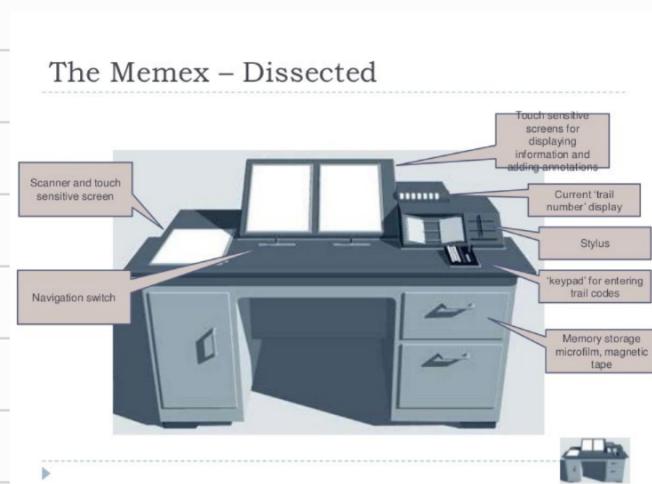


Resumo I IW

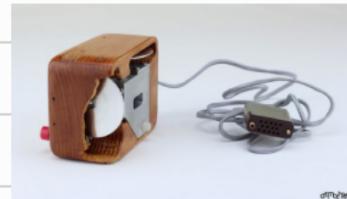
Nota: O que está sublinhado a verde é o que saiu no teste!
exemplo

História da Web

- Vannevar Bush → criou a Memex (guardar conhecimentos)



- Douglas C. Engelbart → pioneiro na interação "humano - computador"



→ 1º mouse

- Ted Nelson → "inventor" dos palavras hipertexto e hipermédia

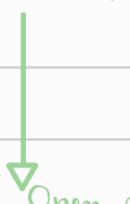
- Tim Berners-Lee → publicou o World Wide Web (www ou web)

- HTTP
- HTML
- URL

→ nascimento da web como serviço público de internet

- Marc Andreessen → apresentou o primeiro navegador gráfico (Mosaic ou Netscape)

Netscape Communication Corporation:



- SSL - Secure Sockets Layer Protocol → protocolo para transmissão segura de dados
- Javascript → linguagem de programação para web

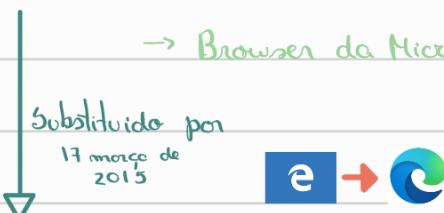
Open source

Mozilla Organization:



- Gecko engine → a partir do Netscape Open source
- Firefox

Internet Explorer:



Microsoft Edge:

- Utilizado como browser principal no Windows 10

8 de abril de 2019

↳ → Pensa a ser baseado em Chromium



Chromium:

- projeto browser de código aberto
(base do web Chrome)

Chrome:

- lançado pela Google em 2 de setembro de 2008
- baseado em Chromium, mas contém recursos proprietários, atualizações,...

7
0

Importante:

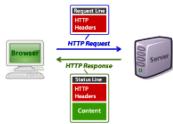
Tim Berners Lee:

- World Wide Web (www ou web)
- Uniform Resource Locator (URL)
- Hypertext Transfer Protocol (HTTP)
- Hypertext Markup Language (HTML)

HTTP - Hypertext Transfer Protocol

- Protocolo utilizado para transferir documentos de hipertexto e seus recursos (imagens, ...) entre máquinas remotas.

→ Modelo de funcionamento baseado em pedido-resposta

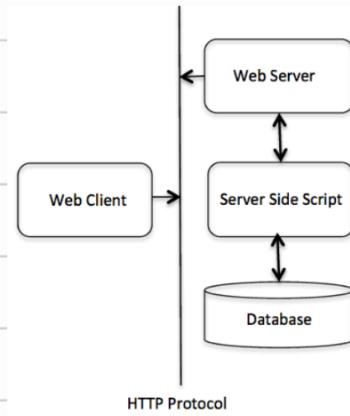


→ Cabeçalho dos mensagens é texto puro (não binário)

→ Não orientado a conexões (cada conexão é sempre uma nova conexão)

Transmissão quebrada a meio

- Protocolo http: recomeca a transmissão do inicio
- Protocolo ftp: permite retorná-la a partir do ponto em que foi interrompida.



Sintaxe:

<protocolo>://<servidor>:<porta>/<caminho>/<recurso>?<dados>

Defauts:
→ HTTP: 80
→ HTTPS: 443

→ Separa o <recurso> dos <dados>
→ Para separação entre os dados usamos &

http://www.ua.pt

Protocolo: http
Servidor: www.ua.pt
Porta: 80

https://www.ua.pt/pt/curso/383

Protocolo: https
Servidor: www.ua.pt
Porta: 443
Caminho: pt/curso/383

http://www.ua.pt/deti/PageCourse.aspx?id=383&p=4&a=9

Separador de dados
codificado como "%26"
x queremos enviar como símbolo
Protocolo: http
Servidor: www.ua.pt
Porta: 80
Caminho: deti
Recurso: PageCourse.aspx
Dados: id=383
p=4
a=9

http://localhost:6067/Aula01/index.html

Protocolo: http
Servidor: localhost
Porta: 6067
Caminho: Aula01
Recurso: index.html

Pedido e resposta a um servidor web

Cliente

GET /hello.html

GET /t.html

Servidor

200 OK

404 Not Found

HTTPS - Hypertext Transfer Protocol Secure

- O Hypertext Transfer Protocol Secure (protocolo de transferência de hipertexto seguro) é uma implementação do protocolo HTTP sobre uma camada adicional de segurança que utiliza o protocolo SSL/TLS

→ Desenvolvida pela Netscape

- Permite que os dados sejam transmitidos através de uma conexão criptográfica que garante a autenticidade do servidor e do cliente por meio de certificados digitais
- "Cria um canal seguro sobre uma rede insegura"
- Mais proteção para escutas ilegais (eavesdropping attacks) e de ataques homem-na-mídia (man-in-the-middle)
- A confiança fornecida é baseada em autoridades de certificação que vêm pré-instalados no navegador

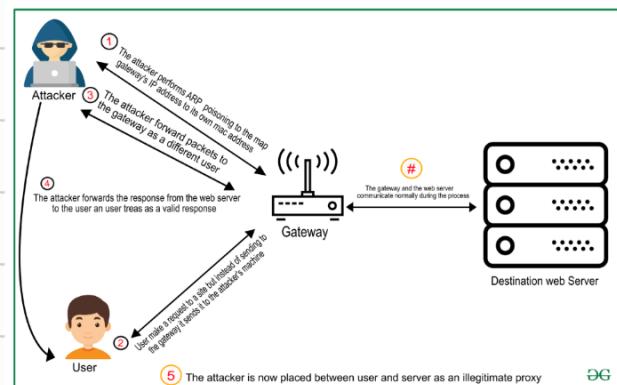
Eavesdropping attacks → escutas ilegais

- Gravar comunicações baseadas em IP
- Intercupação dos dados (quando vão para os routers/gateways)

Uma forma:

Usando a função do alto-falante ou microfone de um dispositivo

Man-in-the-middle



Princípio de funcionamento

- O SSL usa um sistema de criptografia que utiliza dois chaves para criptografar os dados
 - Pública (conhecida por todos)
 - Privada (conhecida apenas pelo destinatário)

↓
é a única e eficaz maneira de obter segurança de dados em comércio eletrônico

HTTP: Não Criptografado (sem SSL)



HTTPS: Conexão Segura e Barata com SSL

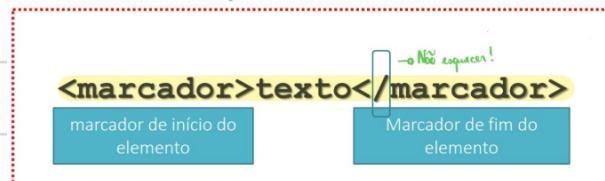


- Quando um SSL - Certificado Digital está instalado no website, um ícone de um cadeado aparece no navegador e o endereço começa em <https://> informando que os dados estão criptografados

HTML - Hypertext Markup Language (linguagem de marcação de texto)

↳ Criado para publicações e disseminação de informação científica

Elemento html: (por de marcadores)



• As etiquetas "<" e ">" são responsáveis pela formatação da linguagem

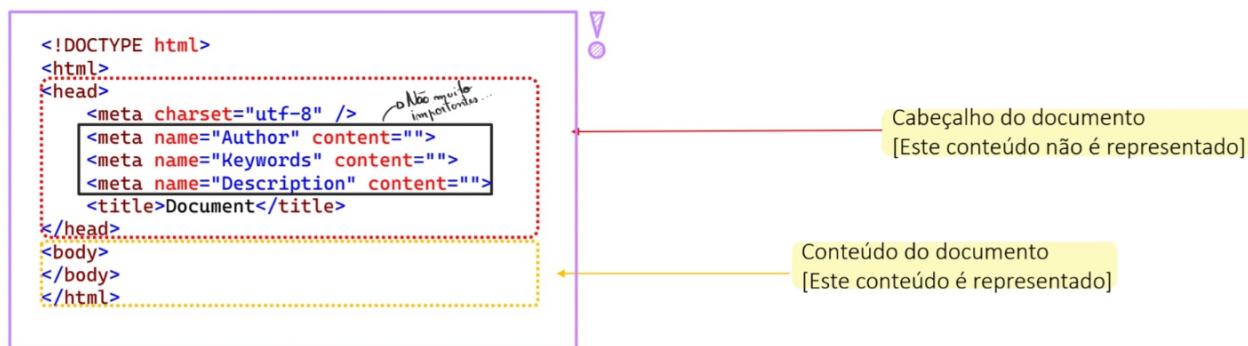
Pode conter:

- Atributos
- Valores
- Filhos
- (...)

Exemplo:

- Simples: <hr/>
(não possui filhos)
- Com atributos: UA
- Com filhos e atributos:
<p>Teste UA Teste</p>

Estrutura base



Elementos básicos:

<html> - Define o início do documento html (pode ter atributos)

<html lang="pt"> (...) </html>

<head> - Define o cabeçalho do documento (não é representado)

Dentro do head
↳ <title> - Define o título da página

<style type = "text/css"> - Para CSS

<script src = "script.js"/>

<script type = "text/javascript"> - Para JavaScript

<link rel = "stylesheet" href = "style.css"/>

<link> - Define ligações da página com outros arquivos (css, scripts, ...)

<meta charset = "utf-8"/>

<meta> - Define propriedades da página (codificação de caracteres, autor, ...)

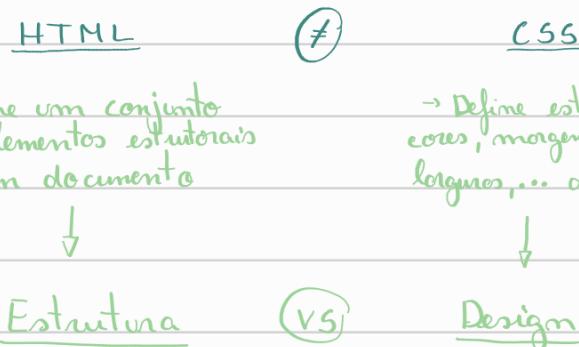
<body> - Define o conteúdo principal do documento e é a parte exibida no navegador
(pode ter atributos) <body style = "background-color: black"> (...) </body>

↓
Existe uma infinidade de elementos para o corpo [slides aulas ②, ③ e ④]

CSS - Cascading Style Sheets

(folhos de estilo encadeados)

- Os estilos CSS permitem fazer a separação entre a estrutura do documento HTML e a sua representação



- Necessidade de encontrar meios de representar a informação mais atrativa ?

↳ Foram criados novos marcadores HTML:

-
- <div>
-

↳ Criado o CSS para separar a estrutura da representação

- É possível que o mesmo documento, quando submetido a folhos de estilos diferentes, seja representado distintamente

Estrutura CSS

- Global - ficheiro externo que depois pode ser associado a vários documentos HTML
- Document - dentro de um marcador <style>...</style> localizado no <head>
- In-line - na linha de um marcador html

- A instrução que prevalece é a que está mais próxima do elemento

Global

```
estilos.css
seletor {propriedade : valor;
          propriedade : valor}

index.html
<head>
  <link rel="stylesheet" href="estilos.css"/>
</head>
```

Document

```
index.html
<head>
  <style type="text/css">
    seletor {propriedade : valor;
              propriedade : valor}
  </style>
</head>
```

In-line

```
index.html
<marcador style="propriedade : valor;
                  propriedade : valor">
  ...
</marcador>
```

- Mais sobre CSS nos slides ③ e ④

Id vs Classe

- Os Ids são únicos
⑦
- As classes não são únicas
- É possível combinar classes com Ids
 - #meu-id - elemento com o id = "meu-id"
 - .minha-classe - elemento com a classe = "minha-classe" (pode ter mais!)
 - div - todos os divs do documento
 - div p - todos os parágrafos dentro de divs
 - p #meu-id - todos os parágrafos com id = "meu-id"
 - a.minha-classe - todos os hiperligações com a classe = "minha-classe"
 - p.class1.class2 - todos os parágrafos com a classe = "class1 class2"
 - (...)

Fontes de texto

- Solução inicial: criar imagens e disponibilizar esses textos como imagens (gasto de dados maior)
- Solução ideal: enviar fontes de texto para os computadores dos utilizadores remotos
 - ↳ Importar do Google fonts é uma solução boa!

```
<head>
<link rel="stylesheet" href="..."/>
<...>
<style>
  body { font-family: 'Open Sans', sans-serif; }
</style>
</head>
```

Twitter Bootstrap

- Grid System
- Menus
- Jumbotron
- Carousel
- Modal
- Tables
- Buttons
- (...)

- Biblioteca que, de forma inteligente, permite adaptar o design às dimensões do dispositivo utilizado para a apresentação (RWD)
- Web design responsivo (RWD) é uma abordagem web que cria mudanças dinâmicas na aparência de um site, dependendo do tamanho e da orientação do dispositivo utilizado na visualização (utiliza breakpoints para determinar o layout)

baseados na largura e utilizando media queries

"Apenas uma única base de código que oferece suporte a utilizadores com diferentes dispositivos"

Framework CSS responsiva:

- biblioteca que facilita a criação para um desenvolvedor
- "coleção de folhos de estilo CSS pré-instalados e prontos para uso"

Exemplos:

Bootstrap by Twitter

Pos: Responsivo, customizável e excelente documentação
HTML, CSS e Javascript framework

Cons: Sites muito parecidos
Necessita de JavaScript

Ideal para: iniciantes

Blumna

Pos: Responsivo, muito customizado e simples

Cons: Pouco suporte e pouca documentação

Ideal para: iniciantes até profissionais

Zurb

Pos: Responsivo, customizável, moderno e bem documentado

Cons: Pesado e pouco suporte

Ideal para: profissionais

Bootstrap by Twitter (focamos neste)

Estrutura

```
<!doctype html>
<html lang="en">
  <head>
    <meta nome="viewport"/>
    <link rel="stylesheet" href="..."/>
    <script src="..."/>
```

CSS e JS

...
...
...

Ativar todos os funcionalidades da responsabilidade

O bootstrap é desenvolvido primeiro para dispositivos móveis e depois vai aumentando

Layout - Grid System (usamos o .container para largura fixa ou .container-fluid para a largura toda)

```
<div class="container">  
  <div class="row">  
    <div class="col-12 col-md-6 col-lg-4 col-xl-3">
```

linha dividida em 12 

Mais baixo de todos Médio Longo Gigante

Funciona também para outros classes:

d-none , d-md-block , d-xl-none

! Nâo aparecer em médio e longo
md, lg

Os níveis aplicam-se
a esse nível e a todos
para cima
.col-sm-4 aplica-se aos
pequenos, médios e grandes...



Cores:

Primary Secondary Success Danger Warning Info Light Dark Link

Muito mais... Slide ⑤ e o site do bootstrap

Linguagem de Programação

! HTML e CSS não são linguagens de programação

• Conceitos básicos:

→ Linguagem compilada: programas são convertidos para código máquina e é verificada a sintaxe, após disso o programa é executado muitas vezes de uma forma rápida

→ Linguagem interpretada: programa é executado e traduzido linha a linha, comando a comando, torna mais lento mas mais flexível

Linguagem interpretada: JavaScript

→ Linguagem tipada: operações são realizadas sobre estruturas de dados bem definidos e cada operação define o tipo de dados que deve receber (oposto de linguagens fracamente tipadas)

Fracamente tipadas: JavaScript

→ Linguagem "case-sensitive": faz distinção entre lettres maiúsculas e minúsculas.

Case-sensitive: JavaScript e JSON



Case-insensitive: HTML e CSS

Java Script

• Inspirada em C e parecida a Java

- Linguagem de script (interpretada) case-sensitive e orientada por objetos e que funciona em múltiplos plataformas - tanto do lado do cliente como do lado do servidor.
 - Kinectout.js
 - React.js
 - Node.js

- Permite ser aumentada com objetivos adicionais com uma variedade de propósitos
- Executado em um browser permite controlar o browser e o seu DOM
 - Ver mais anexos

Vantagens:

- Como é interpretada, basta executar o código escrito pelo programador

Desvantagens:

- Como é interpretada, muitos erros só são detectados quando o fluxo de execução atinge a linha onde o erro está presente (pode provocar paragens na execução)

Um Script de JavaScript:

- Controlar o web browser
- Realizar comunicações assíncronas
- Alterar o conteúdo do documento exibido de modo dinâmico

Incluir no HTML:

Ambos estão corretos mas no
<head> temos de ter
cuidado para carregarmos
antes de ser necessário
utilizá-lo

Fim do <head>

ou

Fim do <body>

} temos de ter
cuidado com o
DOM

<script>
(...)
</script>

(ou)

<script src="script.js" />

! Para não termos problemas com o DOM!
podemos usar window.onload = função !

JavaScript tem:

- elevada versatilidade: executada em qualquer browser de qualquer sistema operativo/dispositivo

- segurança reduzida: código é sempre enviado ao cliente na sua forma textual (rapidamente copiado)

↳ minimiza-se ("minified") o arquivo para evitar
uso e para diminuir o tamanho
Eg: bootstrap.min.js

Importante:

Operações:

+ → Concatenação e Incremento

- , *, /, % → Subtração, Multiplicação, Divisão e Resto da divisão

** → Exponenciação

++, -- → Incremento e decremento

NaN → Quando dá erro (Not a Number)

```

var a = 2
console.log(a++) // 2
? console.log(a) // 3
  
```

Comparações:

<, >, >=, <=, != → Operações normais

==, === → Converte os valores e compara, compara o valor e o tipo

Condições:

```

do {
  ...
} while (condição)
  
```

```

while (condição) {
  ...
}
  
```

```

for (var i = 0; i < 10; i++) {
  ...
}
  
```

Enquanto for verdadeira.

```

if (condição) {
  ...
} else {
  ...
}
  
```

```

switch (a) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
  
```

Existem mais... (slide ⑥)

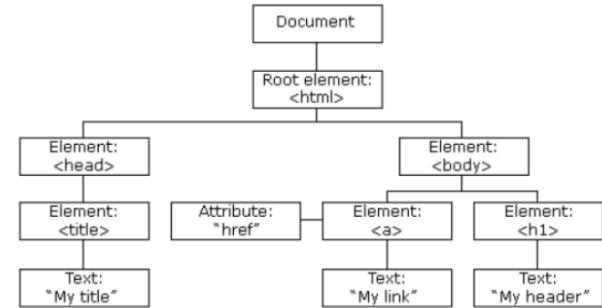
Document Object Model - DOM

- Quando o JavaScript é executado no browser \Rightarrow possibilidade de acceder aos elementos da página HTML
Aceder ao DOM a

Document Object Model (DOM)

→ Todo é feito de mós
Documento HTML \Rightarrow objeto do tipo document

→ O objeto document é o nó raiz do documento
HTML é o "dono" de todos os mós: element's,
text's, atribute's, comment's...



! É importante saber colocar o código JavaScript ou no <head> ou no <body>
conforme o DOM está organizado, para não termos problemas!
(sai sempre no teste!)

⊕ Muito mais sobre JavaScript & DOM no slide ⑥

1 Teste ↑

//

jQuery - "write less, do more"

→ JavaScript Simplificado

- jQuery é uma biblioteca JavaScript, projetada para simplificar a programação
- Fácil a navegação dos elementos de um documento
 - Selecionar elementos do DOM
 - Criar animações
 - Manipular eventos
 - Desenvolver aplicações AJAX

- Todos as ações realizadas na página web geram eventos que simbolizam essas ações
 - Movê o rato sobre um elemento / Selecionar um botão de opção / Clicar num elemento ...

Vantagens:

- Separação entre JavaScript e o HTML (puromente JavaScript)
- Elimina incompatibilidades entre navegadores (interface consistente que funciona nos diferentes navegadores)
- Muito extensível

Desvantagens:

- Biblioteca grande (~ 88kb ou ~284 kb)
- Esconde os partes complexos do JavaScript => dificuldade em aprender JavaScript
- Performance: O JavaScript puro é mais rápido a aceder ao DOM

- É um único ficheiro JavaScript <script src="...js"></script>

Sintaxe - feita a pensar na seleção de elementos HTML e na execução de algumas ações sobre os mesmos

```
$("seletor").action()
```

\$ → para aceder à biblioteca jQuery

"seletor" → para "consultar/encontrar" elementos HTML no documento

"action" → ação jQuery a ser executada no elemento "seletor"

Seletores jQuery

→ São usados para "encontrar" (ou selecionar) elementos HTML baseados no nome, id, classes, tipos, atributos, valores de atributos e muito mais

Nome do marcador	Id de um elemento	Classes
\$("#button").click(function(){});	\$("#erroMessage").show();	\$(".banner").css({"color": "red"});
\$(".p").hide();	Um id deve ser único dentro de uma página; ⇒ utilizado para encontrar um elemento único	

Outros

→ `$(*)`
 → `$(this)`
 → `$(p.intro)`
 → `$("p:first")`
 → `$("ul li:first")` primeiro do primeiro elemento
 → `$("ul li:first-child")` primeiro de todos os elementos

→ `$("[href])")`
 → `$("[target='_blank'])")`
 → `$("a[target='_blank'])")` Existem mais...
 → `$("a[target!= '_blank'])")`
 → `$(":button")`
 → `$("tr:even") ou $("tr:odd")`

Código importante

```

<select id="fruta" class="form-control">
  <option value="1"> Banana </option>
  <option value="2"> Maça </option>
  <option value="3"> Pera </option>
</select>
<script type="text/javascript">
  $(document).ready(function() {
    $("#fruta").change(function() {
      var retVal = $("#fruta option:selected").text();
      alert(retVal);
    });
  });
</script>
    
```

Pode-se utilizar o jQuery para alterar/alterar os elementos

SET

E.g.: `$("#fruta").val(3)` para dar SET
`$(".banner").css({color: "red"})`
 `.text("Banana")`
 `.html("<p>ABC</p>")`
 `.attr("href", "http://www.outro")`

Get

`.val()`
`.css("color")`
`.text()`
`.html()`
`.attr("href")`

↓ Funciona assim quando queremos mudar só 1.

`addClass()`
`removeClass()`
`toggleClass()`

Evento `$(document).ready()`

→ colocar os métodos dentro do ... evita que qualquer código jQuery seja executado antes do documento carregar completamente (ou seja, o documento "isready")

E.g.: esperar que os bibliotecas sejam totalmente carregadas

$$\boxed{\$().ready()} = \boxed{(\text{document}).ready()}$$

Conclusão: Assim, se for colocado dentro de `\$().ready()` o `<script>` tanto pode ficar no início do documento HTML como no final

JavaScript Object Notation - JSON

- Formato de troca/intercâmbio de dados simples que é independente da linguagem de programação utilizada
- Linguagem auto-descritiva \Rightarrow Fácil de entender
- Linguagem "case-sensitive"
- Usa a sintaxe JavaScript, mas JSON é somente texto!
- Pode ser utilizado (o formato de dados) por qualquer outra linguagem de programação

Sintaxe: `{"Key": "value"}` (formato correto, embora o JS aceite tudo!)

```
{"nome": "Cristiano",
  "email": "cristiano@gmail.com",
  "employers": [
    {"firstname": "João", "Salary": "100€"},
    {"firstname": "Maria", "Salary": "50€"}
  ]}
```

LD
Também pode ter números e booleanos:
`{"Key": 24}`
`{"Key": false}`

Asynchronous JavaScript and XML - AJAX

(ver dentro do jQuery)

- Permite que páginas HTML troquem dados com um servidor para atualizar apenas partes dessa página mas sem ser necessária recarregar toda a página (é feito em segundo plano) "Carregados dinamicamente"

AJAX: `$.ajax()`

```
var data = "abc";
$.ajax({
  type: "GET",
  url: "http://somewhere/somepage/somedetails",
  data: {
    "data": data
  },
  dataType: "json",
  success: function (data, textStatus, jqXHR) {
    //if received a response from the server
  },
  error: function (jqXHR, textStatus, errorThrown) {
    //if there was no response from the server
  },
  beforeSend: function (jqXHR, settings) {
    //capture the request before it was sent to server (in send calls)
  },
  complete: function (jqXHR, textStatus) {
    //this is called after the response or error functions are finished
    //so that we can take some action
  }
});
```

GET: `$.get(URL, callback)`

```
$("button").click(function () {
  $.get("getPageAddress", function (data, status) {
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

POST: `$.post(URL, data, callback)`

```
$("button").click(function () {
  $.post("postPageAddress",
  {
    name: "Biden, Joe",
    city: "Washington"
  },
  function (data, status) {
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

jQuery User Interface - jQuery UI

- Coleção de widgets de interface gráfica, efeitos visuais animados e temas implementados com jQuery, CSS's e HTML
- Alguns funcionalidades são cobertos pelo Bootstrap

Importantes

- Autocomplete
- Date picker
- draggable

Existem em jQuery e Bootstrap

jQuery UI	Bootstrap
Accordion	Accordion / Collapse
Button	Buttons
Dialog	Modal
Tabs	Tabs
Tooltips	Tooltips / Popover
Sliders	Carousel

Knockout.js → O professor adora isto => sai sempre 8º

- Biblioteca JavaScript que ajuda a criar interfaces de utilizador de exibição e edição ricos e responsivos com modelos de dados subjacentes limpos
- Atualizações dinâmicas (Eg.: fonte de dados externa é alterada)

Principais características

- Vinculações declarativas (sintaxe concisa e legível)
- Atualização automática da interface com o utilizador
- Acompanhamento de dependências (cadeias de relação entre dados do modelo)
- Templating
- Livre, código aberto
- JavaScript puro (funciona com qualquer framework web)
- Pequeno e leve
- Supporta todos os navegadores
- Totalmente documentado

Criar um viewmodel com KO : (básico !!!)

```
(script)
var viewModel = {"PersonName": "Pedro", "PersonAge": 18}
VM simples
ko.applyBindings(viewModel)
```

O meu nome é

— //

- Uma das principais características: atualiza a interface (view) do utilizador automaticamente quando o View Model muda
 - Para utilizar => declarar as propriedades do modelo como observáveis
- Os observáveis => "notificam os assinantes sobre as alterações e podem detectar dependências automaticamente"

```
var viewModel = { nome: ko.observable("Pedro"),
VM simples
                    idade: ko.observable(45) }
```

- Problema: Nem todos os browsers suportam operações de leitura (GET) e escrita (SET) de JS, logo, por questões de compatibilidade, os objetos Ko.observable são funções!

! → viewModel.personName() => Retorna "Pedro"
→ viewModel.personName("Maria") => Altera o valor do nome para "Maria"

Arrays de observáveis

- Detetar e responder a alterações numa coleção de objetos utilizamos: observableArray

```
var vm = ko.observableArray([  
  {...},  
  {...}])
```

Observáveis calculados

- Funções que dependem de um ou mais observáveis e serão atualizados automaticamente sempre que alguma dependência mude.

Outra forma melhor e mais complexa de representar um View Model:

```
↳ function ViewModel() {  
  var self = this; ← Importante  
  self.firstName = ko.observable("Pedro")  
  self.secondName = ko.observable(18)  
  self.fullname = ko.computed(function() {  
    return self.firstName + " " + self.lastName  
  });  
}  
  
ko.applyBindings(new ViewModel());
```

KO Bindings (cuidado para não confundir com os do jQuery!)

- `text()` - `data-bind="text: PersonName"`
- `html()` - `data-bind="html: details"`
- `css()` - `data-bind="css: PersonStatus"` → um estilo CSS
- `style()` - `data-bind="style: { color: 'red' }"`
- `attr()` - `data-bind="attr: { href: PersonURL, title: 'titulo' }"`
- `visible()` - `data-bind="visible: PersonAge() >= 18"` → Bom para listas e tabelas
- `foreach()` - `data-bind="foreach: PersonList"`

- if() - data-bind = "if : PersonAge() >= 18" → Coloca-se em uma div pai para condicionar os filhos
- ifnot() - data-bind = "ifnot : PersonAge() < 18"
- with() - data-bind = "with : PersonData" → Entra dentro desse Array e agora data-bind = "text : nome" seria igual a data-bind = "text : PersonData.nome"
- click() - data-bind = "click : PersonFunction"
- (...) data-bind = "value : someValue, valueUpdate : 'input'"

Em quase todos os bindings anteriores podemos aplicar condições:

data-bind = "style : { color: PersonAge() >= 18 ? 'red' : 'blue' }" →

se for verdadeiro
se for falso

data-bind = "text : PersonAge() >= 18 ? PersonName : 'menor'"

Outros bindings importantes:

- \$parent - vai para o elemento pai
- \$data - mostra toda a data de algo
- \$index - diz o index do elemento no Array

Quando chamas uma função no Kendo binding ele envia o item a que esse parâmetro estava associado... "foreach: people" → "click: remove"

Um array lista para cada pessoa tem um membro self.people

remove = function(item){
self.people.remove(item)}

(ou) Eg.: self.people(self.people().concat(item))

Application Programming interface - API

• Google API's

• Google Maps :

- Google Maps Embed API (simples)
- Google Maps JavaScript API v3 (complexa)

Embed API

<iframe src = "http://www.google.com/maps/.../ Modo?Key=Chave & Parâmetros >
Obrigatório!

Modo = place | directions | search | view

Chave = chave gratuita de acesso à API

Parâmetros = Dependem do Modo escolhido

↳ Saber todos!

Modos do Embed API

→ Tudo!

Place - Mapa de um lugar

Parâmetros:

Obrigatórios:

q - lugar de pesquisa

... place? key=... & q=Forum Aveiro

Google Maps | Place

q=Forum Aveiro, Aveiro, PT



Directions - Trajeto definido entre um conjunto de pontos

Parâmetros:

Obrigatórios:

origin - origem do trajeto

destination - destino do trajeto

Opcionais:

waypoints - pontos de passagem no trajeto

mode - modo de viagem (driving, walking, bicycling, transit, flying)

avoid - questões a evitar (tolls, highways)

units - unidade de medida (metric, imperial)

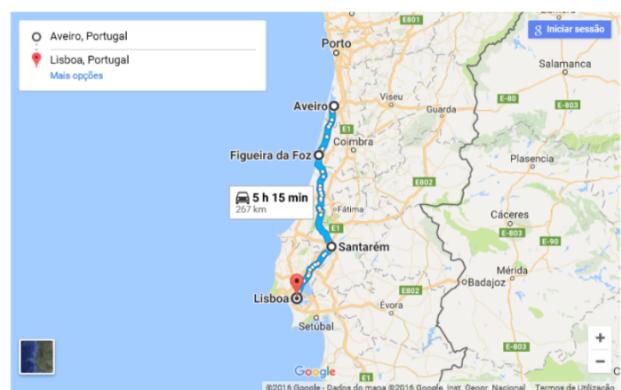
... directions? key=... & origin=Aveiro & destination=Lisboa

& waypoints=Figueira+Foz | Santarem

& avoid=tolls | highways

Google Maps | Directions

origin=Aveiro & destination=Lisbon & waypoints=Figueira+Foz | Santarem & avoid=tolls



Search - pesquisa de características num lugar (livrarias, farmácias, restaurantes)

Parâmetros:

Obrigatórios:

q - lugar de pesquisa e os
caracteristicas a pesquisar

... search? key=... & q=Aveiro+Portugal+Farmácias

Google Maps | Search

q=Portugal+Aveiro+farmácias



View - centro num par de coords.GPS

Parâmetros:

Obrigatórios:

center - coords.

Opcionais:

zoom - nível de ampliação do
mapa (0 - 21)

maptype - tipo de mapa (roadmap,
satellite)

language - linguagem a usar na interface

region - mostra os limites com
base na sensibilidade geo-
políticos

... view? key=... & center=40.63181,-8.6595

& maptype=satellite

Google Maps | View

center=40.63181,-8.6595



JavaScript API v3

Diferenças:

- Pode ser inserido em qualquer elemento HTML (não precisa de ser num "iframe")
- Grau muito mais elevado de controlo e personalização
- Exige JavaScript ⇒ Não funciona jQuery
 - ↳ Preciso instalar mais uma biblioteca...

• Open Street Maps - OSM

- Projeto colaborativo para criar um mapa livre e editável do mundo

Existem outros: NagVis Geomap

Leaflet: A biblioteca JavaScript líder para mapas interativos em computadores desktop e em dispositivos móveis

- Extensível
- Bem documentada
- Código simples e legível

• Serviços de localização atual do utilizador

→ Interface Geolocation:

- ↳ Precisa de acesso ao local do dispositivo

- Geolocation.getCurrentPosition()
- Geolocation.watchPosition()
- Geolocation.clearWatch()

• Gráficos

- Duas abordagens possíveis:

Abordagem local: criado e manipulado diretamente no computador em que está

Eg.: Chart.js, Matplotlib a ser executado o programa



Vantagens: Mais rápido e não depende da velocidade da conexão de rede. Mais seguro!

Abordagem remota: criado e manipulado em um servidor remoto, e é exibido

Eg.: Google Charts, Plotly através de uma conexão de rede



Vantagens: Pode ser acessado de qualquer lado e de qualquer dispositivo. Compartilhar facilmente, com um link.

• Chart.js - Abordagem local

- Os dados estão SEMPRE do nosso lado, utiliza uma biblioteca local

- Gráficos mais complexos e que saem do comum

• Google Charts API - Abordagem remota

- Cria remotamente um gráfico a partir de alguns dados e incorporá-lo numa página web

- Cria uma imagem .png a partir de um pedido HTTP