

Linguagem Javascript

Baseado parcialmente em conteúdos
do W3C (<http://www.w3.org/>)

Javascript : conceitos fundamentais

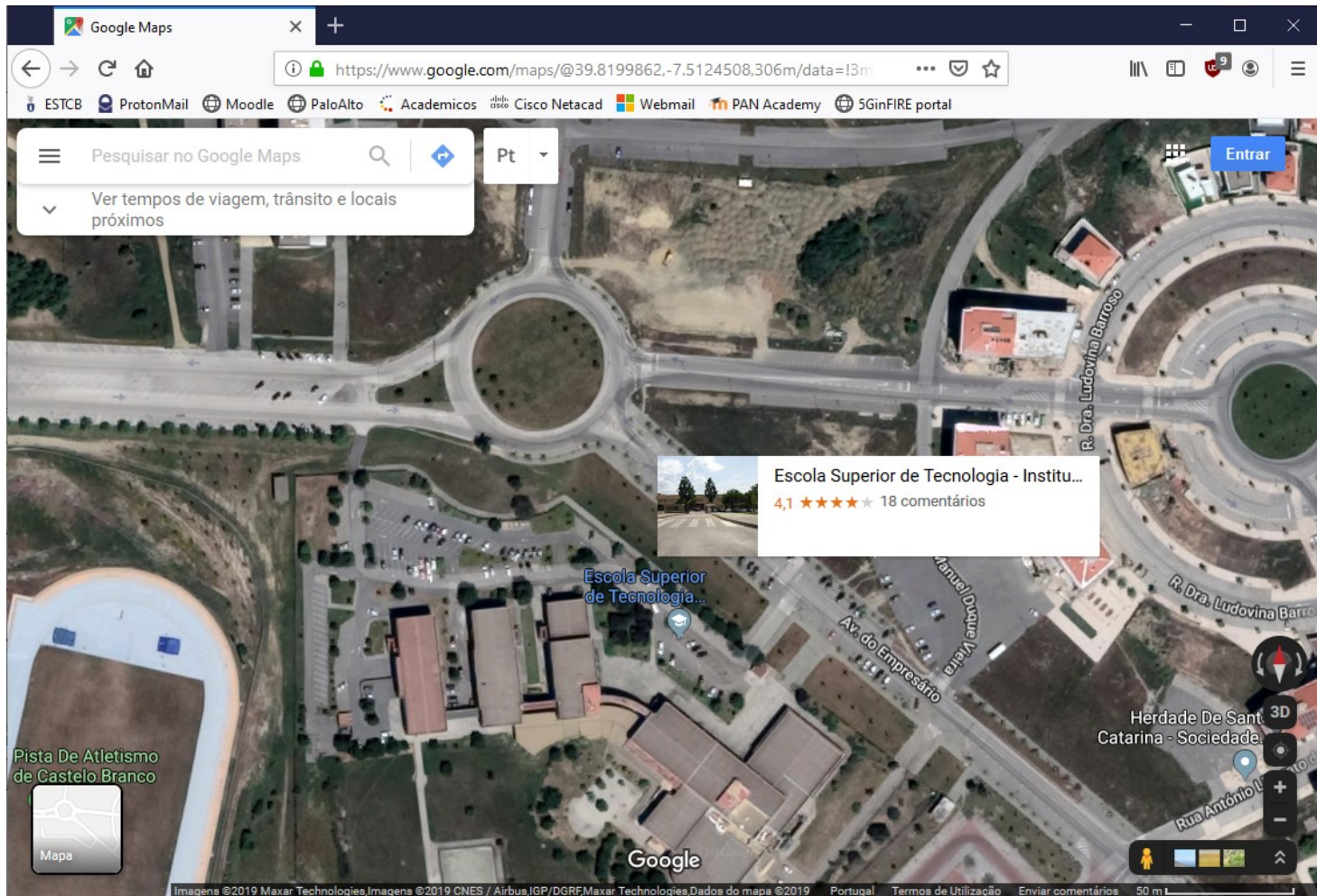
- ❑ É uma linguagem de *scripting*, que foi desenvolvida para permitir páginas HTML interactivas (Client side);
- ❑ É uma linguagem interpretada (pelos browsers);
- ❑ Não é necessária nenhuma licença para usar esta linguagem;
- ❑ O código é embutido nas próprias páginas HTML;
- ❑ É suportada pelos principais Browsers incluindo Edge, Internet Explorer, Firefox, Opera, Safari, Chrome;
- ❑ Javascript e Java são duas linguagens diferentes!
- ❑ O verdadeiro nome da linguagem Javascript é na realidade ECMAScript ;
- ❑ A linguagem está normalizada (norma ISO/IEC 16262 ou ECMA262).

Usando Javascript é possível...

- Ler e escrever no documento HTML;
 - Alterar dinamicamente o documento HTML;
 - Reagir a eventos;
 - Detetar as características do browser;
 - Validar dados em formulários;
 - Enviar e receber dados do servidor, de forma assíncrona
-
- Tudo isto localmente no browser, sem intervenção do servidor Web

Exemplo prático 1 do potencial JS

❑ Google Maps



Exemplo prático 2 do potencial JS

Office online

The screenshot shows an Excel spreadsheet in a web browser. The browser address bar shows the URL: https://ipcb-my.sharepoint.com/:x/r/personal/monicaac_ipcb_pt/_layouts/15/. The Excel ribbon is visible, showing the 'Home' tab. The spreadsheet content is as follows:

Escola Superior de Tecnologia			
Calendário de Provas de Avaliação - Época de Frequência			
1º Semestre - Ano Letivo 2018/19			
Curso Mestrado em:		Desenvolvimento de Software e Sistemas Interativos	
Coordenador de curso:		Mónica Isabel Teixeira da Costa	
Representante no CP:		Mónica Isabel Teixeira da Costa	
1º Ano			
Unidade Curricular	Data(s) Prova(s) de Avaliação (TP/T)	Data(s) Prova(s) de Avaliação (P/PL)	Data(s) Entrega Trabalho(s)/Projecto/Outro(s)
Ambientes Interativos	12 Janeiro 2019	15 dezembro 2018	12 janeiro 2019
Data Warehousing			TP1: 03 novembro 2018 TP2: 12 janeiro 2019
Sistemas eLearning		10 novembro 22 dezembro	22 dezembro 2018

Versões JavaScript

Versão	Nome oficial	Descrição
1	ECMAScript 1 (1997)	First Edition.
2	ECMAScript 2 (1998)	Editorial changes only.
3	ECMAScript 3 (1999)	Added Regular Expressions. Added try/catch.
4	ECMAScript 4	Never released.
5	ECMAScript 5 (2009)	added "strict mode". Added JSON support. Added String.trim(). Added Array.isArray(). Added Array Iteration Methods.
5.1	ECMAScript 5.1 (2011)	Editorial changes.
6	ECMAScript 2015	Added let and const. Added default parameter values. Added Array.find(). Added Array.findIndex().
7	ECMAScript 2016	Added exponential operator (**). Added Array.prototype.includes.
8	ECMAScript 2017	Added string padding. Added new Object properties. Added Async functions. Added Shared Memory.
9	ECMAScript 2018	Added rest / spread properties. Added Asynchronous iteration. Added Promise.finally(). Additions to RegExp.
10	ECMAScript 2019	Array.prototype.flat, Array.prototype.flatMap, changes to Array.sort and Object.fromEntries
11	ECMAScript 2020	BigInt

Suporte dos browsers

Browser Support for ES5 (2009)

Browser	Version	From Date
Chrome	23	Sep 2012
Firefox	21	Apr 2013
IE	9*	Mar 2011
IE / Edge	10	Sep 2012
Safari	6	Jul 2012
Opera	15	Jul 2013

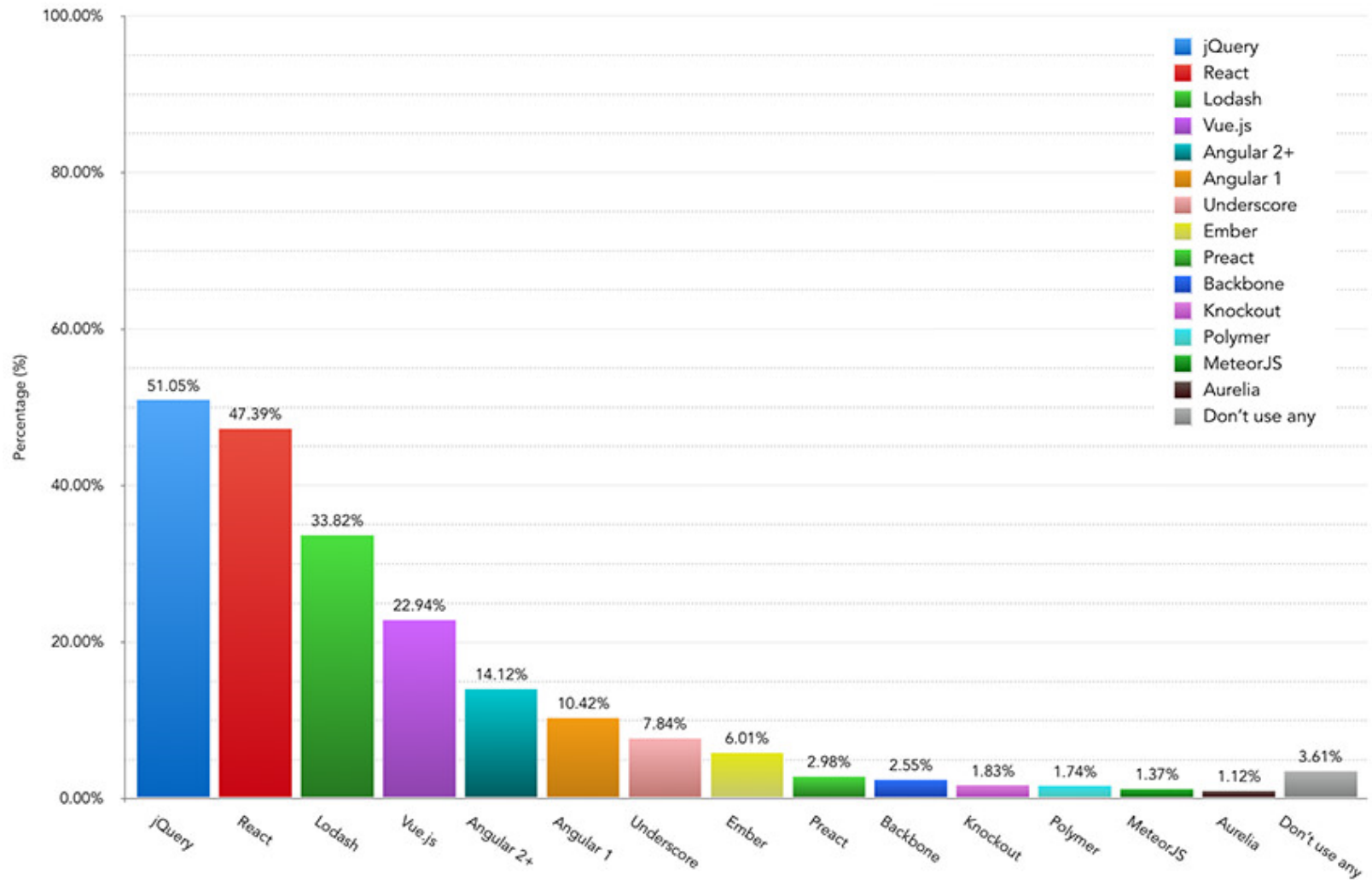
Browser Support for ES6 (ECMAScript 2015)

Browser	Version	Date
Chrome	51	May 2016
Firefox	54	Jun 2017
Edge	14	Aug 2016
Safari	10	Sep 2016
Opera	38	Jun 2016

Browser Support for ES7 (ECMAScript 2016)

Browser	Version	Date
Chrome	68	May 2018
Opera	55	Aug 2018

Bibliotecas/frameworks JS



Fonte: The State of Front-End Tooling 2018 – Results

Disponível em <https://ashleynolan.co.uk/blog/frontend-tooling-survey-2018-results>

Como embutir código Javascript

```
<!DOCTYPE html>  
<html>
```

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

```
<body>
```

```
<script type="text/javascript">  
document.write("<h1>Olá Mundo!</h1>");  
</script>
```

```
</body>
```

```
</html>
```

O código Javascript fica dentro do elemento "script"

Exemplo

Desafio aos alunos

Criar uma página HTML, com código Javascript que mostre a seguinte mensagem no écran, com o estilo heading 1 (<h1>)

"Este é o meu primeiro programa Javascript"

Como embutir código Javascript

```
<html>

<head>
<script type="text/javascript">
function message()
  {
    alert("Olá cá estou eu!");
  }
</script>
</head>

<body>
<script type="text/javascript">
document.write("Olá mundo");
message();
</script>
</body>

</html>
```

O código Javascript definido na secção <head> só é executado quando é invocado

O código Javascript definido na secção <body> é automaticamente executado quando a página é carregada

Exemplo

Desafio aos alunos

Criar uma página HTML, com uma função Javascript declarada na secção <head>, que mostre a seguinte mensagem de alerta: "Entrei!"

Como embutir código Javascript externo

```
<html>

<head>
<script type="text/javascript" src="codigo_externo.js"></script>
</head>

<body>
<script type="text/javascript">
message();
</script>
</body>

</html>
```

```
function message()
{
    alert("Olá cá estou eu!");
}
```

Ficheiro
codigo_externo.js

Exemplo

Desafio aos alunos

Criar uma página HTML, com uma função Javascript declarada num ficheiro externo chamado **funcoes.js**, que mostre a seguinte mensagem de alerta: "Este Código está num ficheiro separado"

Comentários em Javascript

```
function mensagens()  
{  
    alert("Olá cá estou eu!"); //isto é um comentário  
    //alert("Mensagem 2");  
    alert("Mensagem 3");  
    /* alert("Mensagem 4");  
    alert("Mensagem 5");  
    alert("Mensagem 6");  
    */  
    alert("Mensagem 7");  
}
```

Para comentar uma linha, usa-se //

Para comentar um bloco, usa-se /* no início e */ no fim

Exemplo

Variáveis

- ❑ Têm que começar por uma letra ou underscore (_);
- ❑ São “case sensitive”;
- ❑ Podem ser inicializadas com um valor;
- ❑ Devem ser escolhidos nomes auto explicativos;
- ❑ Para criar uma variável não inicializada, tem que ser usado **let** ou **var**;
- ❑ Para criar uma variável inicializada com um valor, não é necessário usar a diretiva **var** ou **let**
- ❑ O tipo da variável depende do seu conteúdo e não de uma declaração prévia do tipo. O tipo pode mudar ao longo do tempo (depende do conteúdo).

```
var contador;  
  
let nome_aluno="João Pedro";  
  
nome_prof="Pardal";  
  
contador=3;
```


Variáveis - usar ou não **var** ou **let** ?

- ❑ Uma variável criada com var existe apenas na função em que foi criada.
- ❑ Uma variável criada com let existe apenas no bloco onde foi criada.
- ❑ Se uma variável for inicializada sem var ou let (com um valor) :
 - ❑ O interpretador vai subindo todos os níveis de escopo até encontrar uma variável ou propriedade com o mesmo nome e vai colocar lá esse valor;
 - ❑ Se não for encontrada nenhuma variável ou propriedade com o mesmo nome, é criada uma propriedade no escopo global e é-lhe atribuído o valor;
 - ❑ !!! É preciso ter muito cuidado com variáveis definidas desta forma, pois variáveis globais podem ser alteradas sem querer !!!
- ❑ Assim, para evitar corromper variáveis que estejam num escopo de nível superior, dentro de funções devemos declarar as variáveis sempre com o var ou let

Variáveis – exemplo de escopo

Programa principal

```
var msg="Ola juventude!";  
var valor=15;  
  
mensagem();  
  
alert("msg: " + msg);  
alert("valor: " + valor);  
alert("dia: " + dia);
```

Função invocada

```
function mensagem()  
{  
    var msg="Sejam bem-vindos";  
  
    valor=1234;  
  
    dia=1;  
}
```

❑ Os valores finais das variáveis no programa principal irão ser:

- msg: "Ola juventude!"
- valor: 1234
- dia: 1

Desafio aos alunos

Implementar o código do slide anterior, experimentando definir as variáveis dentro da função mensagem com var e sem var, analisando de seguida o resultado

Variáveis – let versus var

- ❑ O escopo de uma variável criada com var é a função
- ❑ O escopo de uma variável criada com let é o bloco

```
function teste(x,y)
{
  if ( x > y ) {
    var result=x;
  }
  else {
    var result=y;
  }
  return result;
}
```

result é
aquele
declarado nos
blocos acima

```
function teste(x,y)
{
  if ( x > y ) {
    let result=x;
  }
  else {
    let result=y;
  }
  return result;
}
```

result não é conhecido
aqui – só dentro dos
blocos anteriores

Desafio aos alunos

Implementar o código do slide anterior, experimentando definir as variáveis dentro da função com var e com let, analisando de seguida o resultado

Variáveis – resumo do escopo

```
var a=3;

function teste(x)
{
  while (x < 10) {
    var b= x * x;
    let c=x + 4;
    x++;
    //aqui existem a , b, c, d, x
  }
  d=x;
  //aqui existem a , b, d, x
}

//aqui existem a , d
```

Não é erro, a
variável d existe
mesmo naquele
escopo

Constantes: escopo

- ❑ É possível definir constantes com `const`
- ❑ O escopo de uma constante é o bloco

```
function teste(x)
{
  while (x < 10) {
    const PI = 3.141592653589793;
    var b= x * PI;
    //PI é conhecido aqui
  }
  d=x;
  //PI não é conhecido aqui
}

//nem aqui
```

Funções: exemplo de função

```
function menor_numero(a, b, c)
{
  if (a <= b && a <= c)
  {
    return a;
  }
  else if (b <= c )
  {
    return b;
  }
  else
  {
    return c;
  }
}
```


Operadores matemáticos

Assumindo que **y=5**

Operador	Descrição	Exemplo	Resultado
+	adição	$x=y+2$	$x=7$
-	subtração	$x=y-2$	$x=3$
*	multiplicação	$x=y*2$	$x=10$
/	divisão	$x=y/2$	$x=2.5$
%	resto da divisão	$x=y\%2$	$x=1$
++	incrementar	$x=++y$	$x=6$
--	decrementar	$x=--y$	$x=4$

Operadores de atribuição

Assumindo que **x=10** e **y=5**

Operador	Exemplo	Equivalente a	Resultado
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
=	x=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

Operações com strings

Sempre que uma string entra numa operação,
o resultado é uma string

Exemplo	Resultado
<code>x=5+1</code>	<code>x=6</code>
<code>x="5"+"1"</code>	<code>x="51"</code>
<code>x="5"+1</code>	<code>x="51"</code>
<code>x=5+"1"</code>	<code>x="51"</code>

Operadores de comparação

Assumindo que **x=5**

Operador	Descrição	Exemplo
==	É igual a?	x==5 é verdade x=="5" é verdade
===	É exactamente igual a? (valor e tipo)	x===5 é verdade x==="5" é falso
!=	É diferente?	x!=8 é verdade
>	É maior que?	x>8 é falso
<	É menor que?	x<5 é falso
>=	É maior ou igual a?	x>=8 é falso
<=	É menor ou igual a?	x<=5 é verdade

Operadores lógicos

Assumindo que **x=6** e **y=3**

Operador	Descrição	Exemplo
&&	e	(x < 10 && y > 3) é falso
	ou	(x==6 y==5) é verdade
!	negação	!(x==y) é verdade

If...else if...else

```
function mensagem()  
{  
  var data_hora = new Date(); //objecto do tipo Date  
  var hora_do_dia = data_hora.getHours();  
  
  if (hora_do_dia < 12)  
  {  
    document.write("Bom dia!");  
  }  
  else if (hora_do_dia < 20)  
  {  
    document.write("Boa tarde!");  
  }  
  else  
  {  
    document.write("Boa noite!");  
  }  
}
```

Exemplo

switch

```
var data_hora = new Date();  
var dia_da_semana=data_hora.getDay();  
  
switch (dia_da_semana)  
{  
  case 0:  
    document.write("Domingo: que dia de preguiça!");  
    break;  
  
  case 5:  
    document.write("Já cheira a fim de semana!");  
    break;  
  
  case 6:  
    document.write("Fantástico Sábado!");  
    break;  
  
  default:  
    document.write("Nunca mais chega o fim de semana...");  
}
```

Exemplo

Desafio aos alunos

Conceber uma página HTML com Javascript, com os seguintes requisitos:

- O código Javascript (excepto a chamada de funções) deve ser localizado num ficheiro independente;
- No topo da página, deve aparecer a data de hoje, no seguinte formato: “Sexta-feira, 15 de Janeiro de 2010”;
- Se o dia da semana for Sábado ou Domingo, deve aparecer um popup (alert) no início com uma mensagem alusiva a esse facto.

Ciclo for

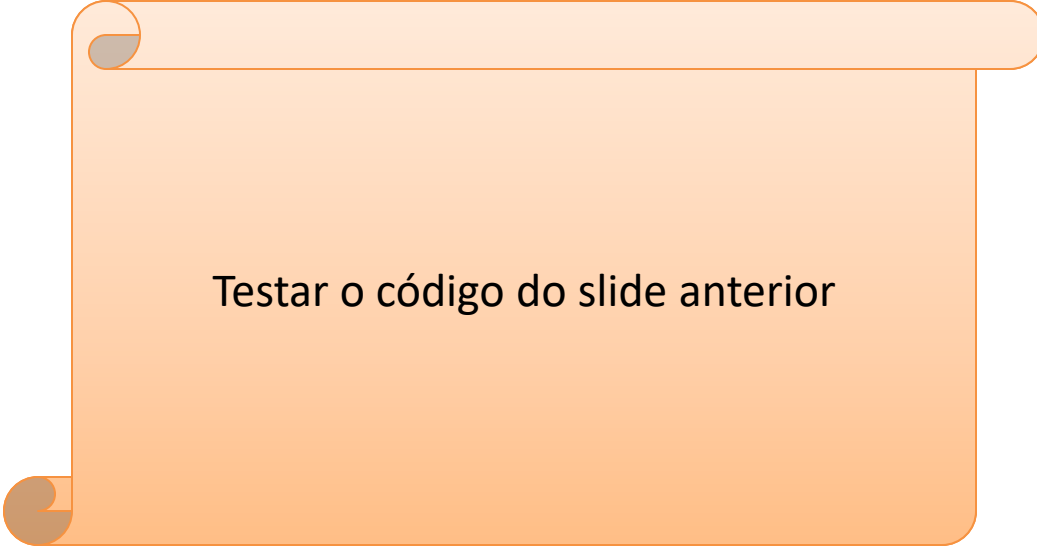
```
<html>
<body>

<script type="text/javascript">
for (i=0; i<=100; i++)
  {
    document.write(" "+i);
  }
</script>

</body>
</html>
```

Exemplo

Desafio aos alunos

A large, light orange scroll with a darker orange border and rounded corners. It has a small tab at the top left and a small circular element at the bottom left, giving it the appearance of a rolled-up document.

Testar o código do slide anterior

Ciclo while

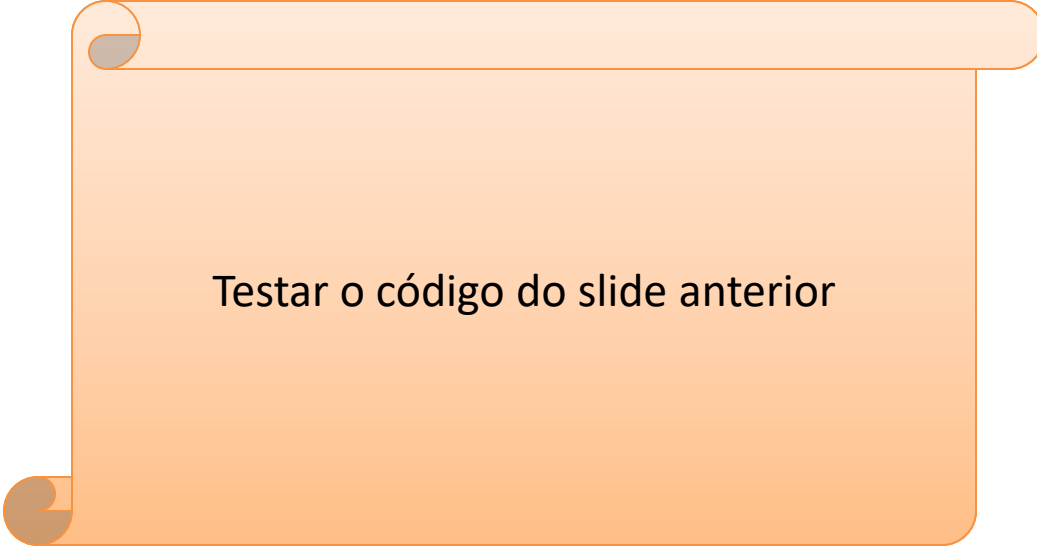
```
<html>
<body>

<script type="text/javascript">
var i=0;
while (i<=5)
{
    document.write("<h1>iteracção número " + i + "</h1>");
    i++;
}
</script>

</body>
</html>
```

Exemplo

Desafio aos alunos

A large orange scroll graphic with a light orange header and a darker orange body. The text is centered in the body.

Testar o código do slide anterior

Arrays

- Um array é usado para guardar vários valores numa única variável
- Cada valor fica numa posição numérica, que é o índice
- Os valores são acedidos através do índice
- Os índices começam em zero
- Os valores não precisam de ser todos do mesmo tipo

```
var frutos = ["Banana","Pera","Laranja"];  
var animais = [];  
var pessoa = ["Carlos", 46, 1.75];  
  
animais[0] = "Elefante";  
animais[1] = "Galinha";  
animais.push("Tubarão");  
  
document.write(animais[1]);  
document.write(frutos[0]);  
document.write(pessoa[2]);
```

Para acrescentar um elemento ao array pode usar-se o método push

Percorrer um array com **for...in**

```
var x;  
var disciplinas = [];  
  
disciplinas[0] = "Matemática";  
disciplinas[1] = "Português";  
disciplinas[2] = "Filosofia";  
  
for (x in disciplinas)  
{  
    document.write(disciplinas[x] + "<br />");  
}
```

Exemplo

Percorrer um array com **forEach**

```
function myFunction(valor) {  
  {  
    document.write(valor);  
  }  
  
var disciplinas = [];  
  
disciplinas[0] = "Matemática";  
disciplinas[1] = "Português";  
disciplinas[2] = "Filosofia";  
  
disciplinas.forEach(myFunction);
```

Desafio aos alunos

Testar o código dos dois slides anteriores

Desafio aos alunos

Conceber uma página HTML com Javascript, com os seguintes requisitos:

- O código Javascript (excepto a chamada de funções) deve ser localizado num ficheiro independente;
- Quando a página é carregada no browser, deve surgir um popup a perguntar: “Deseja calcular a tabuada de que número?”;
- Em seguida, a página deve apresentar a tabuada do número que for introduzido;

Nota: para solicitar algo ao utilizador numa janela popup, pode usar o seguinte código:

```
var numero = prompt("Por favor introduza um número");
```

Arrays: propriedades e métodos

- `length` : propriedade que contém o número de elementos do array
- `toString()` : método que converte o array para uma String com todos os elementos, separados por vírgula
- `join()` : faz o mesmo que o `toString`, mas pode-se especificar o separador
- `push()` : método que acrescenta um elemento ao array (no fim do array)
- `pop()` : método que devolve e elimina o último elemento do array
- `unshift()` : método que acrescenta um elemento no início do array e desloca todos os outros uma posição para a frente
- `shift()` : método que devolve e elimina o primeiro elemento do array e desloca todos os outros uma posição para trás
- `splice()` : método que permite acrescentar elementos em qualquer posição do array, apagando ou não os elementos existentes nessa posição
- `concat()` : método que concatena (junta) vários arrays num só
- `slice()` : método que extrai um subconjunto de elementos de uma array

Desafio aos alunos

Criar um exemplo de código para cada um dos métodos do slide anterior

Arrays: ordenar elementos

- `sort()` : ordena os elementos por ordem alfabética
- `reverse()` : inverte a ordem dos elementos
- O método `sort()` pode ser usado com uma função de comparação para ordenar um array de acordo com as nossas necessidades

```
var numeros = [400, 100, 1, 5000, 25, 10];  
  
numeros.sort(function(a, b){return a - b});
```

Desafio aos alunos

A partir destes dois arrays:

```
animais=["Elefante","Ornitorrinco","Zebra","Baleia"];  
idades=[23,120,34,7,18,45];
```

Ordene o array animais por ordem alfabética

Ordene o array animais por ordem alfabética inversa

Ordene o array idades por ordem crescente de idades

Ordene o array idades por ordem decrescente de idades

Ordene o array animais, do que tem menos letras no nome
para o que tem mais letras no nome

Objetos

- Objetos permitem guardar valores que são acedidos através de um nome, e não através de um índice como os arrays
- Os valores podem ser acrescentados em "run time"

```
var pessoa = new Object;  
pessoa.nome="Carlos Santos"  
pessoa.idade=28;  
  
alert(pessoa.nome);    //esta é uma forma de aceder a um valor  
alert(pessoa["idade"]); //esta é outra forma  
  
pessoa.nacionalidade="Cabo Verde";  
  
index="nacionalidade";  
  
alert(pessoa[index]); //podemos usar variáveis para indicar o nome  
  
var outra_pessoa = {nome:"Kevin de Bruno", idade:32, nacionalidade:"Belgica"};
```

Desafio aos alunos

Implementar o código do slide anterior,
experimentando as características dos objetos,
principalmente o acesso aos valores através dos nomes

Desafio aos alunos

Conceber uma página HTML com Javascript, com os seguintes requisitos:

- São criados 3 objetos globais, com propriedades para identificar pessoas: nome, dia de nascimento, mês de nascimento e ano de nascimento;
- Preencha essas propriedades com valores à sua escolha;
- Crie uma função que aceite como parâmetros 2 destes objetos e mostre o nome da pessoa mais velha;
- O código Javascript (excepto a chamada de funções) deve ser localizado num ficheiro independente;

Debugging

- ❑ Desde os primórdios da informática que as ferramentas de debugging são essenciais para detetar e corrigir bugs no código
- ❑ No caso do Javascript, são os próprios browsers que têm essas ferramentas, nomeadamente:
 - ❑ Firefox Web Developer
 - ❑ Chrome Programmer tools
 - ❑ Edge Developer Tools

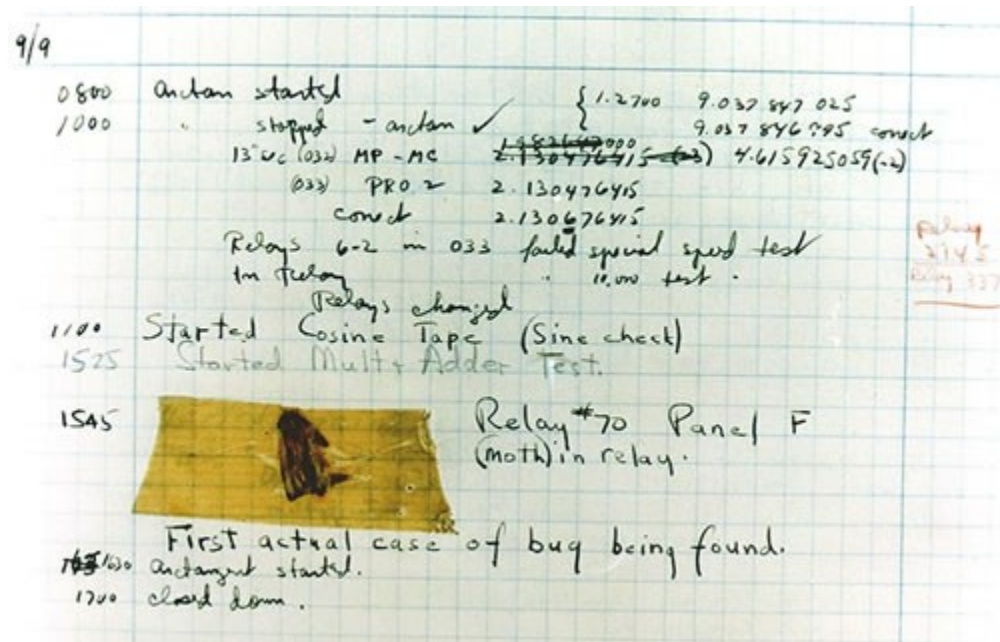
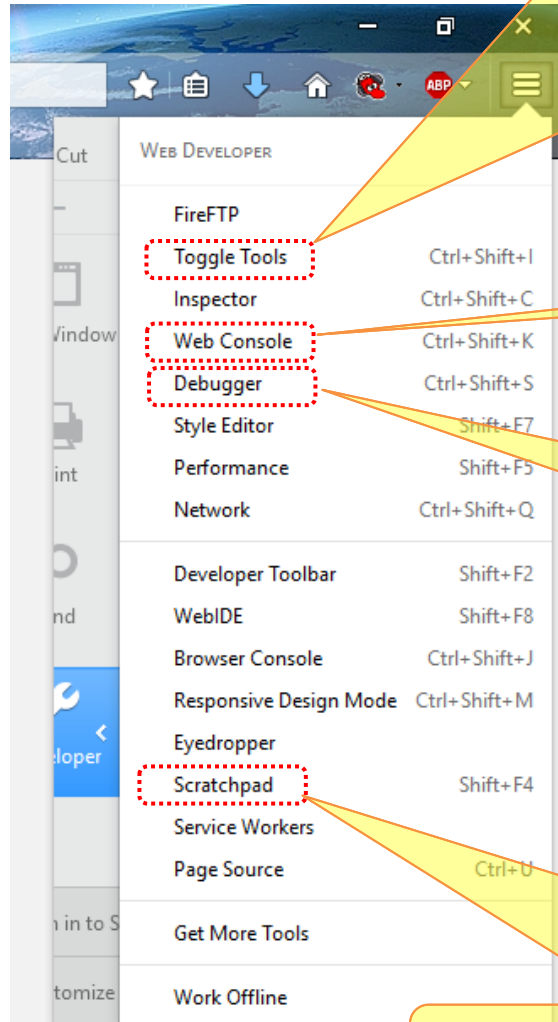


Imagem: a origem da palavra bug

Firefox Web Developer

Mostra/esconde a caixa de ferramentas (tecla F12 faz o mesmo)

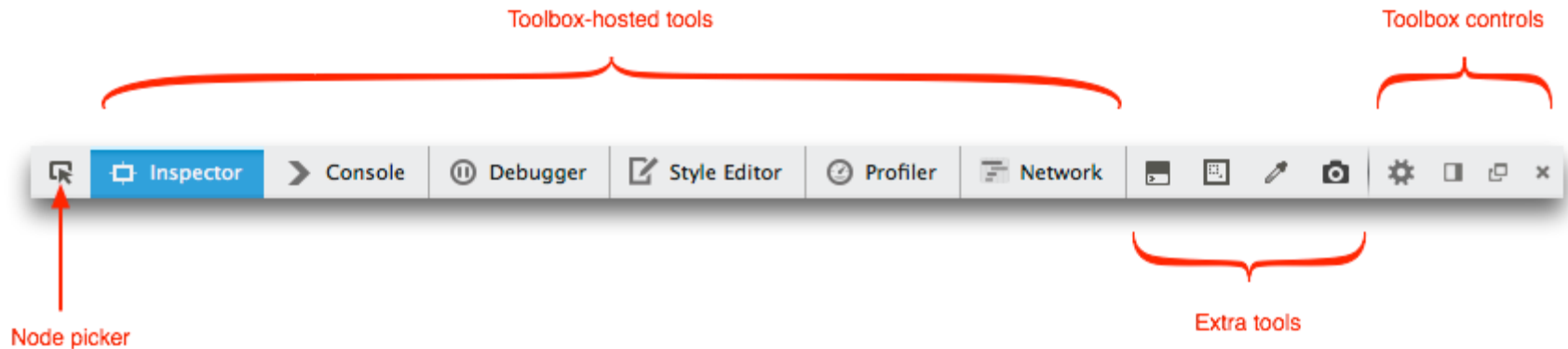


Permite ver mensagens de debugging

Permite examinar o comportamento de código Javascript, correndo esse código linha a linha

Permite experimentar código Javascript e ver o resultado na página

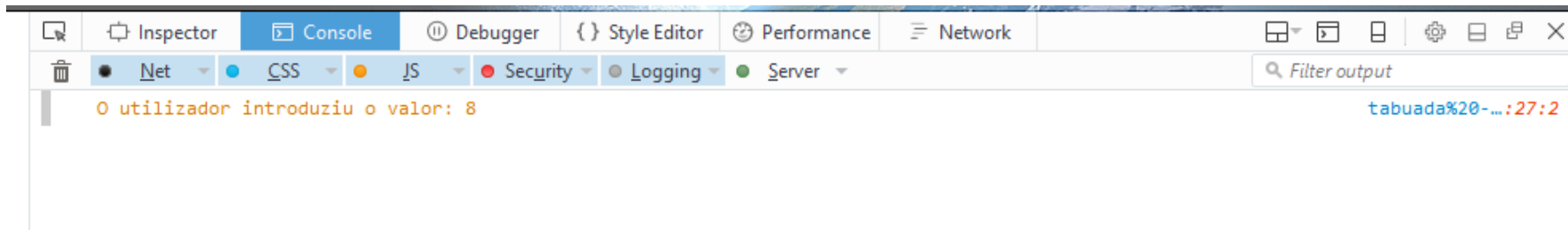
Firefox Web Developer Tools



- **Node picker:** permite selecionar um elemento da página a inspecionar;
- **Page inspector:** permite examinar e modificar o HTML e CSS da página;
- **Console:** permite visualizar mensagens de vários tipos, incluindo network requests, JavaScript, CSS, security errors/warnings bem como mensagens explícitas codificadas pelo programador;
- **Debugger:** permite correr o código javascript passo a passo e examinar e modificar as variáveis;
- **Profiler/Performance:** permite gravar e analisar dados de desempenho do browser, de modo a identificar o código que eventualmente estará a prejudicar o desempenho global da página;
- **Network:** mostra todos os “network requests” que o browser realiza (por exemplo, quando carrega uma página, ou devido a AJAX), quanto tempo demora o pedido/resposta e detalhes do pedido e resposta;

Exemplo do uso da consola

```
<script type="text/javascript">  
    var numero=prompt("Deseja calcular a tabuada do: ");  
    console.log("%s", "O utilizador introduziu o valor: " + numero);  
    calculaTabuada(numero)  
</script>
```



Exemplo de breakpoints no código

```
<script type="text/javascript">  
    var numero=prompt("Deseja calcular a tabuada do: ");  
    debugger;  
    calculaTabuada(numero)  
</script>
```

- Se o debugger estiver ligado, sempre que aparece a diretiva “debugger” no código, é realizado um breakpoint nesse ponto do código;

Exemplo do uso do debugger

1 x 7 = 7

2 x 7 = 14

3 x 7 = 21

4 x 7 = 28

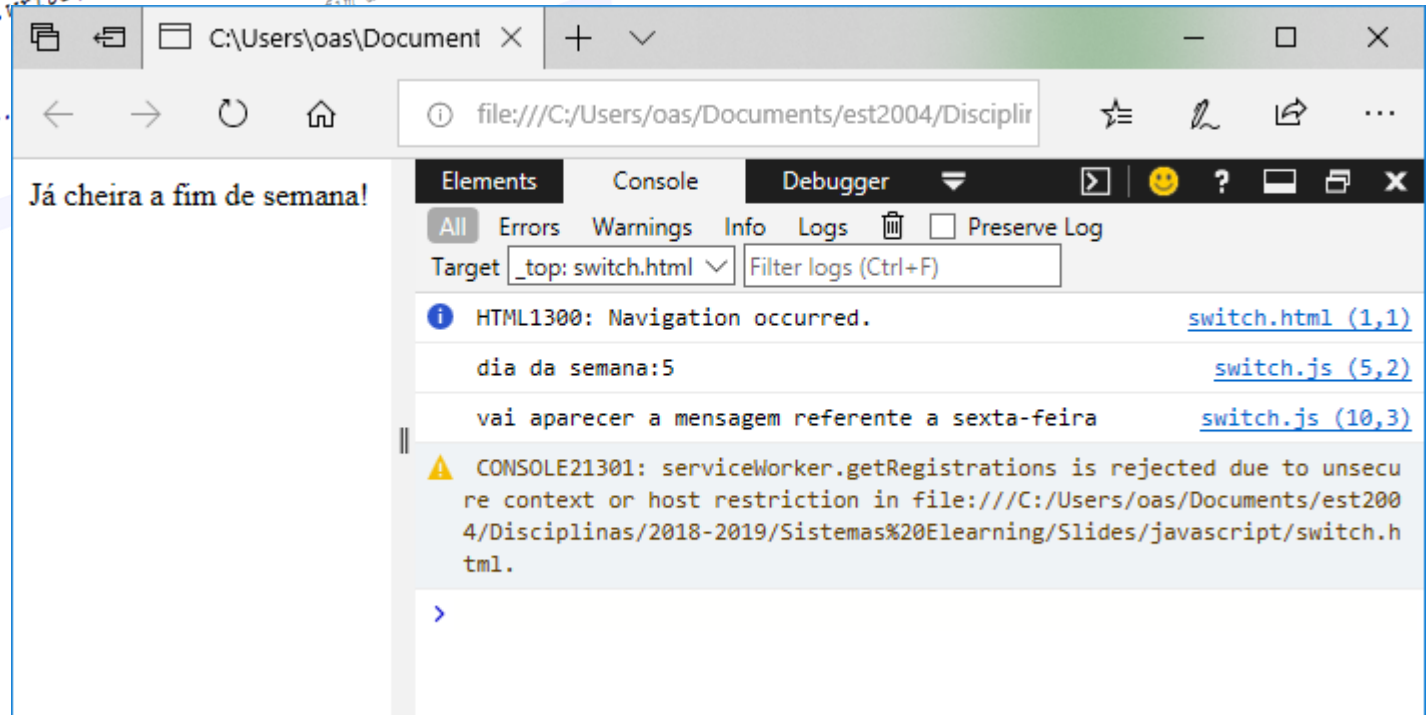
The screenshot displays a web browser's developer console with the 'Debugger' tab active. The code being debugged is a JavaScript function named `calculaTabuada` that generates a multiplication table for a given number. The function iterates from `i=1` to `i=10` and calculates the result `resultado=i*numero`. The debugger is currently paused at line 12, `resultado=i*numero;`. The 'Variables' panel on the right shows the current state of the variables: `i` is 5 and `resultado` is 35. The 'Function scope' panel shows the function's context, including `this` (Window) and `numero` (7). The 'Sources' panel on the left shows the file `tabuada - versao para debugging.html` with line 12 selected.

```
1 <html>
2
3 <head>
4   <meta charset="UTF-8">
5
6   <script type="text/javascript">
7
8     function calculaTabuada(numero) {
9       for (i=1;i<=10;i++)
10      {
11        { resultado=i*numero;
12          textResultado=i + " x " + numero + " = " + resultado;
13          document.write("<h1>" + textResultado + "</h1>");
14        }
15      }
16    }
17  }
18 </script>
19 </head>
20
21
22
23
```

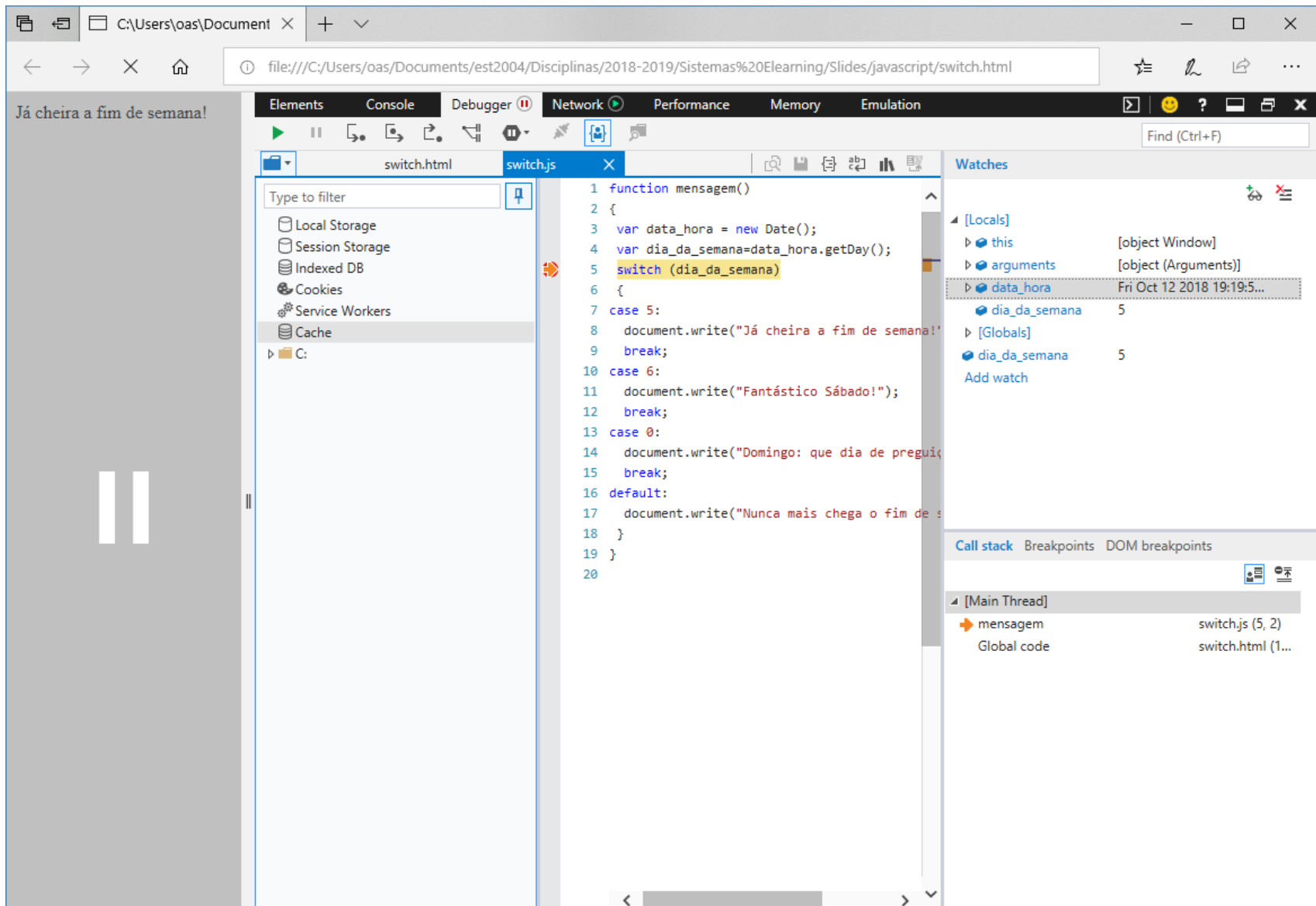
Sources	Call Stack	Variables	Events
file://		Add watch expression	
tabuada - versao para debugging.html		▼ Watch expressions	
12 resultado=i*numero;		i → 5	
		resultado → 35	
		▼ Function scope [calculaTabuada]	
		▶ this: Window → tabuada%20- %20versao%20para%20debugging.html	
		numero: "7"	
		▶ arguments: Arguments	
		▶ Block scope	
		▶ Global scope [Window]	

A consola do browser Edge

```
function mensagem()  
{  
  var data_hora = new Date();  
  var dia_da_semana=data_hora.getDay();  
  console.log("dia da semana:" + dia_da_semana);  
  
  switch (dia_da_semana)  
  {  
    case 5:  
      console.log("vai aparecer a mensagem referente a sexta-feira");  
      document.write("Já cheira a fim de semana!");  
      break;  
    case 6:  
      document.write("Fantástico Sábado!");  
      break;  
    case 0:  
      document.write("Domingo: que dia de preguiça!");  
      break;  
    default:  
      document.write("fim de semana...");  
  }  
}
```



Breakpoints no browser Edge



Exercício

Conceber uma página HTML com Javascript, com os seguintes requisitos:

- O código Javascript (excepto a chamada de funções) deve ser localizado num ficheiro independente;
- Quando a página é carregada no browser, deve surgir um popup a perguntar: “Deseja calcular os números primos até que número?”;
- Em seguida, a página deve apresentar a lista de números primos até ao número que for introduzido;

Nota: um número primo é um número que só é divisível (divisão inteira) por ele próprio e pela unidade;

Desafio aos alunos

Conceber uma página HTML com Javascript, com os seguintes requisitos:

- O código Javascript (excepto a chamada de funções) deve ser localizado num ficheiro independente;
- Quando a página é carregada no browser, deve surgir um popup a perguntar: “Deseja calcular os números primos até que número?”;
- Em seguida, a página deve apresentar a lista de números primos até ao número que for introduzido;

Nota: um número primo é um número superior a um, que só é divisível (divisão inteira) por ele próprio e pela unidade;

São números primos: 2, 3, 5, 7, 11, 13, 17, 19, 23,

Eventos Javascript

Evento	Descrição
onLoad	Invocado quando o objecto é carregado
onUnload	Invocado quando o objecto é eliminado
onFocus	Invocado quando o objecto ganha o focus
onBlur	Invocado quando o objecto perde o focus
onChange	Invocado quando o elemento perde o focus, tendo sido alterado
onSubmit	Invocado quando o utilizador submete um objecto, normalmente um formulário
onMouseOver	Invocado quando o cursor passa em cima do objecto
onMouseOut	Invocado quando o cursor sai de cima do objecto
onClick	Invocado quando há um clique do rato em cima do objecto
onKeyDown	Invocado quando uma tecla é premida para baixo
onKeyUp	Invocado quando uma tecla é largada
onKeyPress	Invocado quando uma tecla é premida e depois largada

Exemplo de eventos Javascript

```
<html>
<body onUnload="alert('Então, adeusinho!');">

<h1 onClick="alert('Olá');">Teste</h1>
<h2 onMouseOver="alert('Olé');">Outro teste</h2>

Last name: <input type="text" value="Mouse" onChange="alert('Mudou!');" />

</body>
</html>
```

Acesso aos elementos DOM HTML

```
<html><head><script type="text/javascript">

function teclaPremida()
{
  var caixaTexto=document.getElementById("benfica");
  var n=caixaTexto.value.length;

  var paragrafos=document.getElementsByName("paragrafo");
  paragrafos[1].innerHTML="escreveu " + n + " letras";

  var paragrafos_h2=document.getElementsByTagName("h2");
  paragrafos_h2[1].innerHTML="escreveu a palavra " + caixaTexto.value;
}

</script></head><body>

Nome: <input id="benfica" type="text" onKeyUp="teclaPremida();" />
<h1 name="paragrafo">linha 1</h1>
<b2 name="paragrafo">linha 2</h2>
<b2 name="paragrafo">linha 3</h2>

</body></html>
```

Exercício

Conceber uma página HTML com Javascript, com os seguintes requisitos:

- O código Javascript (excepto a chamada de funções) deve ser localizado num ficheiro independente;
- A página deve conter a mensagem : “carregue no botão para ganhar o Euromilhões”. Deve aparecer um botão “Ok” por baixo da mensagem;
- Quando o cursor do rato passar por cima do botão, este deve mudar de sítio.

Dica: para mudar a posição de um botão, pode ser usado o seguinte código:

```
var d = document.getElementById('btn');  
  
d.style.position = "absolute";  
d.style.left = Math.random()*300 + "px";  
d.style.top = Math.random()*200 + "px";
```

Exercício

Conceber uma página HTML com Javascript, com os seguintes requisitos:

- O código Javascript (excepto a chamada de funções) deve ser localizado num ficheiro independente;
- A página deve conter um formulário de registo, para introduzir os seguintes campos:
 - Nome;
 - Telefone;
 - Senha;
 - Senha (repetida);
- Quando o utilizador passa com o cursor do rato sobre o texto (label) do campo, deve abrir-se um popup a explicar o que deve ser introduzido nesse campo;
- Os campos devem ser validados em tempo real (o fundo dos campos deve ser vermelho/verde de acordo com os requisitos):
 - O nome deve conter pelo menos 10 letras;
 - O telefone deve conter exactamente 9 dígitos numéricos;
 - As duas senhas devem ser iguais devem ter 8 dígitos no mínimo;

Processamento de exceções

```
<html>
<head>
<script type="text/javascript">
function message()
{
  try
  {
    alertaaaaaaaaaa("Bom dia!");
  }
  catch(err)
  {
    alert('Ocorreu um erro grave!');
  }
}
</script>
</head>

<body>
<input type="button" value="Ver mensagem" onclick="message()" />
</body>
</html>
```

Exemplo

Objectos em Javascript

- ❑ Javascript é uma linguagem orientada a objectos, que suporta classes, objectos, métodos e propriedades;
- ❑ Possui vários objectos predefinidos;
- ❑ Permite a criação de novas classes e objectos;

Objecto “document”

Propriedade “backgroundColor”

```
<script type="text/javascript">  
document.body.style.backgroundColor="#AAAAFF";  
document.write("<h1>Olá mundo");  
</script>
```

Método “write”

Algumas classes predefinidas

Classe	Descrição
String	Usada para manipular texto
Date	Usada para manipular Datas e horas
Array	Usada para guardar várias instâncias de dados numa única variável
Math	Usada para aceder a funções matemáticas
Boolean	Usada para guardar valores booleanos
RegExp	Usada para guardar padrões de caracteres (expressões regulares)

Para obter detalhes (métodos, propriedades e exemplo) sobre estas e outras classes, consultar <http://www.w3schools.com/jsref/default.asp>

Alguns objectos predefinidos

Objecto	Descrição
document	Permite o acesso ao documento HTML e a todos os seus objectos filhos. Ver HTML DOM para mais pormenores
Navigator	Contém informações sobre o browser
Window	Permite o acesso a uma janela aberta no browser
Screen	Contém informação sobre o écran do utilizador
History	Contém informação sobre as páginas visitadas e permite a navegação nessas páginas (back, forward, ...)
Location	Contém informação sobre o URL da página e permite carregar uma nova página

Para obter detalhes (métodos, propriedades e exemplo) sobre estes e outros objectos, consultar <http://www.w3schools.com/jsref/default.asp>

Exercício

Conceber uma página HTML com Javascript, com os seguintes requisitos:

- O código Javascript (excepto a chamada de funções) deve ser localizado num ficheiro independente;
- A página deve conter o seguinte texto: “templates: A , B, C”;
- A página deve apresentar um parágrafo com o nome do Browser Web que está a ser utilizado (obtido de forma dinâmica);
- Quando o utilizador passa com o cursor do rato sobre cada uma das letras (A, B ou C), o layout da página deve ser mudado de acordo com estes requisitos:

	A	B	C
Cor de fundo:	Azul	Verde	Vermelho
Título da página:	Viva o Porto!	Viva o Sporting!	Viva o Benfica!
Imagem de fundo:	Plantel do Porto	Plantel do Sporting	Plantel do Benfica

- Dica: `document.body.style.backgroundImage = "url('equipa.png')";`

Javascript orientado a objetos: opção 1

- É possível criar uma “classe” através de constructor functions
- A classe pode ter “propriedades” e “métodos”
- Normalmente o nome da “classe” começa com uma letra maiúscula
- É boa prática definir cada “classe” no seu próprio ficheiro js

```
function Cliente(nome) {  
  this.nome = nome;  
  
  this.getNome = function() {  
    return this.nome;  
  };  
  
  this.setNIF = function(nr_fiscal) {  
    this.NIF=nr_fiscal;  
  };  
  
  this.getNIF = function() {  
    return this.NIF;  
  };  
  
}
```

```
var cliente1= new Cliente('Paulo');  
var cliente2= new Cliente('Maria');  
var cliente3= new Cliente('Ana');  
  
cliente2.setNIF(173427345);  
  
console.log('Nome:' + cliente2['nome']);  
  
console.log('NIF:' + cliente2.getNIF());
```

Javascript orientado a objetos: opção 2

- A partir da Ecma Script versão 6 há suporte integral a classes e objetos
- A classe pode ter “propriedades” e “métodos” e construtor
- Esta é a forma preferível de usar classes

```
class Cliente {  
  constructor(clientName) {  
    this.nome = clientName;  
  }  
  
  getNome() {  
    return this.nome;  
  }  
  
  setNIF(nr_fiscal) {  
    this.NIF=nr_fiscal;  
  }  
  
  getNIF() {  
    return this.NIF;  
  }  
}
```

```
var cliente1= new Cliente('Paulo');  
var cliente2= new Cliente('Maria');  
var cliente3= new Cliente('Ana');  
  
cliente2.setNIF(173427345);  
  
console.log('Nome:' + cliente2['nome']);  
  
console.log('NIF:' + cliente2.getNIF());
```

Javascript orientado a objetos: herança

- A herança é suportada através de extends

```
class Pessoa {  
  
    constructor(nome) {  
        this.nome = nome;  
    }  
  
    getNome() {  
        return this.nome;  
    }  
  
}
```

```
class Cliente extends Pessoa {  
  
    constructor(nome,NIF) {  
        super(nome);  
        this.NIF=nr_fiscal;  
    }  
  
    setNIF(nr_fiscal) {  
        this.NIF=nr_fiscal;  
    }  
  
    getNIF() {  
        return this.NIF;  
    }  
  
}
```

Exercício

- Crie uma classe chamada “Aluno” num ficheiro separado
- A classe deve ter as seguintes propriedades:
 - Nome completo;
 - Data de nascimento;
 - NIF;
- O nome completo deve ser introduzido no construtor
- As restantes propriedades devem ser acedidas através de get / set
- Deve ser criado um método chamado getIdade, que calcule a idade (em anos) a partir da data atual e da data de nascimento
- No programa principal devem criar vários (pelo menos 3) objetos desta classe e testar os vários métodos da classe