (name)
4. has Attribute (name)

classList: propriedade que retorna uma coleção "live" de classes  
para modificar o atributo "class" de um Element

1. add (class, ...)
2. remove (class, ...)
3. replace (oldClass, newClass)
4. toggle (class)

createElement: método de Document para criar novos elementos

HTML Element: herda de Element

Propriedades:

- style
- hidden

Métodos:

- focus()
- blur()
- click()

HTML\*Element: para cada tag HTML, há uma classe que implementa a interface  
HTML Element

1. HTML Input Element: name, type, value, checked, auto..., default..., disabled, required..., select()
2. HTML Select Element: name, multiple, required, size, length, add(...), item(...), remove(...)
3. HTML Option Element: disabled, selected, defaultSelected, text, value
4. HTML Anchor Element: href, host, hostname, port, hash, pathname, protocol, text, user, pass
5. HTML Image Element: alt, src, width, height

Nó: objeto que representa um nó na árvore do documento

• Ao adicionar nó, se o nó já tem um pai, primeiro é removido da sua localização atual

## Métodos:

1. appendChild (child)
2. replaceChild (newChild, oldChild)
3. removeChild (child)
4. insertBefore (newNode, referenceNode)
5. remove()

Node List: objeto que se comporta como um array de elementos

## Traversias

### Propriedades do Node:

- firstChild
- lastChild
- childNodes
- previousSibling
- nextSibling
- parent node
- nodeType

### Propriedades do Element:

- firstElementChild
- lastElementChild
- children
- previousElementSibling
- nextElementSibling

## Eventos

### Arquitetura Dirigida a Eventos:

- Eventos são ocorrências que acontecem no sistema
- Eventos específicos em objetos específicos podem ter "event handlers" anexados
- Quando o evento acontece, o "handler" anexado é chamado

Ex: Mouse Event (click, dblclick,...); Keyboard Event (keypress, keydown, keyup); Forms

Como anexar event handlers?

HTML: atributo nomeado `on{eventtype}` num elemento

Propriedade: define a propriedade correspondente

Função: addEvent Listener (evento, handle)

- A função que lida com o evento pode receber um parâmetro que representa o evento
- Dependendo do tipo, o evento pode ter diferentes métodos e propriedades

preventDefault(): impõe o comportamento padrão do browser

Bubbling: quando um evento ocorre num elemento, primeiro corre handlers anexados a ele, depois ao pai e assim sucessivamente até à raiz - em cada passo, o handler pode saber o alvo atual (event.currentTarget) e o estado inicial (event.target)

event.stopPropagation(): para bubbling

• O processamento de eventos tem duas fases:

1. Capturing: "mergulha" até ao elemento

2. Bubbling: o evento "sube" desde o elemento

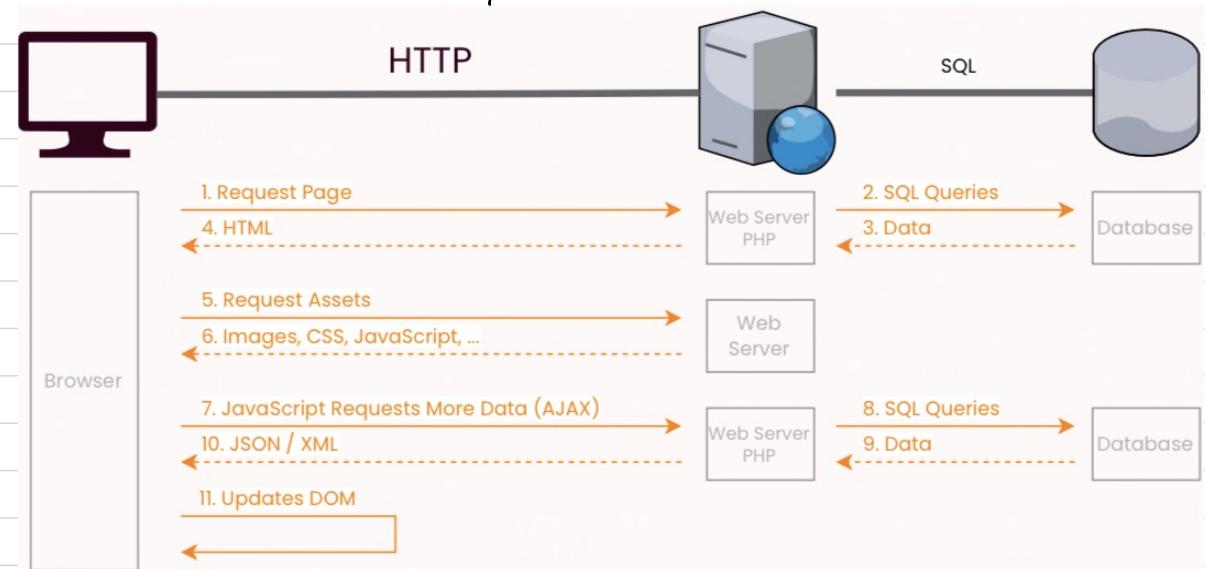
useCapture: permite definir a event handler na fase de captura

Nota Final: `window.addEvent Listener ('load', function() {}...{})`

faç o esperar que o DOM seja completamente carregado antes de adicionar eventos a elementos **MAS** o atributo def é melhor

# AJAX

JavaScript Asíncrono + XML



**XMLHttpRequest**: `.open(metodo, URL, async)`

Para enviar informações para o servidor, é necessário codificá-la primeiro

**XML**: response XML é um Objeto DOM que contém um documento XML parseado

**JSON**: response Text

**Fetch API**: interface para buscar recursos remotos → `fetch(...)`

- retorna uma Promessa que vai eventualmente ser "fulfilled" como uma Resposta
- Não vai ser rejeitada se houver um erro de rede ou de permissão
- Pode receber um objeto com parâmetros extra

Resposta: 1. ok    2. status    3. redirected    4. url    5. body

json(): retorna uma versão de uma resposta JSON

text(): retorna uma versão de uma resposta texto

## Timers

window.setInterval(function, delay): define um temporizador que executa uma função depois de um dado delay (retorna um id para cancelar o temporizador)

window.setInterval(function, interval): executa a função até ser parada com um tempo fixo entre chamadas (retorna um id para cancelar o temporizador)

## DOM Avançado

1. Fecho e Eventos
2. Bind e Eventos
3. Funções Parciais
4. Seletor com Maps
5. Seletor para Atuays
6. Atributos de Dados HTML5

data-id = "..."

Request		
Method	Path	Version
Headers		
Body		

# HTTP

Version	Status Code	Status Message
Headers		
Body		

- Protocolo de Transferência de Hiper-Texto
- Protocolo "application-layer" para transmitir documentos hypermedia
- Modelo cliente-servidor
- Protocolo "stateless"

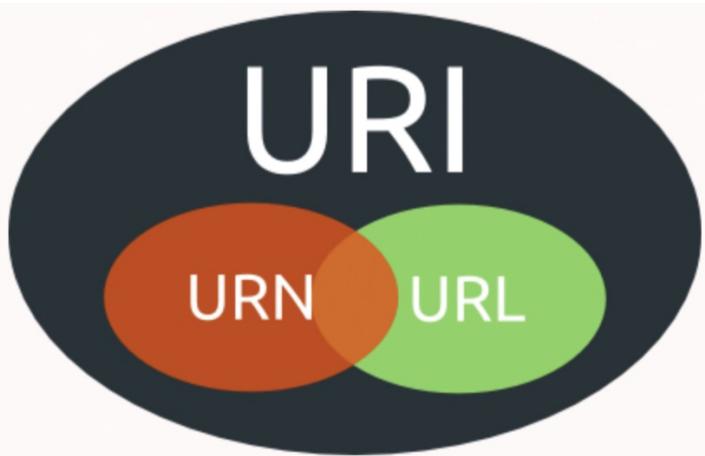
- O browser quer um recurso
- O browser estabelece uma conexão (TCP) ao servidor
- O browser envia um pedido
- O servidor retorna uma resposta

URI: Identificador Uniforme de Recursos

Identificador: objeto que pode atuar como referência para algo que tem identidade

No caso de um URI, o objeto é uma sequência de caracteres com uma sintaxe restrita

- |                                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><u>URI</u> {</p> <ul style="list-style-type: none"> <li>· <u>URN</u>: none</li> <li>· <u>URL</u>: localizado</li> </ul>                                            |
| <p><u>URI</u> {</p> <ul style="list-style-type: none"> <li>· esquema</li> <li>· autoridade</li> <li>· caminho</li> <li>· interrogação</li> <li>· fragmento</li> </ul> |



URN: Nomes Uniformes de Recursos — identificador de recursos persistente e independente da localização

URL: Localizador Uniforme de Recursos — identifica recursos via representação do mecanismo de acesso primário, em vez de pelos atributos (name, ...)

• O URN é o nome da pessoa que mora em URL!

URL HTTP: scheme://host:port/path?query #fragment

1. scheme: HTTP ou HTTPS
2. host: nome do domínio ou endereço IP
3. port: por omisão, 80
4. query: contém dados (pares nome - valor) para o servidor
5. fragment: especifica uma parte ou posição no documento

Pedido HTTP: conjunto de linhas

1. método do pedido seguido dos seus parâmetros
  - a. caminho do documento (URL absoluto)
  - b. versão do protocolo HTTP
- 2...: cada linha, um cabeçalho HTTP
- : linha vazia
- ...3...: bloco opcional de dados

Método: indica a ação a ser realizada pelo servidor

- O significado é semântico
- Os links usam sempre GET; os formulários usam GET ou POST

Métodos Seguros: não tem efeitos colaterais no servidor (não obrigatórios)

GET: pede uma representação do recurso identificado pelo URL (pedido com dados)

HEAD: só pede os cabeçalhos da resposta (o servidor não manda dados)

Métodos Idempotentes: os efeitos colaterais no servidor de vários pedidos idênticos são os mesmos efeitos colaterais que os de um só pedido

PUT: pede para a representação de um recurso ser armazenada no URL

DELETE: pede para o recurso identificado pelo URL ser apagado

- GET
- HEAD

POST: pede para a representação de um recurso ser armazenada no URL

O tipo do corpo é controlado pelo cabeçalho Content-Type, que pode ser alterado em formulários com o atributo "enctype"

application/x-www-form-urlencoded  
multipart/form-data  
text/plain

OPTIONS: pede opções de comunicação para um dado URL no servidor

Outros:

TRACE: realiza um teste "log-back"

CONNECT: pode ser usado para abrir um túnel

PATCH: aplica modificações parciais a um recurso

Resposta HTTP: conjunto de linhas

### 1. linha de estado

- a. número da versão HTTP
- b. código de estado
- c. frase da razão

<u>Resposta</u>	<u>1XX</u> : informação
	<u>2XX</u> : sucesso
	<u>3XX</u> : redirecionamento
	<u>4XX</u> : erro do cliente
	<u>5XX</u> : erro do servidor

### Códigos de Resposta:

2XX: 200 OK, 201 Criado, 202 Aceite, 204 Sem Conteúdo,  
206 Conteúdo Parcial

3XX: 301 Moved Permanently, 304 Não Modificado

4XX: 400 Mal Pedido, 401 Não Autorizado, 403 Proibido

404 Não Encontrado, 405 Método Não Permitido, 408 Timeout

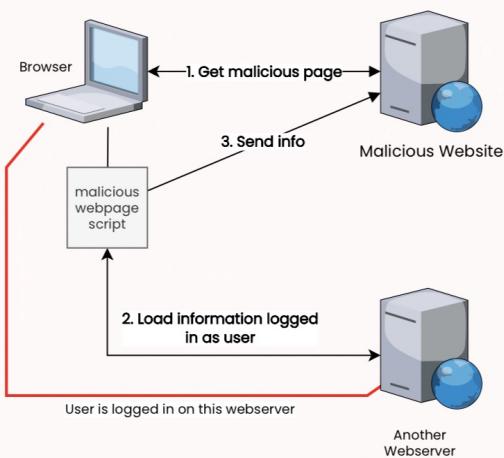
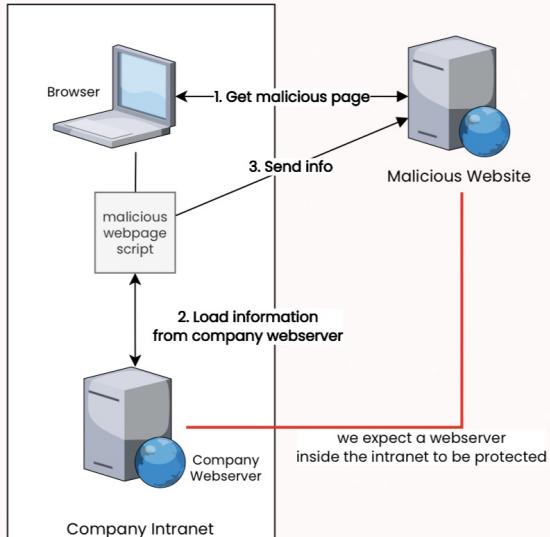
5XX: 500 Erro Interno do Servidor, 502 Bad Gateway, 503 Indisponível

Cabecalhos do Cliente: Accept, Accept-Charset, Accept-Encoding, Accept-Language,  
Connection, Cookie, Content-Length, Content-Type, Date, Host, If-Modified-Since,  
Range, User-Agent

Cabecalhos do Servidor: Accept-Ranges, Allow, Cache-Control, Content-Encoding,  
Content-Language, Content-Length, Content-Location, Content-Range, Content-Type,  
Expires, Last-Modified, Location, Set-Cookie

SOP - Same-Origin Policy: mecanismo de segurança que restrição como é que  
um documento/script carregado por uma origem pode interagir com um recurso  
de outra origem.

• A origem é protocolo - ambiente - porto



CORS - Cross-Origin Resource Sharing: mecanismo baseado em cabeçalhos HTTP para servidores web indicarem de que origens um browser deve permitir carregar recursos

REST - REpresentational State Transfer: estilo de arquitetura baseado em recursos para desenvolver aplicações em rede

Interface Uniforme: o sistema é constituído por recursos nomeados acedidos usando uma interface genérica

- Coisas (recursos) em vez de ações
- Recursos individuais são identificados em pedidos usando URLs
- Quando um cliente obtém a representação de um recurso, incluindo metadados anexos, tem informação suficiente para modificar/agregar o recurso no servidor
- Usa o Standard HTTP para descrever comunicação

Cliente-Servidor: separa a interface do utilizador do armazenamento de dados

Stateless: cada pedido do cliente ao servidor deve conter toda a informação necessária para compreender o pedido e não pode tirar vantagem de nenhum contexto armazenado no servidor

- A comunicação deve ser "stateless" por natureza
- O estado da sessão é mantido interamente no cliente

Cacheable: para melhorar a eficiência da rede, as respostas devem ser cacheadas se forem etiquetadas como "cacheable" ou não

- Se uma resposta é "cacheable", então a cache do cliente tem o direito de reutilizar os dados da resposta para pedidos equivalentes mais tarde

Sistema em Camadas: intermediários, tais como servidores de proxy, servidores de cache, gateways, ..., podem ser inseridos entre cliente e recursos para apoiar desempenho, segurança, ...

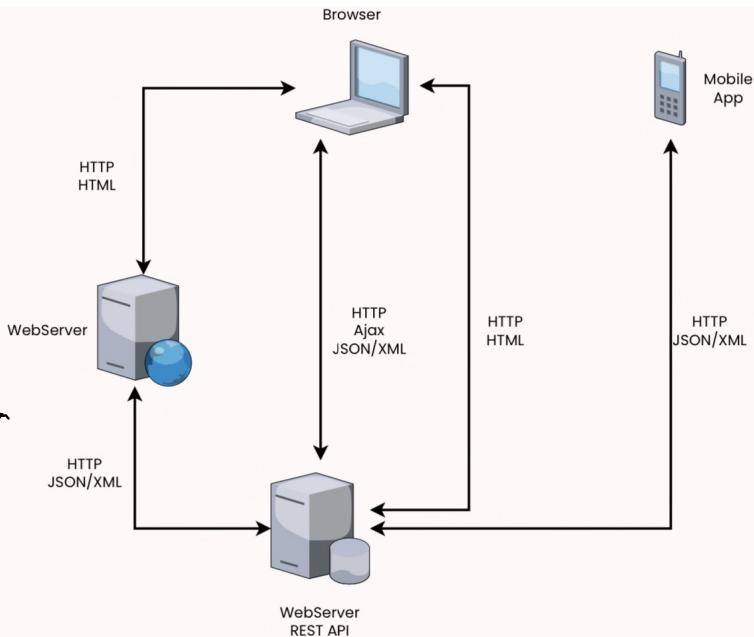
## PHP

header(...): adiciona cabeçalhos

\$\_SERVER['REQUEST-METHOD']

\$\_SERVER['HTTP-ACCEPT']

Pede uma representação particular de um recurso



# Segurança Web

Vulnerabilidade: buraco ou fraqueza na aplicação, que pode ser falha de design ou bug da implementação, que permite a um atacante causar dano aos stakeholders de uma aplicação

Ataque: técnica que os atacantes usam para explorar vulnerabilidades em aplicações

Ataque de Traversia do Caminho: usar os símbolos .. e / para obter acesso aos ficheiros e diretórios que não devem ser acedidos

Prevenção: substituir `include($_GET['page'])` por `if ($page == 'news') include ('news.php')`

Injeção SQL: inserção de uma interrogação via os dados de input do cliente para a aplicação

Prevenção: usar interrogações pré-preparadas

"Cross-Site Scripting" (XSS): tipo de injeção em que scripts maliciosos são injetados em sites anteriormente benignos e confiáveis

Tipos:

1. Persistente: o input do utilizador é armazenado no servidor
2. Refletido: o input do utilizador é imediatamente retornado pelo site
3. Baseado em DOM: o fluxo de dados nunca sai do browser

**"FILTER INPUT, ENCODE OUTPUT"**

Prevenir: nunca forneça dados não confiáveis diretamente num raiz, num comentário HTML, num nome de atributo/tag, diretamente em CSS ou em valores de atributos.

1. Validar
2. Filtrar
3. Codificar
4. "Escape"
5. HTTP Only

**"Cross-Site Request Forgery" (CSRF)**: a aplicação permite ao utilizador submeter um pedido de mudança de estado que não inclui nada secreto, se a vítima visita um dos sites do atacante enquanto já está autenticado em ... os pedidos submetidos não incluem automaticamente a informação de sessão do utilizador, autorizando o pedido do atacante.

Prevenir:

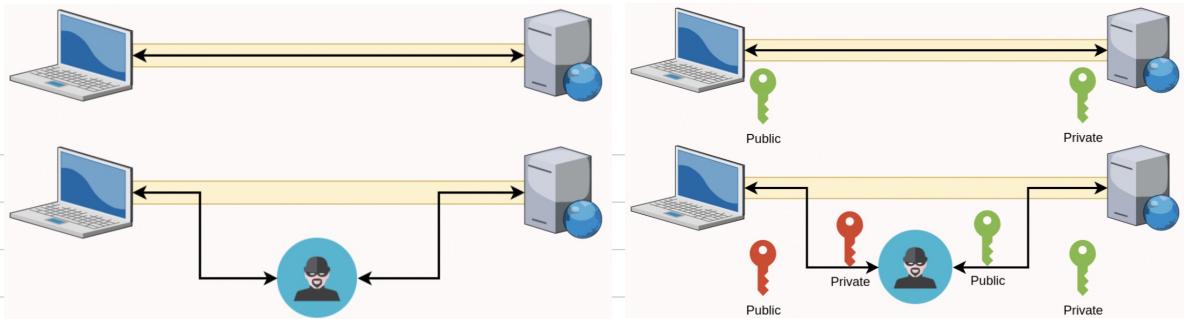
1. Gera um token aleatório por sessão
2. Armazena o token como uma variável da sessão
3. Envia o token como parte de cada pedido (resível)
4. Verificar se o token está contido em todas as páginas

**"Man-in-the-Middle"**: interceptar uma comunicação entre sistemas

**Criptografia de Chave Pública**: criptografia assimétrica que requer duas chaves seguras, uma privada e uma pública

· Se o emissor arma uma mensagem com a sua chave privada, qualquer receptor pode verificar que a mensagem foi enviada por ele

· Se o emissor encripta a mensagem com a chave pública, só o receptor com a chave privada pode ler a mensagem



MAS... usar encriptação não é suficiente!

SOLUÇÃO: usar chaves privadas armadas por uma autoridade certificada (CA)

Afirmativa Digital: esquema que permite demonstrar a autenticidade de uma mensagem

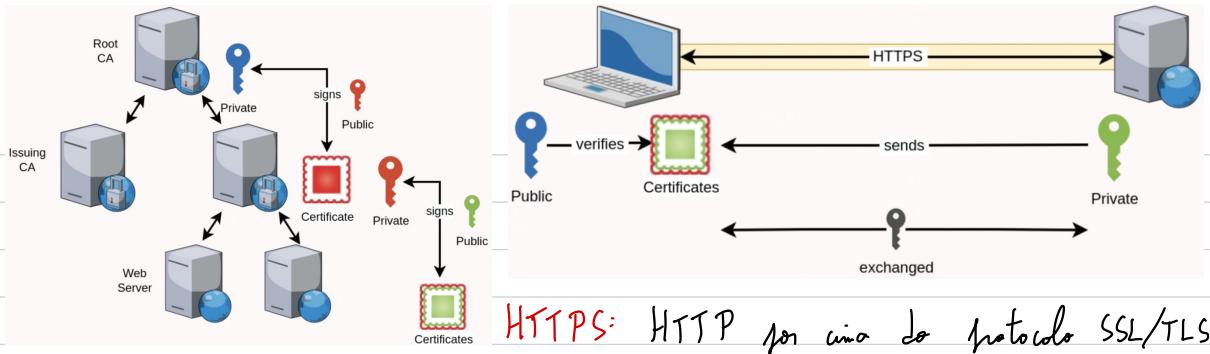
Certificado: ficheiro de dados que liga uma chave criptográfica a uma organização



Autoria Certificada: ao armazenar um certificado, verifica a informação de uma organização

- Os browsers confiam em sites baseados em CA's pré-instalados
- O utilizador confia que o CA só verifica sites legítimos
- O site fornece um certificado válido, armazenado por uma autoridade confiável
- O certificado identifica corretamente o site
- O utilizador confia que o protocolo da camada de encriptação é seguro

Cadeia de Confiança: lista ordenada de certificados, em que cada um certifica o próximo até à raiz ser alcançada



**HTTPS:** HTTP por cima do protocolo SSL/TLS

- O browser usa os certificados de CAs pré-montados para verificar a autenticidade da chave pública do servidor
- Por razões de eficiência, a criptografia de chave pública troca uma chave assimétrica que é usada para o resto da sessão

## Senhas & Credenciais

**NUNCA:** 1. Enviar por HTTP/GET

2. Encriptar no browser

**Hashing:** MD5, SHA1, SHA256, SHA512, bcrypt

1. A senha é cifrada e armazenada na base de dados

2. Numa tentativa de login, compararam-se as hashes das senhas

INTRODUÇÃO  
ARMAZENADA

**"Cracking":** 1. Força-Branca 2. Dicionário 3. Pesquisa 4. Arco-Íris

A função de cifra deve ser lenta o suficiente para impedir ataques

**Salt:** adicionam uma string única/diferente e longa a cada password

**PHP:** password\_hash (...) & password\_verify (...)



# Expressões Regulares

Expressão Regular: sequência de caracteres que forma um padrão de pesquisa

Matching: processo de aplica um padrão de expressão regular para uma string de texto e encontrar strings que não representadas por esse padrão

Validar: toda a string deve coincidir com o padrão

Procurar: uma sub-string deve coincidir com o padrão

Caracteres Literais: primeira ocorrência do caractere na string

Classe de Caractere: só um de vários caracteres [ ... ]

Alcance: ...

Negação: [ ^ ... ]

\d: dígito

\D: NÃO

\w: caracte de palavra

\W: NÃO

\^: espaço

\\$: NÃO

: qualquer caractere exceto mudanças de linha

Ancores: especificam a posição da string

^: string tem de estar no início

\\$: string tem de estar no fim

\b: fronteira / limite de uma palavra

Alternância: uma expressão regular de entre várias expressões regulares possíveis |

## Quantificadores

- ? : torno optional o token precedente na expressão regular
- \* : permite repetir o token 0 ou mais vezes
- + : permite repetir o token 1 ou mais vezes
- ? : permite repetir o token 0 ou 1 vez

{... , ...} : especificar o número mínimo e máximo de repetição

Por defeito, as expressões regulares são "greedy", mas podem ser tornadas "lazy" com um ? depois do operador de repetição

it group : por um grupo dentro de parêntesis cria um grupo (...)

- Os grupos não automaticamente capturados e numerados
- O match completo é sempre o grupo 0

? : cria um grupo, mas não o captura

\... : refere-se a grupos capturados

- ? = : match de algo requisido por ...
- ? != : match de algo não requisido por ...

? <= : "positive lookbehind"

? <!= : "negative lookbehind"

Expressões Regulares não NFA's

• Em HTML, os elementos de input têm um atributo "pattern" que pode conter uma expressão regular para especificar os valores permitidos no campo

PHP  
P.E.

preg-match: procura "subject" para uma match para a expressão regular dada em "pattern"

preg-match-all: procura todas as matches

preg-replace: substitui

preg-match: valida

Java Script  
P. E.

.test: retorna T/F

.match: retorna um array

.search: procura

.replace: substitui

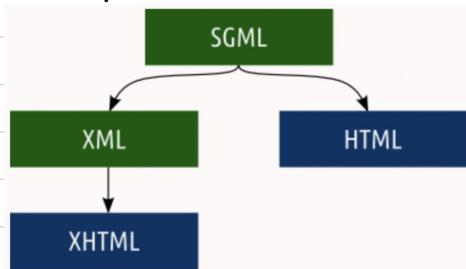
# XML

Linguagem de Marcação: conjunto de palavras e símbolos para descrever a identidade ou função de partes componentes de um documento

Extensible Markup Language: linguagem de marcação que define um conjunto de regras para codificar documentos de forma legível para humanos e máquinas → metalinguagem extensível (formato não fixo)

Standard Generalized Markup Language: padrão internacional para definir marcação para descrever a estrutura de diferentes tipos de documentos eletrônicos → grande, poderosa e complexa

- SGML e XML são metalinguagens; permitem aos utilizadores desenvolverem linguagens
- HTML e XHTML são linguagens concretas



XML: bem-formado se  
(2) contém exatamente uma raiz (1) contém 1 ou mais elementos  
(3) os elementos aninharam corretamente aplicações

<? instrução de processamento ?> permite ao documento conter instruções para  
<!-- comentário -->

<! [CDATA[...]]> secção usada para escapar blocos de texto que  
contêm caracteres que de outra forma seriam reconhecidos como marcação

< elemento > </ fecho > OU < fecho /> se forem vazios  
atributo = ... usado para metadados

<book> <card> (name) ... </name> <email> ... </email> </card> ... </book>

Metalinguagem: não impõe o uso de um conjunto restrito de elementos

- Para restringir a estrutura do documento a um formato usam-se esquemas de linguagem
- Um documento XML é válido se for bem-formado e de acordo com o esquema

Document Type Definition (DTD): modelo que define a estrutura de um documento XML válido especificando os nomes e tipos (do conteúdo) de elementos e atributos, bem como a sequência dos elementos no documento e a estrutura da árvore do documento `!DOCTYPE book [<!ELEMENT book (card*)> <!ELEMENT card (name, email)> <!ELEMENT name (#PCDATA)> ...]`

XML Schema Definition (XSD): mais expressivo do que DTD, com notação XML, auto-explicativa, simples e com documentação embutida; com suporte para "naming", tipos de dados definidos pelo utilizador, herança e evolução  
`<ns:schema> <ns:element name="book"> <ns:complexType> <ns:sequence> <ns:element name="card"> <ns:complexType> <ns:sequence> <ns:element name="name" type="ns:string"/> ... </ns:sequence> </ns:complexType> ... </...>`

Relax NG: alternativa a esquemas XML simples de gerar, com sintaxe XML, suporte para "naming" e auto-explicativa  
`<element name="book" xmlns="..."> <zeroOrMore> <element name="card"> <element name="name"> <text/> </element> ... </element> </zeroOrMore> </element>`

- Um único documento XML deve ser capaz de conter elementos e atributos que não definidos e usados por múltiplos módulos de software — N.S.
- Um naming XML é identificado por um URI  
definição: `xmlns = "..."` prefixo: `xmlns = prefixo`

**XSLT** (Extensible Stylesheet Language Transformation): linguagem para transformar documentos XML nouros documentos XML

**XSL-FO (XSL Formatting Objects)**: linguagem de marcação para formatação de documentos XML

**XPath**: linguagem de interrogação para selecionar nós de um documento XML

**XQuery**: linguagem de interrogação e programação funcional projetada para interrogar e transformar colecções de dados estruturados ou não, normalmente em XML

## XPath

• Linguagem que modela um documento XML como uma árvore de nós

- Típos de Nós*
- { 1. Documento: raiz da árvore (ter sempre um filho elemento)
  - 2. Elemento: tag XML
  - 3. Atributo
  - 4. Texto
  - 5. Comentário
  - 6. Instrução de Processamento <?...?>

- Típos de Nós*
- { 1. conjunto de nós
  - 2. booleanos
  - 3. números
  - 4. strings

**Caminho de Localização**: seleciona um conjunto de nós relativos ao nó de contacto  
/...: Caminho absoluto cujo nó de contacto é a raiz do documento

• Um caminho de localização é composto por passos de localização separados por /

Passo de Localização {  
1. Eixo  
2. Nó de Teste  
3. Predicado (zero ou mais)}

**Eixo:** especifica a relação na árvore entre os nós selecionados pelo passo de localização e o nó de contexto `self::`, `child::`, `parent::`, `descendant::`, `ancestor::attribute::`, `preceding::`, `following::`, `...-ibling::`, `...-or-self::`

• Cada eixo tem um tipo de nó principal - o das elementos que pode conter atributo → atributo      namspace → namspace      outro → element

**QName:** um nó de teste que é um QName é verdadeiro se o nó é o tipo do nó é o tipo do nó principal e tem um nome igual ao especificado  
`...::*`    `...::text()`    `...::comment()`    `...::processing-instruction()`    `...::node()`

**Predicado:** uma expressão `e1[e2]`, em que `e2` é um predicado, especifica uma expressão `e2` que seleciona nós de um conjunto maior `e1`

**Funções:** `last()`, `position()`, `count(node-set)`, `false()`, `true()`

`child::` forca se omitido num passo de localização (é o eixo por defeito)  
`//e` `descendant::e`

• `self::node()`

.. `parent::node()`

`@e` `attribute::e`

**JS:** `document.evaluate(...)`

# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
12 de Janeiro de 2015

Duração: 2h / Sem Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

- 1 val.** 1. Num site de notícias, qual a forma mais semanticamente correcta de criar uma notícia importante?

- <div id="noticia" class="important">...</div>
- <div id="important" class="noticia">...</div>
- <article class="important">...</article>
- <article id="important">...</article>

- 1 val.** 2. Ao examinar os logs do servidor, um administrador de sistemas reparou no seguinte pedido HTTP:

```
post_comment.php?title=&body=<script>window.onload = function() {var link=document.getElementsByName("a");link[0].href="http://do-something.com/";}</script>
```

Que tipo de vulnerabilidade estava a ser explorada?

- Cross-Site Scripting
- SQL Injection
- Javascript Vulnerability
- Cross-Site Request Forgery

- 1 val.** 3. Uma password deve ser sempre cifrada usando um algoritmo de *hashing* seguro. Que tipo de ataque é prevenido ao usar *salts* diferentes para cada password?

- Man in the Middle Attack
- Rainbow Tables
- Brute Force
- Dictionary Attack

- 1 val.** 4. Em PHP, onde são guardadas as variáveis de sessão?

- Na barra de endereços do browser
- Cookies no browser
- Ficheiros no servidor
- Em servidores proxy

- 1 val. 5. O que é impresso ao executar o seguinte código PHP?

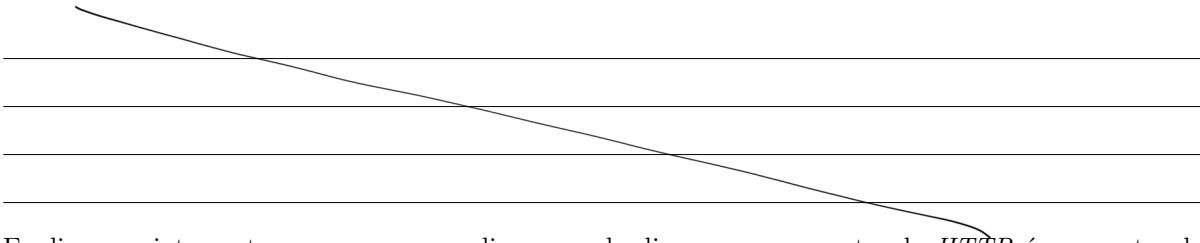
```
1 | <?php  
2 |   $foo = 10;  
3 |  
4 |   function add_to_foo($value) {  
5 |     $foo = $foo + $value;  
6 |   }  
7 |  
8 |   add_to_foo(5);  
9 |   echo $foo;  
10| ?>
```

10

---

- 1 val. 6. Porque é que o código de inicialização do jQuery deve ser chamado dentro de um *event handler* do tipo *ready*, e não directamente?

```
1 | $(document).ready(setUp);  
2 |  
3 | function setUp() {  
4 |   // code goes here  
5 | }
```



- 1 val. 7. Explique sucintamente o que queremos dizer quando dizemos que o protocolo *HTTP* é um protocolo *stateless*.

*não guarda estado*

---

---

---

---

- 1 val. 8. Qual é o principal objectivo da linguagem XSD?

- Validar ficheiros XML
- Gerar ficheiros XML
- Imprimir ficheiros XML
- Compilar ficheiros XML

# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
12 de Janeiro de 2015

Duração: 2h / Com Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

1. Considere o seguinte código HTML:

```
1 <article class="post" id="first">
2   <header>
3     <h1>Title</h1>
4   </header>
5   <p>First paragraph</p>
6   <p>Second paragraph</p>
7   <footer>
8     <p>This is a footer</p>
9   </footer>
10 </article>
```

E o seguinte código CSS:

```
1 article p {color: red;}          /* R1 */
2 article > p {color: blue;}        /* R2 */
3 .post p:first-child {color: green;} /* R3 */
4
5 p + p {color: yellow;}          /* R4 */
6 #first footer {color: magenta;}    /* R5 */
7 .post {color: cyan;}             /* R6 */
```

1½ val.

- (a) Indique a especificidade de cada uma das regras:

R1	R2	R3	R4	R5	R6
0, 0, 2	0, 0, 2	0, 2, 1	0, 0, 2	1, 0, 1	0, 1, 0

1 val.

- (b) Considerando apenas as regras de **R1** a **R3**, indique a cor de cada um dos textos:

Title	1st Par	2nd Par	Footer
invent	blue	blue	green

1 val.

- (c) Considerando **todas as regras**, indique a cor de cada um dos textos:

Title	1st Par	2nd Par	Footer
cyan	blue	yellow	green

2. Considere a seguinte *string*: Peter Piper picked a peck of pickled peppers.

Para cada uma das expressões regulares apresentadas de seguida, sublinhe qual o primeiro *match*:

1/2 val.

- (a) /pi.\*ck/

Peter Piper picked a peck of pickled peppers.

1/2 val.

- (b) /[a-k]{3}/

Peter Piper picked a peck of pickled peppers.

1/2 val.

- (c) /(\w{4}).\1/+

Peter Piper picked a peck of pickled peppers.

1/2 val.

- (d) /ck\b/

Peter Piper picked a peck of pickled peppers.

1/2 val.

- (e) /(pick|peck)(?=l)/

Peter Piper picked a peck of pickled peppers.

1/2 val.

- (f) /(?<=pep)per/

Peter Piper picked a peck of pickled peppers.

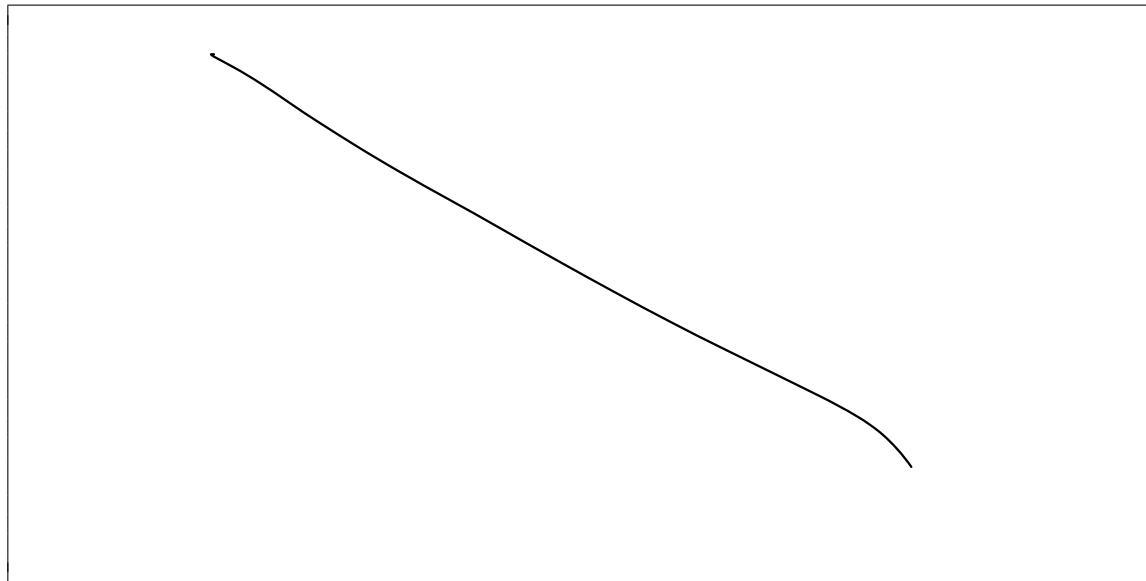
3. Considere o seguinte excerto HTML:

```
1 <input name="color" type="text" value="#336699">
2 <button id="copy" value="Copy">
3 <button id="send" value="Send">
4 <div class="box"></div>
```

Considere que pode haver outros elementos *input*, *button* e *div* no documento. Escreva o código *jQuery* necessário para que:

1 val.

- (a) Quando o botão *copy* é *clicado*, a cor de fundo do *div* mude para a cor especificada no texto do *input*.



# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
30 de Janeiro de 2015

Duração: 2h / Sem Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

- 1 val.** 1. Num site de notícias, qual a forma mais semanticamente correcta de mostrar o título de uma notícia?

- <article><h1 class="header">Título</h1>...</article>
- <article><span class="header"><h1>Título</h1></span>...</article>
- <article><div class="header"><h1>Título</h1></div>...</article>
- <article><header><h1>Título</h1></header>...</article>

- 1 val.** 2. Um utilizador recebeu um email contendo o seguinte link:

```
<a href="http://www.example.com/user/profile/action_delete.php">Funny Cats</a>
```

Que tipo de vulnerabilidade estava a ser explorada?

- Cross-Site Scripting
- Javascript Vulnerability
- Cross-Site Request Forgery
- SQL Injection

- 1 val.** 3. Qual destas estratégias é uma boa estratégia para prevenir ataques do tipo *brute force* a passwords?

- Não permitir passwords repetidas
- Usar dois *salts* para cada password
- Usar funções de *hashing* lentas
- Fazer *hash* das passwords várias vezes

- 1 val.** 4. Em PHP, a função *strpos* retorna a posição, começando em 0, da primeira ocorrência de uma *string* dentro de outra *string*; ou *false* se não a encontrar. Após o seguinte código, qual a forma correcta de verificar se nenhuma ocorrência foi encontrada?

```
$pos = strpos($search, $test);
```

- if (\$pos != true) echo "Not found";
- if (!\$pos) echo "Not found";
- if (\$pos == false) echo "Not found";
- if (\$pos === false) echo "Not found";

- 1 val. 5. Em PHP, quais os valores do array `$a` após o seguinte código ser executado?

```
1 | <?php
2 |   $a = array(0, 1, 2, 3);
3 |   foreach ($a as $k => $v)
4 |     $a[$k]++;
5 | ?>
```

$$\{ [0] \Rightarrow 1, [1] \Rightarrow 2, [2] \Rightarrow 3, [3] \Rightarrow 4 \}$$

- 1 val. 6. Explique, sucintamente, o que significa o símbolo `$` em jQuery.

---

---

---

---

- 1 val. 7. No protocolo *HTTP*, os métodos *GET* e *HEAD* são considerados *idempotentes*. Explique, sucintamente, o que isso significa.

(é igual (tem o mesmo efeito) fazer um ou vários pedidos  
os efeitos colaterais no servidor de vários pedidos idênticos não  
os mesmos efeitos colaterais de um só pedido)

- 1 val. 8. Em XSD, como é que definimos um elemento que pode conter outros elementos e/ou texto?

- Com um *mixed simpleType*
- Com um *mixed complexType*
- Com um *simpleType*
- Com um *complex type*

# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
30 de Janeiro de 2015

Duração: 2h / Com Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

1. Considere o seguinte código HTML:

```
1 <div id="selection">
2   <p><a href="">Two Lists</a></p>
3   <ul>
4     <li class="selected">First</li>
5     <li>Second</li>
6   </ul>
7   <ol class="other">
8     <li>Third</li>
9   </ol>
10 </div>
```

E o seguinte código CSS:

```
1 .selected {color: blue;}          /*R1*/
2 li + li {color: green;}           /*R2*/
3 #selection .other li {color: yellow;} /*R3*/
4
5 ul :first-child {color: magenta;}    /*R4*/
6 #selection p {color: red;}           /*R5*/
7 #selection li:first-child {color: cyan;} /*R6*/
```

1½ val.

- (a) Indique a especificidade de cada uma das regras:

R1	R2	R3	R4	R5	R6
0, 1, 0	0, 0, 2	1, 1, 1	0, 1, 1	1, 0, 1	1, 1, 1

1 val.

- (b) Considerando apenas as regras de **R1** a **R3**, indique a cor de cada um dos textos:

	R1	R2	R3
Two Lists	1st First	2nd Second	Third
inhe <del>n</del> t	blue	green	yellow

1 val.

- (c) Considerando **todas as regras**, indique a cor de cada um dos textos:

Two Lists	First	Second	Third
inhe <del>n</del> t red?	cyan R1 R4 R6	green R2	cyan R3 R6

2. Considere a seguinte *string*: The thirty-three thieves thought that they thrilled the throne throughout Thursday.

Para cada uma das expressões regulares apresentadas de seguida, sublinhe qual o primeiro *match*:

1/2 val.

- (a) /led.+ro/

The thirty-three thieves thought that they thrilled the throne throughout Thursday.

1/2 val.

- (b) /[thir]+[^e]/

The thirty-three thieves thought that they thrilled the throne throughout Thursday.

1/2 val.

- (c) /(\w{3}.+\1)/

The thirty-three thieves thought that they thrilled the throne throughout Thursday.

1/2 val.

- (d) /l1.\*e\b/

The thirty-three thieves thought that they thrilled the throne throughout Thursday.

1/2 val.

- (e) /(h|r|t){3}/

The thirty-three thieves thought that they thrilled the throne throughout Thursday.

1/2 val.

- (f) /(?<!h)o(?=u)/

The thirty-three thieves thought that they thrilled the throne throughout Thursday.

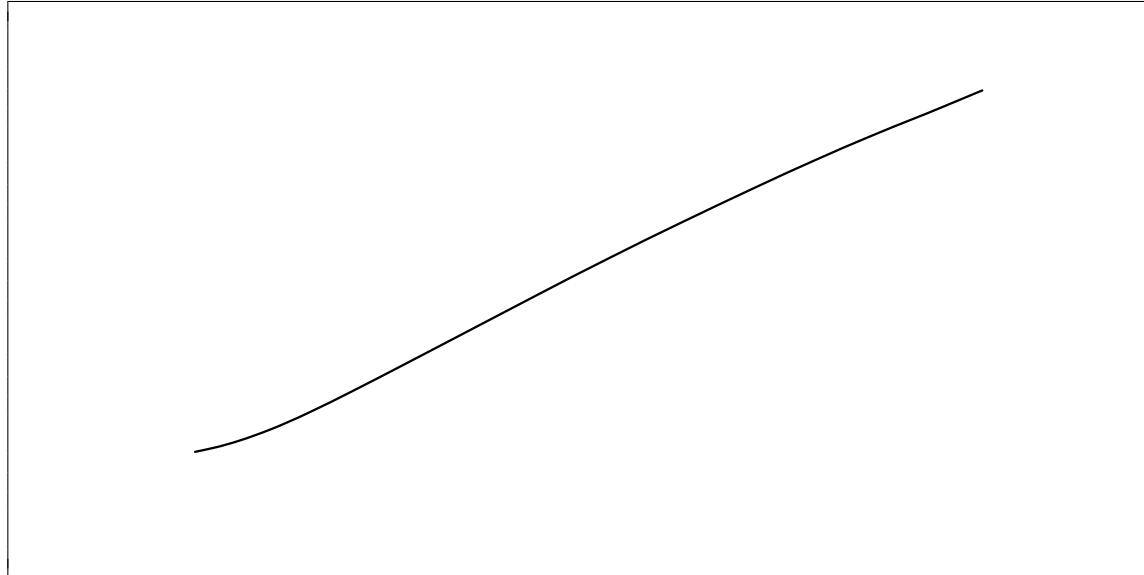
3. Considere o seguinte excerto HTML:

```
1 <script>
2 var secret = Math.floor((Math.random() * 100) + 1); // generates random number
3 var tries = 0;
4 </script>
5 <input name="username" type="text" placeholder="username">
6 <input name="guess" type="text">
7 <input id="guess" type="button" value="Guess">
```

Considere que pode haver outros elementos *input* no documento. Escreva o código *jQuery* necessário para que:

1 val.

- (a) Quando o botão *guess* é *clicado*, se o valor do *input* com o nome *guess* for menor do que a variável *secret*, um diálogo com a frase "go up" apareça, se for maior, que a frase seja "go down" e se for igual, que a função *correct* seja chamada. A variável *tries* deve ser incrementada em qualquer dos casos.



# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
19 de Janeiro de 2016

Duração: 2h / Sem Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

- 1 val.** 1. Em que situações se deve usar o elemento HTML *aside*?

- Quando o elemento vai ficar num dos lados da página.
- Quando o elemento não faz parte do conteúdo principal da página.
- Quando contém elementos de navegação.
- Quando não contém um *header*.

- 1 val.** 2. Qual a forma correcta de prevenir um ataque do tipo *Cross-Site Request Forgery*?

- Usar sempre formulários do tipo POST.
- Fazer os pedidos usando AJAX.
- Usar um *token* de segurança diferente para cada sessão.
- Ter sempre um diálogo de confirmação.

- 1 val.** 3. Uma password deve ser sempre cifrada usando um algoritmo de *hashing* seguro. Que tipo de ataque é prevenido ao usar uma função de *hash* lenta?

- SQL Injection
- Dictionary Attack
- Brute Force
- Rainbow Tables

- 1 val.** 4. No protocolo HTTP, como são enviadas as *cookies*?

- Numa resposta AJAX usando JSON.
- Na barra de endereços do *browser*.
- Ficam guardadas no servidor.
- Num header HTTP.

- 1 val.** 5. Em PHP, ao declararmos uma variável como *global* dentro de uma função, essa variável:

- Fica disponível em qualquer outra função.
- Fica disponível em qualquer outro script dentro da directória.
- Refere-se a uma variável global do *script*.
- Não pode ser usada noutra função.

(continua do outro lado...)

- 1 val. 6. Assuma que o seguinte código *jQuery* se encontra definido dentro do evento *ready* de um documento HTML. Analise, com atenção, o código e indique qual o comportamento esperado:

```
1 | $(‘a’).click(say(‘Hello’));  
2 |  
3 | function say(message) {  
4 |     alert(message);  
5 | }
```

- É mostrado um alerta com a mensagem *Hello* assim que a página carrega.
- É mostrado um alerta com a mensagem *Hello* quando se carrega num link.
- Não acontece nada.
- A página não carrega.

- 1 val. 7. Explique sucintamente quais nodos são seleccionados pela seguinte expressão *XPath*:

.//person/job

Ox nos job, filhos de person, descendentes do nó atual

---

---

- 1 val. 8. Em XSD, um *complexType* pode conter texto?

- Sim, desde que não contenha outros elementos XML.
- Sim, desde que o atributo *mixed* tenha o valor *true*.
- Sim, desde que o atributo *text* tenha o valor *true*.
- Não, só elementos *simpleType* podem conter texto.

# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
19 de Janeiro de 2016

Duração: 2h / Com Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

1. Considere o seguinte código HTML:

```
1 <div class="post" id="first">
2   <div class="header">
3     <h1>Title</h1>
4   </div>
5   <p>First paragraph</p>
6   <p>Second paragraph</p>
7   <div class="footer">
8     <p>This is a footer</p>
9   </div>
10 </div>
```

E o seguinte código CSS:

```
1 #first div {color: blue;}          /* R1 */
2 div .header {color: magenta;}      /* R2 */
3 p + p + div :first-child {color: red;} /* R3 */
4
5 h1 {color: green;}               /* R4 */
6 div + p {color: cyan;}          /* R5 */
7 p {color: yellow;}              /* R6 */
```

1½ val.

- (a) Indique a especificidade de cada uma das regras (e.g. 0,2,2,1):

R1	R2	R3	R4	R5	R6
1, 0, 1	0, 1, 1	0, 1, 3	0, 0, 1	0, 0, 2	0, 0, 1

1 val.

- (b) Considerando apenas as regras de **R1** a **R3**, indique a cor de cada um dos textos:

Title	1st Par	2nd Par	Footer
inher	inher	inher	red

1 val.

- (c) Considerando todas as regras, indique a cor de cada um dos textos:

R4	R5	R6	R7
Title	1st Par	2nd Par	Footer
green	cyan	yellow	red

2. Considere a seguinte *string*:

When you write copy you have the right to copyright the copy you write

Para cada uma das expressões regulares apresentadas de seguida, sublinhe qual o primeiro *match*:

1/2 val.

(a) /copy.\*right/

When you write copy you have the right to copyright the copy you write

1/2 val.

(b) /[write]/

When you write copy you have the right to copyright the copy you write

1/2 val.

(c) /(\w{4}).\*\1/

When you write copy you have the right to copyright the copy you write

1/2 val.

(d) /write\$/

When you write copy you have the right to copyright the copy you write

1/2 val.

(e) /(ri|py)(?!t)/

When you write copy you have the right to copyright the copy you write

1/2 val.

(f) /(\w{3,}?) .\*?\1/

When you write copy you have the right to copyright the copy you write

3. Considere o seguinte excerto HTML:

```
1 <form id="register" action="register.php" method="post">
2   <input name="username" type="text">
3   <input name="password" type="password">
4   <input type="submit" value="Register">
5 </form>
```

Considere que pode haver outros elementos *input* e *submit* no documento. Escreva o código *jQuery* necessário para que:

1 val.

(a) Quando o *input password* perde o foco, se verifique se esta tem pelo menos 8 caracteres e que pelo menos um deles seja um símbolo que não uma letra, um número ou um *underscore*. Caso tal não se verifique, o *border* do input deve passar a vermelho.

# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
1 de Fevereiro de 2016

Duração: 2h / Sem Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

**1 val.** 1. Que atributo é usado para definir que *inputs* do tipo *radio* pertencem ao mesmo grupo?

- O atributo *group*.
- O atributo *name*.
- O atributo *label*.
- O atributo *id*.

**1 val.** 2. Os codificadores dependentes do contexto (*context aware encoders*) são usados quando queremos fazer *output* de dados em que não confiamos. Que tipo de ataque tentam prevenir?

- CSRF (*Cross-site Request Forgery*).
- SQL Injection.
- XSS (*Cross-site Scripting*).
- Path Traversal Attack.

**1 val.** 3. Quando nos ligamos a um servidor usando o protocolo HTTPS, é usada um esquema de chave pública e chave privada. Como é que se garante que a chave pública transmitida pelo servidor é realmente desse servidor e que não estamos a sofrer um *man in the middle attack*?

- A chave foi obtida através de um servidor de chaves seguro.
- As chaves públicas contêm um *checksum* que permite verificar a sua validade.
- A chave foi assinada por uma entidade certificadora reconhecida pelo *browser*.
- A chave é transmitida através de um mecanismo seguro (TLS).

**1 val.** 4. No protocolo HTTP, qual dos seguintes métodos **não** é idempotente?

- GET
- DELETE
- POST
- HEAD

**1 val.** 5. O que é impresso pelo seguinte código PHP:

```
1 | $a = array('a', 'b', 'c');  
2 | foreach ($a as $b => $c)  
3 |   echo $b;
```

012

(continua do outro lado...)

**1 val.** 6. Assuma que o seguinte código *jQuery* se encontra definido dentro do evento *ready* de um documento HTML. Analise, com atenção, o código e indique qual o comportamento esperado:

1 | `alert($('a').css('color'));`

- Aparece uma caixa de diálogo com a cor de todos os *links* do documento.
- Aparece uma caixa de diálogo para cada *link* do documento com a sua cor.
- Aparece uma caixa de diálogo com a cor do primeiro *link* do documento.
- Aparece um erro na consola pois não podemos fazer *alert* de um *array*.

**1 val.** 7. Em XPath, qual é o eixo por omissão?

- element
- parent
- comment
- child

**1 val.** 8. Em XSD, o que significa o atributo *fixed* no seguinte código?

1 | `<xs:attribute name="value" type="xs:integer" fixed="5"/>`

- O atributo tem de estar sempre presente e o valor tem de ser sempre 5.
- O valor do atributo é 5 por omissão.
- Se o atributo estiver presente, tem de ter o valor 5.
- O atributo tem de aparecer 5 vezes.

# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
1 de Fevereiro de 2016

Duração: 2h / Com Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

1. Considere o seguinte código HTML:

```
1 <div id="profile" class="box">
2   <h2 class="name">John Doe</h2>
3   <ul>
4     <li><a href="www.johndoe.com">Website</a></li>
5     <li class="phone">111-222-333</li>
6     <li class="address">Saint Doe Street , 123</li>
7   </ul>
8 </div>
```

E o seguinte código CSS:

```
1 ul li:first-child {color: red;}          /* R1 */
2 #profile.box .name {color: blue;}        /* R2 */
3 #profile li ~ li ~ li {color: green;}    /* R3 */
4
5 #profile .address {color: magenta;}      /* R4 */
6 div ul li.address {color: yellow;}       /* R5 */
7 a {color: cyan;}                        /* R6 */
```

- 1½ val. (a) Indique a especificidade de cada uma das regras (e.g. 0,2,2,1):

R1	R2	R3	R4	R5	R6
0, 1, 2	1, 2, 0	1, 0, 3	1, 1, 0	0, 1, 3	0, 0, 1

- 1 val. (b) Considerando apenas as regras de **R1** a **R3**, indique a cor de cada um dos textos:

R2

John Doe	Website	111-222-333	Saint Doe Street, 123
blue	inherit	inherit	green

- 1 val. (c) Considerando **todas as regras**, indique a cor de cada um dos textos:

R2      R6      R3      R4      R5

John Doe	Website	111-222-333	Saint Doe Street, 123
blue	cyan	inherit	magenta

2. Considere a seguinte *string*:

How many yaks could a yak pack pack if a yak pack could pack yaks

Para cada uma das expressões regulares apresentadas de seguida, sublinhe qual o primeiro *match*:

1/2 val.

(a) /pack.\*pack/

How many yaks could a yak pack pack if a yak pack could pack yaks

1/2 val.

(b) /[pack]{2}/

How many yaks could a yak pack pack if a yak pack could pack yaks

1/2 val.

(c) /(yak|pack).\*\1/

How many yaks could a yak pack pack if a yak pack could pack yaks

1/2 val.

(d) /[^aeiou]{3}/

How many yaks could a yak pack pack if a yak pack could pack yaks

1/2 val.

(e) /(?!ya)k/

How many yaks could a yak pack pack if a yak pack could pack yaks

1/2 val.

(f) /(\w{3,}?) .\*?\1/

How many yaks could a yak pack pack if a yak pack could pack yaks

3. Considere o seguinte excerto HTML que representa um teclado virtual que pretende impedir ataques usando *keyboard loggers*:

```
1 <form id="pin" method="post">
2   <input type="text" name="username">
3   <input type="text" name="pin">
4   <input type="submit" value="Verify">
5 </form>
6 <div id="keypad">
7   <a href="#">1</a> <a href="#">2</a>   <a href="#">3</a><br>
8   <a href="#">4</a> <a href="#">5</a>   <a href="#">6</a><br>
9   <a href="#">7</a> <a href="#">8</a>   <a href="#">9</a><br>
10 </div>
```

Considere que pode haver outros elementos *a*, *input* e *submit* no documento. Escreva o código *jQuery* necessário para que:

1 val.

(a) Quando o utilizador *clique* num dos números do teclado virtual, o valor desse número seja acrescentado ao valor do *input* com o nome *pin*.

# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
12 de Janeiro de 2017

Duração: 2h / Sem Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

**Nota:** Nas perguntas de escolha múltipla, cada resposta errada desconta  $\frac{1}{3}$  da cotação mas pode escolher mais do que uma opção.

- 1 val.** 1. Qual destes elementos **não** é obrigatório num documento HTML válido?

- O elemento *head*.
- O elemento *body*.
- O elemento *title*.
- O elemento *section*.

- 1 val.** 2. Que elementos selecciona o seguinte selector CSS?

```
header h1, h2 { ... }
```

- Elementos *h1* e *h2* dentro de um *header*.
- Elementos *h1* dentro de um *header* e qualquer elemento *h2*.
- Elementos *h1*, *h2* e *header*.
- Nunca selecciona nenhum elemento.

- 1 val.** 3. Um ataque do tipo *Session Fixation* consiste em...

- Decorar e copiar a cookie de sessão de um utilizador.
- Roubar uma cookie de sessão enviada de forma insegura.
- Tirar partido de cookies não cifradas.
- Forçar uma certa cookie de sessão a um utilizador.

- 1 val.** 4. É uma boa ideia fazer *hash* das passwords no *browser*, usando *Javascript*, em vez de o fazer no servidor?

- Não, porque é mais lento.
- Não, porque se perdem algumas das vantagens de se fazer hash.
- Sim, porque previne que a password seja interceptada.
- Sim, porque normaliza o tamanho das passwords.

- 1 val.** 5. No protocolo HTTP, podemos dizer que um método seguro é idempotente?

- Só se forem usadas ligações seguras.
- Só se não forem passados parâmetros por GET.
- Sempre.
- Nunca.

(continua do outro lado...)

**[1 val.]** 6. Em PHP, ao declararmos uma variável como *global* dentro de uma função, essa variável:

- Fica disponível em qualquer outro script dentro da directória.
- Não pode ser usada noutra função.
- Refere-se a uma variável global do *script*.
- Fica disponível em qualquer outra função.

**[1 val.]** 7. Em Javascript, a palavra chave *this* refere-se sempre...

- Ao objecto actual.
- A um objecto que representa o *documento*.
- A um objecto que representa o *browser*.
- Ao contexto de execução.

**[1 val.]** 8. Em *XPath*, o tipo de nodo principal...

- Depende do eixo usado.
- É sempre *elemento*.
- Depende do nodo de contexto.
- Tem de ser sempre especificado.

# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
12 de Janeiro de 2017

Duração: 2h / Com Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

1. Considere o seguinte código HTML:

```
1 <div class="widget">
2   <ul id="todo">
3     <li>Buy Bread</li>
4     <li>Learn Guitar</li>
5     <li class="important">Pay Bills</li>
6     <li class="postponed">Wash Car</li>
7   </ul>
8 </div>
```

E o seguinte código CSS:

```
1 li:first-child {color: blue}          /* R1 */
2 div li {color: red}                  /* R2 */
3 div.widget ul#todo {color: cyan}    /* R3 */
4
5 div > ul#todo .important {color: green} /* R4 */
6 ul li.postponed {color: inherit}      /* R5 */
7 li + li + li {color: magenta}        /* R6 */
```

1½ val.

- (a) Indique a especificidade de cada uma das regras (ex: 0,2,2,1):

R1	R2	R3	R4	R5	R6
0, 1, 1	0, 0, 2	1, 1, 2	1, 1, 2	0, 1, 2	0, 0, 3

1 val.

- (b) Considerando apenas as regras de **R1** a **R3**, indique a cor de cada um dos textos:

Buy Bread	Learn Guitar	Pay Bills	Wash Car
blue	red	red	red

1 val.

- (c) Considerando **todas as regras**, indique a cor de cada um dos textos:

Buy Bread	Learn Guitar	Pay Bills	Wash Car
blue	red	green	inherent cyan

2. Considere a seguinte *string*:

Washing the washing machine while watching the washing machine washing washing

Para cada uma das expressões regulares apresentadas de seguida, sublinhe qual o **primeiro match**:

1/2 val.

(a) /wa.\*ing/

Washing the washing machine while watching the washing machine washing washing

1/2 val.

(b) /[a-z]{3}\b/

Washing the washing machine while watching the washing machine washing washing

1/2 val.

(c) /(ing).\*?\1/

Washing the washing machine while watching the washing machine washing washing

1/2 val.

(d) /^.{3}/

Washing the washing machine while watching the washing machine washing washing

1/2 val.

(e) /(sh|ch)(?!ing)/

Washing the washing machine while watching the washing machine washing washing

1/2 val.

(f) /( .+?) .\*?\1/

Washing the washing machine while watching the washing machine washing washing

3. Considere o seguinte excerto HTML:

```
1 <div id="products">
2   <ul>
3     <li>Apple: <span class="qty">3</span> <a href="#">+</a></li>
4     <li>Banana: <span class="qty">5</span> <a href="#">+</a></li>
5     <li>Pear: <span class="qty">6</span> <a href="#">+</a></li>
6   </ul>
7   <a href="#" class="buy">Buy</a>
8   <p class="total">0</a>
9 </div>
```

Considere que pode haver outros elementos *ul*, *li* e *a* no documento. Escreva o código *jQuery* necessário para que:

1 val.

(a) Quando o *link* no final de cada linha seja *clicado*, a quantidade nessa linha seja incrementada uma unidade.

4. Considere o seguinte documento em XML:

```
1 <authors>
2   <author country="Spain" name="Miguel de Cervantes">
3     <book year="1605" type="Novel">Don Quixote</book>
4   </author>
5   <author country="England" name="William Shakespeare">
6     <book year="1599" type="Tragedy">Hamlet</book>
7     <book year="1606" type="Tragedy">Macbeth</book>
8   </author>
9   <author country="Russia" name="Leo Tolstoy">
10    <book year="1865" type="Novel">War and Peace</book>
11  </author>
12  <author country="Portugal" name="Jose Saramago">
13    <book year="1995" type="Novel">Ensaio sobre a Cegueira</book>
14    <book year="1997" type="Novel">Todos os Nomes</book>
15  </author>
16 </authors>
```

Considerando que o nodo de contexto é a raiz do documento, escreva as expressões XPath que seleccionam os seguintes elementos:

1/2 val.

- (a) O nome de todos os autores.

/authors/author/@name

1/2 val.

- (b) O título dos livros do tipo *Novel*.

/authors/author/book[@type = "Novel"]/text()

1/2 val.

- (c) O nome dos autores que escreveram mais do que um livro.

/authors/author[count(book) > 1]/@name

1 val.

- (d) O país do autor do livro *Ensaio sobre a Cegueira*.

/authors/author[book/text() = 'Ensaio ... Cegueira']/@country

# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
26 de Janeiro de 2017

Duração: 2h / Sem Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

**Nota:** Nas perguntas de escolha múltipla, cada resposta errada desconta  $\frac{1}{3}$  da cotação mas pode escolher mais do que uma opção.

- 1 val.** 1. Para que dois *inputs* do tipo *radio* sejam considerados do mesmo grupo, é necessário que...
- Estejam em linhas consecutivas.
  - Tenham o mesmo valor no atributo *name*.
  - Tenham o mesmo valor no atributo *value*.
  - Tenham o mesmo valor no atributo *group*.
- 1 val.** 2. Que selector CSS se deve usar para seleccionar o título do primeiro artigo com classe *featured*?
- article.featured :first-child h2 { ... }.
  - article.featured ~~h2~~:first-child { ... }.
  - article/~~first-child~~ .featured h2 { ... }.
  - article.featured:first-child h2 { ... }.
- 1 val.** 3. Um ataque do tipo *Session Hijacking* consiste em...
- Alterar a cookie de um utilizador usando CSRF.
  - Forçar uma certa cookie de sessão a um utilizador.
  - Obter uma cookie de sessão enviada de forma insegura.
  - Tirar partido de cookies guardadas no browser sem serem cifradas.
- 1 val.** 4. Uma password cifrada sem *salt* mas com um algoritmo criptograficamente seguro é virtualmente impossível de descobrir?
- Só se a password for grande e complexa.
  - Nunca é possível de descobrir.
  - É sempre fácil de descobrir.
  - Só se o algoritmo usado não for conhecido.
- 1 val.** 5. Ao navegar para um site cujo URL começa por *https://* temos sempre a certeza de que...
- Não está a existir um ataque do tipo *Man in the Middle*.
  - O servidor usa a versão 2.0 do protocolo HTTP.
  - A comunicação é cifrada.
  - A versão do HTML é superior a 4.

(continua do outro lado...)

- 1 val. 6. Em PHP, a função *strpos* retorna a posição, começando em 0, da primeira ocorrência de uma *string* dentro de outra *string*; ou *false* se não a encontrar. Após o seguinte código, qual a forma correcta de verificar se nenhuma ocorrência foi encontrada?

```
$pos = strpos($search, $test);
```

- if (\$pos == false) echo "Not found";
- if (!\$pos) echo "Not found";
- if (\$pos === false) echo "Not found";
- if (\$pos != true) echo "Not found";

- 1 val. 7. Em Javascript, uma função é...

- Um protótipo.
- Uma classe.
- Um contexto.
- Um objecto.

- 1 val. 8. Em XPath, qual é o eixo por omissão?

- element
- child
- parent
- comment

# Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto  
26 de Janeiro de 2017

Duração: 2h / Com Consulta

Nome: \_\_\_\_\_

Número: \_\_\_\_\_

1. Considere o seguinte código HTML:

```
1 <div class="widget">
2   <ul id="todo">
3     <li>Buy Bread</li>
4     <li>Learn Guitar</li>
5     <li class="important">Pay Bills</li>
6     <li class="postponed">Wash Car</li>
7   </ul>
8 </div>
```

E o seguinte código CSS:

```
1 #todo li { color : magenta }          /* R1 */
2 div.widget ul#todo { color : green }    /* R2 */
3 #todo li:first-child { color : cyan }   /* R3 */
4
5 #todo li:nth-child(2) ~ li { color : red } /* R4 */
6 #todo li ~ li { color : blue }           /* R5 */
7 li + li + li { color : purple}         /* R6 */
```

1½ val.

- (a) Indique a especificidade de cada uma das regras (ex: 0,2,2,1):

R1	R2	R3	R4	R5	R6
1, 0, 1	1, 1, 2	1, 1, 1	1, 1, 2	1, 0, 2	0, 0, 3

1 val.

- (b) Considerando apenas as regras de **R1** a **R3**, indique a cor de cada um dos textos:

<b>R1</b>	<b>R3</b>	<b>R1</b>	<b>B1</b>	<b>R1</b>
Buy Bread	Learn Guitar	Pay Bills	Wash Car	
cyan	magenta	magenta	magenta	

1 val.

- (c) Considerando **todas as regras**, indique a cor de cada um dos textos:

<b>R3</b>	<b>R1</b>	<b>R5</b>	<b>B1</b>	<b>R4</b>
Buy Bread	Learn Guitar	Pay Bills	Wash Car	
cyan	blue	red	red	

*R5 R6*

2. Considere a seguinte *string*:

Washing the washing machine while watching the washing machine washing washing

Para cada uma das expressões regulares apresentadas de seguida, sublinhe qual o **primeiro match**:

1/2 val.

(a) /w.\*[a-z]/

Washing the washing machine while watching the washing machine washing washing

1/2 val.

(b) /a[^s]/

Washing the washing machine while watching the washing machine washing washing

1/2 val.

(c) /([a-z]{3}).\*?\1/

Washing the washing machine while watching the washing machine washing washing

1/2 val.

(d) /^.\*?\$/

Washing the washing machine while watching the washing machine washing washing

1/2 val.

(e) /(?!the )washing/

Washing the washing machine while watching the washing machine washing washing

1/2 val.

(f) /(?:w)(a).\*\1/

Washing the washing machine while watching the washing machine washing washing

3. Considere o seguinte excerto HTML:

```
1 <div id="photos">
2   </li>
3   <ul>
4     <li></li>
5     <li></li>
6     <li></li>
7   </ul>
8   <a href="#" class="load">Load More</a>
9 </div>
```

Considere que pode haver outros elementos *ul*, *li*, *img* e *a* no documento, e que a lista de imagens pode ser diferente. Escreva o código *jQuery* necessário para que:

1 val.

(a) Quando o utilizador *clique* numa imagem da lista, a *src* da imagem com a class *large* passe a ser a *src* dessa imagem com o prefixo *large/*.

4. Considere o seguinte documento em XML:

```
1 <authors>
2   <author country="Spain" name="Miguel de Cervantes">
3     <book year="1605" type="Novel">Don Quixote</book>
4   </author>
5   <author country="England" name="William Shakespeare">
6     <book year="1599" type="Tragedy">Hamlet</book>
7     <book year="1606" type="Tragedy">Macbeth</book>
8   </author>
9   <author country="Russia" name="Leo Tolstoy">
10    <book year="1865" type="Novel">War and Peace</book>
11  </author>
12  <author country="Portugal" name="Jose Saramago">
13    <book year="1995" type="Novel">Ensaio sobre a Cegueira</book>
14    <book year="1997" type="Novel">Todos os Nomes</book>
15  </author>
16 </authors>
```

Considerando que o nodo de contexto é a raiz do documento, escreva as expressões XPath que seleccionam os seguintes elementos:

1/2 val.

- (a) O título de todos os livros.

/authors/author/book/text()

1/2 val.

- (b) O título dos livros escritos depois de 1900.

/authors/author/book[@year > 1900]/text()

1/2 val.

- (c) Os anos em que foram escritos livros por autores Ingleses.

/authors/author[@country = 'England']/book/@year

1 val.

- (d) O nome dos autores que escreveram livros do tipo *Novel*.

/authors/author[book/@type = 'Novel']/@name

# EXAME

## HTML:

- Um id por elemento
- Ids não repetidos
- Muitos classes por elemento e podem ser repetidas
- Elementos obrigatórios: html, head, title, body
- Elementos semânticos: main, section, article, nav, aside
- Elementos não-semânticos: span, div
- Label relaciona-se com input se estiver aninhado ou for... → id
- Botões radio são agrupados pelo name

## CSS:

- seletores: id (#), class (.), elemento
- como funcionam os seletores
- calcular especificidade
- br em grid
- conjuntos complexos de regras (especificidade + herança + cascading)
- unidade relativas à fonte: rem e em

## PHP:

- inicializar arrays
- global
- tipagem coerciva e estrita
- arrays \$COOKIE e \$SESSION (guardados onde?)
- trabalhar com arrays (adicionar elementos, iterar, ...)
- retornar false ou 0; o operador == e a sua importância
- concatenação de strings é entre simples VS. duplas
- O que é que a função header realmente faz?
- logout

## Java Script:

- funções não são objetos ; function f...; f.<sup>\*</sup> = ...;
- formas de declarar funções
- passar funções como argumentos
- O que é que bind (call e apply) realmente faz?
- arrays (criar, inicializar e iterar)
- arrays não são objetos
- como funcionam Promises ; o que é que "async function" significa?
- Como funcionam objetos (protótipos, cadeia de protótipos e classes)?
- O que não são fechos?
- Como funciona o "this"?

## DOM:

- O que é o DOM?
- objetos document e window
- métodos querySelector e querySelectorAll (diferenças e o que fazem)
- o elemento <script> (defer e async)
- usar bind com eventos
- O que é que o fetch faz (comandos, protocolo, onde corre)?
- O que é que stopPropagation e preventDefault?
- O que é bubbling e capture?
- fetch VS XMLHttpRequest

Segurança: vai ser diferente

- XSS e CSRF (como prevenir e como funciona)
- como funciona HTTPS
- Sal: importância? o que priva? ↗ guardamos junto com a password?
- como armazenar passwords em PHP? ↗ h(P+S), S

## Expresões Regulares: palavras - cunhadas

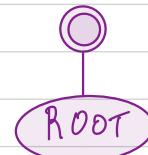
- quantificações "greedy"
- "lookaround", grupos e referências para trás
- o mesmo símbolo pode significar muitas coisas diferentes
- prática!

## XPath: exemplos

· não fácil é:

- caminhos de localização separados por /
- caminho de localização: eixo: nó de teste [predicado]
- o eixo é a direção e especifica o tipo do nó principal
- o nó de teste seleciona nós
- o predicado filtra nós
- o que é o resultado de uma expressão (uso em predicados)
- começa no nó de contexto
- existe uma "raiz virtual" antes da raiz do documento

· atalhos



**HTML:** html, head, title, body

**Semântica:** main, section, article, nav, aside  
Um id único por elemento  
(label for = "...") > ... </label> <input id = "...>  
<label> <input> </label>

**Não-Semântica:** span, div  
Uma ou mais classes por elemento  
botões radio agrupados pelo nome

**CSS:** (#id), .classe : pseudo-classe [atributo], elemento = pseudo-elemento  
Numa grid, grid é uma fração retante e podem criarse células implícitas  
rem: representa o tamanho da fonte do elemento pai da página  
em: representa o tamanho da fonte do elemento pai ou do elemento atual

**PHP:** \$a = array(...); \$a[] = ...; foreach (\$a as \$v) echo \$v;  
global refere-se a uma variável global 0 == false ≠ 0 != false  
Tipagem Lenta: converte os tipos Tipagem Rápida: força os tipos  
header: envia um header de retorno para o browser, o processamento continua  
\$COOKIE: cookies (guardadas no browser) \$SESSION: informações da sessão  
setcookie("name", "value", "time") ≠ setcookie('name', 'value', time()); session\_start(); session\_destroy()

**Java Script:** funções não objetos - f = function (...) => function f (...) => ...  
Objetos têm protótipos e podem ser fornecidos como parâmetros de funções  
**bind:** fixa o contexto e retorna uma nova função call(this, ...) apply(this, ...)  
arrays são objetos - const arr = []; for (const a of arr) console.log(a)  
**Promessa:** resultado eventual de uma operação assíncrona (satisfita/rejeitada)  
**async:** retorna uma promessa **await:** espera até um promessa concluir  
Quando uma função é criada, rateia o ambiente lexical e cria contexto  
**this** { execução global: global ou window  
função: undefined  
seta: contexto envolvente  
**apply/call:** primeiro argumento  
**metodo:** objeto  
**evento:** objeto que disparou

DOM: representação da página web como árvore de nós - querySelect([All])  
document: documento HTML window: janela do browser addEventListeners(...)  
<script>: pausa o HTML, busca o script e corre-o ↗ f.bind(e.target,...)  
defer: não pausa e só corre no final async: não pausa e corre logo  
fetch: interface para ir buscar recursos remotos que retorna uma promessa  
event.stopPropagation(): para "bubbling" event.preventDefault(): impede pré-definição  
bubbling: o evento propaga-se para os pais capturing: desce até ao elemento  
XMLHttpRequest: open(method, url, async) fetch: fetch(url, {method, headers, body})

Segurança: pssowords com password-hash() e password-verify() ↗ (P+S)  
XSS: injetar script maliciosos - persistente/refletido - valida, filtra, codifica  
CSPF: aproveitar o facto do utilizador estar logado para realizar ações - token  
Man-in-the-Middle: Autoridade Certificada - Certificados Assinados - HTTPS  
sal: sal diferente para cada palavra "lookay/rainbow tables" - PHP

Expressões Regulares: os quantificadores não "greedy"; "lookaround"; grupos; referências  
XPath: caminho/echo::nó [teste [predicado]] / direciona::seleciona [filtra] - NÓS

HTTP: protocolo de transferência de hipertexto - GET/HEAD/POST/PUT/DELETE/OPTIONS  
Pedido: método, caminho, versão; cabeçalhos; corpo - browser  
SResposta: versão, código, mensagem; cabeçalhos; corpo - servidor  
Segurança: nem efeitos colaterais no servidor - GET; HEAD (obrigatórios!)  
Idempotente: os efeitos colaterais de muitos pedidos são únicos - PUT; DELETE  
POST: pede para um recurso ser armazenado no URL - cria/substui  
OPTIONS: pede opções de comunicação para um URL ou servidor  
SOP: mecanismo de segurança que restringe a interacção de recursos entre origens  
CORS: mecanismo de headers para indicar origens que o browser deve permitir