MÓDULO

JAVASCRIPT/AJAX

UNIDADE

FORMULÁRIOS EM JAVASCRIPT

iTMD
Instituto Tecnológico

ÍNDICE

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Compreender melhor como funcionam os formulários.
- Gerenciar não apenas o formulário, mas também os seus elementos e eventos de JavaScript.
- Criar validações de formulário.
- Criar formulários dinâmicos.

INTRODUÇÃO

Os formulários são a principal ferramenta de interação com os utilizadores. Irá começar por rever como funciona um formulário HTML e o que pode fazer com JavaScript para o controlar, validar e até mesmo criar de modo dinâmico.

Se, por exemplo, quiser construir um website que precise de validar dados de entrada com um formulário, criar pesquisas ou interagir com o utilizador, etc., irá aprender como o fazer, nesta unidade didática.

O estudo desta unidade é importante, pois vai explicar o controlo e o acesso desde o JavaScript aos formulários HTML, o que é muito útil para programar websites que, mais do que um cartão de visita, servem para interagir com o utilizador, sendo mais dinâmicos do que um simples mostruário de informações.

1. CRIAÇÃO DE FORMULÁRIOS

Esta unidade começa com a revisão do que é um formulário em HTML, como pode criá-lo e quais são as suas propriedades, elementos e eventos principais que pode controlar a partir do JavaScript.

1.1. FORMULÁRIOS EM HTML

O formulário HTML é um objeto da própria linguagem HTML declarado dentro da tag <form> e é utilizado para criar formulários de recolha de dados do utilizador. Do ponto de vista do JavaScript, um formulário é um objeto dependente do objeto document que, por sua vez depende do objeto window. A sintaxe de um formulário na página web é:

```
window.document.forms.nomeFormulario
```

Os formulários HTML têm uma série de propriedades que podem ser atribuídas ao criá-los. Estas propriedades estão dentro da tag <form> e são as seguintes:

- **name**: é o nome do formulário e deve ser exclusivo para lhe poder aceder através do JavaScript.
- **action**: é o link ou código para o qual o formulário é enviado.

- **method**: é o método de envio dos dados do formulário; os seus valores podem ser definidos por GET, que envia os parâmetros como uma string visível na URL ou por POST, que os envia de forma invisível.
- **target**: indica em qual janela ou frame os dados serão processados; por defeito o seu valor é self, o que indica que será processado na mesma página. Outro valor possível é blank, para o carregar numa nova janela.

Resta a criação de um formulário que, por exemplo, envie os dados obtidos para uma nova página PHP:

```
<form name="teste" action="pagina.php" method="POST"
target="_blank">

... conteúdo...

</form>
```

Dentro do formulário pode criar diferentes tipos de objetos para interagir com o utilizador. São eles:

- **text**, campo de texto.
- **password**, campo de texto em formato oculto.
- **hidden**, campo de texto invisível.
- **textarea**, campo de texto com várias linhas.
- **radio**, botão de seleção (circular).
- **checkbox**, caixa de marcação.
- **option**, lista desdobrável.
- **file**, ficheiros.
- **submit**, botão de envio do formulário.
- **reset**, botão reset do formulário.
- **button**, botão para eventos.
- **image**, botão tipo imagem.

1.2. CAMPOS DE TEXTO E JAVASCRIPT

Os campos de texto (text, password, hidden e textarea) possuem várias propriedades e métodos a que pode aceder pelo JavaScript para criar efeitos ou eventos no formulário.

1.2.1. PROPRIEDADES

As propriedades deste tipo de elementos são:

- **name**, nome do campo, deve ser único no formulário.
- **value**, valor do campo, não é obrigatório.
- **disabled**, permite bloquear e desbloquear o campo: se o valor for "true", o campo fica bloqueado e se for "false", desbloqueado. O seu valor padrão o seu valor é "false", mesmo que a propriedade não esteja declarada.
- **readOnly**, esta propriedade não permite modificar o valor do campo e, à semelhança da propriedade anterior, os seus valores podem ser "true" ou "false".
- **size**, é a largura visual do campo (não limita o número de caracteres).
- **length**, é uma propriedade que armazena o número de caracteres que o campo possui.
- **maxlegth**, é o número máximo de caracteres no campo; não afeta a largura que é exibida.

Para aceder a qualquer uma das propriedades destes elementos do formulário em JavaScript, basta chamar o nome do formulário, o nome do campo e o nome da propriedade, por exemplo:

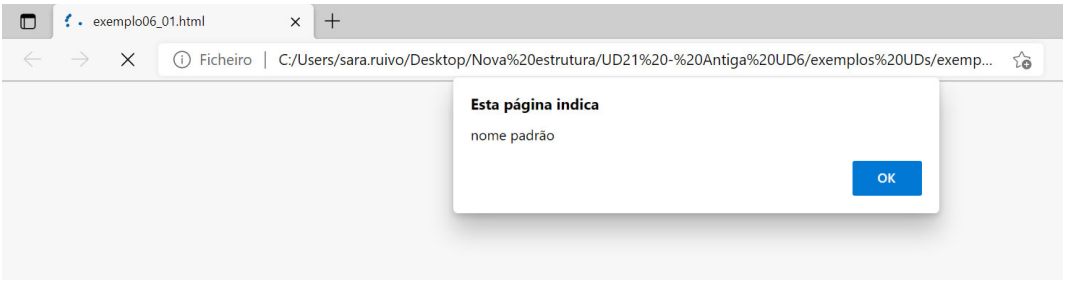
```
document.formulario.campo.propriedade
```

Exemplo 01

O exemplo a seguir mostra na página web um formulário com um elemento de texto com um valor padrão, e uma mensagem JavaScript com esse valor, usando o método de acesso.

Exemplo:

```
<html>
<head></head>
<body>
<form name="teste" method="POST" action=""/>
<input type="text" name="nome" value="nome padrão" />
</form>
<script type="text/javascript">
alert(document.teste.nome.value);
</script>
</body>
</html>
```



Visualização da página web: mensagem inicial.

1.2.2. MÉTODOS

Por outro lado, os objetos de tipo de texto também têm vários métodos a que pode aceder através do JavaScript:

- **focus**, coloca o cursor no campo.
- **select**, seleciona o conteúdo do campo de texto.
- **toUpperCase**, converte o texto do campo em maiúsculas.
- **toLowerCase**, converte o texto do campo em minúsculas.

Para aceder a estes métodos do JavaScript, utiliza-se uma sintaxe semelhante às propriedades:

```
formulario.campo.metodo()
```

Por exemplo:

```
testes.nome.focus()
```

1.2.3. EVENTOS

Os objetos de tipo de texto também têm vários eventos a que pode aceder através do JavaScript:

- **onFocus**, permite executar uma ação ao inserir o cursor no campo.
- **onBlur**, permite executar uma ação ao retirar o cursor do campo.
- **onSelect**, permite executar uma ação quando o texto do campo é selecionado.
- **onKeyUp**, permite executar uma ação quando uma tecla é pressionada e solta (enquanto o cursor estiver no campo).
- **onKeyDown**, permite executar uma ação quando uma tecla é pressionada, mas não solta (apenas uma vez).

- **onKeyPress**, permite executar uma ação quando é pressionada uma tecla e continua a executar se não a soltar.
- **onClick**, permite executar uma ação quando o campo é pressionado com o botão esquerdo do rato.
- **onChange**, permite executar uma ação quando o valor do campo muda.
- **onMouseOver**, permite executar uma ação quando o cursor passa sobre o campo.
- **onMouseOut**, permite executar uma ação quando o cursor deixa de estar sobre o objeto.

Em todos os eventos anteriores, a ação a ser lançada pode ser definida em JavaScript. Para isso, basta colocar o código desejado dentro da chamada do evento, por exemplo:

```
<input type=“text” name=“teste” onclick=“alert(‘olá!’);”/>
```

Como pode ver no exemplo, é muito fácil criar ações para os eventos; se a ação a ser executada for mais complexa, pode chamar uma função criada anteriormente em JavaScript.

Seguem-se os casos mais utilizados no JavaScript para eventos de elementos do tipo texto:

Exemplo 02

Este exemplo mostra um formulário com dois campos, em que o que é escrito no primeiro campo é duplicado no segundo.

Exemplo:

```
<html>

<head>

<script type="text/javascript">

function duplicar(valor){
```

```
teste.copia.value = valor;

}

</script>

</head>

<body>

<form name="teste" method="post" action=""/>

Nome <input type="text" onKeyUp="duplicar(this.value)" name="nome"
value="" /><br/>

Nome duplicado<input type="text" name="copia" value="" />

</form>

</body>

</html>
```

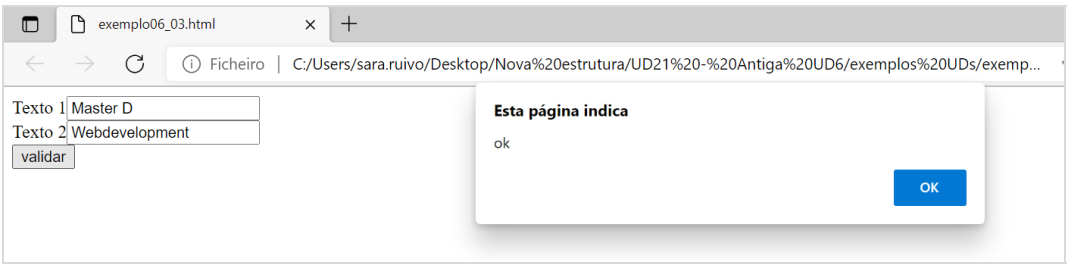
Visualização da página web: duplicação de conteúdo de campos de formulário.

Exemplo 03

No exemplo a seguir, num formulário com dois campos, verifica-se se algum texto foi inserido em ambos os campos; em caso afirmativo, é mostrada uma mensagem "ok", e se algum campo não estiver preenchido, mostra uma mensagem diferente.

Exemplo:

```
<head>
<script type="text/javascript">
  function validar() {
    if ((teste.texto1.value == '') || (teste.texto2.value == '')){
        alert('um dos campos está vazio');
    } else{
    alert('ok');
    }
  }
</script>
</head>
<body>
    <form name="teste" method="POST" action="">
      Texto 1<input type="text" name="texto1" value="" /> <br>
      Texto 2<input type="text" name="texto2" value="" /> <br>
      <input type="button" onclick="validar()" value="validar" />
    </form>
</body>
</html>
```



Visualização da página web: ambos os campos preenchidos.

Exemplo 04

Neste exemplo, será utilizado o evento onFocus dos campos para mostrar uma ajuda sobre o que escrever no campo. Eles serão exibidos numa outra caixa de texto com o tipo "ReadOnly", que estará localizada abaixo.

Exemplo:

```
<html>

<head>

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

</head>

<body>

  <form name="teste" method="POST" action="">

    Texto 1<input type="text" name="texto1" value=""
onFocus="teste.ajuda.value = 'Aqui fica o texto de ajuda do texto
1'"/><br/>

    Texto 2<input type="text" name="texto2" value=""
onFocus="teste.ajuda.value = 'Aqui fica o texto de ajuda do texto
2'"/><br>

    Ajuda :<input type="text" name="ajuda" value="" size="90"/><br>

  </form>

</body>

</html>
```

Visualização da página web, com onfocus no primeiro campo.

Exemplo 05

Este exemplo é um formulário de início de sessão no qual o utilizador e a senha são solicitados. O campo da senha ficará bloqueado até que o campo do utilizador seja preenchido.

Exemplo:

```
<html>

<head>

<script type="text/javascript">

  Function bloqueio(){

    utilizador = teste.utilizador.value;

    if (utilizador != "") { utilizador = true; }

    else{ utilizador = false; }

    if (utilizador == true) { teste.senha.disabled = false; }

    else{ teste.senha.disabled = true; }

  }

</script>

</head>


<body>

  <form name="teste" method="POST" action="" >

    Nome do utilizador: <input type="text" name="utilizador"
size="10" onKeyUp="bloqueio()"> <br>

    Senha: <input type="password" name="senha" size="10" disabled>

  </form>

</body>

</html>
```

Visualização da página web, com campo password desbloqueado.

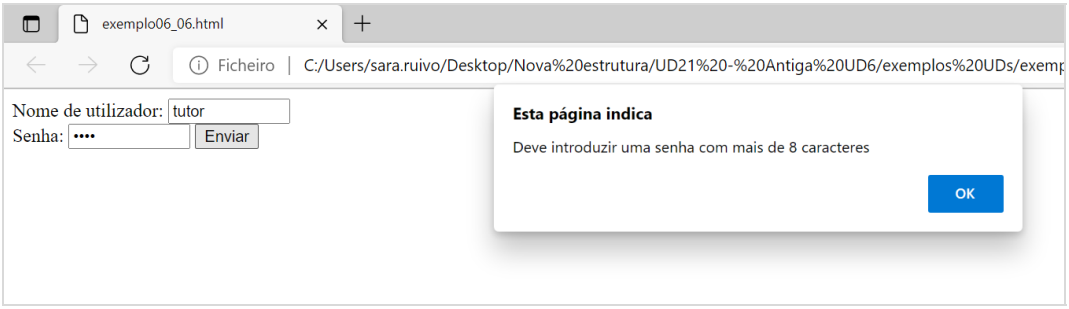
Exemplo 06

Neste exemplo, pretende-se controlar o tamanho mínimo de um campo; por exemplo, ao escrever a senha de acesso no formulário, ela tem de ter pelo menos oito caracteres.

Exemplo:

```
<html>


<head>
    <script type="text/javascript">
        function validarChave(){
            if (teste.senha.value.length < 8) {
                alert('Deve introduzir uma senha com mais de 8
caracteres');
                teste.senha.focus();
                return true;
            }
        }
    </script>
</head>


<body>
```

```
    <form name="teste" method="POSTpost" action="" >

    Nome de utilizador: <input type="text" name="utilizador"
size="10"><br>

    Senha: <input type="password" name="senha" size="10">

    <input type="button" onClick="validarChave()" value="Enviar">

    </form>

</body>


</html>
```



Visualização da página web no caso de não ser introduzida a senha de acordo com os requisitos.

Exemplo 07

O exemplo a seguir força o utilizador a escrever o texto em maiúsculas utilizando o método toUpperCase.

Exemplo:

```
<html>

<head></head>

<body>

    <form name="teste" method="POST" action="">
```

```
        Nome: <input type="text" name="nome" size="20"
onKeyUp="this.value = this.value.toUpperCase();"> <br>

    </form>

</body>

</html>
```

Visualização da página web.

1.3. ELEMENTOS COM CHECKBOX E RADIO BUTTON

Estes elementos de seleção são representados por um círculo, no caso do radiobutton, e por um quadrado, no caso da checkbox. A única diferença entre ambos, ao nível de utilização, é que a checkbox permite marcar vários elementos, enquanto o radiobutton é usado para seleções únicas (apenas um elemento pode ser marcado). Seguem-se as suas propriedades e eventos (visto que não possui métodos):

1.3.1. PROPRIEDADES

As propriedades com acesso a partir do JavaScript para este tipo de elementos são:

- **name**, nome que identifica uma caixa de seleção ou um grupo de radiobuttons.
- **value**, valor associado ao elemento; não é o valor relativo a se está marcado ou não.
- **disabled**, bloqueia o elemento.

- **checked**, indica se está marcado ou não, devolvendo "true", se estiver marcado, ou "false", se não estiver.
- **length**, só é utilizado nos radiobuttons e devolve o número de elementos com o mesmo nome (do mesmo grupo).
- **index**, só é utilizado nos radiobuttons; é um array numérico que possui tantos elementos como radiobuttons no mesmo grupo, e é acessível tal como qualquer outro array.

1.3.2. EVENTOS

Os elementos **checkbox** e **radiobutton** têm quatro eventos importantes que são muito úteis em JavaScript:

- **onFocus**, quando o cursor está sobre o elemento.
- **onBlur**, quando o cursor sai do elemento.
- **onClick**, ao clicar no elemento com o rato.
- **onChange**, quando o valor marcado/desmarcado muda.

Seguem-se alguns exemplos das ações mais usadas nestes elementos HTML:

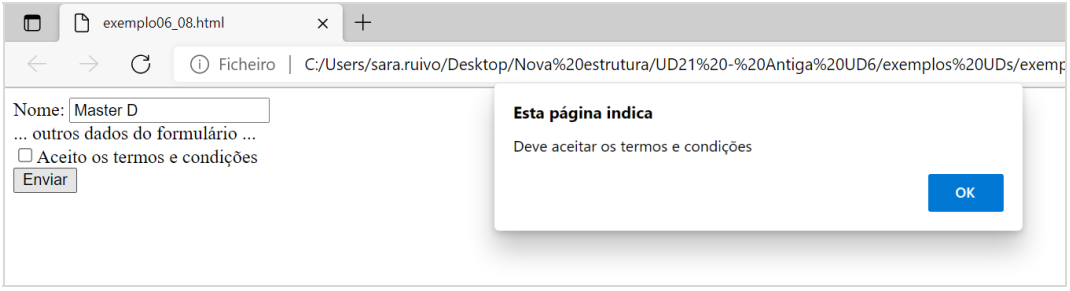
Exemplo 08

O exemplo seguinte é um formulário de registo contendo uma checkbox que aceita os termos de registo. Este formulário criará um código JavaScript para validar se a checkbox foi selecionada, devolvendo mensagens de alerta.

Exemplo:

```
<html>


<head>
    <script type="text/javascript">
        function validar(form) {
```

```
            if (form.aceite.checked == false) {
                alert("Deve aceitar os termos e condições");
                form.aceite.focus();
                return true;
            }
            alert('Aceitou os termos e as condições!');
        }
    </script>
</head>

<body>
    <form name="teste" method="post" action="" >
    Nome: <input type="text" name="nome" size="20"><br>
    ... outros dados do formulário ...<br />
    <input type="checkbox" name="aceite">Aceito os termos e
condições<br />
    <input type="button" value="Enviar"
onClick="validar(this.form)">
    </form>
</body>

</html>
```



Visualização da página web, quando não são aceites os termos e condições, ou seja, quando a checkbox não está selecionada.

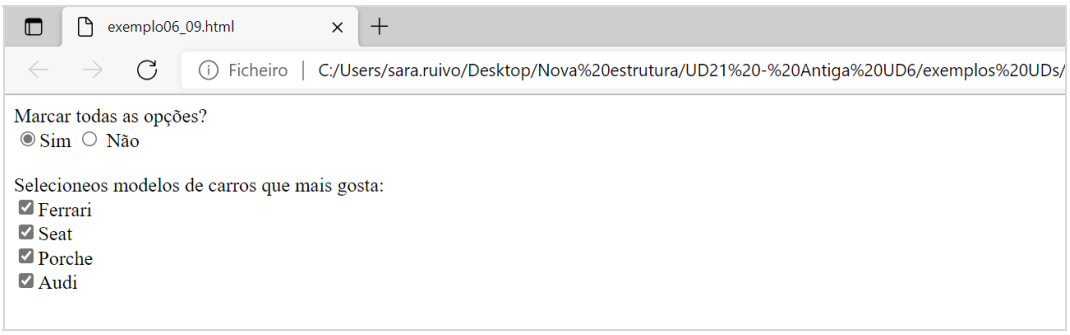
Exemplo 09

No exemplo seguinte, pretende-se utilizar um radiobutton para marcar ou desmarcar todas as checkboxes da lista posterior. Ao selecionar a checkbox "sim", todos os elementos da lista serão selecionados e ao selecionar a checkbox "não", esta seleção fica sem efeito.

Exemplo:

```
<html>


<head>
    <script type="text/javascript">
        function marca(form) {
            for (i = 0; i < form.modelos.length; i++) {
                form.modelos[i].checked = true;
            }
        }
        function desmarca(form) {
            for (i = 0; i < form.modelos.length; i++) {
                form.modelos[i].checked = false;
            }
        }
    </script>
</head>


<body>
```

```
    <form name="teste" method="postPOST" action="">

        Marcar todas as opções?<br>

        <input type="radio" name="opcao" value="sim"
onClick="marca(this.form)">Sim

        <input type="radio" name="opcao" value="nao"
onClick="desmarca(this.form)"> Não

        <br><br>

        Selecioneos modelos de carros que mais gosta:<br>

        <input type="checkbox" name="modelos"
value="Ferrari">Ferrari<br>

        <input type="checkbox" name="modelos" value="Seat">Seat<br>

        <input type="checkbox" name="modelos"
value="Porche">Porche<br>

        <input type="checkbox" name="modelos" value="Audi">Audi<br>

    </form>

</body>


</html>
```



Visualização da página web, quando a checkbox "sim" está selecionada.

1.4. OBJETOS SELECT/OPTION

Estes objetos são listas ou menus suspensos de dados que podem ser selecionados. Select é o elemento selecionado e option são as diferentes opções de lista que este contém.

1.4.1. PROPRIEDADES

Propriedades do elemento select

- **name**, nome do elemento.
- **size**, número de elementos option filhos que serão visíveis à primeira vista.
- **option**, permite o acesso a cada option filho dentro do select.
- **disabled**, permite bloquear o elemento.
- **multiple**, permite que a página de controlo selecione mais do que um subelemento de option.
- **type**, contém informações sobre se é um elemento múltiplo ou simples.

Propriedades do elemento option

- **value**, valor associado à option.
- **text**, texto mostrado pela option.
- **selected**, indica se este item está selecionado.
- **selectedIndex**, indica qual é a option selecionada.

index, é o valor que permite aceder aos dados de um select como um array.

- **length**, é o número de options dentro do mesmo select.

1.4.2. MÉTODOS

- O objeto select tem apenas um método JavaScript:
  - focus, coloca o cursor no elemento.
- Não existem métodos para o subelemento option.

1.4.3. EVENTOS

- Os eventos select acessíveis para a execução de JavaScript são:
  - **onFocus**, quando o cursor entra no elemento.
  - **onBlur**, quando o cursor sai do elemento.
  - **onChange**, quando é feita uma alteração na seleção da option.
- Não há eventos associados à option.

1.4.4. ELEMENTO DE CONSTRUÇÃO DO OPTION

Os subelementos option em JavaScript têm um sistema de criação de programação que permite construir elementos select dinâmicos. A sua sintaxe é:

```
elementoOpcao = newOption(“text”, “value”, “selecaoPadrao”,
“selecao”)
```

Uma vez que o elemento "elementoOpcao" é criado, ele é introduzido no select:

```
formulario.select.options[ultimo_index + 1] = elementoOpcao
```

Também existe a possibilidade de remover options de um select; para isso, basta utilizar a seguinte instrução:

```
formulario.select.options[index] = null
```

Exemplo 10

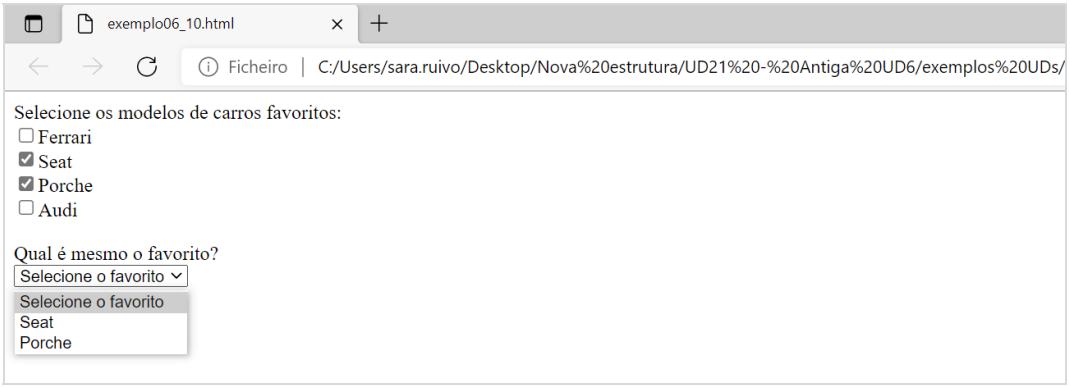
O exemplo seguinte mostra um conjunto de checkboxes com os modelos de carros favoritos. Quando um modelo é selecionado, ele deve ser inserido por baixo no select, para selecionar aqueles de que o utilizador mais gosta (todos os modelos que forem marcados).

Exemplo:

```
<html>


<head>
    <script type="text/javascript">
        function inserir(modelo) {
```

```
            opt = newOption(modelo, modelo);
            teste.favorito.options[teste.favorito.options.length] =
opt;
        }
        function eliminar(modelo) {
            for (i = 0; i < teste.favorito.options.length; i++) {
                if (teste.favorito.options[i].text == modelo) {
                    teste.favorito.options[i] = null;
                }
            }
        }
        function newOption(modelo, modelo) {
            // Criar um object Option
            var opt = document.createElement("option");

            // Assignar texto e valor
            opt.text = modelo;
            opt.value = modelo;

            return opt;
        }

    </script>
</head>

<body>
    <form name="teste" method="POST" action="">
```

```
        Selecione os modelos de carros favoritos:<br>

        <input type="checkbox" name="modelo1" value="ferrari"

            onChange="if
(this.checked){inserir(this.value);}else{eliminar(this.value);}">Fe
rrari<br>

        <input type="checkbox" name="modelo2" value="Seat"

            onChange="if
(this.checked){inserir(this.value);}else{eliminar(this.value);}">Se
at<br>

        <input type="checkbox" name="modelo3" value="Porche"

            onChange="if
(this.checked){inserir(this.value);}else{eliminar(this.value);}">Po
rche<br>

        <input type="checkbox" name="modelo4" value="Audi"

            onChange="if
(this.checked){inserir(this.value);}else{eliminar(this.value);}">Au
di<br><br>

        Qual é mesmo o favorito?<br />

        <select id="favorito" name="favorito">

            <option value="">Selecione o favorito</option>

        </select>

    </form>

</body>



</html>
```



Visualização da página web, quando são selecionados alguns modelos.

Exemplo 11

Neste exemplo existem dois selects diferentes. No primeiro aparecem diferentes países e no segundo idiomas. O segundo select estará disabled até que um valor do primeiro select seja selecionado.

Exemplo:

```
<html>


<head>
    <script type="text/javascript">
        function habilitarSelect(form) {
            if (form.paises[0].selected == true) {
                form.idioma.disabled = true;
            } else {
                form.idioma.disabled = false;
            }
        }
    </script>
</head>
```

```
<body>

    <form name="teste" method="POST" action="">

        <select name="paises"
onChange="habilitarSelect(this.form)">

            <option value="">Selecione um país</option>

            <option value="portugal">Portugal</option>

            <option value="espanha">Espanha</option>

            <option value="franca">França</option>

            <option value="alemanha">Alemanha</option>

        </select>

        <select name="idioma" disabled>

            <option value="">Idioma</option>

            <option value="portugues">Português</option>

            <option value="espanhol">Espanhol</option>

            <option value="frances">Francês</option>

            <option value="alemao">Alemão</option>

        </select>

    </form>

</body>


</html>
```

Visualização inicial da página web.

2. VALIDAÇÃO DE UM FORMULÁRIO

A validação de formulários é uma das ferramentas mais utilizadas em JavaScript, pois torna possível controlar quais os dados inseridos, se estão corretos ou não, e obter maior eficiência nos formulários, à semelhança dos formulários de uma página web que pede registo e avisa se faltarem dados ou se os dados introduzidos estiverem incorretos.

A primeira coisa a fazer é criar um controlo de envio de formulário. Para isso, será utilizado um método que consiste em passar sempre pela validação dos dados antes de enviar o formulário. Basta substituir o típico botão submit por um button que permite aceder aos eventos, mas não envia o formulário.

A este botão é ainda adicionado um evento onClick com a chamada da função de validação.

```
<input type="button" name="envio" value="Enviar"
onClick="validar(this.form);" />
```

Seguem-se os casos mais usados na validação de dados e uma explicação de como criar a função de validação.

Primeiro, é definida a função com uma variável que ajudará a saber se o formulário é válido ou não; finalmente, uma instrução if... else que permite enviar o formulário ou exibir um erro no ecrã.

```
function validar(formulario){
    var valido = ‘s’;
    var mensagem = ‘’;
    ... código ...
    if (valido == ‘s’){
      formulario.submit();
    } else{
      alert(mensagem);
    }
}
```

2.1. VALIDAÇÃO DE CAMPOS VAZIOS

O caso mais típico de validação de formulário é a validação de campos vazios, ou seja, se tiver um formulário com quatro campos e todos tiverem de ser pre-enchidos, não será permitido o envio do formulário com campos vazios. Para fazer isso, utilizará uma condição que verifica o value dos elementos de texto indicados.

```
if (formulario.nome.value == ‘’){
valido = ‘n’;
mensagem = mensagem + ‘O campo do nome não pode estar vazio \n’;
}
```

Na variável "mensagem" é criada uma mensagem para exibir, caso o campo esteja vazio.

2.2. VERIFICAÇÃO DE INTRODUÇÃO DE NÚMEROS

Outro caso bastante utilizado é o de inserir a idade ou qualquer campo apenas com valores numéricos. Para o validar, utilizará funções isNaN e parseInt:

```
idade = parseInt(formulario.idade.value);
if (isNaN(idade)){
  valido = 'n';
  mensagem = mensagem + 'Só pode inserir números \n';
}
```

2.3. VALIDAÇÃO DE CAMPOS COM COMPRIMENTO FIXO

Este caso é utilizado quando se pretende, por exemplo, que o utilizador insira o seu número de telemóvel, pois já se sabe que os números de telemóvel têm nove dígitos (excluindo o indicativo). Utiliza-se então a função "teste" e uma string para validar este tipo de dados.

A função “teste” é lançada numa string de formato ou expressões regulares que vão entre duas barras (//), passando um parâmetro, que será o valor do campo a ser testado.

Seguem-se as opções que as expressões regulares permitem:

	Significado	Exemplo	Resultado
****	Marca de carácter especial	/\$casa/	Procura a palavra “$casa”
^	Início de uma linha	/^X/	Linhas que começam com “X”
$	Fim de uma linha	/z$/	Linhas que terminam por “z”
.	Qualquer carácter	/m.r/	Pode ser “mar”, “mer”, “mjr”, “mdr”...
|	Indica opções	/(L|l|f|)ocal/	Procura “Local”, “local”, “focal”
()	Agrupa caracteres	/(casa)/	Procura “casa”
[]	Conjunto de caracteres Opcional	/escrev[aoe]/	Pode ser “escreva”, “escrevo”, “escreve”

É ainda possível adicionar modificadores a estas expressões para atuar nos possíveis resultados:

	Descrição	Exemplo	Resultado
*****	Repetir 0 ou mais vezes	/l*234/	Pode ser “234”, “1234”, “11234”...
+	Repetir 1 ou mais vezes	/a+mar/	Pode ser “amar”, “aamar”, “aaamar”...
?	1 ou 0 vezes	/a?mar/	Pode ser “amar”, “mar”
{n}	Exatamente n vezes	/a{2}traso/	Será “aatraso”
{n,}	Pelo menos n vezes	/(m){2}onte/	Pode ser “mmonte”, “mmmonte”...
{m,n}	Entre m e n vezes	/cas{1,3}a/	Será “casa”, “cassa”, “casssa”

Além das expressões anteriores, há uma série de caracteres reservados que são escritos depois de uma barra invertida (\), cuja função é indicar que estes não devem ser devolvidos nos resultados:

	Significado	Exemplo	Resultado
\b	Início ou fim da palavra	/\bver\b/	Encontra "ver de", mas não “verde"
\B	Fronteira entre não-palavras	/\Bver\B	Corresponde com "Valverde" mas não com "verde"
\d	Um dígito	/[A-Z]\d/	Não falha em "A4"
\D	Alfabético (não numérico)	/[A-Z]\D/	Falharia em "A4"
\O	Carácter nulo		
\t	Carácter ASCII 9 (tabulador)		
\f	Saltar a página		
\n	Quebra de linha		
\w	Qualquer número ou letra	[a-zA-Z0-9_]	
/\w+/			Encontra a frase em "frase.", mas não o ponto (.)
\W	O oposto de \w ([^a-zA-Z0-9_])	/\W/	Encontraria apenas o ponto (.)
\s	Carácter tipo espaço	/\sSe\s/	Encontra "Apenas Se", mas não iria encontrar "Sentir"
\S	O oposto de \s		

	Significado	Exemplo	Resultado
\cX	Carácter de controlo X	\c9	A guia
\xhh	O hexadecimal hh	/\x41/	Encontra o "A" (ASCII Hex41) na "letra A"

Portanto, a expressão regular para o número de telemóvel seria:

```
/^[0-9]{9,9}$/
```

Onde com "[0-9]" são indicados os tipos de caracteres permitidos e com "{9,9}" o número de caracteres. Neste caso, ao colocar "{9,9}" define-se que deve ter 9 caracteres, mas também podia ser definido, por exemplo, por "{1,9}",o que indicaria que tem entre 1 e 9 caracteres.

Para o exemplo, define-se na função de validação o código para verificar o campo do número de telemóvel:

```
var numero = /(^([0-9]{5,5})|^)$/;

if(!numero.test(formulario.numTel.value)) {

valido = 'n';

mensagem = mensagem + 'Número de telemóvel inválido \n';

}
```

Também será adicionada uma verificação de e-mail usando este sistema de expressão regular. No caso dos e-mails, são strings que podem ter vários caracteres iniciais, seguidos de arroba (@), posteriormente seguidos de um número indefinido de caracteres para terminar com um ponto final e outro número indefinido de caracteres, por exemplo, cursodejavascript@masterd.pt. A expressão regular para o controlo de e-mail será:

```
/^(.+\@.+\..+)$/
```

O código para a função de validação é o seguinte:

```
var email = /^(.+\@.+\..+)$/;

if(!email.test(formulario.email.value)) {

valido = 'n';

mensagem = mensagem + 'Email inválido \n';

}
```

2.4. VALIDAÇÃO DE DATAS

Para a validação das datas, irá criar uma função JavaScript externa, para a qual passará a string de data inserida pelo utilizador, perante a qual ela devolve se a função é válida ou não. Para criar a função basta utilizar o objeto Date e extrair separadamente da string indicada o dia, mês e ano para trabalhar com eles.

```
functionvalidarData(dataInd){

var data = new String(dataInd);

var dataReal= newDate();

var ano= new String(data.substring(data.lastIndexOf("-
")+1,data.length));

var mes= new String(data.substring(data.indexOf("-
")+1,data.lastIndexOf("-")));

var dia= new String(data.substring(0,data.indexOf("-")));

if (isNaN(ano) || ano.length<4 || parseFloat(ano)<1900){

return false;

}

if (isNaN(mes) || parseFloat(mes)<1 || parseFloat(mes)>12){

return false;

}
```

```
if (isNaN(dia) || parseInt(dia)<1 || parseInt(dia)>31){
return false;
}
if (mes==4 || mes==6 || mes==9 || mes==11 || mes==2) {
if (mes==2 &&dia > 28 || dia>30) {
return false;
}
}
return true;
}
```

Uma vez criada a função, insira na função de validação o código que chama esta função “validarData" para verificar se ela está correta.

```
if(!validarData(formulario.nascimento.value)){
  valido = 'n';
  mensagem = mensagem + 'Data de nascimento inválida \n';
}
```

CONCLUSÃO

A validação de formulário é uma ferramenta muito útil para interagir com o utilizador. Permite criar aplicações e páginas de contacto ou registo sem erros de conteúdo, pois torna possível a validação dos dados introduzidos antes de serem enviados.

Nesta unidade didática, também aprendeu a interagir com os elementos dos formulários para criar efeitos ou validações de campos e elementos checkbox, select, etc.

AUTOAVALIAÇÃO

1. **O elemento forms depende diretamente dos elementos HTML:**

   a) document e window.

   b) document e page.

   c) page e window.

   d) Não depende de nenhum elemento.

2. **Que propriedade permite bloquear um campo text no formulário?**

   a) disabled.

   b) value.

   c) length.

   d) maxLength.

3. **Como é que se pode aceder à propriedade de um elemento do formulário?**

   a) campo.propriedade.

   b) formulario.campo.propriedade.

   c) formulario(campo).propriedade.

   d) formulario.propriedade.campo.

4. O que é que o método toUpperCase nos elementos text permite fazer?

a) Converter o texto em minúsculas.

b) Converter o texto em maiúsculas.

c) Colocar o cursor no elemento.

d) Selecionar o conteúdo do campo.

5. Quando é que o evento onKeyUp num elemento text permite o lançamento de uma ação?

a) Quando o texto do campo é selecionado.

b) Ao passar o rato sobre o elemento.

c) Ao clicar com o rato no elemento.

d) Quando uma tecla é pressionada e solta com o cursor sobre o campo.

6. Qual é a propriedade dos elementos checkbox que informa se ela está marcada ou não?

a) selected.

b) value.

c) checked.

d) index.

7. Qual é o evento checkbox que permite que uma ação seja iniciada quando o cursor sai do elemento?

a) onFocus.

b) onOut.

c) onMouseOut.

d) onBlur.

8. Qual é o evento que permite lançar uma ação quando o valor da caixa de seleção muda de marcado para desmarcado?

a) onChecked.

b) onChange.

c) onUnChecked.

d) Não existe esse evento.

9. Qual é o método que permite enviar o formulário de JavaScript?

a) submit.

b) send.

c) formSend.

d) sendForm.

10. O que é que a propriedade selectedIndex do elemento select indica?

a) Qual é a opção selecionada.

b) O valor associado ao elemento.

c) A quantidade de options dentro de um select.

d) É uma propriedade dos subelementos option e não do select.

11. Qual é a sintaxe do elemento construtor de options dentro do select?

a) option = newOption (‘text’, ’value’, ’seleçãoPadrão’, ’seleção’).

b) option = new Select.option(‘text’, ’value’).

c) option = Option(‘text’, ’value’ , ’seleçãoPadrão’, ’seleção’).

d) option = newOption (‘text’, ’value’, ’seleção’).

12. Qual é a declaração para validar se um campo está vazio num formulário?

a) if (formulario.campo.value == ‘ ’).

b) if (formulario.campo.value != ‘ ’).

c) if (formulario.campo.value).

d) if (isEmpty(formulario.campo.value)).

13. Como é que se pode validar se um campo é numérico?

a) Convertendo o valor num número com parseInt e verificando com isNaN se é um valor numérico.

b) isNumber.

c) isNumeric.

d) if (campo.isNaN == true).

14. Quais são as expressões regulares para usar na função test?

a) \\.

b) //.

c) ^^.

d) [].

15. Qual é a expressão regular utilizada para fazer uma quebra de linha?

a) \l.

b) /n.

c) Basta clicar na tecla enter.

d) \n.

16. Na lista de expressões regulares, o que indica o símbolo "$"?

a) Início da linha.

b) Marca de caracteres especiais.

c) Fim de linha.

d) Opções.

17. Qual é o modificador que permite indicar que um carácter pode ser repetido uma ou mais vezes numa expressão regular?

a) *.

b) ?.

c) +.

d) {}.

18. O que é que o carácter reservado "\b" numa expressão regular indica?

a) Uma quebra de página.

b) Uma quebra de linha.

c) Carácter de tabulação.

d) Início ou fim da palavra.

19. O que é que a seguinte expressão regular "/^(.+\@.+\..+)$/" indica?

a) Strings do estilo (... @ ...).

b) Strings como abcde@edsfr.

c) Strings como abcde@efghi.edf.

d) A expressão está mal formada e não indica nada.

20. Qual a expressão correta para indicar numa expressão regular um conjunto de caracteres opcionais em maiúsculas?

a) [A-Z].

b) (A-Z).

c) {A-Z}.

d) \A-Z\.

SOLUÇÕES

1.	a	**2.**	a	**3.**	b	**4.**	b	**5.**	d
6.	c	**7.**	d	**8.**	b	**9.**	a	**10.**	d
11.	a	**12.**	a	**13.**	a	**14.**	b	**15.**	d
16.	c	**17.**	c	**18.**	d	**19.**	c	**20.**	a

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Para mais informações sobre os formulários em HTML consulte os seguintes websites:

- https://developer.mozilla.org/pt-BR/docs/Web/Guide/HTML/Forms
- https://www.w3schools.com/html/html_forms.asp

BIBLIOGRAFIA

- Refsnes Data (1999) "JavaScript Form Validation". Disponível em:

  https://www.w3schools.com/jS/js_validation.asp. Consultado a 4 de fevereiro de 2021.

- VV. AA. (2010). *JavaScript.* Madrid: AnayaMultimedia.