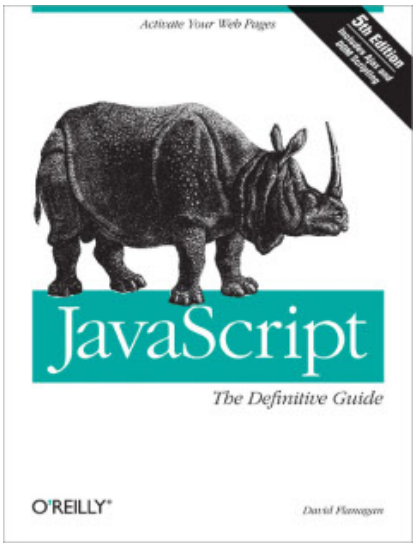
Linguagens e Ambientes de Programação (2018/2019)

Teórica 17 (13/mai/2019)

Introdução à linguagem JavaScript.
Porque vamos trabalhar com a linguagem JavaScript?
JavaScript embebido numa consola.
JavaScript embebido num WEB browser.
DOM - Document Object Model.

Introdução à linguagem JavaScript



Brendan Eich

Algumas características

- O principal responsável pelo desenho e implementação foi Brendan Eich. Esse trabalho começou a ser realizado no final de 1995 na empresa Netscape Communications. Inicialmente ele foi contratado para tornar a linguagem Java, já integrada no browser Netscape, mais fácil de usar por não-programadores. Mas rapidamente ele decidiu que que era necessário criar uma linguagem de scripting nova para facilitar a gestão dos elementos das páginas WEB. Acima de tudo era importante tornar a linguagem acessível a WEB designers com poucos conhecimentos de programação.

- O grande objetivo inicial do Java e depois do JavaScript foi adicionar interatividade a páginas WEB. O JavaScript foi concebido para a execução de tarefas simples, para as quais o Java era demasiado pesado. Contudo, por várias razões, o JavaScript é usado atualmente em projetos grandes e de formas que ultrapassam as intenções originais. As características da linguagem que a tornam fácil de usar acabam por se revelar problemáticas nessas situações. (Votaremos a este assunto numa próxima aula).

- Hoje em dia, a linguagem JavaScript está integrada em diversos tipos de aplicações, não só em browsers. Só alguns exemplos: a aplicação Acrobat usa-a para manipular ficheiros PDF; os controlos remotos programáveis topo-da-gama da Philips usam-na; potencialmente, qualquer aplicação escrita em Java 6.0, ou versão mais recente, pode integrar JavaScript usando o package `javax.script`.

- É uma linguagem que foi padronizada pela organização ECMA em 1997 e esse padrão tem evoluído: a versão atual é designada por 5.1. Há diferentes dialetos desse standard ECMA que, para além do núcleo padronizado, suportam diversas extensões. Para além do JavaScript, existe o JScript, o ActionScript, etc.

- A especificação ECMA diz que a linguagem deve poder ser interpretada, por forma a que seja possível correr scripts gerados dinamicamente.

- O corpo das funções tem uma sintaxe semelhante à do Java e outros aspetos superficiais imitam o Java (e.g. classe Math). Contudo, todo o resto da linguagem é diferente do Java: trata-se duma linguagem com tipificação dinâmica, que suporta programação funcional, e ao nível dos objetos baseia-se nas ideias invulgares da linguagem Self. A escolha do nome JavaScript foi apenas uma decisão de marketing tomada em 1995.

- A maioria das implementações assume que os programas correm dentro dum ambiente, e.g. um browser, que disponibiliza os objetos e os métodos com os quais a linguagem vai interagir. Só por si, a linguagem nem sequer possui operações de entrada (input) ou de saida (output).

- Suporta o uso de expressões regulares na manipulação de texto (inspirado no Awk e Perl).

- Tem gestão automática de memória, tal como o Java, mas opcionalmente pode ser usada uma primitiva `delete` para apagar explicitamente objetos, tal como em C++.

- O JavaScript é uma linguagem híbrida que suporta os paradigmas imperativo (procedimental), funcional e orientado pelos objetos.

Versões do JavaScript

A implementação original feita por Brendan Eich evoluiu com o tempo, e deu origem a duas implementações que atualmente são *open source* e são mantidas pela Mozilla Foundation:

- SpiderMonkey - Escrito em C.
- Rhino - Escrito em Java para permitir usar scripting dentro de programas Java. Serviu de base ao novo package `javax.script`, introduzido no Java 6.0.

Outra implementação mais que importa referir é a seguinte:

- node.js - Escrito pela Google e usado no Google Chrome. Muito rápido.

O JavaScript foi padronizado em 1997. O padrão chama-se ECMA-262, sendo ECMAScript o nome oficial da linguagem. No entanto, a linguagem continua a ser conhecida informalmente como JavaScript.

- JavaScript 1.0 - 1995
- JavaScript 1.1 - 1996
- ECMAScript 1 - 1997
- ECMAScript 2 - 1998
- ECMAScript 3 - 1999 - Introduz expressões regulares e exceções
- ECMAScript 5 - 2009 - Introduz JSON, geradores, iteradores, *compreensões* usando arrays, expressões `let`, etc.
- ECMAScript 5.1 - 2010
- ECMAScript 6 - 2015 - Introduz classes e modulos. As classes seguem o modelo do Java.
- ECMAScript 7 - 2016 - Introduz Array.prototype.includes.
- ECMAScript 8 - 2017 - Introduz await/async.

Bibliotecas e frameworks

Existem bibliotecas e frameworks que ajudam a escrever programas JavaScript sofisticados, com interfaces ambiciosas. Eis alguns das mais usados:

- JQuery
- Prototype Framework
- script.aculo.us

Porque vamos trabalhar com a linguagem JavaScript?

Na nossa cadeira, vamos estudar e trabalhar com a linguagem JavaScript por diversas razões. Eis as principais:

- Para tomarmos contacto com uma linguagem de scripting, para percebermos as características destas linguagens, e para sabermos quais as situações em que é vantajoso usá-las.
- Para estudarmos uma linguagem com um sistema de tipos dinâmico.
- Para estudarmos uma linguagem em que o sistema de objetos é baseado em protótipos e não em classes.
- Para revisitarmos o paradigma orientado pelos objetos.
- Para fazermos uma pequena introdução à programação Web do lado do cliente e do lado do servidor.

JavaScript embebido numa consola

O JavaScript só pode ser usado na prática, integrado num ambiente de execução que forneça objetos e métodos para interação com o exterior.

A versão de consola do Node.js corre num ambiente que fornece um interpretador interativo e uma consola, que disponibiliza uma operação de escrita `log`.

Examine bem esta pequena sessão de trabalho com a versão de consola do interpretador Node.js. Constitui um bom primeiro contacto com a linguagem.

```
// Correr nodejs
$nodejs

//Matemática simples
> 1 + 1
2

> d = Math.PI * 2 * 2
12.566370614359172

> Math.random()
0.25688508804887533

> typeof(1)
number

> typeof(1.0)
'number'

// Strings
> "hello, world"[0]
'h'

> "hello, world".replace("hello", "goodbye")
'goodbye, world'

> x = 12.4 + "144"
'12.4144'

> typeof(x)
'string'

> typeof(parseInt("123", 10))
'number'

> parseInt("aaa", 10)    // conversão base 10
NaN

> parseInt("aaa", 16)    // conversão base 16
2730

// Ciclos
> for (i = 0; i < 5; i++)
        console.log(i)
0
1
2
3
4

// Arrays
> a = ["dog", "cat", "hen"]
[ 'dog', 'cat', 'hen' ]

> a.length
3

> a[0] = 123.45
123.45

> a
[ 123.45, 'cat', 'hen' ]

> typeof(a[0])
'number'
```

```
> typeof(a)
'object'


// Funções
> function f(x) { return x + 1; }

> typeof(f)
'function'

> f(7)
8

> function curriedAdd(x)
    { return function(y) { return x + y; } }

> var g = curriedAdd(5)

> g(1)
6

// Fim
> .exit    // ou CTRL-D

$
```

Com apenas duas exceções, as expressões anteriores também funcionam no "JavaScript Shell" (disponível na coluna da esquerda da página da cadeira) e também na consola do "Firebug", uma extensão do Firefox que pode ser instalada a partir do site http://getfirebug.com.

JavaScript embebido num WEB browser

Virtualmente todos os browsers atuais têm um interpretador de JavaScript embebido. Cada browser proporciona ao JavaScript um ambiente de execução com objetos e métodos que permitem usar essa linguagem para criar páginas WEB interativas.

Chama-se DOM - Document Object Model ao ambiente que qualquer browser é obrigado a disponibilizar ao JavaScript, caso queira suportar a linguagem. O DOM implementa a visão que o JavaScript tem das páginas escritas em HTML e do estado interno do browser. Usando o DOM, o JavaScript consegue examinar e modificar dinamicamente qualquer página WEB e ainda examinar e modificar o estado do browser.

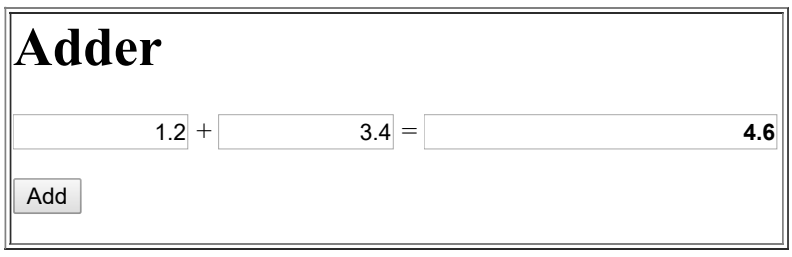
DOM é um padrão controlado pela organização W3C - World Wide Web Consortium, a mesma organização que controla o padrão HTML, XML e muitos outros padrões da WEB.

Para usar JavaScript integrado numa página WEB é necessário saber um pouco de HTML. Nas nossas aulas vamos restringuir-nos à programação de formulários HTML (HTML forms).

Convém saber que há incompatibilidades entre diferentes browsers em diferentes plataformas. Conseguir fazer um programa em JavaScript que funcione em todos os browser existentes é uma tarefa complicada.

Exemplo 1

O seguinte exemplo é um formulário simples que permite somar números. O formulário é constituída por três caixas de texto, a terceira das quais é *read-only*, e por um botão. Por favor, brinque um pouco com o formulário para perceber o seu comportamento.



Agora examine o código HTML e JavaScript que implementa o formulário anterior. Repare que no botão "Add" o atributo ONCLICK tem associado código que será executado sempre que o botão gerar um **evento de click**.

```
<HTML>
```

```
<HEAD>
<TITLE>MyDocument</TITLE>

<SCRIPT TYPE="text/javascript">
function compute(n1, n2) {
    return n1 + n2;
}
function runForm1(form) {
    form.text3.value = compute(parseFloat(form.text1.value), parseFloat(form.text2.value));
}
</SCRIPT>
</HEAD>

<BODY>
<H1>Adder</H1>
<FORM ID="form1">
     <INPUT TYPE="text" ID="text1" VALUE="1.2" SIZE=10 style="text-align: right">
   + <INPUT TYPE="text" ID="text2" VALUE="3.4" SIZE=10 style="text-align: right">
   = <INPUT TYPE="text" ID="text3" VALUE="4.6" SIZE=25 READONLY style="text-align: right; font-weight: bold">
 <p> <INPUT TYPE="button" ID="button1" VALUE="Add" ONCLICK="runForm1(form)">
</FORM>
</BODY>

</HTML>
```

Exemplo 2

O seguinte exemplo é uma *form* contendo botões de rádio e uma área de texto. Por favor, teste o formulário.

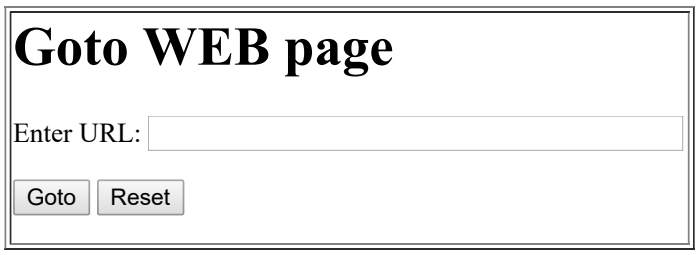
Eis o código HTML e JavaScript correspondentes:

```
<HTML>

<HEAD>
<TITLE>MyDocument</TITLE>

<SCRIPT>
function runForm2(form) {
    var msg = "Sent by: ";
    msg += form.firstname.value + " " + form.lastname.value;
    msg += " (" + form.email.value + ") ";
    if( form.sex1.checked ) {
        msg += " [male]";
    }
    else if( form.sex2.checked ) {
        msg += " [female]";
    }
    else {
        msg += " []";
    }
    msg += "\nMesg: " + form.textarea1.value;
    alert(msg);
}
</SCRIPT>
</HEAD>
```

```
<BODY>
<H1>Send Messages</H1>
<FORM ID="form2">
    First name: <INPUT TYPE="text" ID="firstname"><BR>
    Last name: <INPUT TYPE="text" ID="lastname"><BR>
    email: <INPUT TYPE="text" ID="email"><BR>
    <INPUT TYPE="radio" ID="sex1" VALUE="Male" ONCLICK="sex2.checked=false"> Male
    <INPUT TYPE="radio" ID="sex2" VALUE="Female" ONCLICK="sex1.checked=false"> Female<BR>
    <P><B>Your Message</B><BR>
    <TEXTAREA ID="textarea1" ROWS="5" COLS="50">Supercalifragilisticexpialidocious.</TEXTAREA>
    <P><INPUT TYPE="button" ID="button1" VALUE="Send" ONCLICK="runForm2(form)">
    <INPUT type="reset">
</FORM>
</BODY>

</HTML>
```

Exemplo 3

O seguinte exemplo mostra como se pode carregar um novo URL, mudando assim completamente o conteúdo do documento corrente. Também ilustra a criação duma caixa de alerta.

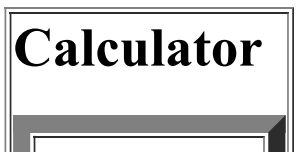


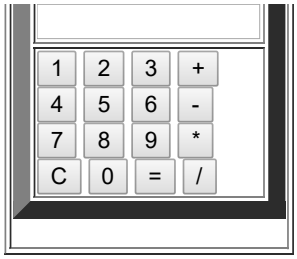
Eis o código HTML e JavaScript correspondentes:

```
<HTML>

<HEAD>
<TITLE>MyDocument</TITLE>

<SCRIPT>
function runForm3(form) {
    var url = form.url.value;
    if( url == "" ) {
        alert("Error: Empty URL");
    }
    else {
        document.location = form.url.value;   // Change URL
    }
}
</SCRIPT>
</HEAD>

<BODY>
<H1>Goto WEB page</H1>
<FORM ID="form3">
    Enter URL: <INPUT TYPE="text" ID="url" SIZE=40>
    <P><INPUT TYPE="button" ID="button1" VALUE="Goto" ONCLICK="runForm3(form)">
    <INPUT type="reset">
</FORM>
</BODY>

</HTML>
```

Exemplo 4

O seguinte exemplo serve para ilustrar a operação eval do JavaScript. Como já foi dito antes, todas as implementações de JavaScript devem ser capazes de interpretar código gerado dinamicamente. Isso pressupõe a existência duma função `eval` para correr código JavaScript arbitrário.



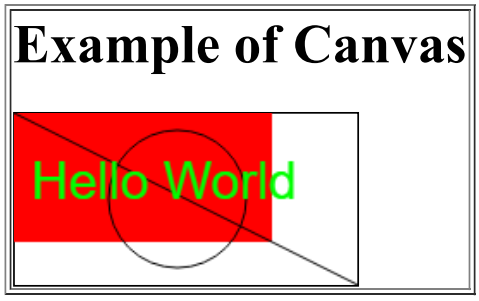


Eis o código HTML e JavaScript que implementa a calculadora. A implementação é verdadeiramente simples. Constrói no mostrador da calculadora, a pouco e pouco, a expressão introduzida pelo utilizador sob a forma de string. No final, quando o utilizador carrega na tecla '=', invoca a função `eval` para avaliar a o conteúdo da string. Este truque funciona bem porque as expressões aritméticas do JavaScript têm a sintaxe habitual da matemática.

```
<HTML>

<HEAD>
<TITLE>MyDocument</TITLE>
</HEAD>

<BODY>
<H1>Calculator</H1>
<FORM ID="form4">
    <TABLE BORDER=10>
    <TR><TD>
        <INPUT TYPE="text" ID="display" Size="14">
    </TD></TR>
    <TR><TD>
        <INPUT TYPE="button" ID="1" VALUE="1" ONCLICK="form.display.value += '1'">
        <INPUT TYPE="button" ID="2" VALUE="2" ONCLICK="form.display.value += '2'">
        <INPUT TYPE="button" ID="3" VALUE="3" ONCLICK="form.display.value += '3'">
        <INPUT TYPE="button" ID="+" VALUE="+" ONCLICK="form.display.value += '+'">
    <br>
        <INPUT TYPE="button" ID="4" VALUE="4" ONCLICK="form.display.value += '4'">
        <INPUT TYPE="button" ID="5" VALUE="5" ONCLICK="form.display.value += '5'">
        <INPUT TYPE="button" ID="6" VALUE="6" ONCLICK="form.display.value += '6'">
        <INPUT TYPE="button" ID="-" VALUE="-" ONCLICK="form.display.value += '-'">
    <br>
        <INPUT TYPE="button" ID="7" VALUE="7" ONCLICK="form.display.value += '7'">
        <INPUT TYPE="button" ID="8" VALUE="8" ONCLICK="form.display.value += '8'">
        <INPUT TYPE="button" ID="9" VALUE="9" ONCLICK="form.display.value += '9'">
        <INPUT TYPE="button" ID="*" VALUE="*" ONCLICK="form.display.value += '*'">
    <br>
        <INPUT TYPE="button" ID="C" VALUE="C" ONCLICK="form.display.value = ''">
        <INPUT TYPE="button" ID="0" VALUE="0" ONCLICK="form.display.value += '0'">
        <INPUT TYPE="button" ID="=" VALUE="=" ONCLICK="form.display.value = eval(form.display.value)">
        <INPUT TYPE="button" ID="/" VALUE="/" ONCLICK="form.display.value += '/'">
    <br>
    </TD></TR>
    </TABLE>
</FORM>
</BODY>

</HTML>
```

Exemplo 5

O seguinte exemplo ilustra o uso dum **canvas** em JavaScript. Um canvas é simplesmente um retângulo onde se podem desenhar gráficos.



Examine o código HTML e JavaScript que implementa o retângulo anterior.

O preenchimento do canvas só pode ser feito depois do canvas ter sido criado. Uma forma de fazer isso é colocar o código JavaScript nas linhas imediatamente a seguir à definição do canvas. Mas é considerado pouco elegante misturar código JavaScript com código HTML no body da página. É preferível colocar o código JavaScript dentro do header da página, como temos feito sempre, e invocar esse código depois da página ter sido carregada. Existe um **evento de onload** que é ativado quando a página é carregada. Esse evento é capturado usando o atributo ONLOAD no elemento BODY.

```
<HTML>

<HEAD>
<TITLE>Canvas</TITLE>

<SCRIPT>
function fillCanvas() {
    var c=document.getElementById("canvas1");
    var ctx=c.getContext("2d");

    ctx.fillStyle="#FF0000";
    ctx.fillRect(0,0,150,75);

    ctx.moveTo(0,0);
    ctx.lineTo(200,100);
    ctx.stroke();

    ctx.beginPath();
    ctx.arc(95,50,40,0,2*Math.PI);
    ctx.stroke();

    ctx.fillStyle="#00FF00";
    ctx.font="30px Arial";
    ctx.fillText("Hello World",10,50);
}
</SCRIPT>
</HEAD>

<BODY ONLOAD="fillCanvas()">
<H1>Example of Canvas</H1>
<CANVAS id="canvas1" width="200" height="100" style="border:1px solid #000000">
    This text is displayed if your browser does not support HTML5 Canvas
</CANVAS>
</BODY>
</HTML>
```

Exemplo 6

Este exemplo mostra como se pode mudar um pedaço de texto arbitrário dentro duma página Web.

Exemplo: blá blá Click Me 0 blá blá.

Requer um pouco de planeamento, porque precisamos primeiro de associar um ID ao texto a alterar.

Se o texto já estiver marcado de alguma forma, por exemplo com a marca de bold assim **`<b>o meu texto</b>`**, então basta adicionar o ID desta forma, **`<b id='o meu id'>o meu texto</b>`**.

Se o texto não estiver marcado, então temos de o marcar com a marca **span** e adicionar o ID desta forma, **`<span id='o meu id'>o meu texto</span>`**.

Agora que o texto já está marcado e tem um ID, para lhe aceder em JavaScript basta usar a expressão **`document.getElementById('o meu id').innerHTML`**.

Veja o código do exemplo:

```
Exemplo: blá blá
<INPUT TYPE="button"
 VALUE="Click Me"
 ONCLICK="var el = document.getElementById('txt0'); el.innerHTML = parseInt(el.innerHTML,10) + 1"
>
<SPAN id='txt0'>0</SPAN>
blá blá.
```

#100