

Introdução ao JavaScript





Sumário

- 3 - O que é JavaScript?
- 4 - Iniciando com JavaScript
- 8 - A tag `<script>`
- 10 - DOM: estrutura base para alterações
- 12 - Eventos e funções DOM
- 14 - Manipulando o CSS com JavaScript
- 16 - Validando formulários com JavaScript
- 18 - Calculadora simples com JavaScript
- 22 - Bônus: JavaScript com jQuery
- 26 - Referências



Autor:

João Victor Correia de Oliveira

com orientação do professor

Aderbal Botelho Leite Neto

qualquer dúvida, sugestão ou crítica:

jvcorreia@sempreceub.com

ou

aderbal.neto@ceub.edu.br

O que é JavaScript?

O JavaScript surgiu com a necessidade de proporcionar mais dinamismo ao ambiente web, o qual era muito estático. Sendo uma linguagem leve, o JS se relaciona diretamente com componentes dentro do browser, fornecendo um controle pragmático sobre eles.

Algumas informações que devem ser entendidas antes de se começar a desenvolver em JS:

- O JS é uma linguagem híbrida, que dá liberdade ao desenvolvedor para desenvolver de forma estruturada, orientada a objeto ou até mesmo estruturada.
- O JS é uma linguagem interpretada, ou seja, ela não passa por um processo de compilação sendo papel do interpretador converter esse código de alto nível.
- O JS é uma linguagem dinâmica, onde a declaração de tipo de variável não é obrigatória (tipagem fraca).
- Apesar de terem nomes parecidos, o JavaScript não deve ser confundido de nenhuma forma com Java pois são coisas totalmente diferentes.
- Antes que o JS se tornasse popular, para que a linguagem evoluísse obedecendo certos padrões e normas, os criadores se associaram ao ECMA (European Computer Manufactures Association) em 1996. Após essa associação, ele passou a se chamar ECMASCRIPT. Porém, o nome JAVASCRIPT já estava na boca da galera e aí acabou ficando, e o ECMASCRIPT acabou ficando para designar apenas a versão.

Iniciando com o JavaScript

Existe muitas formas para se executar um código JS dentro de uma página, e uma delas é usando o console do navegador. A maioria dos navegadores já vem com essa função habilitada.

No Google Chrome:

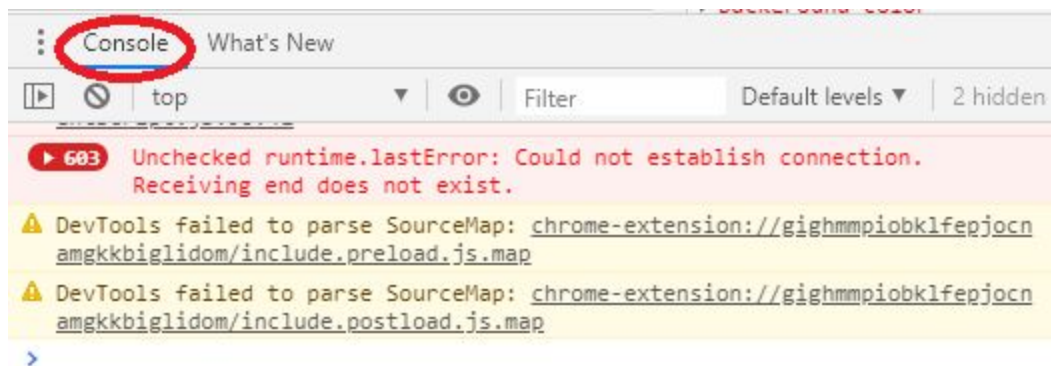
F12 → Acessar a aba Console

ou

Control + Shift + C

No Firefox:

Control + Shift + K



Faça um teste! Digite você mesmo no console o seguinte código:

```
alert("o ceub me deixa feliz!");
```

Operações

Observe que nessa linguagem as operações são muito simples, como em qualquer outra linguagem:

>16 + 13 ← Input

29 ← Output do console

>15 * 3

45

>10 - 1

9

>25 / 5

5

>23 % 2

1

Variáveis

Para armazenarmos um valor para uso posterior, podemos criar uma variável:

```
>var divisao = 10 / 5;
```

```
undefined
```

No exemplo acima, guardamos o resultado de 10 dividido por 5 na variável 'divisao'. O output quando se cria uma variável é sempre undefined. Podemos declarar qualquer tipo de variável:

```
>var nome = "joao victor correia";
```

```
undefined
```

```
>var numeropi = 3.141;
```

```
undefined
```

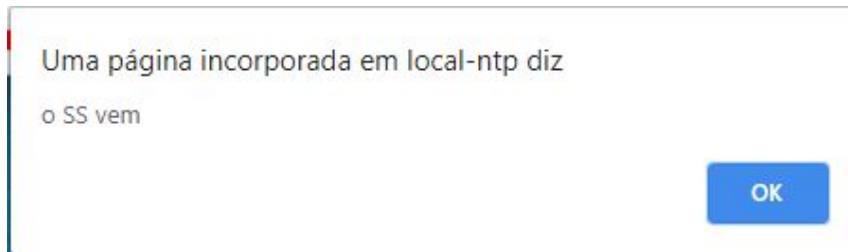
```
>var soulindo = true;
```

```
undefined
```

Alert

Esse comando serve para exibir alertas fora do console. Ele abre um popup no navegador com o conteúdo que digitarmos entre os parênteses.

```
>alert("o SS vem");
```



OBS: O PONTO E VÍRGULA APÓS CADA LINHA NÃO É OBRIGATÓRIO, FICA A SEU CARGO.

Array

O array é útil quando precisamos trabalhar com diversos valores armazenados:

```
var professores = ["Aderbal", "Klein"];
```

Note que aqui não é necessário declarar o tipo de lista nem mesmo dizer que é uma lista, basta abrir os colchetes. Podemos também adicionar elementos usando o método push:

```
professores.push("João");
```


For

Muitas vezes precisamos executar um trecho de código até que determinada condição seja estabelecida. Para isso, o JavaScript oferece uma série de blocos de repetição, sendo o for o mais comum.

```
var professores = ["Aderbal", "Klein", "jv", "o lendario", "zuckman"];  
  
for (let i = 0; i < professores.length; i++) {  
    console.log(professores[i]);  
}
```

No bloco acima, vamos imprimir no console o nome de cada professor dentro da lista... veja o resultado:

```
[Running] node "c:\Users\João Victor\Desktop\program\HTML\ProgramaçãoWebCeub\script.js"  
Aderbal  
Klein  
jv  
o lendario  
zuckman
```

A tag <script>

Mais acima, vimos que podemos testar os nossos códigos dentro do console do browser, mas não podemos pedir para que o usuário faça isso toda vez que entrar na página né? Para inserirmos um script dentro de uma página podemos utilizar a tag <script> :

```
<html>
  <head>
    <script>alert('olá mundo');</script>
  </head>
  <body>
    <p>pagina web</p>
  </body>
</html>
```

Esteja ciente de que a tag script também pode ser declarada dentro da tag body, e essas questões estão ligadas diretamente ao carregamento de sua página. Ao se inserir um script no head, ele é carregado antes dos outros componentes, o que pode gerar um travamento inicial. Minha recomendação é que ao colocar scripts escritos diretamente na página, ou seja, os que não são referenciados, é melhor colocá-los ao final do body, mais especificamente dentro do footer.

Exercício 1

```
<html>
  <head>
    <title>página web</title>
  </head>
  <body>
    <p>pagina web</p>

    <footer>
      <script>alert('olá mundo');</script>
    </footer>
  </body>
</html>
```

Para não ter que escrever todos os scripts dentro da página HTML, eles podem ser referenciados também por meio da tag <script>, dando

maior organização ao código e a possibilidade de reaproveitar o código em outras páginas:



(referência e modelo de projeto)

DOM: estrutura base para alterações

Com o objetivo de permitir alterações na página após carregar o HTML, os navegadores carregam em sua memória uma estrutura chamada DOM (Document Object Model) , que pode ser usada através do comando document.

Exercício 2

```
<html>
  <body >
    <p id="paragrafo"></p>

    <footer>
      <script>
        document.getElementById('paragrafo').innerHTML = 'isso é manipulado por JS!';
      </script>
    </footer>
  </body>
</html>
```

Observe que não foi escrito nada no parágrafo, mas o JavaScript escreveu o conteúdo dentro dele. Por enquanto se preocupe apenas com a parte document.getElementById. Se não tivermos um id podemos inserir também a tag (p, h1, h2), porém o JS vai modificar todas as tags então não é muito recomendado. Aqui os comandos para utilizar esses acessos:

Método	O que faz
document.getElementById	Procura elemento por Id
document.getElementsByTagName	Procura elemento pelo nome da tag
document.getElementsByClassName	Procura elemento pela classe

É possível também guardar os elementos acessados em variáveis, para que seja possível acessá-los depois.

```
<script>
  var par = document.querySelector('#paragrafo');
  par.textContent = 'olá';
</script>
```

O `querySelector` é similar ao `GetDocumentByld`, ele usa os seletores CSS para pode encontrar elementos dentro da página.

Eventos e funções para DOM

É interessante a possibilidade de alterar o documento por meio de JavaScript, porém essas ações geralmente são chamadas quando o usuário executa alguma ação, por exemplo: clicar em um botão, passar o mouse por cima de algum lugar ou preencher um formulário. Por padrão, qualquer código colocado dentro da tag `<script>` é executado assim que o navegador lê ele. Para podermos guardar um tipo de código para ser acionado depois podemos criar uma função:

```
function botao(){
  alert('você clicou no botão');
}
```

Esta é uma função bem básica, usamos sempre o function para abrir a função, damos um nome a função depois precisamos abrir os parênteses e nele declaramos as variáveis que serão usadas, no caso acima nenhuma foi usada. Para exemplificar melhor como uma função é chamada, vamos criar um botão para mandar um alert apenas no clique:

```
<html>
  <head>
    <script src="script.js"></script>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <button onclick="botao()">Clique aqui</button>
  </body>
</html>
```

Ao clicar no botão ele deve retornar o alert.

Chamamos a função dentro do componente (no caso o button) por meio do evento onclick. Além desse, temos muitos outros eventos que são muito importantes para manipular os eventos dentro desses componentes:

- onclick : quando ocorre um clique com o mouse;
- ondblclick : quando ocorrem dois cliques com o mouse;
- onmousemove: quando mexe o mouse;
- onmouseover: quando passa o mouse em cima
- onkeypress: quando pressionar e soltar uma tecla;
- onblur: quando um elemento perde o foco;
- onfocus: quando um elemento ganha o foco;
- onchange: quando um input tem o valor alterado;
- onload: quando a página é carregada;
- onsubmit: quando vai enviar dados de um formulário, por exemplo.

Muito utilizado em validações.

Exercício 3: reproduzir o exemplo acima (clique no botão e exibição de alert)

Manipulando o CSS com JavaScript

Podemos usar o JS para manipular o nosso CSS em tempo de execução, no próximo exercício vamos alterar o background do body da página à medida que passamos o mouse em cima da nossa opção de cor. O modelo de manipular o CSS segue o mesmo padrão do exercício acima, então você já estará mais familiarizado. Observe que, todas as propriedades do CSS podem ser manipuladas (fonte, tamanho, visibilidade e etc). O exemplo abaixo vai lhe dar base para manipular as outras propriedades.

Exercício 4

```
function vermelho(){
    document.querySelector('body').style.backgroundColor = 'red';
}

function azul(){
    document.querySelector('body').style.backgroundColor = 'blue';
}

function verde(){
    document.querySelector('body').style.backgroundColor = 'green';
}
```

```
<html>
  <head>
    <script src="script.js"></script>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <p style="background-color: red; onmouseover=vermelho()">Fundo vermelho</p>
    <p style="background-color: blue; onmouseover=azul()">Fundo azul</p>
    <p style="background-color: green; onmouseover=verde()">Fundo verde</p>
  </body>
</html>
```

Fundo vermelho

Fundo azul

Fundo verde

(Quando abrimos a página)

Fundo vermelho

Fundo azul

Fundo verde

(Ao passar o mouse por cima de 'fundo azul')

Fundo vermelho

Fundo azul

Fundo verde

(Ao passar o mouse por cima de 'fundo verde')

Validando formulários com o JavaScript

Abaixo vamos fazer um exemplo com uma validação simples, porém a partir dele você irá ter base para outros exemplos mais complexos. Primeiramente, vamos criar o nosso formulário... Vou pegar um simples da internet pois o foco dessa apostila não é HTML.

Exercício 5

```
<html>
  <head>
    <script src="script.js"></script>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body >
    <form method="POST" name="formulario">
      <input type="text" name="nome" placeholder="digite seu Nome aqui">
      <input type="password" name="senha" placeholder="Digite a senha">
      <input type="submit" name="enviar" value="Cadastrar">
    </form>
  </body>
</html>
```

Após o formulário ter sido montado, e o arquivo de script devidamente referenciado, podemos ir para a criação da nossa função de validação. Após criá-la, vamos pegar o valor dos componentes dentro do formulário:

```
function valida(){
    var x = document.forms["formulario"]["nome"].value;
    var y = document.forms["formulario"]["senha"].value;
}
```

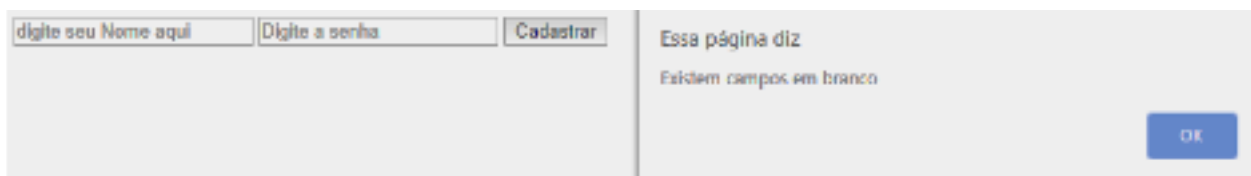
Com os valores devidamente acessados, podemos partir para a condição que definirá a validação do formulário:

```
function valida(){
    var x = document.forms["formulario"]["nome"].value;
    var y = document.forms["formulario"]["senha"].value;

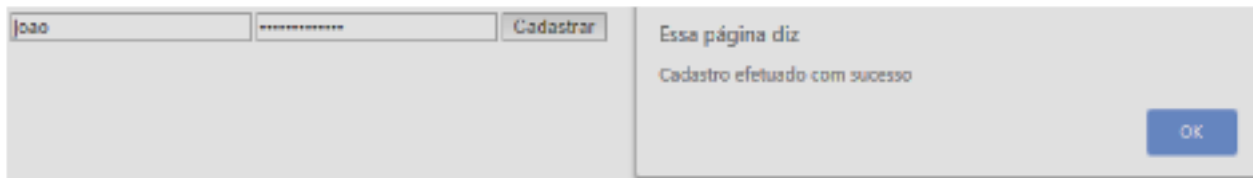
    if (x == "" || y == "") {
        alert("Existem campos em branco");
    }
    else{
        alert("Cadastro efetuado com sucesso");
    }
}
```

A validação diz que se um dos campos não contiver conteúdo ele vai exibir um alert dizendo que existem campos em branco, caso contrário ele exibe um alert indicando o sucesso. Bem, agora só precisamos referenciar a função dentro do form:

```
<form method="POST" onsubmit="return valida()" name="formulario">
    <input type="text" name="nome" placeholder="digite seu Nome aqui">
    <input type="password" name="senha" placeholder="Digite a senha">
    <input type="submit" name="enviar" value="Cadastrar">
</form>
```



(Sem inserir dados)



(Inserindo dados)

Calculadora simples com JavaScript

Nesse último exemplo, faremos uma calculadora bem simples em uma página web utilizando JS. Note que, se você já tem alguma experiência o jeito de fazer pode não parecer tão bom de acordo com os bons modos, mas o objetivo aqui é fazer com que a linguagem seja compreendida. Criaremos uma página web com 4 radio buttons, que são aquelas bolinhas para o usuário poder escolher a operação que vai efetuar na calculado, além de 2 inputs para ele digitar os valores. Vou deixar a parte de HTML aqui para adiantar o trabalho de quem for replicar o exercício:

Exercício 6

```
<html>

  <head>

    <script src="script.js"></script>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

  </head>

  <body >

    <input type="radio" name="op" value=1>

    <label >Soma</label><br>
```

```
<input type="radio" name="op" value=2>
```

```
<label >Subtração</label><br>
```

```
<input type="radio" name="op" value=3>
```

```
<label >Multiplicação</label><br>
```

```
<input type="radio" name="op" value=4>
```

```
<label >Divisão</label><br>
```

```
<label>Valor 1</label>
```

```
<input type="number" id="input1">
```

```
<br>
```

```
<label>Valor 2</label>
```

```
<input type="number" id="input2">
```

```
<br>
```

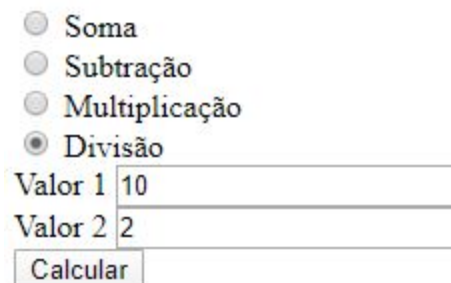
```
<p style="color: red;" id="resultado"></p>
```

```
<button onclick="calcula()">Calcular</button>
```

```
</body>
```

```
</html>
```

Note que será uma página bem simples, o CSS fica a seu cargo:



☐ Soma
☐ Subtração
☐ Multiplicação
☒ Divisão

Valor 1 10
Valor 2 2

Calcular

Agora, nosso trabalho será criar a função... criaremos apenas uma função que vai fazer todos os cálculos da página de acordo com o que o usuário marcar. Primeiro, vamos captar o valor dos inputs do usuário.

```
function calcula(){  
    var a = document.getElementById('input1').value;  
    var b = document.getElementById('input2').value;  
}
```

Após essa parte, precisamos chamar o valor do radio button, porém tenho uma péssima notícia para você..... por algum motivo não é possível chamar o valor diretamente por esse método `getElementById('op').value`. Mas não se preocupe, contornarmos esse problema utilizando um for.

```
function calcula(){
    var a = document.getElementById('input1').value;
    var b = document.getElementById('input2').value;

    for (var i = 0; i < document.getElementsByName('op').length; i++)
    {
        if (document.getElementsByName('op')[i].checked)
        {
            var c = document.getElementsByName('op')[i].value;
        }
    }
}
```

Calma, pode parecer um pouco desesperador no início, mas vou explicar. Vamos percorrer a lista dos radio buttons que estão em 'op', e procurar o que esteja marcado, ou 'checked'. Se ele estiver marcado (checked), vamos atribuir o valor dele para uma variável, desse jeito ele irá captar o valor. Para finalizar, devemos colocar os ifs para definir a operação de acordo com o valor que foi captado. Precisamos usar o parseInt, para converter os valores captados pelos inputs para inteiro. Coloque o if abaixo depois após o for.

```
if(c == 1){
    document.getElementById('resultado').innerHTML = parseInt(a) + parseInt(b);
}
else if(c == 2){
    document.getElementById('resultado').innerHTML = parseInt(a) - parseInt(b);
}
else if(c == 3){
    document.getElementById('resultado').innerHTML = parseInt(a) * parseInt(b);
}
else if(c == 4){
    document.getElementById('resultado').innerHTML = parseInt(a) / parseInt(b);
}
```

Se tiver prestado atenção no código html, deve ter percebido que coloquei um parágrafo sem nenhum tipo de conteúdo dentro.. é nele que exibiremos o resultado!

A página está pronta, agora é só testar:

☒ Soma
☐ Subtração
☐ Multiplicação
☐ Divisão

Valor 1
Valor 2

22

☐ Soma
☐ Subtração
☒ Multiplicação
☐ Divisão

Valor 1
Valor 2

40

Bônus: JavaScript com jQuery

O JQuery veio para facilitar a vida do desenvolvedor JavaScript, sendo uma biblioteca que têm muita coisa para você utilizar. Sendo a biblioteca mais famosa de JavaScript, jQuery é usado por cerca de 70% das páginas web, então veja que aprender jQuery é um grande negócio. Sua sintaxe foi desenvolvida para melhorar o acesso a documentos, manipular eventos, desenvolver aplicações AJAX, entre outros.

Para fazer o download do jQuery, basta entrar na página:

<https://jquery.com/download/>

Após isso, clique no “download the compressed production...”:

Downloading jQuery

Compressed and uncompressed copies of jQuery files are available. The uncompressed file is best used during development or debugging; the compressed file saves bandwidth and improves performance in production. You can also download a [sourcemap file](#) for use when debugging with a compressed file. The map file is *not* required for users to run jQuery, it just improves the developer's debugger experience. As of jQuery 1.11.0/2.1.0 the `/** sourceMappingURL` comment is not included in the compressed file.

To locally download these files, right-click the link and select “Save as...” from the menu.

jQuery

For help when upgrading jQuery, please see the [upgrade guide](#) most relevant to your version. We also recommend using the [jQuery Migrate plugin](#).

[Download the compressed production jQuery 3.4.1](#)

[Download the uncompressed development jQuery 3.4.1](#)

[Download the map file for jQuery 3.4.1](#)

Você vai notar que vai abrir uma página com muito código, aperte CTRL + A para copiar todo o código, crie um arquivo JS dentro do seu projeto e cole lá.

```
/*! jQuery v3.4.1 | (c) JS Foundation and other contributors | jquery.org/license */
!function(e,t){“use strict”;“object”==typeof module&&“object”==typeof module.exports?module.exports=e.document?t(e,!0):function(e){if(!e.docu
window with a document”);return t(e)}:t(e)}(“undefined”!=typeof window?window:this,function(C,e){“use strict”;var t=
[],E=C.document,r=Object.getPrototypeOf,s=t.slice,g=t.concat,u=t.push,i=t.indexOf,n={},o=n.toString,v=n.hasOwnProperty,a=v.toString,l=a.call
{return“function”==typeof e&&“number”!=typeof e.nodeType},x=function(e){return null!=e&&e===e.window},c={type:!0,src:!0,nonce:!0,noModule:!0}
(n=n)[E].createElement(“script”);if(o.text=e,t)for(r in c)(i=t[r]||t.getAttribute&&t.getAttribute(r))&&o.setAttribute(r,i);n.head.appendChild
{return null==e?e+“”:“object”==typeof e||“function”==typeof e?n[o.call(e)]||“object”:typeof e}var f=“3.4.1”,k=function(e,t){return new k.fn.:
[\s\u0000\u000a\u000d]+$/g,function d(e){var t=!e&&“length”in e&&e.length,n=w(e);return!m(e)&&x(e)&&(“array”===n||0===t||“number”==typeof t&&0<t&&
{jquery:f,constructor:k,length:0,toArray:function(){return s.call(this)},get:function(e){return null==e?s.call(this):e<0?this[e+this.length]
t=k.merge(this.constructor(),e);return t.prevObject=this,t},each:function(e){return k.each(this,e)},map:function(n){return this.pushStack(k.
n.call(e,t,e))},slice:function(){return this.pushStack(s.apply(this,arguments))},first:function(){return this.eq(0)},last:function(){return
t=this.length,n+=e<0?t:0;return this.pushStack(0<n&&n<t?[this[n]]:[])},end:function(){return
this.prevObject||this.constructor()},push:u,sort:t.sort,splice:t.splice,k.extend=k.fn.extend=function(){var e,t,n,r,i,o,a=arguments[0]||
{),s=1,u=arguments.length,l=1;for(“boolean”==typeof a&&(l=a,a=arguments[s]||{}),s++,”object”==typeof a||m(a)||((a={),s==u&&(a=this,s--);s<
e)r=e[t],“__proto__”!=t&&a!=r&&(l&&r&&(k.isPlainObject(r)||(!Array.isArray(r)))?(n=a[t],o=i&&Array.isArray(n)?[i]||k.isPlainObject(n)?n
(a[t]=r);return a},k.extend({expando:“jQuery”+(+Math.random()).replace(/D/g,“”),isReady:!0,error:function(e){throw new Error(e)},noop:fun
t,n;return!(!e||“object Object”!=o.call(e))&&(!t=r(e))||“function”==typeof(n=v.call(t,“constructor”)&&t.constructor)&&a.call(n)==1},is
e)return!0;return!0},globalEval:function(e,t){b(e,{nonce:t&&t.nonce}),each:function(e,t){var n,r=0;if(d(e)){for(n=e.length;r<n;r++)if(!1==
e)if(!1==t.call(e[r],r,e[r]))break;return e},trim:function(e){return null==e?“”:e+“”).replace(p,“”),makeArray:function(e,t){var n=t||[];r
k.merge(n,“string”==typeof e?[e]:e).u.call(n,e),n},isArray:function(e,t,n){return null==t?-1:i.call(t,e,n)},merge:function(e,t){for(var
n=t.length,r=0,i=e.length;r<n;r++)e[i++]=t[r];return e.length=i,e},grep:function(e,t,n){for(var r=[],i=0,o=e.length,a=!n;i<o;i++)!t(e[i],i
r),map:function(e,t,n){var r,i,o=0,a=[];if(d(e))for(r=e.length;r<o;r++)null!=(i=t(e[o],o,n))&&a.push(i);else for(o in e)null!=(i=t(e[o],o,n
g.apply([],a)),guid:1,support:y},”function”==typeof Symbol&&(k.fn[Symbol.iterator]=t[Symbol.iterator]),k.each(“Boolean Number String Functi
Symbol”.split(“ ”).function(e,t){n[“fobject”+t+“l”]=t.toLowerCase();})var h=function(n){var e,d,b,o,i,h,f,e,w,u,l,T,C,a,E,v,s,c,v,k=“sizzle”
```


Na sua página HTML, você irá referenciar o arquivo com todo o código jQuery.

```
<footer>
  <!-- É comum se carregar o script do jQuery no footer, por causa da renderização da página-->
  <script src="jquery.js"></script>
  <script src="scripts.js"></script>
</footer>
```

Você também deverá criar um arquivo para poder referenciar os scripts que serão escritos.

A sintaxe básica do jQuery pode parecer meio complicada no início, mas com algumas horas de prática você já estará expert!!!

```
$(function(){
})
```

--> essa é a sintaxe básica de todo código jQuery, dentro dele você irá escrever suas funções.

Exemplo base, escondendo um parágrafo ao apertar botão:

O primeiro passo é criar a página HTML com os componentes:

```
<html>

  <body>

    <p>esse paragrafo vai ser escondido no clique</p>

    <button id="b1">clique aqui para esconder</button>

    <footer>

      <script src="jquery.js"></script>

      <script src="apostila.js"></script>

    </footer>
```

```
</body>

</html>
```

O arquivo jquery.js contém a biblioteca jQuery e o apostila.js vai conter o script que iremos escrever.

```
$(function(){
})
```

-> o primeiro passo é abrir a função padrão. Agora devemos colocar a função que aciona o botão no clique...

```
$(function(){
  $('#b1').click(function(){
  })
})
```

-> o b1 é o id que atribuímos para o botão, .click serve para chamar no clique e o function serve para atribuir a ação que vai acontecer no clique. Agora para finalizar precisamos chamar o parágrafo e escondê-lo:

```
$(function(){
  $('#b1').click(function(){
    $('#p').hide();
  })
})
```

-> o .hide serve justamente para esconder, mas se a intenção for mostrar algo escondido basta utilizar o .show .

Agora é só testar:

esse paragrafo vai ser escondido no clique

clique aqui para esconder

clique aqui para esconder

100% funcional!!!!

Esse exemplo foi apenas para te apresentar o jQuery, note que ele complica muito menos a chamada de eventos do que o JavaScript puro, então é altamente recomendado utilizá-lo a fim de obter um desenvolvimento ágil. Você pode encontrar a documentação do jQuery em: <https://api.jquery.com/>

FIM!!!

Referências:

-<https://www.w3schools.com/js/>

-<https://medium.com/@wellyngtonamaral/o-que-voc%C3%AA-deveria-saber-sobre-javascript-antes-de-programar-em-javascript-884eff23025c>