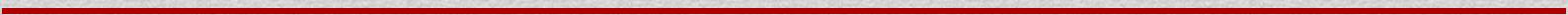


# **Web Forms 2.0**

# **Questionário de Usabilidade**



## Questionários

### ■ Objetivo

- Recolher informação demográfica e as opiniões dos utilizadores

### ■ Vantagens

- Úteis para grandes grupos e dispersos de utilizadores
- Mais rápidos que as entrevistas
- Podem ser analisados com mais rigor

### ■ Desvantagens

- Possibilidade de algumas perguntas não serem respondidas
  - Pouco flexíveis porque só obtemos as respostas às perguntas que estão no questionário
-

## Questionários de Usabilidade

### ■ SUS (System Usability Scale) [John Brooke, 1986]

- Robusto mais utilizado e recomendado
  - 1. I think that I would like to use this website frequently
  - 2. I found the website unnecessarily complex
  - 3. I thought the website was easy to use
  - 4. I think that I would need the support of a technical person to be able to use this website
  - 5. I found the various functions in this website were well integrated
  - 6. I thought there was too much inconsistency in this website
  - 7. I would imagine that most people would learn to use this website very quickly
  - 8. I found the website very cumbersome to use
  - 9. I felt very confident using the website
  - 10. I needed to learn a lot of things before I could get going with this website

## Questionários de Usabilidade: Estrutura

### ■ Estrutura

- Introdução
  - Descrever os objetivos, a informar o tempo estimado para o preenchimento, a explicar as questões de privacidade e a fazer os agradecimentos
- Dados Demográficos
  - Perguntas relacionadas com informação demográfica (género, idade, local de nascimento e outros detalhes para o estudo)
- Tarefas/Tópicos
  - Perguntas específicas de cada tarefa agrupadas em tópicos



## Questionários de Usabilidade : Questões Abertas

### ■ Resposta Livre

- Podem ser respondidas sem qualquer restrição e são utilizadas quando não existem respostas pré-determinadas
- Vantagens
  - Recolha de informação mais rica que as perguntas de escolha múltipla
- Desvantagem
  - Mais difíceis de analisar



## Questionários de Usabilidade : Questões de Escolha Múltipla

### ■ Escolha Múltipla

Género:

- Masculino       Feminino

Qual o seu sistema operativo preferido?

- Windows  
 Mac Os  
 Linux  
 Outro. Qual \_\_\_\_\_ ?



## Questionários de Usabilidade: Escala de Classificação

- **Escala de Likert**

- **Usadas para medir opiniões, atitudes, convicções e reações**

O esquema de cores da aplicação é excelente (1 significa “Discorda completamente e 5 significa “Concorda Completamente)?

1    2    3    4    5

O esquema de cores da aplicação é excelente?

- Concordo completamente
- Concordo
- Não concordo nem discordo
- Discordo
- Concordo completamente

## Formulários HTML: Web Forms

### ■ HTML Forms

- Utilizados para receber o *input* do utilizador
- Elemento **<form>** define um HTML form

text input

First name:

Last name:

radio buttons

Male  
 Female

checkboxes

Checkbox 1:   
Checkbox 2:

drop down list

Option 1  
Option 1  
Option 2  
Option 3

## Formulários HTML: Elemento “form”

Define a ação a realizar quando “form” é submetido

Em geral o “form” é submetido a uma página (“action.php” ou “action.asp”) num servidor Web.

```
<form name="fdpessoais" method="POST" onsubmit="validateForm()"  
" action="Tarefas.html">  
  <fieldset>  
    <legend> Dados Pessoais</legend>  
    <h3> Qual é a sua idade? </h3>  
    <input id = "name1" type="radio" name="idade" value=<18>  
    <label for = "name1"><18</label>  
    <input id = "name2" type="radio" name="idade" value="18-25">  
    <label for = "name2" >18-25</label>  
  
    <input type="submit" value="Next">  
  </fieldset>  
</form>
```

Dados Pessoais

Qual é a sua idade?

<18  18-25  26-33

Botão “submit” é a forma mais comum de submeter um “form” para o servidor

## Formulários HTML: Elementos

### Forms HTML

Tag	Descrição
<form>	Define um formulário HTML para receber <i>input</i> do utilizador
<input>	Define um controlo de <i>input</i>
<textarea>	Define várias linhas de entrada de texto
<label>	Define uma <i>label</i> para um elemento <input>
<fieldset>	Agrupar elementos relacionados num formulário
<legend>	Define uma legenda num formulário
<select>	Define uma lista <i>drop-down</i>
<optgroup>	Define um grupo de opções relacionadas numa lista <i>drop-down</i>
<option>	Define uma opção numa lista <i>drop-down</i>
<button>	Define um botão para interação com o utilizador

## Formulários HTML: HTML5

### ■ Elementos Adicionados no HTML5

Tag	Descrição
<datalist>	Define uma lista pre-definida de opções para controlos de <i>input</i>
<keygen>	Permite gerar um par chave-campo para formulários
<output>	Define o resultado de um cálculo

---

## Formulários HTML: Tipos de <input>

- **Elemento *input***
    - <input type="text">
    - <input type="password">
    - <input type="submit">
    - <input type="radio">
    - <input type="checkbox">
    - <input type="button">
  - **Novos valores para atributo “type” (HTML5)**
    - color
    - date, time, datetime, datetime-local
    - email
    - month, week
    - number, range, tel
    - search, url
-

## Formulários HTML: <input> (Radio Button)

```
<input id = "name1" type="radio" name="idade" value="<18">  
<label for = "name1"><18</label>
```

```
<input id = "name2" type="radio" name="idade" value="18-25">  
<label for = "name2" >18-25</label>
```

```
<input id = "name3" type="radio" name="idade" value="26-33" required>  
<label for = "name3"> 26-33 </label>
```

Qual é a sua idade?

<18    18-25    26-33

Utilizado para referenciar elementos em JavaScript. Deve ser o mesmo em todos os radio buttons pertencentes ao mesmo conjunto

Cada radio button deve ter um “value” diferente. É valor enviado quando é feito o “submit” do “form”.

### BOAS PRÁTICAS (usabilidade)

- (1) Os “radio buttons” são mutuamente exclusivos e devem ser utilizados para permitir que o utilizador escolha um opção entre várias.
- (2) Em geral, são uma boa opção quando nenhum dos outros tipos de “input” é adequado.
- (3) Devem ser apresentados na horizontal.
- (4) Não deve estar selecionada uma opção por omissão porque se o utilizador se esquecer a opção é aceite como a resposta do utilizador.

## Formulários HTML: <input> (Checkbox)

```
<input id="b1" type="checkbox" name="browser1" value="Firefox">  
<label for = "b1"> Firefox </label>
```

```
<input id="b2" type="checkbox" name="browser2" value="Opera">  
<label for = "b2"> Opera </label>
```

```
<input id="b3" type="checkbox" name="browser3" value="Safari">  
<label for = "b3"> Safari </label>
```

Deve ser diferente porque podemos escolher mais do que uma opção e é preciso identificar a opção

- Firefox
- Opera
- Safari

### BOAS PRÁTICAS (usabilidade)

- (1) Os elementos “checkbox” devem ser utilizados para permitir que o utilizador escolha várias opções de entre várias.
- (2) Não devem ser utilizados como mutuamente exclusivos para que o utilizador escolha apenas uma opção.
- (3) Devem ser apresentados na vertical e por ordem alfabética para não influenciar o utilizador.
- (4) Não deve estar selecionada uma opção por omissão porque se o utilizador se esquecer a opção é aceite como a resposta do utilizador.

## Formulários HTML: <input> (Text)

```
<form name="fdp1" method="POST" action="">
    <label for="fn"> First Name: </label>
    <input id="fn" type="text" name="firstname">
    <label for="ln"> Last Name: </label>
    <input id="ln" type="text" name="lastname">
</form>
```

```
<form name="fdp2" method="POST" action="">
    First name:
    <input type="text" name="firstname" value="Mickey">
    Last name:
    <input type="text" name="lastname" value="Mouse">
    <input type="submit" value="Submit" onsubmit="Validate" >
</form>
```

Deve-se evitar colocar texto solto

First Name:	<input type="text"/>
Last Name:	<input type="text"/>
First name:	<input type="text" value="Mickey"/>
Last name:	<input type="text" value="Mouse"/>
<input type="button" value="Submit"/>	

### BOAS PRÁTICAS (usabilidade)

(1) Texto adicional para identificar os “forms” (First Name) deve ser incluído através do elemento <label>. Mais fácil de formatar com as css e melhora a legibilidade do html. **Não se deve fazer como no 2º “form”.**

(2) “textarea” – “form” para receber texto de introduzido pelo utilizador em várias linhas.

(3) Para mais do que uma linha devem utilizar o “form” “textarea”.

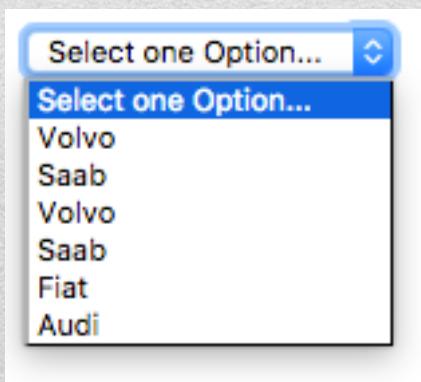
Atributo “value” é utilizado para definir o valor inicial do texto e também aceder ao texto introduzido pelo utilizador

## Formulários HTML: Select list

```
<form name="fdp5" method="POST" action="">
  <select name="cars">
    <option value="omissão">Select one Option...</option>
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
</form>
```

↑  
Opção selecionada por omissão mas que na prática não funciona como uma opção (deve ser tratado em JavaScript).

Serve para ajudar o utilizador (Regras de usabilidade)



### BOAS PRÁTICAS (Usabilidade)

(1) “Select list” ou “Dropdown list” permite que o utilizador selecione uma opção da lista tal como os “radio buttons” contudo, apresenta a lista de opções de forma mais compacta permitindo poupar espaço.

(2) Atributo “multiple” permite selecionar mais do que uma opção tal como um conjunto de elementos do tipo “checkbox”.

(3) Para um número de opções inferior a 5 devem ser utilizados “radio buttons” ou “checkboxes”.

(4) Para um número de opções superior a 15 deve ser utilizado um <input> do tipo “text”.

## Formulários HTML: <input> (Button)

```
<form name="fdp1" method="POST" action="">
    <label for="fn"> First Name: </label>
    <input id="fn" type="text" name="firstname">
    <label for="ln"> Last Name: </label>
    <input id="ln" type="text" name="lastname">

    <input type="button" value="Confirm" onsubmit="Validate">
</form>
<form name="fdp2" method="POST" action="">
    First name:
    <input type="text" name="firstname" value="Mickey">
    Last name:
    <input type="text" name="lastname" value="Mouse">

    <input type="submit" value="Confirm" onsubmit="Validate" >
</form>
```

First Name:

Last Name:

First name:

Last name:

### BOAS PRÁTICAS

(1) A diferença entre os dois botões está na submissão do formulário. Com um elemento <input> do tipo "button" não é feita a submissão do formulário nem são realizadas as validações automáticas dos "forms".

(2) Deve ser utilizado o evento "onsubmit" quando se trata de um "form".

(3) Não deve ser utilizado o "onclick". Quando a submissão do "form" é feita, por exemplo, através de uma tecla, o evento "onclick" não é ativado.

## Formulários HTML: <input> (Novos Tipos)

### Exemplos

Vários tipos de *input*

```
<form name="fdp4" method="POST" action="" onsubmit="Validate">
    <label for="i1">Select your favorite color:</label>
    <input id="i1" type="color" name="favcolor" value="#ff0000">
    <input type="submit">
</form>
```

```
<form name="fdp5" method="POST" action="" onsubmit="Validate">
    <label for = "i2">Points:</label>
    <input id="i2" type="range" name="points" min="0" max="10">
    <input type="submit">
</form>
```

```
<form name="fdp7" method="POST" action="" onsubmit="Validate">
    <label for="i1">Quantity (between 1 and 5):</label>
    <input type="number" name="quantity" min="1" max="3">
    <input type="submit">
</form>
```

## Formulários HTML: Atributos

- **Elemento <input>**
  - `disabled`, `maxlength`, `readonly`, `size`, `value`,...
- **HTML5**
  - `autocomplete`
  - `autofocus`
  - `form`
  - `Formaction`
  - `formmethod`
  - `formnovalidate`
  - `formtarget`
  - `height` and `width`
  - `list`
  - `min` and `max`
  - `multiple`
  - `pattern (regexp)`
  - `placeholder`
  - `required`
  - `step`

## Formulários HTML: Atributos - Exemplos

### Exemplos

The diagram illustrates three examples of HTML form code. Each example includes a yellow box labeled "Atributos" with arrows pointing to specific attributes used in the code.

```
<form name="a1" method="POST" onsubmit="Validate" action="">
    <input type="text" name="fname" placeholder="First name"><br>
    <input type="text" name="lname" placeholder="Last name"><br>
    <input type="submit" value="Submit">
</form>
```

```
<form name="a2" method="POST" onsubmit="Validate" action="">
    <label for = "U1">Username:</label>
    <input id="U1" type="text" name="usrname" required>
    <input type="submit">
</form>
```

```
<form name="a3" method="POST" onsubmit="Validate" action="">
    <label for = "C1">Country code:</label>
    <input id="C1" type="text" name="country_code" pattern="[A-Za-z]{3}" title="Three letter country code">
    <input type="submit">
</form>
```

## Formulários HTML: Validação

- **Tarefas Típicas de Validação**
  - Verificar se o utilizador preencheu todos os campos “`required`”
  - Validar dados (e.g., validar uma data)
  - Verificar se o utilizador introduziu texto em campos numéricos
- **Validação no Servidor**
  - Realizada pelo Web Server depois de enviado o `input` para o servidor
- **Validação no Cliente**
  - Realizada pelo Web Browser antes de enviar o `input` para o web server



## Formulários HTML: Validação por Atributos

Atributo	Descrição
disabled	Não é permitido o <i>input</i> dados
max	Indica o valor máximo de entrada
min	Indica o valor mínimo de entrada
pattern	Define um padrão para os dados de entrada
required	É obrigatório preencher este campo
type	Define o tipo de elementos de entrada

---

## Formulários HTML: Validação – Propriedades e Métodos DOM

Métodos	Descrição
checkValidity()	Retorna “true” se um elemento <i>input</i> tem dados válidos
setCustomValidity()	Atribui a propriedade “validationMessage” de um elemento

Propriedades	Descrição
patterMismatch	“true” se os dados não seguem o padrão
rangeOverflow	“true” se os dados são maiores que o definido pelo <i>max</i>
typeMismatch	“true” se o valor do elemento é válido para o <i>type</i>
valueMissing	“true” se um elemento ( <i>required</i> ) não tem valor
valid	“true” se um valor de um elemento é válido

## Formulário HTML: Questões (1)

### ■ Exemplos 1

2 - Indique, por ordem de preferência, os Web Browsers que mais utiliza. Pode indicar 3. Utilize valores de 1 a 3, sendo 1 o mais preferido e 3 o menos.

Browsers	1ºP	2ºP	3ºP
Chrome	●	●	●
Firefox	●	●	●
Internet Explorer	●	●	●
Opera	●	●	●
Safari	●	●	●

3 - Conhece outros sites de pesquisa de imagem?

Não     Sim. Em caso afirmativo, Quais?

## Formulários HTML: Atributos – “value”, “checked”

### ■ Valor de um elemento “form”

```
function FirstOption() {  
    let fop = document.forms["fdpessoais"]["First"].value;  
    let sop = document.forms["fdpessoais"]["Second"].value;  
    let top = document.forms["fdpessoais"]["Third"].value;  
    if (fop==sop) {  
        document.getElementById("s"+sop).checked = false;  
    }  
    if (fop==top) {  
        document.getElementById("t"+top).checked = false;  
    }  
}
```

Valor colocado no atributo “name” do *form*

Valor colocado no atributo “name” do *input*

Valor da opção selecionada pelo utilizador

Verificar ou selecionar uma opção de um conjunto de “radio buttons” ou “checkboxes” selecionada

## Formulário HTML: Questões (2)

### ■ Exemplos 2

4 - Indique, por ordem de preferência, os Web Browsers que mais utiliza. Deve indicar 3, sendo a opção 1 o de maior preferência e a opção 3, o de menor preferência. \*

Opção 1 Firefox

Opção 2 Chrome

Opção 3 Opera

Selezione uma opção  
Chrome  
**Firefox**  
Internet Explorer  
Opera  
Safari

5 - Conl

pesquisa de ímagens? \*

## Formulários HTML: Select (onchange)

```
<label for="primeiro">Opção 1</label>
<select id="primeiro" name="q4"
onchange="checkBrowser(this)" required>
    <option value="">Selecione uma opção</option>
    <option value="0">Chrome</option>
    <option value="1">Firefox</option>
    <option value="2">Internet Explorer</option>
    <option value="3">Opera</option>
    <option value="4">Safari</option>
</select>
```

Utiliza-se o evento Onchange que é ativado quando é alterada a opção selecionada.  
“this” – serve para identificar o elemento que está a chamar a função “checkBrowser()”

## Formulários HTML: *checkBrowser()*

```
function checkBrowser(elemento) {
    let opcao1 = document.getElementById("primeiro");
    let opcao2 = document.getElementById("segundo");
    let opcao3 = document.getElementById("terceiro");

    if ((elemento.id.localeCompare("primeiro") !== 0) && (elemento.value ===
opcao1.value)){
        opcao1.value= "";
    }

    if ((elemento.id.localeCompare("segundo") !== 0) && (elemento.value ===
opcao2.value)){
        opcao2.value= "";
    }

    if ((elemento.id.localeCompare("terceiro") !== 0) && (elemento.value ===
opcao3.value)){
        opcao3.value= "";
    }
}
```

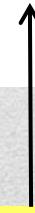
(1) Esta função identifica o “select” e depois verifica se a opção já está escolhida em outro “select”.

(2) Utiliza o atributo “value” para alterar a opção escolhida.

## Formulário HTML: Questões Mistas

5 - Conhece outros sites de pesquisa de imagens? \*

Não  Sim  Quais?



(1) - “disable” inicialmente.

(2) - Se o utilizador escolher “Sim” deve ficar “enable”.

(3) - Se o utilizador escolher “Não” é preciso limpar o texto e fazer “disable” novamente

## Formulários HTML: Atributo “Disabled”

- “Disabled”

```
function Write_Text() {  
    let x = document.forms["fdpessoais"]["pimg"].value;  
    if (x == "Não") {  
        document.forms["fdpessoais"]["outros_pimg"].disabled  
=true;  
        document.forms["fdpessoais"]["outros_pimg"].value="";  
    } else {  
        document.forms["fdpessoais"]["outros_pimg"].disabled  
=false;  
    }  
}
```

Atributo “disabled=true” -> elemento não activo  
Atributo “disabled=false” -> elemento activo



## Web Storage



## Web Storage: LocalStorage

- **LocalStorage**
  - API que permite guardar valores localmente (*Browser*) indexados por uma chave que se mantém até que sejam apagados
- **Objeto e Métodos Principais**
  - `localStorage` (Objeto disponível através do objeto `window`)
    - `setItem()` (Método para guardar no LocalStorage)
    - `getItem()` (Método para ir buscar ao LocalStorage)
    - `removeItem()` (Método para ir apagar um item no LocalStorage)
    - `clear()` (Método para ir apagar o LocalStorage)



## Web Storage: LocalStorage -.setItem

### setItem

Após o preenchimento do formulário o evento “onsubmit” do *form* deve validar, recolher e guardar os dados no localStorage. Não esquecer do botão “submit” do *form*

```
function validateForm() {  
    console.log(document.forms ["fdpessoais"] ["sexo"].value);  
    let d = new Date();  
    localStorage.setItem(d.getTime(),  
document.forms ["fdpessoais"] ["sexo"].value);  
}
```

Guardar os valores no  
localStorage

Para identificar os dados, devemos escolher um chave única (key).  
Aqui estou a utilizar a data e hora do momento em que o utilizador valida os  
dados

## Web Storage: LocalStorage - getItem

- **getItem**

```
function getdataForm() {  
    let todo_index = window.localStorage.length;  
    for (let i = 0; i < todo_index; i++) {  
        let info =  
            window.localStorage.getItem(window.localStorage.key(i));  
        console.log(info);  
    }  
}
```

Função para ir buscar os  
valores no localStorage

## Web Storage: LocalStorage – removeItem e clear

- **removeItem**
  - `window.localStorage.removeItem(key);`
- **clear**
  - `window.localStorage.clear();`



## Web Storage: LocalStorage – Escrita do Dados em XML

### XML no LocalStorage

```
function validateForm() {  
    let d = new Date();  
  
    let xmlRowString = "<Questionario>";  
  
    let Rq= document.forms["fdpessoais"]["idade"].value;  
    xmlRowString += '<q id="q1">' + Rq + '</q>';  
  
    let Rq= document.forms["fdpessoais"]["sexo"].value;  
    xmlRowString += '<q id="q2">' + Rq + '</q>';  
  
    xmlRowString += "</Questionario>";  
    window.localStorage.setItem(d.getTime(), xmlRowString);  
}
```

Gravar no localStorage a resposta à 1<sup>a</sup> pergunta

Gravar no localStorage a resposta à 2<sup>a</sup> pergunta

## Web Storage: LocalStorage – Leitura dos Dados em XML

### Leitura dos Dados do LocalStorage em XML

```
function getdataForm() {  
    let todo_index = window.localStorage.length;  
    for (let i = 0; i < todo_index; i++) {  
        let localStorageRow =  
            window.localStorage.getItem(window.localStorage.key(i));  
        if (window.DOMParser) {  
            let parser = new DOMParser();  
            let xmlDoc=parser.parseFromString(localStorageRow,"text/xml");  
        }  
    }  
    let x = xmlDoc.getElementsByTagName("q");  
    document.write("<p> Respostas dadas no Questionário: Utilizador 1  
    </p>");  
    document.write("<p> Idade:" + x[0].childNodes[0].nodeValue + "</p>");  
    document.write("<p> Género:" + x[1].childNodes[0].nodeValue + "</p>");  
}
```

Utilizamos o objeto “DOMParser” para interpretar a *string* xml como se fosse um ficheiro xml

Utilizamos as funções do DOM para navegar e aceder aos dados na árvore XML