Tecnologias Web

Semana 7

[Jaime Martins](#) • [Mário Saleiro](#)

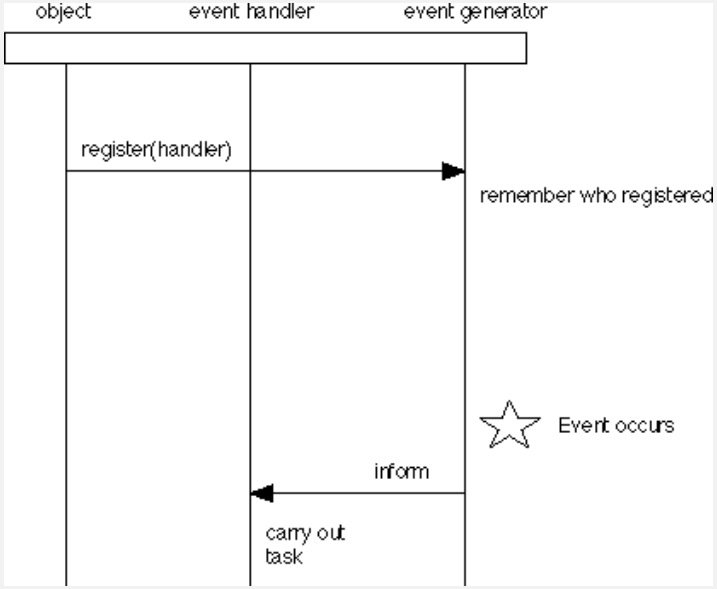
Instituto Superior de Engenharia

Universidade do Algarve

2023/24

JavaScript

Eventos, programação por eventos

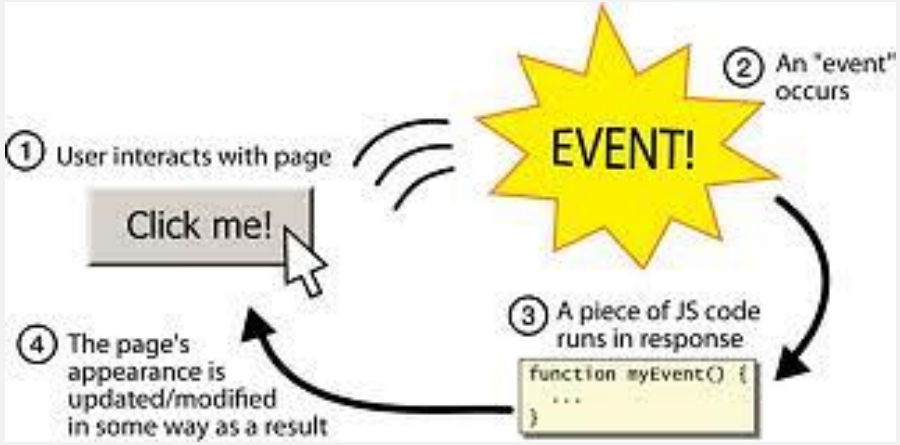


http://csis.pace.edu/~bergin/Java/eventfundamentals.html

JavaScript

Eventos

Click do rato
Premir uma tecla
Load da página
Load de uma imagem
Mov. rato sobre elemento
Campo de entrada é alterado
Submissão de um formulário



jQuery **React**
Vue.js **Angular** **Svelte**

consultar: http://www.w3schools.com/jsref/obj_window.asp

JavaScript

Atributos evento

```
<a href="http://www.ualg.pt" onclick="alert(this.href);
                                      return false">Click Here</a>
```

```
const link = document.getElementsByTagName("a")[0]

link.onclick = function clickHandler() {
  alert(this.href)
  return false
};
```

False!

JavaScript

Contexto do atributo evento

*Ponteiro **this***

```
function whatDidIClickOn() {
  console.log("You clicked on ", this)
  return false
}

const links = document.getElementsByTagName("a")

for (let i = 0, link; link = links[i]; i++) {
   link.onclick = whatDidIClickOn
}
```

Console (ferramentas do programador)
You clicked on <a href="http://www.ualg.pt">

JavaScript

Métodos dos eventos

.addEventListener(eventType, handler, capture)

Alguns *eventType:* **submit, mouseover, keypressed, click, load**

```
<p id="para">Click here</p>

const para = document.getElementById("para")
para.addEventListener("click", eventHandler, false)
para.addEventListener("click", otherEventHandler, false)

function eventHandler(event) {
  alert(this.nodeName + "clicked")
}

function otherEventHandler(event) {
  alert("Evento " + event.type + ".")
}
```

"P clicked" e "Evento click."

HTML DOM Event Object (w3schools.com)

JavaScript

Objecto `event`

Propriedades (atributos)

- **type** Tipo de evento
- **target** Elemento que emitiu o evento. Pode não ser o mesmo elemento ao que o manipulador de eventos está associado.
- **currentTarget** Elemento ao qual está associado ao evento.
- **eventPhase** Numero que indica se o evento está na fase de captura (1), bubbling (3), ou se atingiu o elemento (2).
- **timeStamp** O numero de segundos em que o evento ocorreu.

Metodos

preventDefault() Não permite a execução definida por defeito. Por exemplo: links e submissão de forms.

JavaScript


```
<a id="link" href="http://www.google.com">Don't go to
Google</a>

const link = document.getElementById("link")
link.addEventListener("click", dontFollowLink, false)

function dontFollowLink(event) {
 alert("Not going to " + this.href)
 event.preventDefault()
}
```

.stopPropagation() Impede a propagação do evento pela estrutura do DOM. Mecanismo de captura e *bubbling*

JavaScript

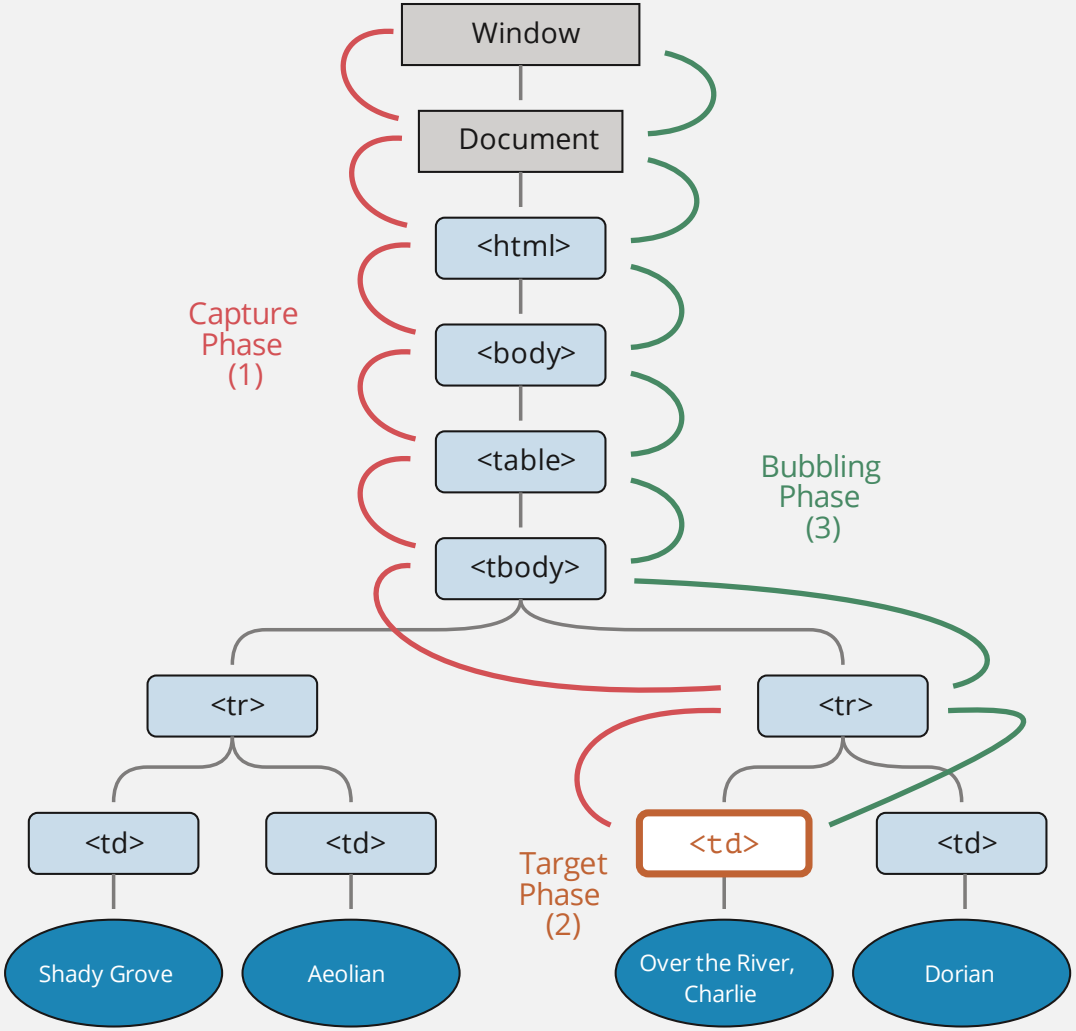
.removeEventListener(eventType, handler, capture)

```
<h1 id="header">Click this multiple times</h1>

const header = document.getElementById("header")

header.addEventListener("click", selfRemovingEvent, false)

function selfRemovingEvent(event) {
  alert("Listening for the event and removing myself")
   this.removeEventListener(event.type, selfRemovingEvent, false)
}
```

consultar: http://docstore.mik.ua/orelly/webprog/jscript/ch12_01.htm
http://docstore.mik.ua/orelly/webprog/jscript/ch19_01.htm

JavaScript

Captura e **Bubbling** do objecto `event`



JavaScript

Captura e Bubbling do objecto event

Bubbling

```
<html>
    <head>
        <title> Exemplo </title>
    </head>
    <body>
        <div>
            <p>
                <a href="http://www.google.com">Click</a>
            </p>
        </div>
        <script> Codigo javascript</script>
    </body>
</html>
```

JavaScript

Código javascript

```
const div = document.getElementsByTagName("div")[0]
const p = document.getElementsByTagName("p")[0]
const a = document.getElementsByTagName("a")[0]

document.addEventListener("click", whatNode, false)
document.body.addEventListener("click", whatNode, false)

div.addEventListener("click", whatNode, false)
p.addEventListener("click", whatNode, false)
a.addEventListener("click", whatNode, false)

function whatNode(event) {
    console.log(this.nodeName)
    event.preventDefault()
}
```

CSS (??)

```
body, div, p {
  padding: 10px;
}
```

BREAK TO
SEE THINGS
HAPPENING!!
false = bubbling

JavaScript

Captura

```
document.addEventListener("click", whatNode, true)
document.body.addEventListener("click", whatNode, true)

div.addEventListener("click", whatNode, true)
p.addEventListener("click", whatNode, true)
a.addEventListener("click", whatNode, true)

function whatNode(event) {
  console.log(this.nodeName, "phase: " + event.eventPhase)
  event.preventDefault()
}
```

Output:

```
#document phase: 1
BODY phase: 1
DIV phase: 1
P phase: 1
A phase: 2
A phase: 2
P phase: 3
DIV phase: 3
BODY phase: 3
#document phase: 3
```

BREAK TO
SEE THINGS
HAPPENING!!
true = captura

JavaScript

Suspensão propagação evento (na captura)

```
function whatNode(event) {
  if (this.nodeName === "DIV") {
    event.stopPropagation()
  }
  console.log(this.nodeName, "phase: " + event.eventPhase)
  event.preventDefault()
}
```

Eventos associados ao BOM

.onload window.onload, também para <img>

~~**.unload** window.unload~~ (usar **.visibilitychange**)

.beforeunload window.beforunload

.resize window.resize

JavaScript

Eventos associados ao rato

clientX, clientY posição do rato no *viewport* da janela. Origem topo esquerdo da janela.

pageX, pageY posição do rato na página. Com possível deslocamento da página. Origem topo esquerdo da janela.

screenX, screenY posição da janela browser. Origem topo esquerdo da janela do desktop

offsetX, offsetY posição do rato no elemento de accionou o evento. Origem topo esquerdo da área do elemento.

altkey, shiftkey, ctrlkey verifica se alguma das teclas está pressionada

button, detail, relatedTarget, mousedown, mouseup, dblclick, mouseover, mouseout

click, accionado depois de um mousedown e mouseup. Também poder ser acc. com tecla *enter* (usando tab para chegar ao elemento)

JavaScript

Eventos associados ao teclado

keydown qualquer tecla faz disparar este evento.

keypress só teclas passiveis de impressão. Alt, shit, ctrl e enter não accionam este evento.

keyup evento accionado quando o utilizador liberta a tecla e depois do keypress e keypress.

Eventos associados ao <form>

change o evento é accionado quando o utilizador altera o conteúdo de uma das entradas do <form>

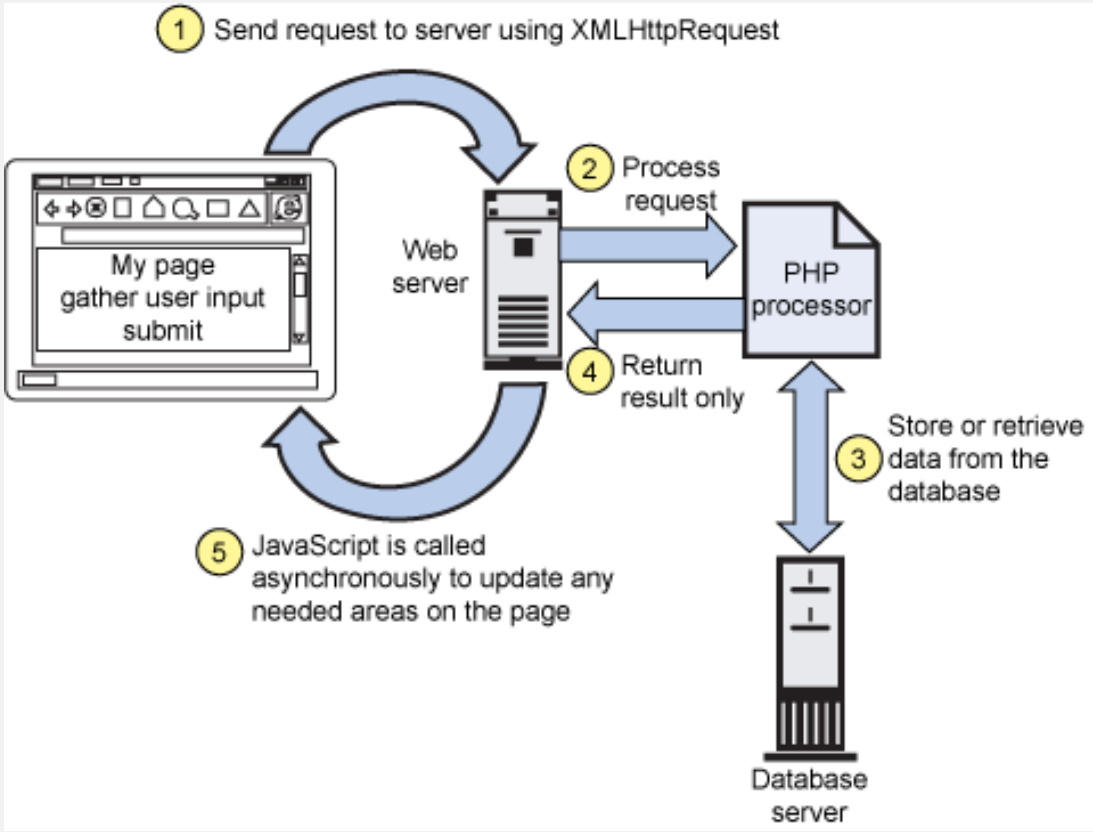
submit o evento só é accionado ao submeter-se o conteúdo do <form>. É muito utilizado para validação de entradas

Outros eventos: **focus** e **blur** (associados ao rato e teclado, nas entradas do elemento <form>)

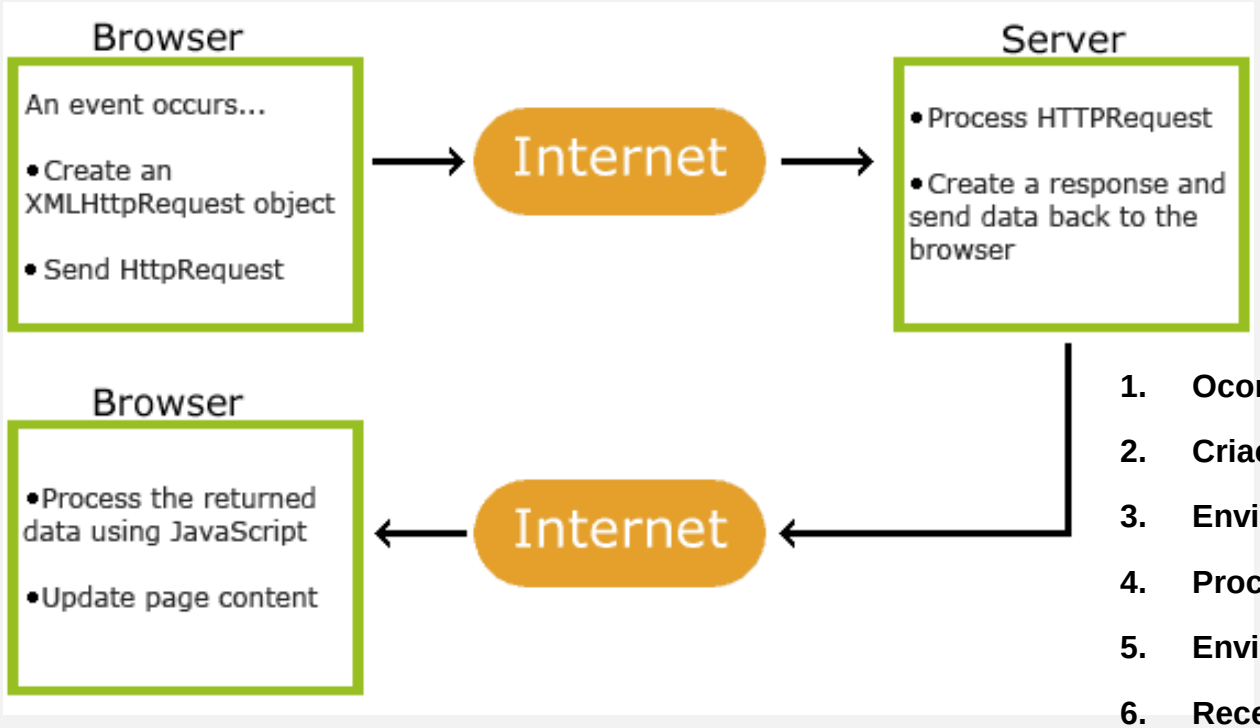
JavaScript com AJAX

AJAX?

Asynchronous **J**avascript **a**nd **X**ML



JavaScript com AJAX



1. **Ocorrência de evento**
2. **Criação objeto XMLHttpRequest**
3. **Envio objeto XMLHttpRequest**
4. **Processamento da solicitação HTTP**
5. **Envio da resposta ao cliente**
6. **Receção e actualização da página por javascript**

Servidor diferente ?

http://www.youtube.com/watch?v=FQ5nj1u2e4c

-WebSockets- ?

http://www.html5rocks.com/pt/tutorials/websockets/basics/

JavaScript com AJAX

Criação, preenchimento do objecto AJAX

```
const xhr = new XMLHttpRequest()
xhr.open("GET", "escalões.txt", false)
```

Envio do objecto AJAX

```
xhr.send()
```

true para solicitação sincrona

Receção e processamento objecto AJAX

```
xhr.onload = function () {
    if (this.status == 200) {
        // obtivemos uma resposta!
        alert(request.responseText)
    } else {
        // algum erro
        alert('Error - ' + request.status + ': ' + request.statusText)
    } // end if
}
```

JavaScript and AJAX for Dummies, Andy Harris, Wiley Publishing, Inc.

https://www.w3schools.com/js/js_ajax_intro.asp

JavaScript com AJAX

Table 1-1 Useful Members of the XMLHttpRequest Object

Member	*Description*	*Basket analogy*
`open(protocol, URL, synchronization)`	Opens up a connection to the indicated file on the server.	Stands under a particular window.
`send(parameters)`	Initiates the transaction with given parameters (or null).	Releases the basket but hangs on to the string.
`status`	Returns the HTTP status code returned by the server (200 is success).	Checks for error codes ("window closed," "balloon popped," "string broken," or "everything's great").
`statusText`	Text form of HTTP status.	Text form of status code, a text translation of the numeric error code returned by status.
`responseText`	Text of the transaction's response.	Gets the contents of the basket.
`readyState`	Describes current status of the transaction (4 is complete).	Is the basket empty, going up, coming down, or here and ready to get contents?
`onReadyStateChange`	Event handler. Attach a function to this parameter, and when the `readyState` changes, the function will be called automatically.	What should I do when the state of the basket changes? For example, should I do something when I get the basket back?

JavaScript com AJAX


```
<html>
    <head>
        <script>

            // código javascript + AJAX a ser colocado

        </script>
    </head>
    <body>
        <div id="myDiv">
            <h2>Escalões: </h2>
        </div>
        <button type="button" onclick="loadStrDoc()">Change Content</button>
    </body>
</html>
```

JavaScript com AJAX

Código JavaScript com AJAX

```
function loadStrDoc() {
    const xhr = new XMLHttpRequest()
     xhr.open('GET', 'escaloes.txt', true)

    xhr.onload = function () {
        if (this.status == 200) {
            document.getElementById('myDiv').innerHTML =
                xhr.responseText
        }
    }

    xhr.send(null)
}
```

Pode ser função explicita!!

JavaScript com AJAX

Código JavaScript com AJAX (solução do ex. 3 da aula prática 6)

```
<!DOCTYPE html>
<html>

<head>
    <title>Exemplo de Formulário</title>
    <script>
        function loadOptions () {
            const xhr = new XMLHttpRequest()
            xhr.onload = function () {
              if (xhr.status === 200) {
                const options = xhr.responseText.split('\n')
                const select = document.getElementById('escaloes')
                options.forEach(function (option) {
                  const optionElement = document.createElement('option')
                  optionElement.text = option
                  select.add(optionElement)
                })
              }
            }
            xhr.open('GET', 'http://localhost/escaloes.txt', true)
            xhr.send()
        }
    </script>
</head>
```

JavaScript com AJAX


```
<body onload="loadOptions()">
    <form>
        <label for="name">Nome:</label>
        <input type="text" id="name" name="name"><br><br>

        <label for="address">Morada:</label>
        <input type="text" id="address" name="address"><br><br>

        <label for="escaloes">Escalões:</label>
        <select id="escaloes" name="escaloes"></select><br><br>

        <input type="submit" value="Submeter">
    </form>
</body>

</html>
```

Web APIs – Javascript Fetch API

Definição e Utilização - O método `fetch()` inicia o processo de busca de um recurso no servidor e devolve uma *Promise* que resulta num objeto *Response*.

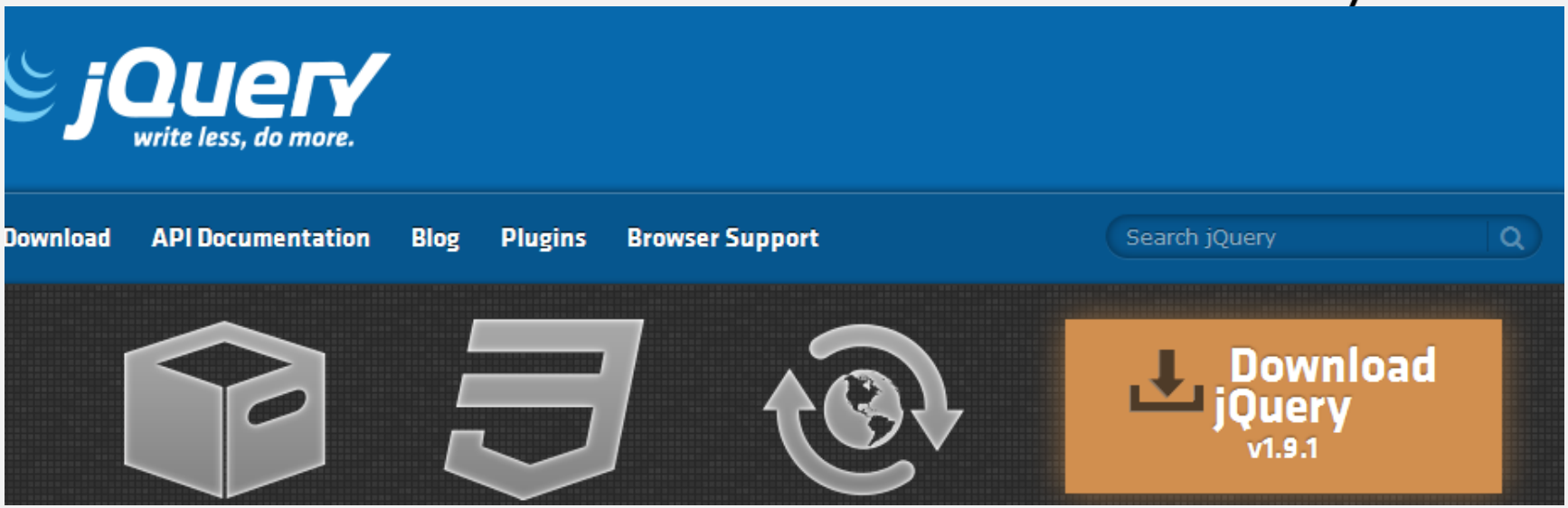
```
fetch(file)
.then(x => x.text())
.then(y => myDisplay(y))
```

```
async function getText(file) {
  let x = await fetch(file)
  let y = await x.text()
  myDisplay(y)
}
```

```
async function getText(file) {
  let myObject = await fetch(file)
  let myText = await myObject.text()
  myDisplay(myText)
}
```

😀 **No need for XMLHttpRequest anymore.**

JQuery



http://jquery.com/download/

Ver capitulo 10

JavaScript and AJAX for Dummies, Andy Harris, Wiley Publishing, Inc.

JQuery

Inclusão do JQuery (no servidor local)

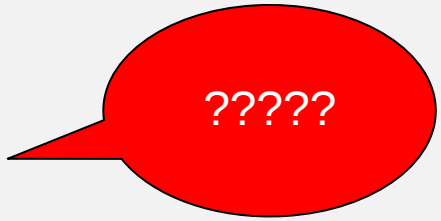
```
<script src="jquery-3.7.1.min.js"></script>
```

Inclusão do JQuery (através do servidor oficial)

```
<script src="https://code.jquery.com/jquery-
3.7.1.min.js"></script>

<script>
  //your JS + jQuery code here

</script>
```

JQuery – Exemplo 1


```
<html>
    <head>
        <meta http-equiv="content-type" content="text/xml; charset=utf-8"/>
        <title>change.html</title>
        <script src="jquery-3.7.1.min.js"/>
        <script>
            function changeMe(){
                $('#output').html("I've changed")
            }
        </script>
    </head>
    <body onload="changeMe()">
        <h1>Basic jQuery demo</h1>
        <div id="output">
            Did this change?
        </div>
    </body>
</html>
```




```
const jqOutput = $('#output')
const domOutput = document.getElementById('output')

jqOutput.html("I've changed") //jQuery
domOutput.innerHTML = "I've changed" //Javascript
```

JQuery

Mais alguns exemplos de jQuery

Manipulation:

`$('p').text('Hello, World!')` sets the text content of all <p> elements to "Hello, World!".

`$('div').html('<h1>Welcome</h1>')` sets the HTML content of all elements to <h1>Welcome</h1>.

`$('input').val('John')` sets the value of all <input> elements to "John".

CSS and Attributes:

`$('p').css('color', 'red')` sets the color of all <p> elements to red.

`$('img').attr('src', 'image.jpg')` sets the src attribute of all <img> elements to "image.jpg".

Event Handling:

`$('button').click(function() { ... })` attaches a click event handler to all <button> elements.

`$('form').submit(function(event) { event.preventDefault(); ... })` attaches a submit event handler to a <form> element and prevents the default form submission.

JQuery – Exemplo 2


```
<html>
    <head>
        <meta http-equiv="content-type" content="text/xml; charset=utf-8"/>
        <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
        <script>
            $(init)  // same as $(document).ready(init)

            function init(){
              $("li").hover(border, noBorder)
            } // end init

            function border(){
              $(this).css("border", "1px solid black")
            }

            function noBorder(){
              $(this).css("border", "0px none black")
            }
```

JQuery – Exemplo 2


```
    </script>
      <title>hover.html</title>
  </head>
  <body>
      <h1>Hover Demo</h1>
      <ul>
          <li>alpha</li>
          <li>beta</li>
          <li>gamma</li>
          <li>delta</li>
      </ul>
  </body>
</html>
```

Perguntas?

👋 Goodbye JavaScript!