DESENVOLVIMENTO DE APLICAÇÕES COMPUTACIONAIS

Documento de Apoio ao Guião W3: Introdução à programação JavaScript

Edição 2024/25

Objetivos

- Introdução à programação JavaScript;

JavaScript

O JavaScript é uma linguagem de *scripting* interpretada pelos navegadores web (*browsers*) e, por isso é normalmente referida com sendo a tecnologia de *scripting* do lado do cliente das aplicações web. Trata-se de uma tecnologia orientada aos objetos cujo funcionamento assenta na manipulação de objetos próprios ou definidos em entidades relacionadas, como são disso exemplos o modelo de objetos dos browsers (BOM – Browser Object Model) e o modelo de objetos dos documentos (DOM – Document Object Model).

O código JavaScript pode ser executado no momento em que a página HTML é carregada pelo browser, ou quando se verifica uma determinada condição ou evento (por exemplo, quando o rato passa em cima de uma zona da página, o utilizador introduz dados num formulário, etc). No primeiro caso, o código deve ser colocado no corpo principal da página (<body>), enquanto que no segundo caso, será invocada uma função definida, normalmente, no cabeçalho da página (<head>).

index1.html

```
<html>
<head>
 <script type="text/JavaScript">
  function displayDate()
  {
   document.getElementById("demo").innerHTML=Date();
  }
 </script>
</head>
<body>
<h1>Função JavaScript definida no HEAD da página/h1>
<p id="demo"></p>
<button type="button" onclick="displayDate()">Display Date</button>
</body>
</html>
```

index2.html

```
<html>
<head>
</head>
<body>

  <p id=”demo”>Texto HTML:</p>

  <script type=”text/JavaScript”>
     document.write(“Hello World!”);
  </script>
</body>
</html>
```

index3.html

```
<html>
<head>
<script src="myscript.js">
</script>
</head>
<body>
<input type="button" onclick="popup()"
value="Click Me!">
</body>
</html>
```

myscript.js

```
function popup() {
alert("Hello World")
}
```

De forma a simplificar as páginas HTML, o código JavaScript pode ser alternativamente guardado externamente num ficheiro com a extensão ***.js***.

Teste as páginas *index1.html, index2.html* e a página *index3.html* construindo o ficheiro associado *myscript.js* com o código que se ilustra na figura acima.

Dentro de uma página HTML, o código JavaScript encontra-se entre as etiquetas <script> e </script> que permitem definir também o tipo de linguagem de extensão. Por se tratar da linguagem de script definida por omissão nos browsers, a definição do tipo com "`type="text/JavaScript"`, não é obrigatória. As linhas entre estas etiquetas são as interpretadas e executadas pelo navegador. Por exemplo, a linha com a instrução JavaScript "`document.write("Hello World!");`", utiliza o método `write` do objeto `document`, para ordenar ao navegador que escreva a mensagem `Hello World!` na página HTML.

A linguagem JavaScript pode ser usada em inúmeras aplicações e permite adicionar às páginas Web funcionalidades tão variadas como a validação dos dados introduzidos por um utilizador num formulário, acrescentar, remover ou alterar elementos (nós) HTML, ou incluir mapas da Google em páginas HTML[1] utilizando a sua API JavaScript. Por se tratar de uma linguagem interpretada pelo browser, poderá haver diferenças na forma como sistemas distintos reagem ao mesmo programa (script) JavaScript.

A linguagem JavaScript é uma linguagem de extensão ("scripting") e interpretada, que distingue entre maiúsculas e minúsculas, isto é, é "*case-sensitive*".

Sintaxe JavaScript

Variáveis

Como em qualquer linguagem de programação, as variáveis em JavaScript podem conter diferentes tipos de dados (ex. numérico, booleano, string). Nesta linguagem o tipo de uma variável é automaticamente definido pelo interpretador do *browser,* com a atribuição do valor, não sendo definido de forma explicita com base numa palavra chave. Isto significa também que ao longo de um programa a mesma variável pode ser utilizada para guardar diferentes tipos de dados.

As variáveis podem ser declaradas explicitamente utilizando a palavra chave `let` (ex. `let i=12;`) ou implicitamente (usando `i=12;`). Contudo o âmbito da variável será diferente nos dois casos. Uma variável tem âmbito global se for declarada fora de uma função ou se for definida implicitamente, mesmo que dentro de uma função; uma variável é local se declarada explicitamente dentro de funções e apenas pode ser acedida dentro desta.

```
let inteiro = 7;
let booleano = true;
let str1 =  "DEAPC";
```

[1] http://code.google.com/intl/pt-PT/apis/maps/documentation/javascript/tutorial.html

```
str2 = "2099";
let ano = parseInt(str2);
```

Diversas operações aritméticas e lógicas podem ser executadas com as variáveis. A tabela seguinte apresenta os principais operadores. A sintaxe destes operadores é semelhante à utilizada na linguagem de programação C.

Tipo	Operadores							Exemplo	Nota
Aritmética	+	-	*	/	%	++	--	X=y+2	
Atribuição	+=	-=	*=	/=	%=	=		x+=y	
Comparação	<	>	<=	>=	==	!=	===	x===5	=== (os operandos são exatamente iguais em valor e tipo)
Lógicos	&&	||	!					x< 10 && y>5	

Em JavaScript o operador '+' também se aplica ao contexto das variáveis do tipo *string* permitindo fazer a concatenação de *strings*. O contexto em que este operador é utilizado vai determinar o resultado, ou seja, se utilizado com operandos numéricos vai resultar também num valor numérico enquanto que se for utilizado com *strings* resultará numa nova *string*.

A utilização do operador '+' entre uma *string* e uma variável do tipo numérico irá resultar numa nova *string*.

```
<html>
<body>
<script type="text/JavaScript">

let str1 =  "DEAPC";
let str2 = "2099/";
let str3 = str1 + " " + str2 + 2100;

document.write(str3);
document.write("<br />");

</script>
</body>
</html>
```

Controlo de execução

O controlo do fluxo de execução determina quais e como as instruções de um programa são executadas. Este controlo é efetuado através da utilização de construções condicionais e de ciclos iterativos. A sintaxe das duas principais expressões condicionais (`if..else if.. else` / `switch`) é apresentada na seguinte tabela.

Nome	Modo de funcionamento	Sintaxe

`if..else if..`	O código de um bloco é apenas executado se a condição for verdadeira	if (condição 1) {
		// código 1}
		else if (condição 2){
		// código 2 }
		else {

		// código }

switch	O código de um dos blocos é executado se se verificar a condição. Para evitar que o código referente à condição seguinte seja executado, utiliza-se, tal como na Linguagem C, a primitiva `break`)	swicth (n) {
		case 1: // código 1
		break;
		case 2: // código 2
		break;
		default: // código
		}

Os ciclos permitem que um bloco de código seja executado um determinado número de vezes ou enquanto uma condição se verificar. A sintaxe dos três principais ciclos (`for` / `while / do… while)` é apresentada na seguinte tabela

Nome	Modo de funcionamento	Sintaxe

for	Um bloco de código é executado um número específico de vezes	for(var=valor_inicial; var<valor_final; modificação de var)
		{
		// código
		}

while	Um bloco de código é executado enquanto a condição for verdadeira	while (condição) {
		// código
		}

do…while		do{
		// código
		} while (condição);

As primitivas `break` e `continue` permitem interromper a normal execução de um ciclo ou expressão condicional. O primeiro termina a execução do bloco de código e continua a execução após este. Por outro lado, a primitiva `continue` termina a execução da execução da iteração atual e continua no próximo valor.

Funções

As funções são blocos de código que executam uma tarefa específica. Em JavaScript estas são invocadas por um evento ou por uma chamada direta da função. As funções podem ser definidas internamente em qualquer parte do documento (usualmente são agregadas na secção HTML `<head>`) ou externamente num ficheiro com extensão .js.

A sintaxe de uma função compreende a utilização da palavra chave `function`, a definição de um nome, parâmetros de entrada e valor de retorno como se mostra de seguida:

```
function nome (argumento_1, argumento_2, …)
{
// código;
       return var;
}
```

Em JavaScript os parâmetros são passados com um vector. Cada função tem duas propriedades que podem ser utilizadas para determinar informações sobre os parâmetros:

- `nome.arguments` – vector contendo os parâmetros que foram passados
- `nome.arguments.length` – devolve o número de parâmetros

De seguida é dado um exemplo da utilização de funções e ciclos nesta linguagem tendo como objetivo fazer a soma de dois números:

```
<html>
<head>
<script>
  function soma(a,b)
  {
    let args = soma.arguments;
    let comp = args.length;

    document.write("<p> Comprimento dos argumentos = " + comp + "</p>");

    for(i=0;i<comp;i++){
        document.write("<p> argumento[" + i +"]= " + args[i] + "</p>");
    }
    return a+b;
  }
</script>
</head>
<body>
<h1>Programa que efetua a soma de dois números</h1>

<script type="text/JavaScript">
document.write("Resultado = " + soma(4,3));
</script>
</body>
</html>
```

Eventos

O Javacript permite detetar determinados eventos resultantes de ações realizadas pelo utilizador de uma dada página Web, e utilizá-los para despoletar uma dada função JavaScript. Alguns dos eventos definidos no JavaScript são os seguintes:

onclick	A função será iniciada quando o utilizador carregar num botão do rato.
onmouseover	A função será iniciada quando o rato se encontrar sobre um determinado objeto (texto, imagem, …)
onchange	A função será iniciada após deteção de alteração do valor predefinido de um campo.
onsubmit	A função será iniciada quando os dados de um FORM forem submetidos. Pode ser usada para ativar a função de validação dos valores submetidos

Experimente alguns destes eventos com a seguinte página

index.html

```
<html>
<head>
<script src="myjs.js">
</script>
</head>
<body>
<input type="button" onclick="popup()" value="Click Me!">
<a href="http://www.isep.ipp.pt" onmouseover="onMoOv()"> ISEP website </a>
<input type="text" value="chave" onchange="valida()">
</body>
</html>
```

myjs.js

```
function popup() {
alert("Hello World")
}
function onMoOv() {
alert("O rato está lá ...")
}
function valida() {
alert("A alterar o valor?")
}
```

Com a evolução do HTML, inúmeros outros eventos foram sendo definidos e associados a diferentes objetos, como são disso exemplos os eventos seguintes:

Eventos para formulários (<FORM> … </FORM>)	
oninput	Deteta quando são inseridos dados
oninvalid	Deteta quando os dados inseridos são inválidos
onreset	A função é executada quando o botão de reset é selecionado
Eventos associados à janela do browser (objeto window do browser)	
onpageshow	A função é executada quando o utilizador navega na página
onresize	A função é executada quando se altera a dimensão da janela
onoffline	A função é executada quando o browser começa a funcionar em offline

A integração da deteção de eventos nas páginas HTML pode ser realizada `inline`, por configuração das propriedades dos eventos nos objetos onde será detetado (como exemplificado acima), ou recorrendo aos métodos de processamento de eventos (`Event Handlers`) de cada objeto. Considerando a crescente interatividade das páginas, uma manutenção dos eventos nelas incluídos quando implementados `inline` nos inúmeros objetos, torna-se uma tarefa complexa sendo por isso preterida esta abordagem em detrimento da utilização de `event handlers`.

A configuração dos eventos com `event handlers`, recorre aos métodos

- `addEventListener('evento',funcProcessEvent)` – que ativa a deteção do `evento` e que, quando esse é detetado, invoca a função `funcProcessEvent` que o processa;
- `removeEventListener('evento',funcProcessEvent)` – desativa a deteção do evento.

JavaScript e DOM

Copie a página seguite para um ficheiro e teste-a no seu browser.

```
<html>
  <head>
    <title>Testes DOM</title>
    <script>
               function setBodyAttr(attr, value){
                 if (document.body) eval('document.body.'+attr+'="'+value+'"');
               }
               function ver() { window.alert("Versão de teste !");  }
    </script>
  </head>
  <body>
    <div style="margin: .5in; height: 400;">
      <p><b><tt>text</tt></b></p>
      <form>
        <select onChange="setBodyAttr('text', this.options[this.selectedIndex].value);" id="sel1">
          <option value="black">preto
          <option value="red">vermelho
        </select>
        <p><b><tt>bgColor</tt></b></p>
        <select onChange="setBodyAttr('bgColor', this.options[this.selectedIndex].value);" id="sel2">
          <option value="lightgrey">cinza
          <option value="lightpink">rosa
        </select>
        <p><b><tt>link</tt></b></p>
       <small> <a href="http://www.qualquer.lado" id="mostra"> (link) </a> </small><br>
      </form>
      <form>
        <input type="button" value="versão" onclick="ver()" />
      </form>
    </div>
    <script> document.getElementById("mostra").href ="http://www.isep.ipp.pt";   </script>
  </body>
</html>
```

a) Para que serve a função `setBodyAttr(...)`?

b) Para que serve a função `getElementById(...)` nesta página?

c) Que valores pode assumir `this.selectedIndex` do objecto `id="sel1"` ?

Referências

[1] "Tecnologia da Web para programadores", https://developer.mozilla.org/pt-PT/docs/Web

[2] "JavaScript Tutorial," http://www.w3schools.com/js/default.asp

[3] "JavaScript Bible," D. Goodman, M. Morrison, P. Novitski and T. G. Rayl.

[4] "Beginning HTML, XHTML, CSS, and JavaScript", Jon Duckett

Histórico

1) 27 de Abril de 2011, Versão 0.1, pmd@isep.ipp.pt
2) 4 de Maio de 2011, Versão 0.3, pmv@isep.ipp.pt, rjc@isep.ipp.pt
3) 11 de Maio de 2011, Versão 1.1, jbm@isep.ipp.pt

4) 14 de Maio de 2012, Versão 1.2, jbm@isep.ipp.pt
5) 13 de maio de 2017, Versão 1.3, jbm@isep.ipp.pt
6) 10 de maio de 2019, Versão 2.0, jbm@isep.ipp.pt
7) 25 de maio de 2023, Versão 2.1, jbm@isep.ipp.pt
8) 13 de maio de 2024, Versão 2.2, jbm@isep.ipp.pt
9) 20 de maio de 2025, Versão 2.3, jbm@isep.ipp.pt