Cruzeiro do Sul Educacional

jQuery, Persistência e Requisições Assíncrona

Conteudista
Prof. Me. Marco Antonio Sanches Anastacio

Revisão Textual
Prof. Me. Claudio Brites

OBJETIVOS DA UNIDADE

- Entender o que é a biblioteca jQuery e como utilizá-la;
- Compreender os principais conceitos associados à biblioteca, aplicando os conhecimentos adquiridos na construção de páginas *web*;
- Conhecer e aplicar o armazenamento por meio do uso da api *WebStorage*.

Atenção, estudante! Aqui, reforçamos o acesso ao conteúdo *on-line* para que você assista à videoaula. Será muito importante para o entendimento do conteúdo.

Este arquivo PDF contém o mesmo conteúdo visto *on-line*. Sua disponibilização é para consulta *off-line* e possibilidade de impressão. No entanto, recomendamos que acesse o conteúdo *on-line* para melhor aproveitamento.

Introdução ao jQuery

O jQuery é uma biblioteca *JavaScript* criada por John Resig e disponibilizada como *software* livre e aberto, ou seja, de código *open source*, fornecida gratuitamente. Sua biblioteca é utilizada para adicionar interatividade e dinamismo às páginas *web* como, por exemplo, com funções de navegação, criação de animações e efeitos, funções para registrar/tratar eventos e funções para desenvolver aplicações AJAX.

A biblioteca jQuery nada mais é do que um arquivo *JavaScript* que deverá ser incluído na página em que serão aplicados os efeitos desejados. Esse arquivo pode ser adicionado a partir de um arquivo público remoto conhecido como CDN, ou a partir de um arquivo local, incluído na pasta do projeto. Observe o exemplo a seguir que faz uso da biblioteca jQuery a partir de um repositório remoto:

```
<html>
<head>
        <title>JQuery</title>
        <script src="https://code.jquery.com/jquery-3.6.4.js"></script>
</head>
</body>
        <h1>Aprendendo jQuery</h1>
<body>
</html>
```

Leitura

O que é uma CDN?

Seletores e Operadores

A finalidade do jQuery é controlar o comportamento de uma página *web* e, para tanto, é necessário encontrar os elementos HTML que constituem a página e

terão seu comportamento modificado. Nesse sentido, a sintaxe do jQuery é bastante simples, veja a Figura 1.

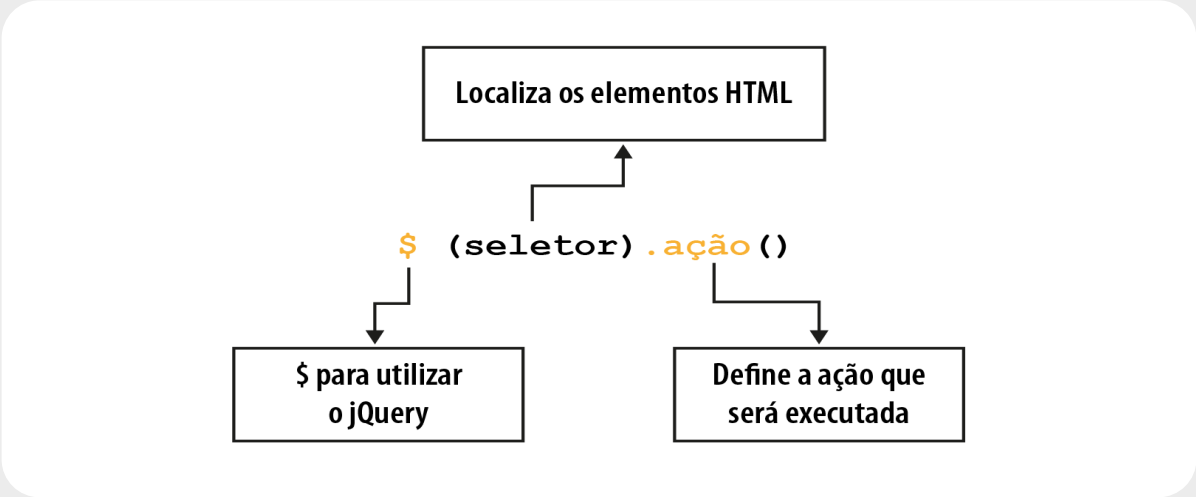

Figura 1 – Sintaxe do jQuery

#ParaTodosVerem: imagem que descreve a sintaxe de uso do jQuery. No centro, há uma descrição da sintaxe. Acima, um retângulo com fundo cinza possui o texto "localiza os elementos HTML" e refere-se ao seletor. Abaixo, outros dois retângulos com fundo na cor cinza. O retângulo à esquerda tem em seu interior o texto "$ para utilizar o jQuery", já o à direita contém o texto: "define a ação que será executada". Fim da descrição.

Vejamos alguns exemplos de como utilizar o jQuery para ter acesso aos elementos de uma página *web:*

- **$("*"):** seleciona todos os elementos da página;
- **$("tipo_do_elemento"):** seleciona todos os elementos de um tipo escolhido, basta colocar o nome do elemento entre as aspas. Exemplo: $("p"). hide() esconde todos os elementos <p>;
- **$("#id_do_elemento"):** seleciona o elemento pelo id. Um elemento do html possui um atributo id, esse atributo id é informado juntamente com o caracter # para o jQuery ter acesso. Exemplo: $("#main").hide() oculta o elemento com id="main";
- **$(".nome_da_classe"):** seleciona os elementos da classe. Dentro de um seletor, o caractere ponto (.) refere-se à classe (atributo "class" da *tag* HTML). Exemplo: $(".main").hide() oculta o elemento com class="main".

Para utilizar as funcionalidades do jQuery, o documento deve estar todo carregado. Isso evita erro quando tentamos executar um código antes que o documento termine de carregar. Observe o exemplo a seguir:

```
1.  <html>
2.  <head>
3.      <title>JQuery</title>
4.      <script src="https://code.jquery.com/jquery-3.6.4.js"></script>
5.      <script>
6.          $(document).ready(function(){
7.          $("h1.main").text("Combinando seletores");
8.          });
9.      </script>
10. </head>
11. <body>
12.     <h1>Aprendendo jQuery</h1>
13.     <h1 class="main">Aprendendo jQuery</h1>
14.     <h1 class="main">Aprendendo jQuery</h1>
15. </body>
16. </html>
```

Se você digitou o código acima, pôde perceber que seu navegador trocou duas ocorrências do título "Aprendendo jQuery" por "Combinando seletores". No Quadro 1, analisaremos o que está acontecendo:

Quadro 1 – Exemplo comentado

Linha(s)	Descrição
Linha 4	Utilizamos a biblioteca jQuery a partir de um repositório *on-line.*
Linha 6	Esperamos a página carregar completamente.
Linha 7	Substituímos o conteúdo do elemento <h1>, cuja classe é *main*, pelo texto "Combinando seletores".

Podemos também substituir $(document).ready(function() pela chamada de sintaxe abreviada, que é a seguinte:

```
$(function() {
        //bloco de códigos
})
```

Reflita

Você percebeu que o primeiro elemento <h1> não foi alterado? Isso aconteceu, pois o código que utilizamos altera somente o elemento cuja classe é *main*.

Com jQuery, podemos selecionar ou ter acesso aos elementos de acordo com os atributos do HTML. Para isso, basta informar o atributo, o operador e o valor dentro de colchetes ([]), conforme a sintaxe: $([atributo operador valor]).

Os operadores disponíveis são listados no Quadro 2:

Quadro 2 – Operadores

Operador	Significado
=	O valor informado precisa ser igual ao valor do atributo.
! =	O valor informado precisa ser diferente do valor do atributo.
^=	O valor do atributo precisa começar ou ser igual ao valor informado.
$ =	O valor do atributo precisa terminar ou ser igual ao valor informado
~ =	O valor do atributo precisa conter o valor informado.

Vejamos como podemos aplicar os operadores no jQuery. Observe os exemplos

abaixo:

- **$("[name='txtNome']"):** seleciona todos os elementos cujo atributo 'name' seja igual 'txtNome';

- **$("[type='text'][name='txtInfo']"):** seleciona e define o valor de todos os elementos do tipo 'input', cujo atributo nome seja igual a 'txtInfo'.

Tratamentos de Eventos

Uma das principais características do jQuery é sua capacidade para responder a eventos em um documento *web*. Os eventos podem ser, por exemplo, uma ação do usuário:

- Movendo o *mouse* sobre um elemento;

- Selecionando um *radio button;*

- Clicando sobre um elemento.

A seguir, vejamos alguns dos eventos mais comuns utilizados com jQuery, que estão listados abaixo:

- **$(document).ready(função):** como visto anteriormente, aguarda a página toda ser carregada para executar uma função;

- **$(seletor).click(função):** quando o elemento indicado no seletor for selecionado e receber o evento *click*, a função é executada;

- **$(seletor).dbclick(função):** quando o elemento indicado no seletor for selecionado e receber o evento duplo *click*, a função é executada;

- **$(seletor).change(função):** quando o elemento indicado no seletor for selecionado e sofrer uma alteração, então a função será executada;

- **$(seletor).focus(função):** quando o elemento indicado no seletor receber o foco, a função é executada.

Agora vamos aplicar esse conceito no exemplo a seguir, no qual o elemento <p> é ocultado quando clicamos sobre ele:

```
<html>
<head>
  <script src="https://code.jquery.com/jquery-3.6.4.js"></script>
  <script>
    $(document).ready(function () {
      $("p").click(function () {
        $(this).hide();
      });
    });
  </script>
</head>
<body>
  <p>Se você clicar aqui, vou desparecer.</p>
  <p>Clique aqui também!!</p>
</body>
</html>
```

No próximo exemplo, o método focus() é executado quando o campo do formulário recebe foco. Perceba como estamos manipulando o CSS do elemento. Veremos mais sobre a manipulação do CSS antes de finalizar esta Unidade.

```
<html>
<head>
  <script src="https://code.jquery.com/jquery-3.6.4.js"></script>
  <script>
    $(document).ready(function () {
      $("input").focus(function () {
        $(this).css("background-color", "red");
      });
    });
  </script>
</head>
<body>
  Nome: <input type="text" name="fullname"><br>
</body>
</html>
```

Métodos

A seguir, vamos listar alguns dos principais métodos utilizados para manipular elementos HTML:

- **$(seletor).text(conteúdo):** acessa o texto dos elementos;
- **$(seletor).val(conteúdo):** acessa os valores dos elementos contidos em um formulário. Os elementos podem ser do tipo input, *select* e radio;
- **$(seletor).html(conteúdo):** recupera ou define o conteúdo dos elementos, por exemplo, o código HTML que está dentro de uma div;
- **$(seletor).append(conteúdo):** insere o conteúdo ao final do elemento;
- **$(seletor).prepend(conteúdo):** insere o conteúdo no início do elemento;
- **$(seletor).after(conteúdo):** insere o conteúdo informado após o elemento;

- **$(seletor).before(conteúdo):** insere o conteúdo informado antes do elemento.

Vejamos um exemplo completo que se utiliza dos métodos citados acima:

```
<html>
<head>
        <script src="https://code.jquery.com/jquery-3.6.4.js"></script>
        <script>
                $(document).ready(function () {
                    $("#b1").click(function(){
                        $("[type='text']").val("Inserindo um valor novo");
                        $("[type='text']").before("- antes -");
                        $("[type='text']").after("- depois -");
                        $("#d1").prepend(" Aprendendo ");
                        $("#d1").append(" jQuery ");
                        $("#d2").html("Programação Web");
                    })
                });
        </script>
        <style>
                div{border: 1px solid saddlebrown;}
        </style>
</head>
<body>
        <div id="d1">Aprendendo jQuery</div>
        <br>
        <input type="text" value="valor inicial">
        <input id="b1" type="button" value="Clique aqui">
        <br><br>
        <div id="d2">Praticando jQuery</div>
</body>
</html>
```

Observe na Figura 2 a execução do código anterior:

Figura 2 – Exemplo antes de clicar no botão

#ParaTodosVerem: imagem da tela do navegador Chrome em fundo na cor branca. Na parte superior, há um retângulo com o texto "Aprendendo jQuery". Abaixo outro retângulo com o texto "valor inicial" e, ao lado deste, há um botão com a inscrição "Clique aqui". Na parte inferior da Figura, um último retângulo contém o texto "Praticando jQuery". Fim da descrição.

Ao clicar no botão, o evento de clique é acionado no elemento b1 e os métodos são executados, obtendo o resultado apresentado na Figura 3:

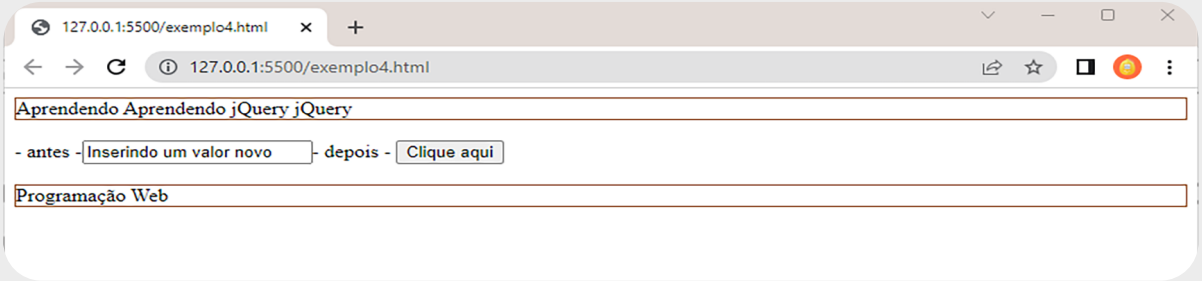

Figura 3 – Exemplo após clicar no botão

#ParaTodosVerem: imagem da tela do navegador Chrome em fundo na cor branca. Na parte superior, há um retângulo com o texto "Aprendendo jQuery". No centro, há o texto "- antes -", seguido de um retângulo com o texto "Inserindo um valor novo", após, o texto "- depois -", seguido de um botão com a inscrição "Clique aqui". Na parte inferior da figura, um último retângulo contém o texto "Programação *Web*". Fim da descrição.

Manipulando CSS com JQuery

O jQuery nos permite manipular com facilidade os seletores alinhados com CSS e CSS3, ou seja, é possível ter acesso às propriedades de alteração do estilo de forma simples e prática, evitando-se as funções padrão do *JavaScript*. Por meio do uso do método css() do jQuery, podemos definir ou retornar uma ou mais propriedades de estilo para elementos selecionados.

Vejamos como isso funciona, começando com a sintaxe para retornar o valor de uma propriedade CSS específica, que é dada por: $(seletor).css("nome_da_propriedade"). Vamos aplicar esse conceito no exemplo a seguir, que retorna a cor de fundo do elemento <p>:

```
<html>
<head>
        <script src="https://code.jquery.com/jquery-3.6.4.js"></script>
        <script>
                $(document).ready(function () {
                        let valor = $("p").css("background-color");
                        $("button").click(function () {
                                alert("Background-color=" + valor);
                        });
                });
        </script>
        <style>
            p {background-color: blueviolet; color: white; padding: 10px;}
    </style>
</head>
<body>
        <p>Clique no botão para revelar a cor de fundo.</p>
        <button>Clique aqui</button>
</body>
</html>
```

Agora vamos definir uma propriedade específica do CSS utilizando a seguinte sintaxe: $(seletor).css("nome_da_propriedade", "valor"). O exemplo que utilizaremos para aplicar o conceito é muito parecido com o anterior. Porém agora vamos mudar a cor de fundo do parágrafo quando clicamos no botão:

```
<html>
<head>
	<script src="https://code.jquery.com/jquery-3.6.4.js"></script>
	<script>
		$(document).ready(function () {
			$("button").click(function () {
				$("p").css("background-color","red");
			});
		});
	</script>
	<style>
		p {background-color: blueviolet; color: white; padding: 10px;}
	</style>
</head>
<body>
	<p>Clique no botão para mudar a cor de fundo.</p>
	<button>Clique aqui</button>
</body>
</html>
```

É possível definirmos várias propriedades CSS para um único elemento utilizando a sintaxe: $(seletor).css({"propriedade1": "valor", "propriedade2": "valor", ...}). Vamos modificar nosso exemplo anterior para que, ao clicarmos no botão, além da cor de fundo (*background-color*), o tamanho da fonte seja alterado. Para isso, teremos que alterar o código para:

```
$("p").css({"background-color":"red", "font-size": "24px"});
```

WebStorage (HTML5)

Como o lançamento do HTML5, a retenção de dados no lado do cliente ficou mais prática e eficiente por meio da API *WebStorage*, que oferece mecanismos para que o navegador possa armazenar dados por meio de um conceito chave/valor, com espaço maior de armazenamento e sem o envio das informações salvas para o servidor.

A compatibilidade da API com os diversos navegadores existentes no mercado é exibida no Quadro 3.

Quadro 3 – Compatibilidade da API com os navegadores *web*

Propriedade	Chrome	Explorer	Mozilla	Safari	Opera
WebStorage	4.0	8.0	3.5	4.0	10.5

Para evitar problemas na implementação de um armazenamento web, uma boa prática é verificar se o navegador possui compatibilidade com a API, o que pode ser feito com o código do exemplo a seguir:

```
<script>
        if (typeof (Storage) !== “undefined”) {
                alert(“Seu navegador possui suporte ao WebStorage!”)
        } else {
                alert(“Desculpe, mas seu navegador não possui suporte!”)
        }
</script>
```

O *WebStorage* fornece dois objetos para armazenar dados no cliente:

- *LocalStorage*, que armazena dados sem data de validade, ou seja, os dados somente serão excluídos pela aplicação ou pelo usuário;
- *SessionStorage*, que armazena dados para uma sessão (os dados são perdidos quando a guia do navegador é fechada).

O armazenamento *web* utiliza-se de um conceito de objetos, que são conjuntos contendo pares de chave/valor. Observe na Figura 4 que a chave “nome” armazena o valor “Marco”.

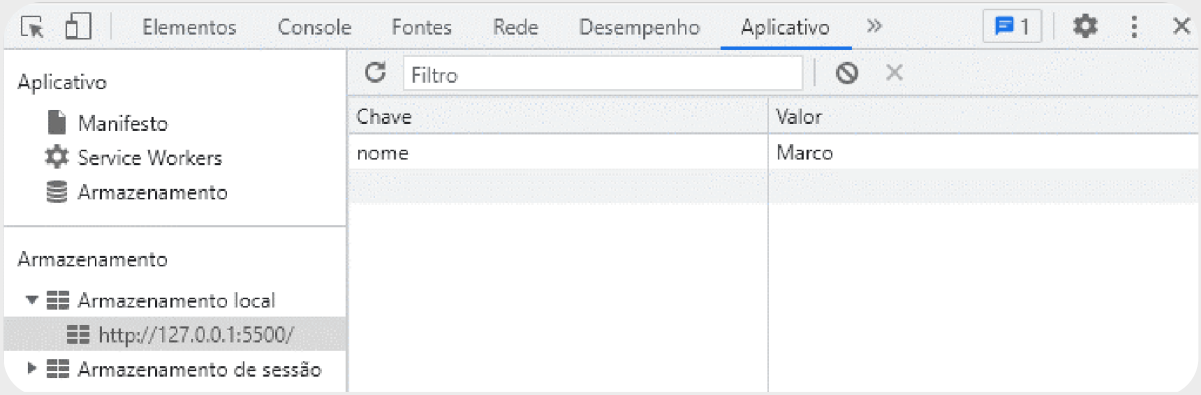

Figura 4 – Armazenamento local

#ParaTodosVerem: imagem da tela de ferramentas do desenvolvedor exibida no navegador Chrome em fundo na cor branca. Na parte superior, há um menu com dois ícones à esquerda das opções e outros quatro ícones à direita. Abaixo, à esquerda, são listadas as opções “Aplicativo” e “Armazenamento”. À direita, há um campo contendo o texto “Filtro” e, abaixo deste, há uma tabela com duas colunas, com os títulos “Chave” e “Valor”. Abaixo do título “Chave” há o valor “nome” e abaixo de “Valor” há o texto “Marco”. Fim da descrição.

Para manipular e acessar os dados armazenados, vamos utilizar os seguintes métodos:

- ***SetItem*(nome, valor):** utilizado para gravar dados. Exemplo: localStorage.setItem(“nome”,”Maria”);
- ***GetItem*(nome):** utilizado para recuperar dados. Exemplo: localStorage.getItem(“nome”);
- ***RemoveItem*(nome):** utilizado para apagar um dado gravado. Exemplo: localStorage.removeItem(“nome”);
- ***Clear*():** utilizado para apagar todos os dados do armazenamento. Exemplo: localStorage.clear().

Vamos aplicar os conceitos estudados em um exemplo para cadastro simples de alunos, por meio de um formulário, como mostra a Figura 5:

Figura 5 – Formulário de cadastro de alunos

#ParaTodosVerem: imagem da tela do navegador Chrome em fundo na cor branca. No centro, há um retângulo com fundo na cor branca contendo um formulário com os campos "RGM", "Nome" e "Curso", sendo que ao lado de cada campo há um retângulo para entrada de dados. Abaixo, três botões: o primeiro, à esquerda, com fundo azul, contém o texto "Gravar", o do centro tem fundo verde e a inscrição "Carregar" e o último, à direita, tem fundo vermelho e o texto "Apagar". Fim da descrição.

Para manipular os dados, vamos utilizar jQuery e criaremos três métodos:

- **GravarDados():** está associado ao botão "Gravar" e é responsável por salvar os dados do formulário no armazenamento local;
- **CarregarDados():** está associado ao botão "Carregar" e é responsável por carregar os dados a partir do armazenamento local para o formulário;
- **ApagarDados():** está associado ao botão "Apagar" e é responsável por apagar todos os dados do armazenamento local.

O código para implementação das funções é mostrado a seguir:

```
<script>
        $("#button1").click(gravarDados);
        $("#button2").click(carregarDados);
        $("#button3").click(apagarDados);

        function gravarDados() {
                localStorage.setItem("rgm", $("#rgm").val());
                localStorage.setItem("nome", $("#nome").val());
                localStorage.setItem("curso", $("#curso").val());
                alert("Dados gravados");
                limparCampos();
        }

        function carregarDados() {
                if (localStorage.length > 0) {
                        $("#rgm").val(localStorage.getItem("rgm"));
                        $("#nome").val(localStorage.getItem("nome"));
                        $("#curso").val(localStorage.getItem("curso"));
                } else {
                        alert("Nenhum registro encontrado");
                }
        }

        function apagarDados() {
                if (localStorage.length > 0) {
                        localStorage.clear();
                } else {
                        alert("Nenhum registro encontrado");
                }
                limparCampos();
}
```

```
        function limparCampos() {
                $("#rgm").val("");
                $("#curso").val("");
                $("#nome").val("");
        }
</script>}


function limparCampos() {
$("#rgm").val("");
$("#curso").val("");
$("#nome").val("");
}
</script>
```

AJAX

O *AJAX* é o acrônimo de *Asynchronous JavaScript and XML*, ou *JavaScript* e XML Assíncronos, em bom português. Trata-se de um conjunto de técnicas para carregamento de conteúdo em uma página *web* que permite que aplicações trabalhem de modo assíncrono, tornando o conteúdo *web* mais interativo para o usuário, uma vez que a troca de dados com o servidor ocorre em segundo plano. Na prática, isso significa que conseguimos atualizar partes de uma página *web* sem a necessidade de recarregar a página inteira.

Quando trabalhamos com AJAX, utilizamos o objeto XMLHttpRequest, que é manipulado por meio do *JavaScript* e que efetua a comunicação com o servidor por meio de requisições HTTP.

O XMLHttpRequest é um objeto nativo padronizado nos navegadores atuais, mas foi implementado originalmente pela *Microsoft* como controle *ActiveX* nas versões antigas do *Internet Explorer*. Desde então, outros navegadores passaram a implementar uma classe XMLHttpRequest, que permite suporte aos métodos e propriedade desse objeto.

A biblioteca jQuery oferece uma vasta documentação para se trabalhar as requisições AJAX. O principal método para executarmos uma solicitação é o $.ajax(), que é utilizado por todos os outros métodos jQuery AJAX.

A implementação com $.ajax(), considerada de nível inferior, oferece um conjunto vasto de opções não disponíveis em funções chamadas de alto nível, como $.get() e $.post(), que são mais fáceis de se utilizar.

Na requisição, precisamos informar um seletor que vai dar início à execução, o endereço do arquivo que contém os dados que serão retornados, o tipo de dado retornado e o que será feito com ele. No exemplo a seguir, utilizamos a função $.get() para alterar o conteúdo de um parágrafo:

```
1.  <html>
2.  <head>
3.      <script src="https://code.jquery.com/jquery-3.6.4.js"></script>
4.      <script>
5.          $(function(){
6.              $("button").click(function(){
7.                  $.get({url: "dados.txt", success: function(result){
8.                      $("#p1").html(result);
9.                  }});
10.             });
11.         });
12.         </script>
13.     </head>
14.     <body>
15.         <h2>AJAX com jQuery</h2>
16.         <p id="p1">Clique no botão para alterar este conteúdo</p>
17.         <button>Clique aqui</button>
18.     </body>
19.     </html>
```

Quando o botão “Clique aqui” for acionado, uma requisição será iniciada e os dados do arquivo “dados.txt” serão carregados para a página *web*, substituindo o conteúdo do parágrafo. Veja a seguir o código comentado no Quadro 4:

Quadro 4 – Exemplo comentado

Linha(s)	Descrição
Linha 6	Anexamos o evento *click* no elemento *button* que será responsável por disparar a requisição.
Linha 7	Iniciamos a função $.get() e passamos o parâmetro url, que requisita o arquivo dados.txt. Observe que *result* armazena os dados recebidos nessa requisição.
Linha 8	O método html() é utilizado para inserir o conteúdo da requisição dentro do <p id = “p1”>, conforme a marcação HTML.

Usar os métodos ajax() do jQuery torna as requisições mais simples do que usar o método nativo do *JavaScript*, o XMLHttpRequest.

Entretanto, uma função mais moderna que tem se tornado muito popular é o Fetch, que fornece uma implementação mais flexível e fácil de se fazer requisições para busca de dados de forma assíncrona pela rede. O Fetch veio para substituir o objeto XMLHttpRequest e tem se tornado padrão nas versões mais atuais dos navegadores *Firefox Chrome*, mas, por se tratar de uma tecnologia nova, navegadores mais antigos não reconhecem essa funcionalidade.

Uma forma de verificar se o navegador possui suporte é a seguinte:

```
if(self.fetch) {
// execute minha solicitação do fetch aqui
} else {
// faça alguma coisa com XMLHttpRequest?
}
```

No exemplo a seguir, vamos fazer uma requisição *Fetch* para obter a lista de repositórios de um usuário do *github*:

```
1.  <html>
2.  <head>
3.      <script src="https://code.jquery.com/jquery-3.6.4.js"></script>
4.  </head>
5.  <body>
6.      <h2>Repositórios do github</h2>
7.      <p>Utilizando o Fetch para listar os repositórios do github</p>
8.      <ul></ul>
9.      <script>
10.         const url = "https://api.github.com/users/msanches/repos";
11.         fetch(url)
12.         .then(response => response.json())
13.         .then(data => {
14.             data.forEach(element => {
15.                 $("ul").append("<li>" + element.name + "</li>");
16.             });
17.         })
18.         .catch(err => console.log(err))
19.     </script>
20. </body>
21. </html>
```

/

O código comentado do exemplo é mostrado abaixo no Quadro 5:

Quadro 5 – Exemplo comentado

Linha(s)	Descrição
Linha 10	Declaramos a url que queremos fazer a requisição.
Linha 11	Usamos o Fetch para fazer a requisição para a url que declaramos.
Linha 12	É a resposta da requisição, que retornará um valor response no formato json.
Linha 13 - 15	Com o valor da requisição retornado, vamos utilizar o jQuery para exibir uma lista com o nome dos repositórios encontrados.

MATERIAL COMPLEMENTAR

Sites

W3school
É um *site* voltado à aprendizagem de conteúdos de tecnologias como HTML, CSS, *Javascript*, *Python* entre outros. Visite o tutorial de jQuery.
https://bit.ly/3PyCtGs

jQuery
Visite a página oficial do jQuery e descubra todos os recursos que a biblioteca dispõe para o desenvolvimento *web*.
https://bit.ly/3sWXdi4

Leituras

O que São as Ferramentas de Desenvolvimento do Navegador
Os navegadores modernos oferecem um conjunto de ferramentas que auxiliam o desenvolvedor a inspecionar o HTML, CSS e *JavaScript* de uma página. Conheça um pouco mais sobre o *devtools* visitando o *link* a seguir.
https://bit.ly/45K6d96

***Fetch* API**
É uma plataforma aberta destinada à aprendizagem de tecnologias *Web*. Acesse a página com os conteúdos de *Fetch* API.
https://mzl.la/3PbcwLD

REFERÊNCIAS BIBLIOGRÁFICAS

CLARK, R. *et al*. **Introdução ao HTML5 e CSS3:** a evolução da *web*. Rio de Janeiro: Alta *Books*, 2014.

SILVA, M. S. **jQuery**: A biblioteca do programador *JavaScript*. São Paulo: Novatec, 2014.

SILVA, M. S. **CSS3:** Desenvolva aplicações ***web*** profissionais com uso dos poderosos recursos de estilização das CSS3. São Paulo: Novatec, 2012.

TERUEL, E. C. **HTML 5:** guia prático. 2. ed. São Paulo: Erica, 2013.

CSS/*Training*. **W3**. Disponível em: < https://www.w3.org/community/webed/wiki/CSS/Training>. Acesso em: 27/02/2023.