

Introdução às tecnologias Web - ITW

Aula 7 – Javascript (continuação)

Sumário

A linguagem javascript

- Variáveis públicas e privadas

Programando com o javascript

- Interacção com o DOM (revisitado)

- Validação de formulários

- Lista de tarefa (to do list)

Temporizadores



Revisões

Ciclos

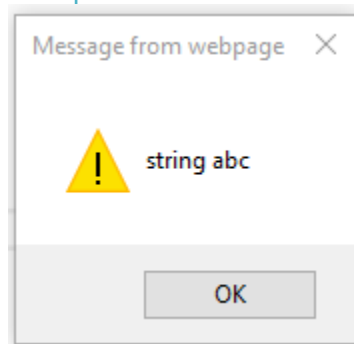
Sintaxe da linguagem Javascript

Condições (switch ... case) - (revisão)

Quando há mais que uma condição para testar, é possível a utilização de um conjunto de instruções `if ... else` encadeadas ou, em alternativa, a utilização da instrução `switch ... case`.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    var a = "abc";
    switch (a) {
      case "abc":
        alert("string abc");
        break;
      case 3:
        alert("inteiro 3");
        break;
      default:
        alert("outro");
    }
  </script>
</head>
<body>
  <!-- Conteúdo html aqui ...-->
</body>
</html>
```

- Para cada comparação há uma instrução `case`.
 - Cada instrução `case` deve ser separada por uma instrução `break`. Caso contrário, o programa continuará a fazer as comparações seguintes.
- A instrução `default` será executada caso nenhuma das instruções de comparação tenha sido válida.
 - Esta instrução não precisa do separador `break`.




Sintaxe da linguagem Javascript

Condições (switch ... case) - (revisão)

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    var a = "abc";
    switch (a) {
      case "abc":
        alert("string abc");
        break;
      case 3:
        alert("inteiro 3");
        break;
      default:
        alert("outro");
    }
  </script>
</head>
<body>
  <!-- Conteúdo html aqui ...-->
</body>
</html>
```

Conversão de um switch
em múltiplos if's



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>document</title>
  <script>
    var a = "abc";
    if (a == "abc") {
      alert("string abc");
    }
    else
      if (a == 3) {
        alert("inteiro 3");
      }
    else {
      alert("outro");
    }
  </script>
</head>
<body>
  <!-- Conteúdo html aqui ...-->
</body>
</html>
```

Sintaxe da linguagem Javascript

Ciclos (revisão)

Para implementar ciclos, a linguagem JS suporta as instruções `while`, `do-while`, e `for`:

```
do {  
    /* instruções */  
} while (condição);
```

```
while (condição) {  
    /* instruções */  
}
```

```
for (início; comparação; incremento) {  
    /* instruções */  
}
```

Diferenças entre os diversos tipos de ciclos:

- `do-while` – as instruções do ciclo são executadas pelo menos uma vez porque a `condição` de comparação é executada no fim do ciclo;
- `while` – as instruções do ciclo são executadas 0 ou mais vezes, pois o ciclo só se realiza se a `condição` se verificar à partida;
- `for` – as instruções do ciclo são executadas um número fixo de vezes – desde o `início` até à `comparação` com um `incremento`.

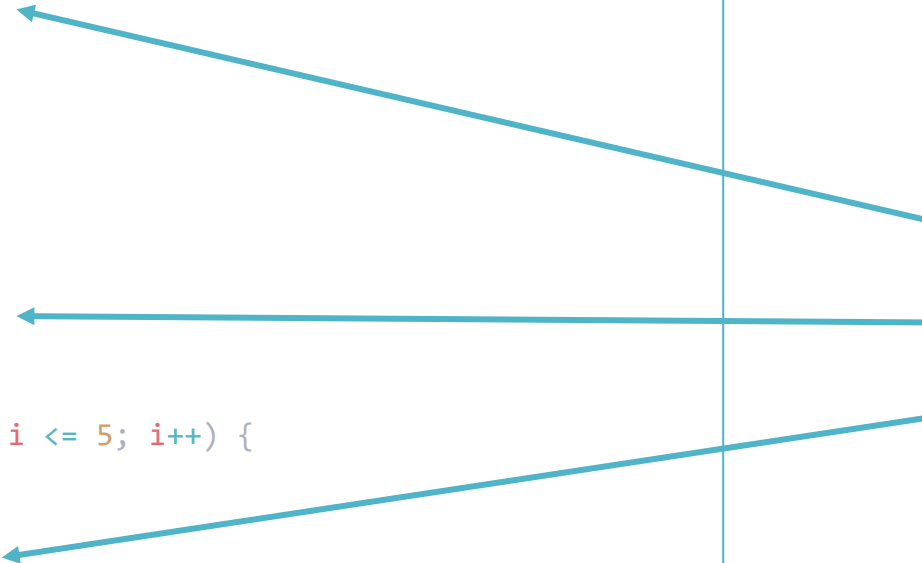
Sintaxe da linguagem Javascript

Ciclos (revisão)

```
var i = 7;
var a = 2;
do {
    a += ++i;
}
while (i < 5);
console.log(a);

var i = 7;
var a = 2;
while (i < 5) {
    a += i++;
}
console.log(a);

var a = 1;
for (var i = 2; i <= 5; i++) {
    a *= i;
}
console.log(a);
```



(Recursividade)

```
function f(x){
    if(x<=1)
        return 1;
    return x * f(x-1);
}
console.log(f(5));
```

?

Variáveis públicas e privadas

Revisitando a aula prática anterior

Variáveis globais e locais

Variáveis globais são variáveis declaradas fora de uma função;
As variáveis globais podem ser usadas em qualquer contexto.

Variáveis locais são declaradas dentro de uma função.
Uma variável local declarada dentro de uma função só poderá ser usada dentro dessa função.

Apesar da facilidade que as variáveis globais oferecem, também têm as suas desvantagens.
Quando uma variável é declarada como global ela ocupará memória até que só será libertada quando o programa finalizar – por isso devem ser usadas apenas quando necessário.
As variáveis locais só ocupam memória enquanto aquela função for executada, sendo destruída, e libertada a memória ocupada pela mesma, no final da execução da função.

```
var operacao = "+";
```

```
function getOperacao() {  
    var e = document.getElementById("operacao");  
    operacao = e.options[e.selectedIndex].value;  
    console.log(operacao);  
}
```

```
function calcula() {  
    /* Vamos precisar de código aqui ... */  
    var op1 = document.getElementById("op1"); /* op1 é o marcador <input id="op1" ...> todo! */  
    var op2 = document.getElementById("op2");  
    var res = document.getElementById("res");  
    switch (operacao) {  
        case "+":  
            res.value = parseFloat(op1.value) + parseFloat(op2.value);  
            //res.value = op1.value + op2.value;  
            break;  
        case "-":  
            res.value = op1.value - op2.value;  
            break;  
        default:  
            alert("Erro: operação não definida ...");  
    }  
}
```

```
console.log(operacao);  
console.log(op1.value);  
console.log(op2.value);
```

Qual o resultado destas operações?

Validação de formulários

O javascript em ação...

Validação de formulários

A validação em formulários é fundamental para garantir a qualidade dos dados recebidos, melhorar a experiência do utilizador e preservar os recursos do servidor.

Quando dados inválidos são enviados ao servidor, o aplicativo pode apresentar erros, falhas de segurança, ou até comprometer a lógica de negócio.

Com validação, garantimos que apenas dados no formato esperado chegam ao servidor, reduzindo a chance de erros e resultados inesperados.

Validar no lado do cliente reduz a necessidade de processamento no servidor, diminuindo o uso de banda e evitando chamadas desnecessárias.

Isso é especialmente importante para grandes aplicações e em picos de acesso, onde a sobrecarga pode causar lentidão ou até interrupções no serviço.

Sem validação, um sistema fica vulnerável a ataques como *SQL Injection*, ou o *Cross-Site Scripting* (XSS), e a injeção de código.

A validação de formulários evita a inserção de dados maliciosos, bloqueando potenciais ameaças antes que elas atinjam o servidor.

A validação fornece feedback imediato ao utilizador, evitando frustrações e reduzindo erros no preenchimento dos formulários.

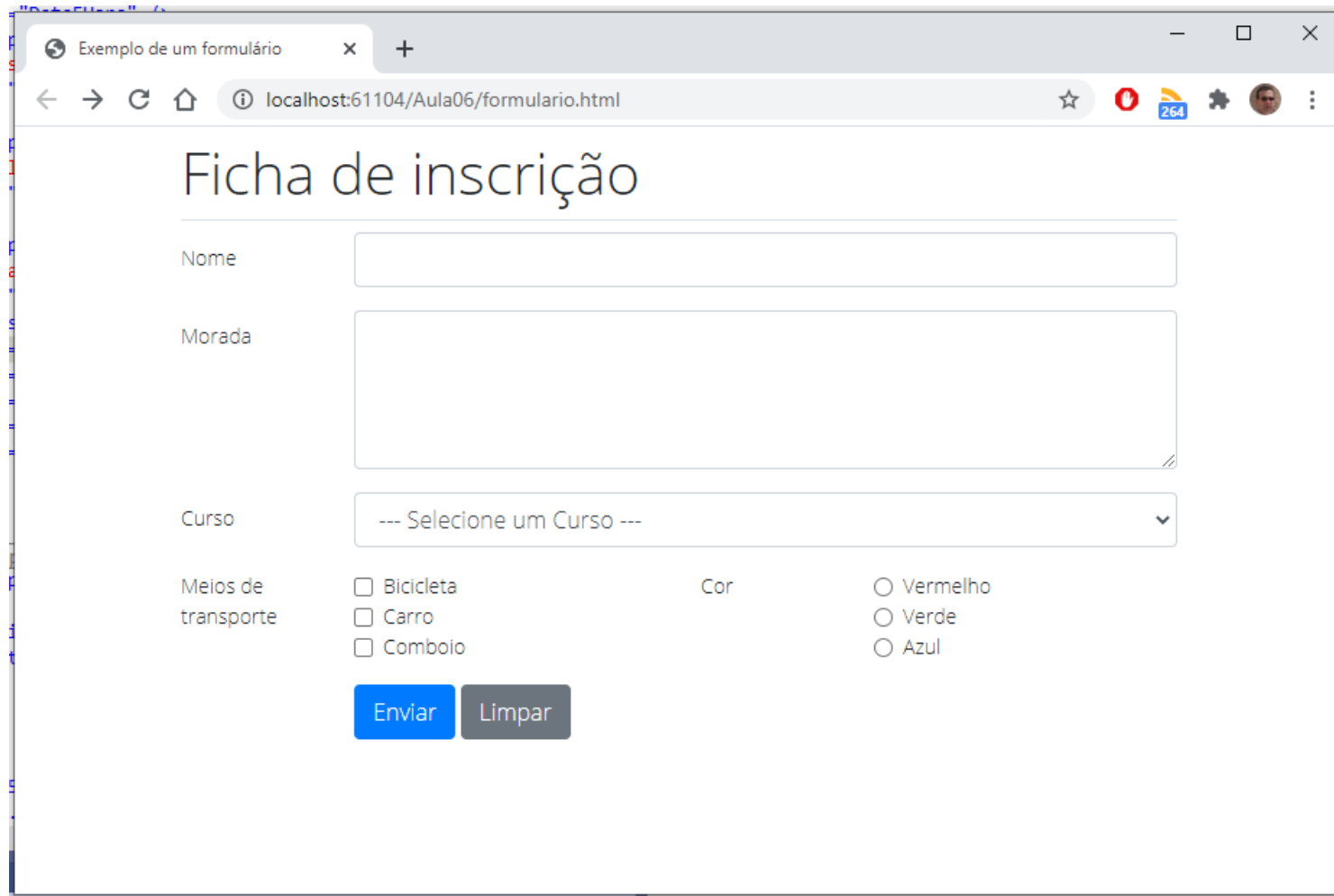
Isso torna o processo mais intuitivo e ágil, pois o utilizador é informado em tempo real sobre erros de preenchimento (por exemplo, e-mails no formato errado ou campos obrigatórios não preenchidos).

Cross-Site Scripting (ou XSS) é uma vulnerabilidade de segurança que permite a um atacante injetar scripts maliciosos em sites ou aplicações web legítimas, explorando falhas na *sanitização* de dados de entrada. Esse tipo de ataque acontece quando uma aplicação web incorpora conteúdo dinâmico, como formulários ou caixas de texto, sem validar e limpar adequadamente as entradas fornecidas pelos utilizadores.

SQL Injection é uma técnica de ataque que permite a um invasor manipular uma consulta SQL para interagir de forma maliciosa com a base de dados de uma aplicação. Esse ataque explora vulnerabilidades nas entradas de dados da aplicação (como formulários, campos de login ou URLs) que não são devidamente validadas ou *sanitizadas*.

Revisitando o exemplo da aula 2

(com a inclusão dos estilos bootstrap)

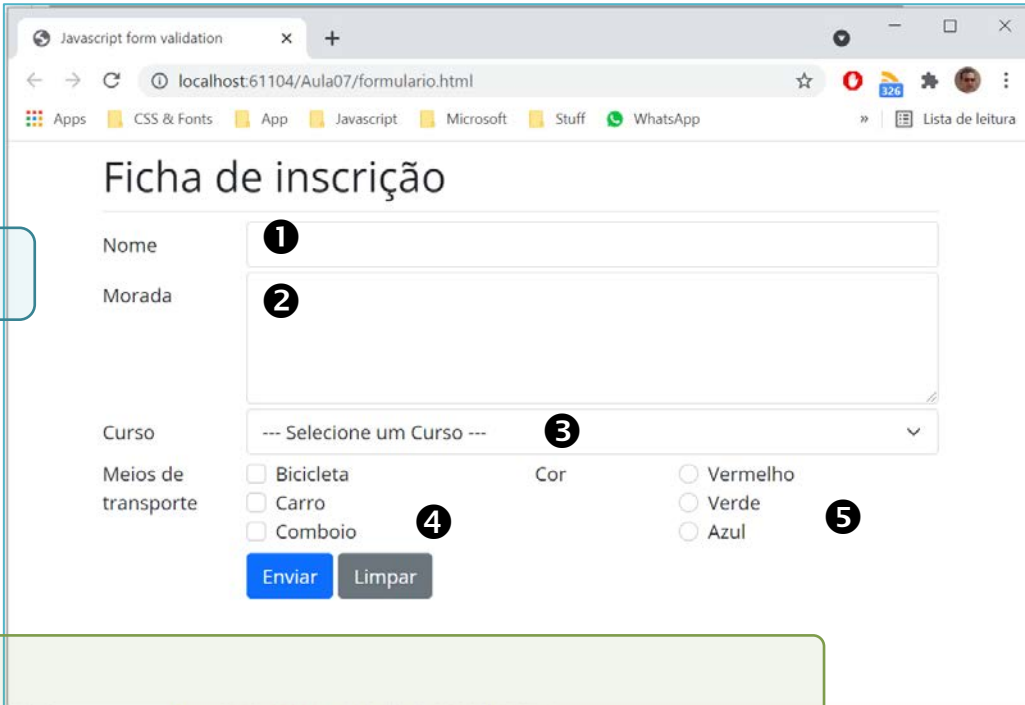


The screenshot shows a web browser window with the title 'Exemplo de um formulário' and the URL 'localhost:61104/Aula06/formulario.html'. The page displays a registration form titled 'Ficha de inscrição'. The form includes the following fields and options:

- Nome:** A text input field.
- Morada:** A text area for address.
- Curso:** A dropdown menu with the placeholder text '--- Selecione um Curso ---'.
- Meios de transporte:** A group of three checkboxes:
 - ☐ Bicicleta
 - ☐ Carro
 - ☐ Comboio
- Cor:** A group of three radio buttons:
 - ☐ Vermelho
 - ☐ Verde
 - ☐ Azul

At the bottom of the form are two buttons: 'Enviar' (blue) and 'Limpar' (grey).

```
<body>
<div class="container">
  <h1 class="border-bottom py-2 mb-2">Ficha de inscrição</h1>
  <form action="http://192.168.160.36/FormEcho.aspx" name="FichaInscricao">
    <div class="row mb-1">
      <label for="Nome" class="col-md-2 col-form-label">Nome</label>
      <div class="col-md-10"><input type="text" name="Nome" id="Nome" class="form-control" /></div>
    </div>
    <div class="row mb-1">
      <label for="Morada" class="col-md-2 col-form-label">Morada</label>
      <div class="col-md-10"><textarea name="Morada" id="Morada" class="form-control" rows="4"></textarea></div>
    </div>
    <div class="row mb-1">
      <label for="Curso" class="col-md-2 col-form-label">Curso</label>
      <div class="col-md-10">
        <select name="Curso" id="Curso" class="form-select">
          <option value="">--- Selecione um Curso ---</option>
          <option value="1">Licenciatura em Engenharia Informática</option>
          <option value="2">Mestrado Integrado em Engenharia de Computadores e Telemática</option>
          <option value="3">Mestrado Integrado em Engenharia Eletrónica e Telecomunicações</option>
          <option value="9">Outro</option>
        </select>
      </div>
    </div>
    <div class="row mb-1">
      <div class="col-md-2">Meios de transporte</div>
      <div class="col-md-4">
        <div class="form-check">
          <input type="checkbox" name="vehicle" class="form-check-input" value="Bicicleta" id="bicicleta"><label class="form-check-label" for="bicicleta">Bicicleta</label><br />
          <input type="checkbox" name="vehicle" class="form-check-input" value="Carro" id="carro"><label class="form-check-label" for="carro">Carro</label><br />
          <input type="checkbox" name="vehicle" class="form-check-input" value="Comboio" id="comboio"><label class="form-check-label" for="comboio">Comboio</label>
        </div>
      </div>
      <div class="col-md-2">Cor</div>
      <div class="col-md-4">
        <div class="form-check">
          <input type="radio" name="cor" class="form-check-input" value="Vermelho" id="vermelho"><label class="form-check-label" for="vermelho">Vermelho</label><br />
          <input type="radio" name="cor" class="form-check-input" value="Verde" id="verde"><label class="form-check-label" for="verde">Verde</label><br />
          <input type="radio" name="cor" class="form-check-input" value="Azul" id="azul"><label class="form-check-label" for="azul">Azul</label>
        </div>
      </div>
    </div>
    <div class="row mb-1">
      <div class="col-md-10 offset-md-2">
        <input type="Submit" name="submitBtn" value="Enviar" class="btn btn-primary" />
        <input type="reset" name="resetBtn" value="Limpar" class="btn btn-secondary" />
      </div>
    </div>
  </form>
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
<script>
/* Função de validação */
function validate() {
}
</script>
</body>
```



Exemplo da aula 2

(com a inclusão dos estilos bootstrap)

JavaScript form validation

localhost:61104/Aula07/formulario.html

Ficha de inscrição

Nome

Morada

Curso

Meios de transporte

☐ Bicicleta

☐ Carro

☐ Comboio

Cor

☐ Vermelho

☐ Verde

☐ Azul

Form Echo Variables

Not secure | 192.168.160.36/FormEcho.aspx?DataEHora=&Nome=&Morada=&Curs...

GET VALUES:	
DataEHora	
Nome	
Morada	
Curso	
submitBtn	Enviar

Form Echo Variables - V-1.0
© jsp / Sep2014-Nov2015

Problema

Os dados foram submetidos ao servidor sem terem sido preenchidos!

Esta situação deve ser evitada pois, tal como referido, implica que o servidor perca tempo a fazer a validação dos dados antes de fazer o processamento desejado ou, pior, pode acarretar problemas de segurança.

Se pensarmos num sistema com milhões de utilizadores, obrigaremos o servidor a uma quantidade de trabalho que poderia ser distribuída por cada um dos computadores individuais (se os clientes validassem os dados antes do seu envio – dividir o processamento deixando que cada um faça um pouquinho).

Para fazer a validação dos campos do formulário do lado do cliente, deve ser utilizado um `<script></script>` javascript de modo a garantir que os dados estão de acordo com o desejado antes de serem enviados ao servidor.

Revisão de formulários html

(agora em conjunto com javascript)

MUITO IMPORTANTE:

1. O CSS e o javascript referem-se aos elementos pelo atributo **ID**
2. Os **label's** referem-se aos elementos pelo atributo **ID**
3. Os dados dos formulários são enviados ao servidor pelo atributo **name**

Revisão de formulários html

(agora em conjunto com javascript)

Solução (2): Utilização simultânea dos atributos **name** e **ID** dos elementos a enviar para o servidor – o que é até aconselhável para se poder utilizar **label's**

```
formulário.html x HtmlPage1.html
16 <body>
17 <div class="container">
18 <h1 class="border-bottom py-2 mb-2">Ficha de inscrição</h1>
19 <form action="http://192.168.160.36/FormEcho.aspx" name="FichaInscricao">
20 <div class="row mb-1">
21 <label for="Nome" class="col-md-2 col-form-label">Nome</label>
22 <div class="col-md-10"><input type="text" name="Nome" id="Nome" class="form-control" /></div>
23 </div>
24 <div class="row mb-1">
25 <label for="Morada" class="col-md-2 col-form-label">Morada</label>
26 <div class="col-md-10"><textarea name="Morada" id="Morada" class="form-control" rows="4"></textarea></div>
27 </div>
28 <div class="row mb-1">
29 <label for="Curso" class="col-md-2 col-form-label">Curso</label>
30 <div class="col-md-10">
31 <select name="Curso" id="Curso" class="form-select">
32 <option value="">--- Selecione um Curso ---</option>
33 <option value="1">Licenciatura em Engenharia Informática</option>
34 <option value="2">Mestrado Integrado em Engenharia de Computadores e Telemática</option>
35 <option value="3">Mestrado Integrado em Engenharia Eletrónica e Telecomunicações</option>
36 <option value="9">Outro</option>
37 </select>
38 </div>
39 </div>
40 <div class="row mb-1">
41 <div class="col-md-2">Meios de transporte</div>
42 <div class="col-md-4">
43 <div class="form-check">
44 <input type="checkbox" name="vehicle" class="form-check-input" value="Bicicleta" id="bicicleta"><label class="form-check-label" for="bicicleta">Bicicleta</label><br />
45 <input type="checkbox" name="vehicle" class="form-check-input" value="Carro" id="carro"><label class="form-check-label" for="carro">Carro</label><br />
46 <input type="checkbox" name="vehicle" class="form-check-input" value="Comboio" id="comboio"><label class="form-check-label" for="comboio">Comboio</label>
```

```
16 <body>
17 <div class="container">
18   <h1 class="border-bottom py-2 mb-2">Ficha de inscrição</h1>
19   <form action="http://192.168.160.36/FormEcho.aspx" name="FichaInscricao">
20     <div class="row mb-1">
21       <label for="Nome" class="col-md-2 col-form-label">Nome</label>
22       <div class="col-md-10"><input type="text" name="Nome" id="Nome" class="form-control" /></div>
23     </div>
24     <div class="row mb-1">
25       <label for="Morada" class="col-md-2 col-form-label">Morada</label>
26       <div class="col-md-10"><textarea name="Morada" id="Morada" class="form-control" rows="4"></textarea></div>
27     </div>
28     <div class="row mb-1">
29       <label for="Curso" class="col-md-2 col-form-label">Curso</label>
30       <div class="col-md-10">
31         <select name="Curso" id="Curso" class="form-select">
32           <option value="">--- Selecione um Curso ---</option>
33           <option value="1">Licenciatura em Engenharia Informática</option>
34           <option value="2">Mestrado Integrado em Engenharia de Computadores e Telemática</option>
35           <option value="3">Mestrado Integrado em Engenharia Eletrónica e Telecomunicações</option>
36           <option value="9">Outro</option>
37         </select>
38       </div>
39     </div>
40     <div class="row mb-1">
41       <div class="col-md-2">Meios de transporte</div>
42       <div class="col-md-4">
43         <div class="form-check">
44           <input type="checkbox" name="vehicle" class="form-check-input" value="Bicicleta" id="bicicleta"><label class="form-check-label" for="bicicleta">Bicicleta</label><br />
45           <input type="checkbox" name="vehicle" class="form-check-input" value="Carro" id="carro"><label class="form-check-label" for="carro">Carro</label><br />
46           <input type="checkbox" name="vehicle" class="form-check-input" value="Comboio" id="comboio"><label class="form-check-label" for="comboio">Comboio</label>
47         </div>
48       </div>
49       <div class="col-md-2">Cor</div>
50       <div class="col-md-4">
51         <div class="form-check">
52           <input type="radio" name="cor" class="form-check-input" value="Vermelho" id="vermelho"><label class="form-check-label" for="vermelho">Vermelho</label><br />
53           <input type="radio" name="cor" class="form-check-input" value="Verde" id="verde"><label class="form-check-label" for="verde">Verde</label><br />
54           <input type="radio" name="cor" class="form-check-input" value="Azul" id="azul"><label class="form-check-label" for="azul">Azul</label>
55         </div>
56       </div>
57     </div>
58     <div class="row mb-1">
59       <div class="col-md-10 offset-md-2">
60         <input type="Submit" name="submitBtn" value="Enviar" class="btn btn-primary" />
61         <input type="reset" name="resetBtn" value="Limpar" class="btn btn-secondary" />
62       </div>
63     </div>
64   </form>
65 </div>
```

Validação de um formulário

Validação de formulário

Neste momento estão criadas as condições para a validação dos valores de um formulário antes do seu envio para o servidor.

Condições de validação do formulário:

1. *O nome tem de ter no mínimo três letras;*
2. *A morada tem de ter no mínimo três palavras;*
3. *Tem de escolher um curso*
4. *Tem de escolher (pelo menos) dois meios de transporte*
5. *Tem de escolher uma cor*

Validação de formulários

o botão de submit

Na linha do botão de submit deve ser acrescentada a instrução

`onclick="return validate()"`

Quando o botão for carregado, a função `validate()` será executada antes do envio do formulário.

se retornar `true` o formulário será enviado;

se retornar `false` o formulário não será enviado e o utilizador poderá corrigir os valores.

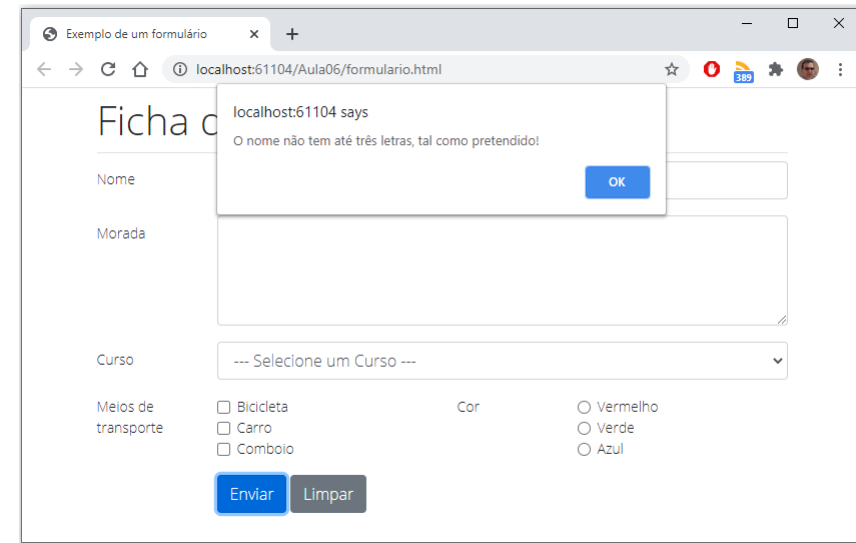
```
<form action="http://192.168.160.36/FormEcho.aspx" name="FichaInscricao">
(...)
  <div class="row mb-1">
    <div class="col-md-10 offset-md-2">
      <input type="Submit" name="submitBtn" value="Enviar" class="btn btn-primary" onclick="return validate()" />
      <input type="reset" name="resetBtn" value="Limpar" class="btn btn-secondary" />
    </div>
  </div>
</form>
```

Validação do formulário

1. O nome tem de ter no mínimo três letras

Ação: limpar os espaços e verificar o tamanho do texto inserido.

Se o formulário for submetido sem nome deve aparecer um **alert** indicando o erro.



```
<script>
  /* Função de validação */
  function validate() {
    var retVal = true; /* Vamos partir do princípio de que o formulário está válido ... */
    var _nome = document.getElementById("Nome");
    if (_nome.value.trim().length < 3) {
      retVal = false;
      alert("O nome não tem o mínimo de três letras! ");
    }
    return retVal;
  }
</script>
```


Funções e métodos utilizados

`trim()` - O método `trim()` remove os espaços em branco (whitespaces) do início e/ou fim de um texto.

É considerado espaço em branco (espaço, tabulação, espaço fixo/rígido, etc.) e qualquer sinal de fim de linha de texto (LF, CR, etc.).

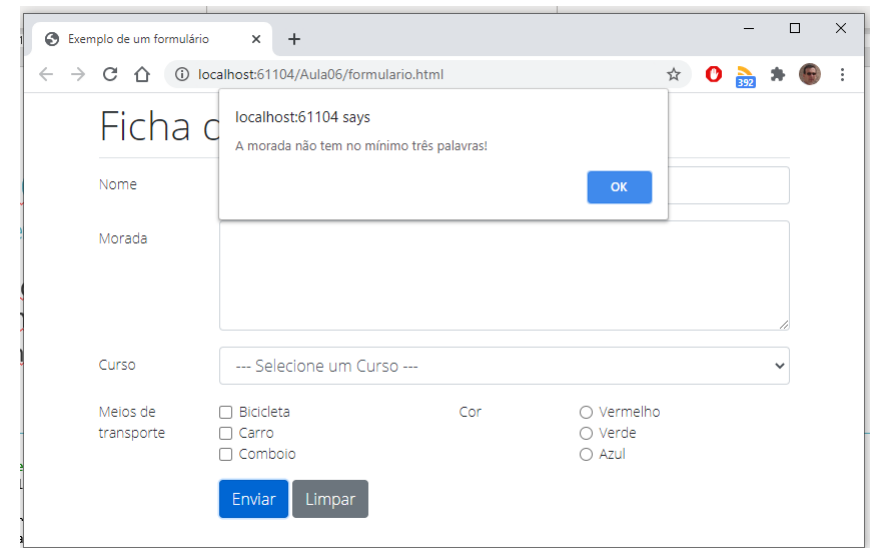
`length` - A propriedade `length` de um objeto String contém o comprimento da string.

Validação do formulário

2. A morada tem de ter no mínimo três palavras

Ação: dividir a morada em palavras e contar o número de palavras.

Se o formulário for submetido sem nome deve aparecer um **alert** indicando o erro.



```
<script>
  /* Função de validação */
  function validate() {
    (...)
    var _morada = document.getElementById("Morada");
    var palavrasArray = _morada.value.split(' ');
    if (palavrasArray.length < 3) {
      retVal = false;
      alert("A morada não tem no mínimo três palavras!");
    }
    return retVal;
  }
</script>
```

Funções e métodos utilizados

`split()` - O método `split()` divide uma `string` em uma lista ordenada de `substrings`, coloca essas substrings num `array` e retorna o array.

A divisão é feita procurando um padrão, onde o padrão é fornecido como o primeiro parâmetro na chamada do método.

`array` - Um array é um conjunto de dados. Arrays são utilizados para armazenar mais de um valor numa única variável. Isso é comparável a uma variável que pode armazenar apenas um valor.

Cada item do array tem índice numérico que permite o acesso ao "valor" armazenado na variável.

Em JavaScript, um array começa no índice zero e pode ser manipulado a partir de vários métodos.

```
const theName = 'Joaquim Sousa Pinto'
// Split the name in parts
var words = theName.split(' ');
// Log to console
console.log(words)
console.log(words.length)
-----
(3) ["Joaquim", "Sousa", "Pinto"]
3
```

```
var myArray = [1, 2, 3, 4];
var nomesArray = ["Joaquim", "Ana", "Pedro"];
// Log to console
console.log(myArray[2])
console.log(nomesArray[0])
-----
3
Joaquim
```

Validação do formulário

3. Tem de escolher um curso (opção 1)

Ação: determinar qual o índice selecionado.
Se o índice for = 0, não há curso selecionado

The screenshot shows a web browser window with the address bar displaying 'localhost:61104/Aula06/formulario.html'. The page contains a form titled 'Ficha de inscrição' with the following fields: 'Nome' (text input), 'Morada' (text area), 'Curso' (dropdown menu with the placeholder text '--- Selecione um Curso ---'), and 'Meios de transporte' (checkboxes for 'Bicicleta', 'Carro', and 'Comboio'). There are also radio buttons for 'Cor' (Vermelho, Verde, Azul) and two buttons at the bottom: 'Enviar' and 'Limpar'. A modal alert box is open, displaying the message 'localhost:61104 says: Selecione um curso!' with an 'OK' button.

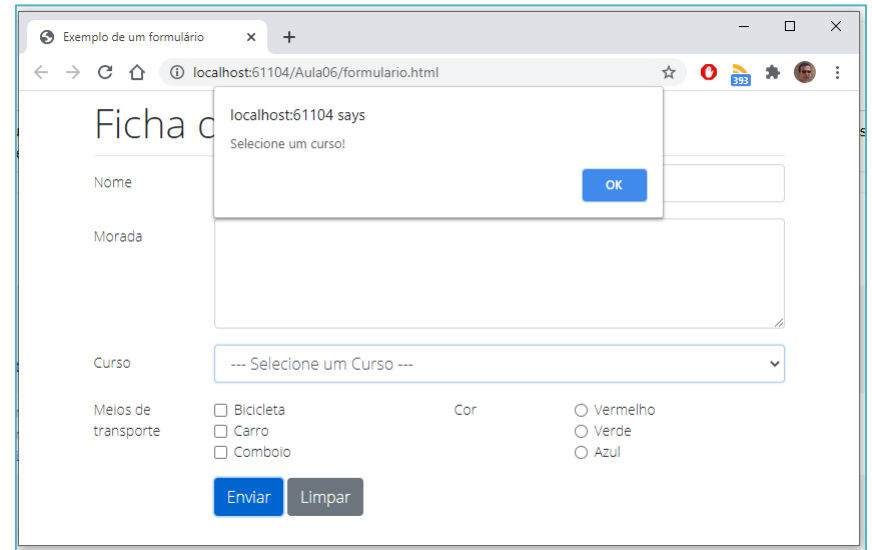
```
<script>
  /* Função de validação */
  function validate() {
    (...)
    var _cursoSelecioneado = document.getElementById("Curso").selectedIndex;
    /* Se o índice é zero, não está nenhum curso selecionado */
    if (_cursoSelecioneado == 0) {
      retVal = false;
      alert("Selecione um curso!");
    }
    return retVal;
  }
</script>
```

Validação do formulário

3. Tem de escolher um curso (opção 2)

Ação: determinar o índice selecionado e verificar nas opções o valor da opção do índice. Se for = "", não há curso selecionado

Se o formulário for submetido sem nome deve aparecer um **alert** indicando o erro.



```
<script>
  /* Função de validação */
  function validate() {
    (...)
    var _cursoSelecioneado = document.getElementById("Curso").selectedIndex;
    var cursosArray = document.getElementById("Curso").options;
    /* Se o valor da opção _cursoSelecioneado tem valor = "", não está nenhum curso selecionado */
    if (cursosArray[_cursoSelecioneado].value == "") {
      retVal = false;
      alert("Selecione um curso!");
    }
    return retVal;
  }
</script>
```

Propriedades utilizadas

`HTMLSelectElement.selectedIndex` - O `HTMLSelectElement.selectedIndex` é um longo (long) que reflete o índice do elemento `<option>` selecionado.

O valor -1 indica que nenhum elemento está selecionado.

`HTMLSelectElement.options` - A propriedade `HTMLSelectElement.options` retorna um `HTMLOptionsCollection` dos elementos `<option>` contidos no elemento `<select>`.

A propriedade `HTMLSelectElement.options` é somente de leitura.

Validação do formulário

4. Tem de escolher (pelo menos) dois meios de transporte (opção 1)

```
<script>
  /* Função de validação */
  function validate() {
    (...)
    /* Recolher todos os inputs */
    var inputElemsArray = document.getElementsByTagName("input");
    var count = 0;
    for (var i = 0; i < inputElemsArray.length; i++) {
      /* Contar quais os que são do tipo checkbox e estão seleccionados */
      if (inputElemsArray[i].type == "checkbox" && inputElemsArray[i].checked == true) {
        count++;
      }
    }
    if (count < 2) {
      retVal = false;
      alert("Selecione dois meios de transporte!");
    }
    return retVal;
  }
</script>
```

Validação do formulário

4. Tem de escolher (pelo menos) dois meios de transporte (opção 1)

```
<script>
  /* Função de validação */
  function validate() {
    (...)
    /* Recolher todos os inputs */
    var inputElemsArray = document.getElementsByTagName("input");
    var count = 0;
    for (var i = 0; i < inputElemsArray.length; i++) {
      /* Contar quais os que são do tipo checkbox e estão seleccionados */
      if (inputElemsArray[i].type == "checkbox" && inputElemsArray[i].checked == true) {
        count++;
      }
    }
    if (count < 2) {
      retVal = false;
      alert("Selecione dois meios de transporte!");
    }
    return retVal;
  }
</script>
```

Exemplo de um formulário

localhost:61104/Aula06/formulario.html

Ficha de inscrição

Nome

Morada

Curso

--- Selecione um Curso ---

Meios de transporte

☐ Bicicleta

☐ Carro

☐ Comboio

Cor

☐ Vermelho

☐ Verde

☐ Azul

Enviar Limpar

localhost:61104 says
Selecione dois meios de transporte!

OK

Validação do formulário

4. Tem de escolher (pelo menos) dois meios de transporte (opção 2)

Exemplo de um formulário

localhost:61104/Aula06/formulario.html

Ficha de inscrição

Nome

Morada

Curso

Meios de transporte

Cor

Enviar Limpar

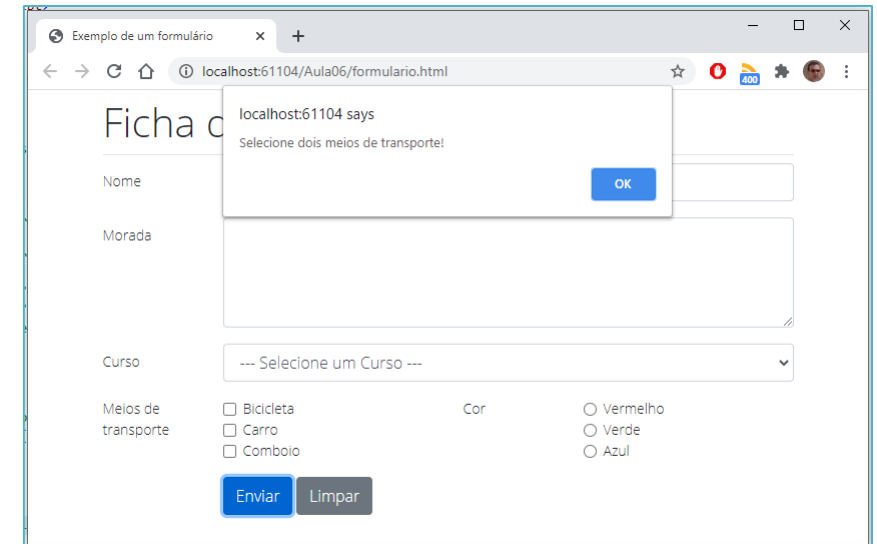
localhost:61104 says
Selecione dois meios de transporte!

OK

```
<script>
  /* Função de validação */
  function validate() {
    (...)
    /* Opção 2: recolher todos os input[type="checkbox"] que estejam checked */
    var _transportesSelecioneados = document.querySelectorAll('input[type="checkbox"]:checked').length;
    if (_transportesSelecioneados < 2) {
      retVal = false;
      alert("Selecione dois meios de transporte!");
    }
    return retVal;
  }
</script>
```

Validação do formulário

4. Tem de escolher (pelo menos) dois meios de transporte (opção 3)



```
<script>
  /* Função de validação */
  function validate() {
    (...)
    /* Opção 2: recolher todos os input[name="vehicle"] que estejam checked */
    var _transportesSelecioneados = document.querySelectorAll('input[name="vehicle"]:checked').length;
    if (_transportesSelecioneados < 2) {
      retVal = false;
      alert("Selecione dois meios de transporte!");
    }
    return retVal;
  }
</script>
```

Métodos utilizados

`Element.getElementsByTagName(tag)` - O método `Element.getElementsByTagName()` retorna uma `HTMLCollection` de elementos com o nome de tag fornecido.

`document.querySelectorAll(seletores)` - O método `document.querySelectorAll(seletores)` retorna uma `NodeList` que representa a lista dos elementos do documento que correspondem ao grupo de seletores especificado.

Validação do formulário

4. Tem de escolher uma cor

```
<script>
  /* Função de validação */
  function validate() {
    (...)
    /* Recolher todos os input[type="radio"] que estejam checked */
    var _cores = document.querySelectorAll('input[name="cor"]:checked').length;
    if (_cores == 0) {
      retVal = false;
      alert("Selecione uma cor!");
    }
    return retVal;
  }
</script>
```

Validação do formulário

Erros assinalados na interface

A validação utilizada implica que para cada falha seja apresentada uma mensagem de alert.

Esta não é a forma mais adequada. A forma mais adequada é assinalar na interface os erros.

The screenshot shows a web browser window with the title 'Exemplo de um formulário' and the URL 'localhost:61104/Aula06/formulario.html'. The page displays a registration form titled 'Ficha de inscrição' with the following fields and error messages:

- Nome:** A text input field with the error message: 'Este campo é de preenchimento obrigatório (Min: 3 letras).'.
- Morada:** A text input field with the error message: 'Este campo é de preenchimento obrigatório (Min: 3 palavras).'.
- Curso:** A dropdown menu with the placeholder text '--- Selecione um Curso ---' and the error message: 'Este campo é de preenchimento obrigatório (Escolha um curso).'.
- Meios de transporte:** A group of checkboxes for 'Bicicleta', 'Carro', and 'Comboio'. The error message is: 'Este campo é de preenchimento obrigatório (Escolha, pelo menos, dois meios de transporte).'.
- Cor:** A group of radio buttons for 'Vermelho', 'Verde', and 'Azul'. The error message is: 'Este campo é de preenchimento obrigatório (Escolha uma cor).'.

At the bottom of the form, there are two buttons: 'Enviar' (blue) and 'Limpar' (grey).

Validação do formulário

Erros assinalados na interface – o código html

```
<form action="http://192.168.160.36/FormEcho.aspx" name="FichaInscricao">
  <div class="row mb-1">
    <label for="Nome" class="col-md-2 col-form-label">Nome</label>
    <div class="col-md-10"><input type="text" name="Nome" id="Nome" class="form-control" />
    <div class="d-none text-danger col-form-label" id="NomeError"><i class="fa fa-exclamation-triangle"></i> Este campo é
de preenchimento obrigatório (Min: 3 letras).</div>
  </div>
</div>
<div class="row mb-1">
  <label for="Morada" class="col-md-2 col-form-label">Morada</label>
  <div class="col-md-10">
    <textarea name="Morada" id="Morada" class="form-control" rows="4"></textarea>
    <div class="d-none text-danger col-form-label" id="MoradaError"><i class="fa fa-exclamation-triangle"></i> Este
campo é de preenchimento obrigatório (Min: 3 palavras).</div>
  </div>
</div>
(...)
</form>
```

```
<script>
  /* Função de validação */
  function validate() {
    var retVal = true; /* Vamos partir do princípio de que o formulário está válido ... */
    var _nome = document.getElementById("Nome");
    var _nomeError = document.getElementById("NomeError");
    if (_nome.value.trim().length < 3) {
      retVal = false;
      _nomeError.classList.add("d-block");
      _nomeError.classList.remove("d-none");
    }
    else {
      _nomeError.classList.remove("d-block");
      _nomeError.classList.add("d-none");
    }
    var _morada = document.getElementById("Morada");
    var _moradaError = document.getElementById("MoradaError");
    var palavrasArray = _morada.value.split(' '); /* Partir a Morada em palavras */
    if (palavrasArray.length < 3) { /* Contar o número de palavras */
      retVal = false;
      _moradaError.classList.add("d-block");
      _moradaError.classList.remove("d-none");
    }
    else {
      _moradaError.classList.remove("d-block");
      _moradaError.classList.add("d-none");
    }
    (...)
    return retVal;
  }
</script>
```

Propriedades e métodos utilizados

`Element.classList` - O `Element.classList` é uma propriedade somente leitura que retorna uma coleção `DOMTokenList` dos atributos de classe do elemento. Isso pode ser usado para manipular a lista de classes.

`tokenList.add(token1[, token2[, ...tokenN]])` - O método `add()` da interface `DOMTokenList` adiciona o(s) token(s) fornecido(s) à lista;

`tokenList.remove(token1[, token2[, ...tokenN]])` - O método `remove()` da interface `DOMTokenList` remove o(s) token(s) fornecido(s) à lista;

`tokenList.contains(token)` - O método `contains()` da interface `DOMTokenList` retorna um Booleano - verdadeiro se a lista contém o token fornecido; caso contrário, é falso.

Exemplo:

```
_moradaError.classList.remove("d-block");  
_moradaError.classList.add("d-none");
```


Validação de formulários utilizado o bootstrap

Validação de formulários

Suponha que queremos enviar para o servidor um formulário do tipo:

Formulário

Nome José Maria	Sobrenome da Silva Pintel	Username
Cidade	Distrito Selecione um distrito	Código Postal
<input type="radio"/> Default radio 1	<input type="radio"/> Styled radio 1	Idade
<input type="radio"/> Default radio 2	<input type="radio"/> Styled radio 2	
<input type="checkbox"/> Concordo com os Termos e Condições.		
<input type="button" value="Enviar Formulário"/> <input type="button" value="Limpar Formulário"/>		



FORM VALUES:	
FirstName	José Maria
LastName	da Silva Pintel
Username	
City	
State	
Zip	
Form Echo Variables - V-1.0	
© jsp / Sep2014-Nov2015	

Se nada for feito, o formulário será enviado para o servidor mesmo que os campos estejam sem qualquer valor.

Isso traz uma sobrecarga imensa ao servidor com milhares de utilizadores, todos eles a enviar formulários por preencher, ...

Solução? Fazer a validação logo no lado do cliente de modo a evitar a sobrecarga do servidor

Validação de formulário com o bootstrap

É possível validar formulários utilizando o bootstrap.

Valida elementos do tipo:

- `<input>`s e `<textarea>`s contendo a classe `.form-control`
- `<select>`s contendo a classe `.form-select`
- `<input>`s contendo a classe `.form-checks`

Como é feita a validação?

O Bootstrap procura pelas classes `.invalid-*` e `.valid-*` filhas da classe pai `.was-validated`, geralmente aplicada ao `<form>`.

Deste modo, qualquer campo obrigatório (atributo `required` no elemento html) será validado quando for carregado o botão de envio do formulário.

Resultado da validação

Formulário

<div>Nome José Maria</div> <div>Perfeito!</div>	<div>Sobrenome da Silva Pincel</div> <div>Perfeito!</div>	<div>Username</div> <div>Insira um email válido.</div>
<div>Cidade</div> <div>Insira o nome de uma cidade (mínimo de 3 letras).</div>	<div>Distrito Selecione um distrito</div> <div>Selecione um distrito.</div>	<div>Código Postal</div> <div>Introduza um código postal válido.</div>
<div><div><input type="radio"/> Default radio 1</div><div><input type="radio"/> Default radio 2</div><div>Tem de seleccionar uma das opções.</div></div> <div><div><input checked="" type="radio"/> Styled radio 1</div><div><input checked="" type="radio"/> Styled radio 2</div><div>Tem de seleccionar uma das opções.</div></div>	<div>Idade</div> <div>Introduza um número entre 18 e 75.</div>	<div>Homepage</div> <div>Introduza uma url válida.</div>
<div><input type="checkbox"/> Concordo com os Termos e Condições. Tem de concordar com os Termos e Condições antes de submeter o formulário.</div>		
<div><div>Enviar Formulário</div><div>Limpar Formulário</div></div>		

Parte do código

```
<h3>Formulário</h3>
<form class="row g-3 needs-validation" novalidate action="http://192.168.160.36/FormEcho.aspx" method="post">
  <div class="col-md-4 form-floating">
    <input type="text" class="form-control" name="FirstName" id="validationCustom01" value="José Maria" pattern=".{3,}" required>
    <label for="validationCustom01" class="form-label">Nome</label>
    <div class="invalid-feedback">Insira o primeiro nome.</div>
    <div class="valid-feedback">Perfeito!</div>
  </div>
  <div class="col-md-4 form-floating">
    <input type="text" class="form-control" name="LastName" id="validationCustom02" value="da Silva Pincel" pattern=".{3,}" required>
    <label for="validationCustom02" class="form-label">Sobrenome</label>
    <div class="invalid-feedback">Insira o último nome.</div>
    <div class="valid-feedback">Perfeito!</div>
  </div>
  <div class="col-md-4 form-floating">
    <input type="email" class="form-control" name="Username" id="validationCustomUsername" required>
    <label for="validationCustomUsername" class="form-label">Username</label>
    <div class="invalid-feedback">Insira um email válido.</div>
  </div>
  <div class="col-md-6 form-floating">
    <input type="text" name="City" class="form-control" id="validationCustom03" pattern="[A-Z,a-z]{3,}" required>
    <label for="validationCustom03" class="form-label">Cidade</label>
    <div class="invalid-feedback">Insira o nome de uma cidade (mínimo de 3 letras).</div>
  </div>
  (...)
  <div class="col-12">
    <button class="btn btn-primary btn-sm" type="submit"><i class="fa fa-save"></i> Enviar Formulário</button>
    <button class="btn btn-danger btn-sm" type="reset"><i class="fa fa-trash-o"></i> Limpar Formulário</button>
  </div>
</form>
```

O script de validação

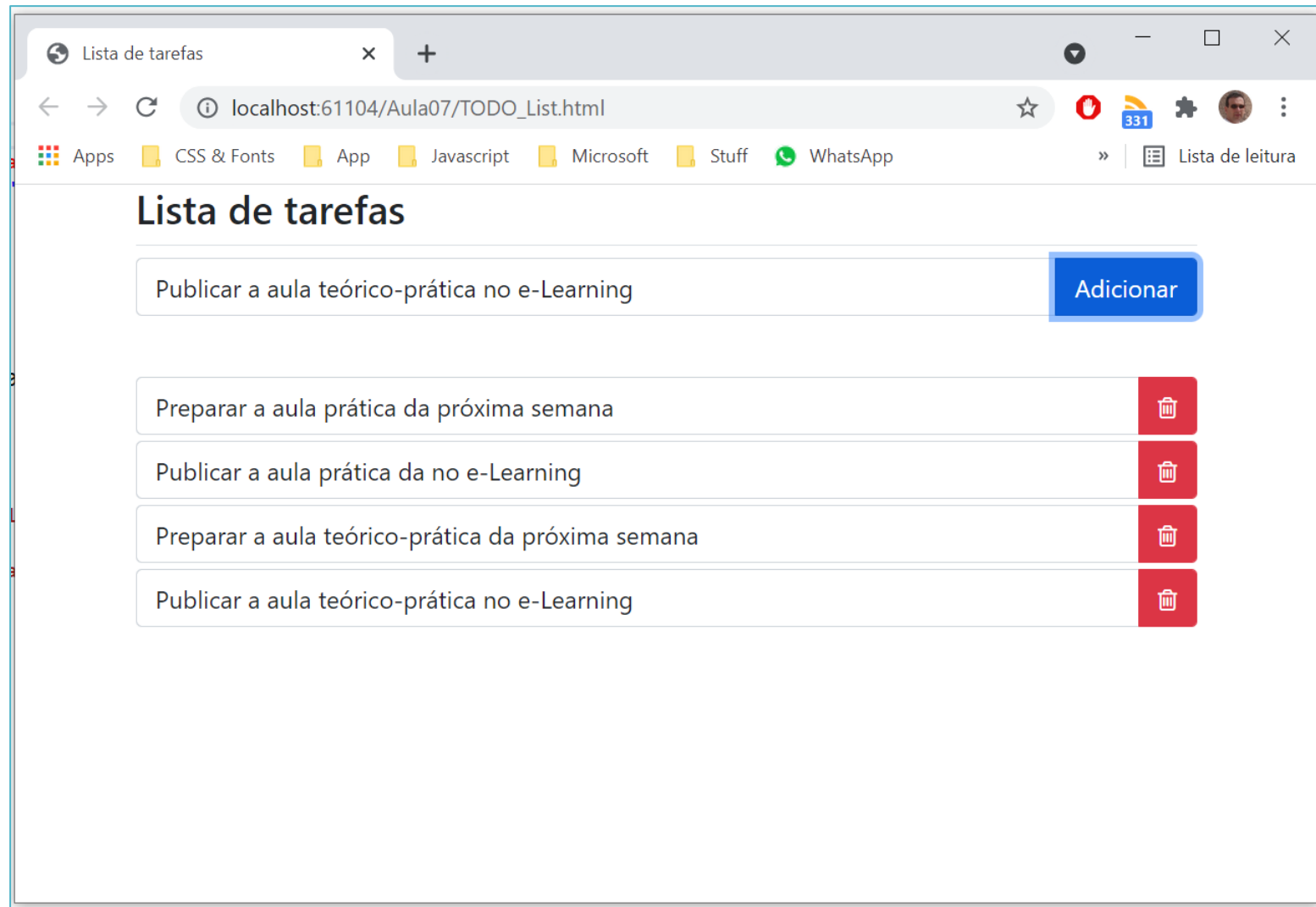
<https://getbootstrap.com/docs/5.3/forms/validation/>

```
(...)  
<!-- Bootstrap Bundle -->  
<script src="../../lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>  
<script>  
  // Example starter JavaScript for disabling form submissions if there are invalid fields  
  (function () {  
    'use strict'  
  
    // Fetch all the forms we want to apply custom Bootstrap validation styles to  
    var forms = document.querySelectorAll('.needs-validation')  
  
    // Loop over them and prevent submission  
    Array.prototype.slice.call(forms)  
      .forEach(function (form) {  
        form.addEventListener('submit', function (event) {  
          if (!form.checkValidity()) {  
            event.preventDefault()  
            event.stopPropagation()  
          }  
  
          form.classList.add('was-validated')  
        }, false)  
      })  
  })()  
</script>  
</body>  
</html>
```

Este código já existe no site do Bootstrap. Tem apenas de ser incorporado no nosso site.

Lista de tarefas

TO DO List




```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Lista de tarefas</title>
  <link href="../lib/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet" />
  <link href="../lib/font-awesome/dist/css/font-awesome.min.css" rel="stylesheet" />
  <link href="TODO_List.css" rel="stylesheet" />
</head>
<body>
  <div class="container">
    <h3 class="border-bottom pb-2 mb-2">Lista de tarefas</h3>
    <div class="input-group">
      <input type="text" class="form-control" id="newtaskId" placeholder="Tarefa a realizar...">
      <button id="push" class="btn btn-primary input-group-text" onclick="addTask()">Adicionar</button>
    </div>
    <p>&nbsp;</p>
    <div id="tasks"></div>
  </div>
  <script>
    function addTask() {
      if (document.querySelector('#newtaskId').value.length == 0) {
        alert("Por favor, indique uma tarefa.")
      }
      else {
        document.querySelector('#tasks').innerHTML +=
          '<div class="input-group mb-1">' +
          '<span id="taskname" class="form-control">' + document.querySelector("#newtaskId").value + '</span>' +
          '<button class="btn btn-danger input-group-text"><i class="fa fa-trash-o"></i></button>' +
          '</div>';
        var current_tasks = document.querySelectorAll(".btn-danger");
        for (var i = 0; i < current_tasks.length; i++) {
          current_tasks[i].onclick = function () {
            this.parentNode.remove();
          }
        }
      }
    }
  </script>
</body>
</html>

```

Javascript - Temporizadores

Javascript para fazer animações

Uma das vantagens da utilização da linguagem javascript em páginas html é permitir a criação de animações e aumentar a interatividade.

Exemplo:

Considere o seguinte código (que faz um elemento diminuir de altura até desaparecer):

```
function diminuirVertical(elemento) {  
    var altura = parseInt(elemento.style.height, 10);  
    for (; altura > 0; altura--) {  
        elemento.style.height = altura + "px";  
    }  
}
```

Problema deste exemplo: Como não há forma de controlar o tempo de execução, quanto mais rápido for o computador, mais rápido o texto desaparecerá ...

Sintaxe da linguagem Javascript

Temporizadores

Para controlar o tempo de execução do código, fazendo com que o comportamento seja igual em todos os computadores a linguagem javascript permite a utilização de **temporizadores**, que são “relógios programáveis” com uma resolução (diferença entre dois estados) de 1 milisegundo.

É assim possível ativar funções de forma periódica, sendo igualmente possível controlar o intervalo entre execuções.

No caso de animações, é possível controlar a duração da animação.

Sintaxe da linguagem Javascript

Temporizadores - Funções relevantes

`setInterval("função", intervalo)` - Define qual a **função** a ser invocada, de modo repetitivo, a cada **intervalo** de tempo.

O intervalo de tempo é expresso em milisegundos.

A função devolve um objeto para que seja possível cancelar o temporizador;

`clearInterval(temporizador)` - Apaga o **temporizador** passado no argumento;

`setTimeout("função", atraso)` - Define qual a função a ser invocada depois do **atraso** especificado, em milisegundos.

Neste caso, a função é executada apenas uma vez.

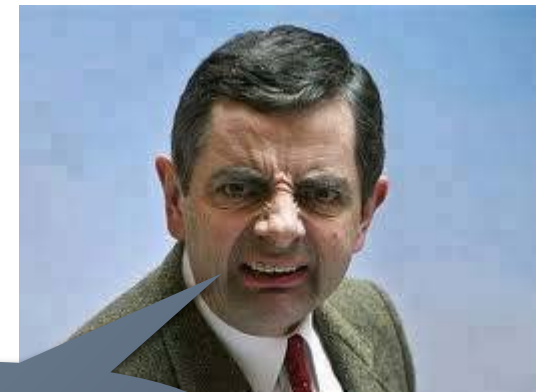
Temporizadores

Exemplo

Neste exemplo altera-se a altura de uma imagem em **10px** de cada vez que o processo é executado – a cada **100ms**.

A função além de reduzir a imagem, deteta se a altura da imagem é zero e cancela o temporizador nessa altura (evitando tamanhos negativos).

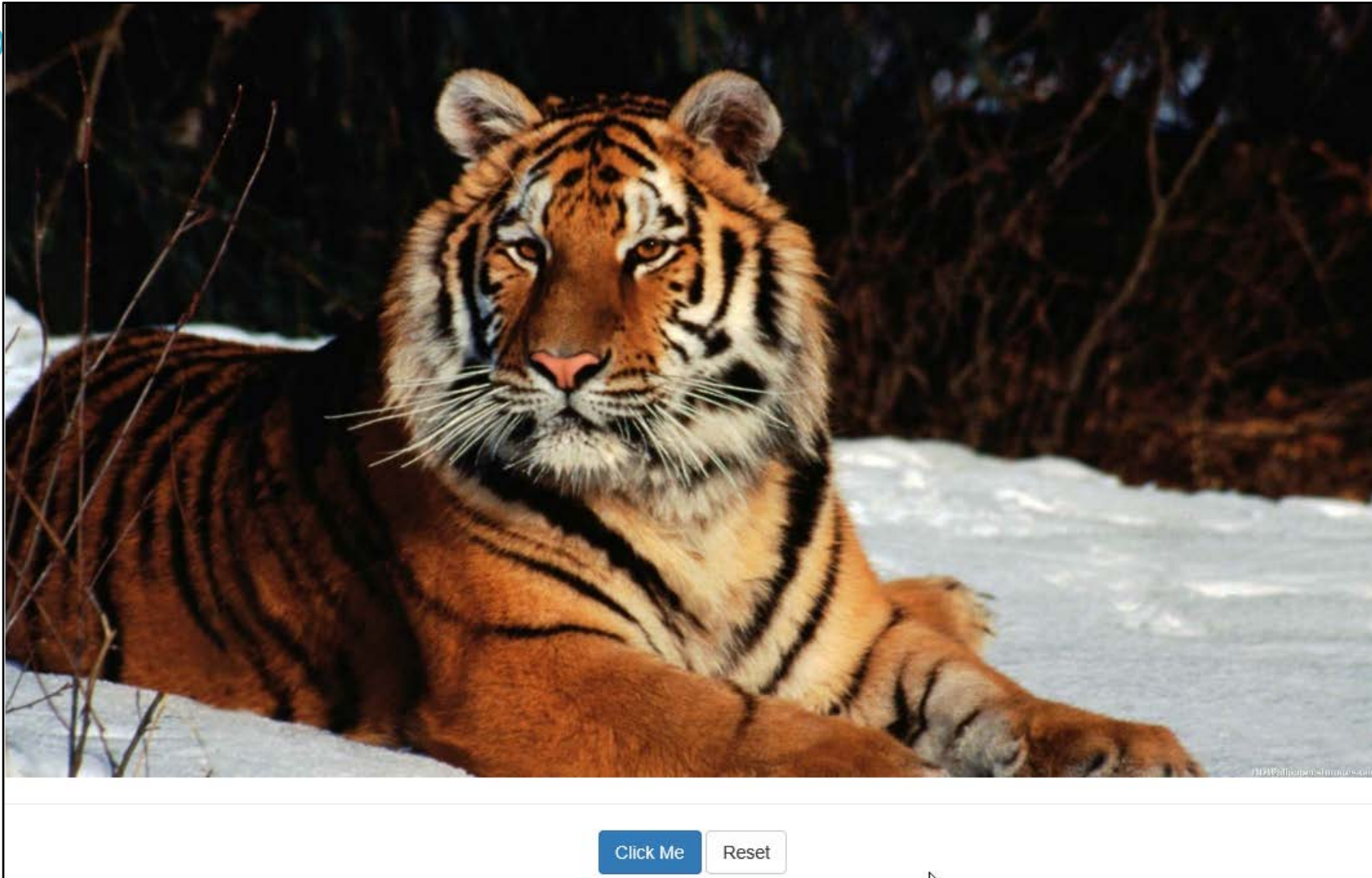
```
var temporizador = null;
function diminuirVertical(elemento) {
  if (temporizador == null) {
    temporizador = setInterval("diminuirVertical(" + elemento.id + ")", 100);
  }
  var altura = parseInt(elemento.style.height) - 10;
  elemento.style.height = altura + "px";
  if (altura == 0) {
    window.clearInterval(temporizador);
    temporizador = null;
  }
}
```



Isto está certo?

Temporizadores

Exemplo



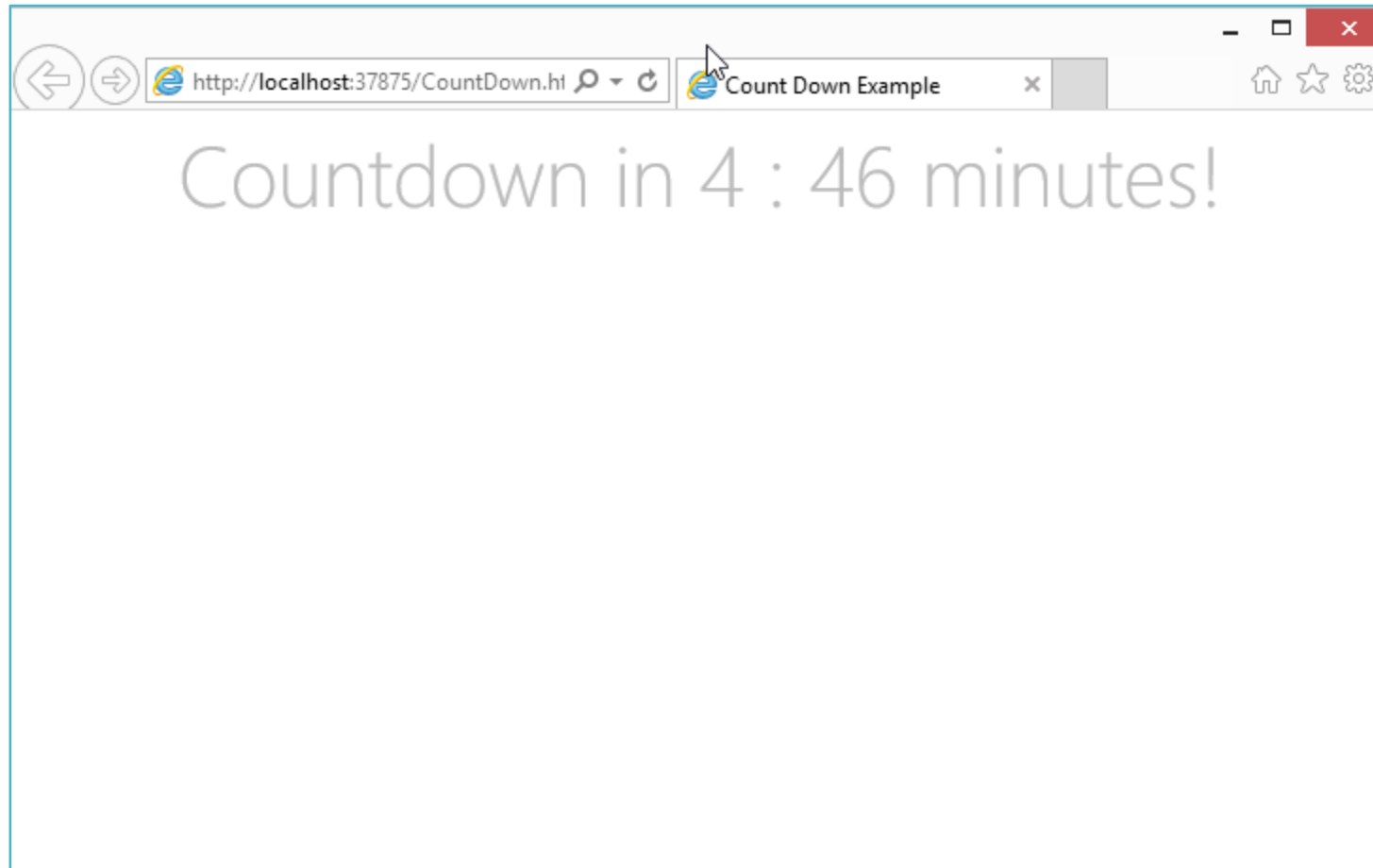
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Exemplo com temporizadores - Imagem resize</title>
  <link href="../../Content/bootstrap.min.css" rel="stylesheet" />
  <link href="../../Content/bootstrap-theme.min.css" rel="stylesheet" />
  <script type="text/javascript">
    var temporizador = null;

    function diminuirVertical(elemento) {
      if (temporizador == null) {
        temporizador = setInterval("diminuirVertical(" + elemento.id + ")", 100);
      }
      var altura = parseInt(elemento.style.height) - 10;
      elemento.style.height = altura + "px";
      if (altura <= 0) {
        elemento.style.display = "none";
        window.clearInterval(temporizador);
        temporizador = null;
      }
    }

    function resetImage(elemento) {
      elemento.style.display = "block";
      elemento.style.height = "600px";
    }
  </script>
</head>
<body>
  <div class="container">
    
  </div>
  <div class="row"><hr /></div>
  <div class="row text-center">
    <input type="button" onclick="diminuirVertical(document.getElementById('myImage'))" value="Click Me" class="btn btn-primary" />
    <input type="reset" value="Reset" class="btn btn-default" onclick="resetImage(document.getElementById('myImage'))" />
  </div>
</body>
</html>
```


Temporizadores

Exemplo 2 – Countdown - interface



Temporizadores

Exemplo 2 – o código

Isto funciona?



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Count Down Example</title>
  <link href="../Content/bootstrap.min.css" rel="stylesheet" />
  <link href="../Content/bootstrap-theme.min.css" rel="stylesheet" />
  <style type="text/css">
    #theMessage { font-family: 'Segoe UI Light'; font-size: 64px; color: #bbb; }
  </style>
  <script>
    window.onload = function () {
      var min = 5;
      var sec = 0;
      setInterval(function () {
        document.getElementById("timer").innerHTML = min + " : " + sec;
        if (sec == 0) {
          sec = 60;
          min--;
          if (min == -1) { //--- aqui acaba
            min = 4;
          }
        }
        sec--;
      }, 50); // normal: 1000 | Colocar em 50 para testar...
    }
  </script>
</head>
<body>
  <div id="theMessage" class="text-center">Countdown in <span id="timer">05:00</span> minutes!</div>
</body>
</html>
```



Para a próxima aula teórico-prática
temos teste (5 de novembro)

O que sai num teste teórico-prático?

Sai tudo o que está nos slides! (tudo, é tudo - sem exceção), história, http, html, css, javascript, etc, etc.

Concretizando:

A) um conjunto de perguntas de múltipla escolha (15-20 valores);

B) uma perguntas de desenvolvimento (0-5 valores)

“Pode sair” um exercício em que é apresentado o aspeto final e é necessário fazer o código html e o css;

“Pode sair” um exercício com um formulário e é necessário escrever a(s) instrução(ões) de validação.

Haverá mais que uma sala a fazer o teste.

O Anf V (aqui ao lado)