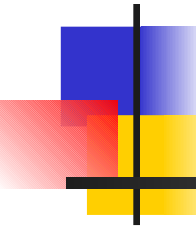
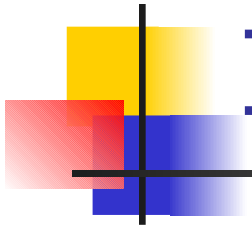
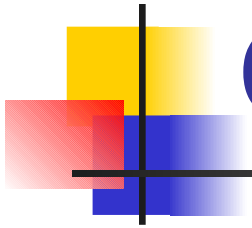


JavaScript



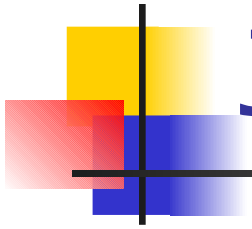
Introdução

- O JavaScript é utilizado por milhões de páginas na web para melhorar o design, validar forms, e muito mais…
- O JavaScript foi inicialmente desenvolvido pela Netscape e é a linguagem de scripting mais popular na Internet
- O JavaScript funciona a partir da versão 3 dos browsers Internet Explorer e Netscape
- Actualmente quase todos os browsers existentes suportam JavaScript



O que é o JavaScript?

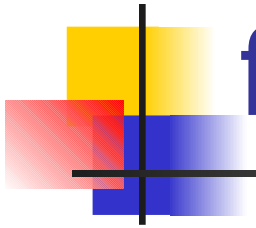
- O JavaScript foi projectado para adicionar interactividade às páginas HTML
- O JavaScript é uma linguagem de script – uma linguagem de script é uma linguagem de programação simples
- O Javascript são linhas de código executáveis pelo computador
- O Javascript é normalmente embebido directamente nas páginas HTML
- O JavaScript é uma linguagem interpretada (quer dizer que funciona sem prévia compilação)
- Qualquer um pode utilizar o JavaScript sem ter que comprar uma licença



Java e JavaScript

- Java e JavaScript são duas linguagens diferentes
- O Java desenvolvido pela Sun é uma linguagem de programação potente e complexa – nas mesmas categorias que o C e o C++

O que é que o JavaScript pode fazer?



- **O JavaScript dá aos projectistas de HTML uma linguagem de programação –** Os autores de HTML muitas vezes não são programadores, mas o JavaScript é uma linguagem de script com uma sintaxe muito simples! Quase qualquer um pode colocar pequenos bocados de código nas suas páginas HTML

- **O JavaScript pode colocar texto dinâmico na sua página HTML –** com uma simples instrução como esta: **`document.write("<h1" + nome + "<h1>")`** pode escrever uma variável de texto na página HTML

O que é que o JavaScript pode fazer (continuação)?

- **O JavaScript pode reagir a eventos –** A JavaScript pode ser configurado para executar quando qualquer coisa acontece, como quando o página acaba de ser carregada ou um utilizador clica em um elemento HTML

- **O JavaScript pode ler e escrever elementos HTML –** O JavaScript pode ler e modificar o conteúdo de um elemento HTML

- **JavaScript pode ser utilzado para validar dados –** O JavaScript pode ser utilizador para validar dados de uma forma antes de a submeter para um servidor, o que guarda o servidor de processamento extra

Como por código JavaScript em uma página HTML?

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

Este código vai produzir o output:

Olá Mundo

Explicação do exemplo

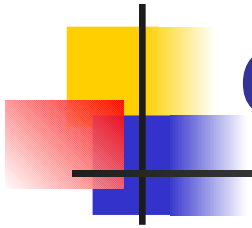
- Para inserir o script em uma página HTML, utiliza-se a tag <script>. Utiliza-se este tipo de atributo para definir a linguagem de script

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

- Então, o JavaScript inicia; O comando de JavaScript para escrever para uma página é **`document.write("Olá Mundo!")`**
- No fim do script é necessário fechar a tag <script>

```
</script>
```

Finalizar instruções com ponto e vírgula?



- Nas linguagens de programação tradicionais, como o C++ e o Java, cada intrução tem que terminar com ponto e vírgula

- Alguns programadores continuam este hábito quando escreve JavaScript, mas em geral, é opcional! De qualquer modo, o ponto e vírgual é necessário se quiser por mais de uma instrução na mesma linha

Como lidar com browsers antigos?

- Browsers que não suportam scripts vão mostrar o script como conteúdo da página. Para prevenir isso de acontecer, pode-se utilizar a tag de comentário do HTML

```
<script type="text/javascript">
<!-
Algumas instruções
//-->
</script>
```

- As duas barras (//) no fim da linha de comentário são um símbolo de comentário do JavaScript. Isso previne o JavaScript de interpretar aquela linha

Onde colocar o JavaScript?

- Os scripts na secção BODY vão ser executados enquanto a página carrega
- Os scripts na secção head vão ser executados quando chamados
- Quando se coloca um script na secção head, asseguramos que o script é carregado antes de alguém o utilizar

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

Onde colocar o JavaScript?

- **Scripts na secção BODY:** Script para serem executados quando a página carrega e vai para a secção BODY.

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

- É possível ter scripts no BODY e no HEAD

Como executar JavaScript externo

- Às vezes pode-se querer executar o mesmo script em várias páginas, sem ter que escrever o script em cada página
- Para simplificar, pode-se escrever o script em um ficheiro externo e guardá-lo com a extensão de ficheiro .js
- Document.write("Este script é externo")
- Guarde o ficheiro externo como exp.js, e note que o script externo não contém a tag <script>
- Agora pode-se chamar o script utilizando o atributo "src" em qualquer página

```
<html>
<head>
</head>
<body>
<script src="xxx.js">
</script>
</body>
</html>
```

Exemplo de script na secção HEAD

- Scripts que contenham funções vão para a secção head do documento. Então pode-se ter a certeza que os scripts são carregados antes da função ser chamada

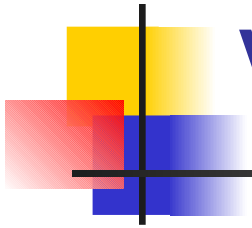
```
<html>
<head>
<script type="text/javascript">
function mensagem()
{
   alert("Esta caixa de alerta foi chamada através do evento
   onload")
}
</script>
</head>

<body onload="mensagem()">

</body>
</html>
```

Exemplo de script na secção BODY

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

<body>

<script type="text/javascript">
  document.write("This message is written
  when the page loads")
</script>

</body>
</html>
```

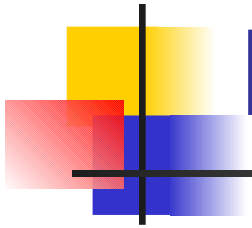
Como aceder a um script externo

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

<script src="exp.js">
</script>

<p>
    O script actual é um ficheiro externo chamado "exp.js".
</p>

</body>
</html>
```



Variáveis JavaScript

- Uma variável é um "contentor" de informação que deseja guardar
- O valor de uma variável pode mudar durante o script
- Pode-se referir a uma variável pelo nome para ver o seu valor ou para modificar o seu valor
- Regras para os nomes de variáveis
  - Os nomes de variáveis são case sensitive
  - Devem começar com uma letra ou com o caracter underscore



Declarar uma variável

- Pode criar uma variável com a instrução **`var`**

  `var strnome = `*`algumnome`*

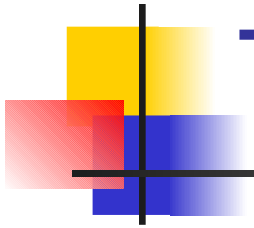
- Pode também criar uma variável sem a instrução **`var`**

Atribuir um valor a uma variável

- Atribui um valor a uma variável assim:

  `var strnome = "Marco"`
- Ou assim

  `strnome = "Marco"`
- O nome da variável fica no lado esquerdo da expressão e o valor que pretende atribuir à variável fica na direita
- Agora a variável "strnome" tem o valor "Marco"



Tempo de vida das variáveis

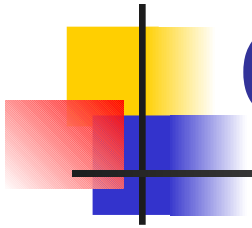
- Quando declara uma variável dentro de uma função, a variável somente pode ser acedida dentro da função.
- Quando sai da função, a variável é destruída.
- Estas variáveis chamam-se variáveis locais
- Pode-se ter variáveis locais com o mesmo nome em diferentes funções, porque são reconhecidas somente pela função onde estão declaradas
- Se declarar uma variável fora de uma função, todas as funções na sua página web podem aceder a essa variável
- O tempo de vida destas ultimas variáveis começa quando são declaradas, e termina quando a página web é fechada

Operadores aritméticos em Javascript

Operador	Descrição	Exemplo	Resultado
+	Adição	x=2 x+2	4
-	Subtracção	x=2 5-x	3
*	Multiplicação	x=4 x*5	20
/	Divisão	15/5 5/2	3 2.5
%	Resto da divisão	5%2 10%8 10%2	1 2 0
++	Incremento	x=5 x++	x=6
--	Decremento	x=5 x	x=4

Exemplo da utilização de variáveis JavaScript

```
<html>
<body>

<script type="text/javascript">
   var nome = "Hege"
   document.write(nome)
   document.write("<h1>" + nome + "</h1>")
</script>

<p>Este exemplo declara uma variável, dá um valor a essa
   variável, e mostra essa variável.</p>

</body>
</html>
```



Operadores de atribuição

Operador	Exemplo	É o mesmo que
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

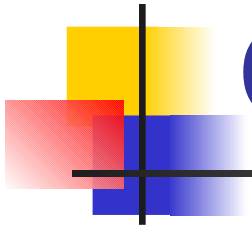
Operadores de comparação

Operador	Descrição	Exemplo
==	É igual a	5==8 retorna falso
!=	Não é igual a	5!=8 retorna verdadeiro
>	É maior que	5>8 retorna falso
<	É menor que	5<8 retorna verdadeiro
>=	É maior que ou igual a	5>=8 retorna falso
<=	É menor ou igual a	5<=8 retorna verdadeiro

Operadores lógicos

Operador	Descrição	Exemplo
&&	and	x=6 y=3 (x < 10 && y > 1) retorna verdadeiro
||	or	x=6 y=3 (x==5 || y==5) retorna falso
!	not	x=6 y=3 !(x==y) retorna verdadeiro

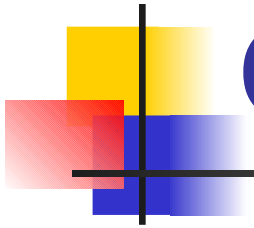
Operador de strings



- Uma string (cadeia de caracteres) é frequentemente texto
- Por exemplo "Olá Mundo!"
- Para juntar duas ou mais strings, é utilizado o operador **+**

```
txt1 = "Que dia"
txt2 = "bonito está hoje"
txt3 = txt1 + txt2
```

- A variável txt3 agora contem "Que diabonito está hoje"



Operador de strings

- Para adicionar um espaço entre as duas strings, insira um espaço na expressão, ou em uma das strings

```
txt1 = "Que dia"
txt2 = "bonito está hoje"
txt3 = txt1 + " " + txt2
```

- Ou

```
txt1 = "Que dia "
txt2 = "bonito está hoje"
txt3 = txt1 + txt2
```

Funções em JavaScript

- Uma função é um bloco de código reutilizável que pode ser executado através de um evento, ou quando a função é chamada
- Uma função é um conjunto de instruções
- As funções são definidas no início do ficheiro (na secção head), e chamadas mais tarde no ficheiro
- Função pré-definida do JavaScript

```
alert("Isto é uma mensagem"
```

Exemplo de uma função

```
<html>
<head>
<script type="text/javascript">
function minhafunc()
{
   alert(“Olá")
}
</script>
</head>
<body>
<form>
<input type="button" onclick="minhafunc()" value=“Chama">
</form>
<p>Carregando no botão, a função será chamada. A função
   será um alert com uma mensagem.</p>
</body>
</html>
```

Funções em JavaScript

- Para criar uma função, são definidos o nome, os argumentos, e algumas instruções:

```
minhafunc(argumento1, argumento2, etc)
{
   Algumas instruções
}
```

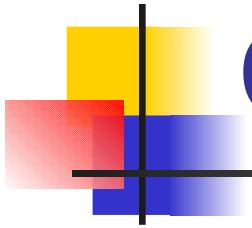
- Uma função sem argumentos, deve incluir parêntesis

```
minhafunc()
{
   Algumas instruções
}
```

Funções em JavaScript

- Os argumentos são variáveis utilizadas na função. Os valores destas variáveis são passados na chamada da função
- Colocando as funções na secção head, certifica-se que o código da função já foi carregado antes de se chamar a função
- Algumas funções retornam um valor para a expressão de chamada

```
function resultado(a, b)
{
   c = a + b
   return c
}
```



Como chamar uma função

- Uma função não é executada antes de ser chamada
- Pode-se chamar uma função contendo argumentos:

```
minhafunc(argumento1, argumento2, etc)
```

- Ou sem argumentos

```
minhafunc()
```

A instrução return

- As funções que retornam um resultado têm de utilizar a instrução "return"
- Esta instrução especifica o valor que vai ser retornado para o local onde a função foi chamada
- Imagine uma função que retorna a soma de dois números:

```
function total(a, b) {
   resultado = a + b
   return resultado
}
```

- Quando chamar a função tem que enviar os dois argumentos

```
soma = total(2, 3)
```

- O valor retornado pela função fica guardado na variável soma

Função com argumentos

```
<html>
<head>
<script type="text/javascript">
function minhafunc(txt)
{
   alert(txt)
}
</script>
</head>
<body>
<form>
<input type="button" onclick="myfunction('Hello')" value="chama">
</form>
<p>Carregando no botão, a função com um argumento vai ser
   chamada, a função vai enviar um alert com esse argumento.</p>
</body>
</html>
```

Função com argumentos

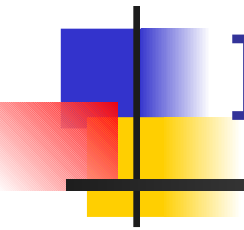
```
<html>
<head>
<script type="text/javascript">
function myfunc(txt)
{
   alert(txt)
}
</script>
</head>
<body>
<form>
<input type="button" onclick="myfunc('Bom dia!')" value="Manhã">
<input type="button" onclick="myfunc('Boa tarde!')" value="Tarde">
</form>
<p>A função vai alertar uma mensagem diferente conforme o botão
   que carregar.</p>
</body>
</html>
```

Uma função que retorna um valor

```
<html>
<head>
<script type="text/javascript">
function myFunction() {
   return ("Olá, tenha um bom dia!")
}
</script>
</head>
<body>
<script type="text/javascript">
   document.write(myFunction())
</script>
<p>O script na secção body chama a função.</p>
<p>A função retorna um texto.</p>
</body>
</html>
```

Uma função que faz a soma de 2 argumentos e retorna o resultado

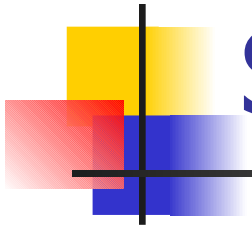
```
<html>
<head>
<script type="text/javascript">
function total(numeroA,numeroB)
{
   return numeroA + numeroB
}
</script>
</head>
<body>
<script type="text/javascript">
   document.write(total(2, 3))
</script>
<p>O script na secção body chama a função com dois argumentos,
   2 e 3.</p>
<p>A função retorna a soma dos dois argumentos.</p>
</body>
</html>
```



Instruções Condicionais

Instruções condicionais

- Frequentemente quando se programa, pretende-se efectuar acções diferentes conforme decisões diferentes, para isso, utiliza-se instruções condicionais
- Em JavaScript tempos três tipos de instruções condicionais:
  - A instrução **if** – utilize esta instrução se pretender efectuar um determinado bloco de código quando uma condição é verdadeira
  - A instrução **If ... else** – utilize esta instrução quando pretender seleccionar um de dois blocos de código para executar
  - A instrução **switch** – utilize esta instrução se pretender seleccionar um de muitos blocos possíveis de execução



Sintaxe da instrução **if**

if (*condição*)

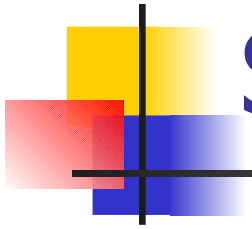
{

Código para ser executado se a condição for verdadeira

}

Exemplo da instrução **if**

```
<script type="text/javascript">
// Se as horas forem menores do que 12,
// vai receber um "Bom dia"

var d = new Date()
var horas = d.getHours()
if (horas < 12)
{
  document.write("<b>Bom dia</b>")
}
</script>
```

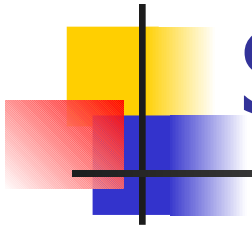


Sintaxe do **If ... else**

```
if (condição)
{
   Código a ser executado se a condição for verdadeira
}
else
{
   Código para ser executado se a condição for falsa
}
```

Exemplo do **If ... else**

```
<script type="text/javascript">
// Se as horas forem menores do que 12,
// vai receber um "Bom dia"
// Senão vai receber uma "Boa tarde"

var d = new Date()
var horas = d.getHours()
if (horas < 12)
{
   document.write("Bom dia!")
}
else
{
   document.write("Boa tarde!")
}
</script>
```



Sintaxe da instrução **switch**

```
switch (expressão)
{
case label1:
   Código a ser executado se a expressão = label1
   break
case label2:
   Código a ser executado se a expressão = label2
   break
default:
   Código a ser executado se a expressão for
   diferente de label1 e de label2
}
```

Exemplo da instrução **switch**

```
<script type="text/javascript">
// Receberá uma mensagem diferente conforme o dia, note que o
Domingo = 0, Segunda = 1, terça = 2, etc.
var d = new Date()
diaSemana = d.getDay()
switch (diaSemana)
{
case 5:
   document.write("Finalmente sexta")
   break
case 6:
   document.write("Super sábado")
   break
case 0:
   document.write("Sonolento domingo")
   break
default:
   document.write("Nunca mais chega a sexta")
}
</script>
```

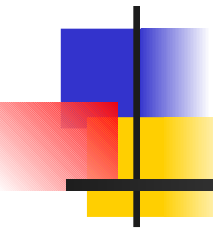
Operador condicional

- JavaScript também contem um operador condicional que atribui um valor a uma variável baseado em uma condição
- No exemplo a seguir, se a variável visita for igual a PRES, então é posta a mensagem “Caro presidente” na variável com o nome msg. Se a variável visita não for igual a PRES, então é posta a string “Caro “ na variável com o nome msg
- Sintaxe:

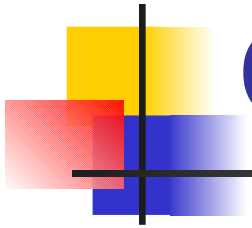
```
nomevariável = (condição) ? valor1 : valor2
```

- Exemplo

```
msg = (visita == "PRES") ? “Caro Presidente “ : “Caro "
```

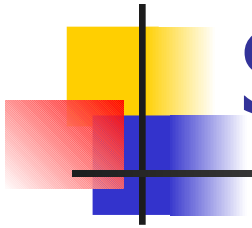


Ciclos em JavaScript



Ciclos em JavaScript

- Os ciclos em JavaScript são utilizados para repetir um determinado bloco de código um número especificado de vezes, para isso, existem as seguintes instruções para ciclos:
- **While** – repete um bloco de código enquanto uma condição for verdadeira
- **do...while** – Executa o código uma vez, e depois repete esse código enquanto uma condição for verdadeira
- **for** – executa um bloco de código um determinado nº de vezes

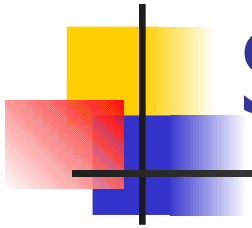


Sintaxe do **While**

```
while (condição)
{
   código para ser executado
}
```

Exemplo do **while**

```
<html>
<body>
<script type="text/javascript">
i = 0
while (i <= 5)
{
  document.write("O número é " + i)
  document.write("<br>")
  i++
}
</script>
</body>
</html>
```

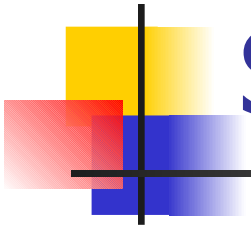


Sintaxe do **do...while**

```
do
{
   código a ser executado
}
while (condição)
```

Exemplo do **do...while**

```
<html>
<body>
<script type="text/javascript">
i = 0
do
{
  document.write("O número é " + i)
  document.write("<br>")
  i++
}
while (i <= 5)
</script>
</body>
</html>
```



Sintaxe do **for**

for (*inicialização*; *condição*; *incremento*)

{

código a ser executado

}

Exemplo do **for**

```
<html>
<body>
<script type="text/javascript">
for (i = 0; i <= 5; i++)
{
  document.write("O número é " + i)
  document.write("<br>")
}
</script>
```