



PERNAMBUCO  
GOVERNO DO ESTADO

# Web Design Avançado

Ewerton Mendonça



**Curso Técnico em Informática**

Educação a Distância

2019



# Web Design Avançado

Ewerton Mendonça

**Curso Técnico em Informática**

Educação a Distância

2.ed. | Ago. 2019



Licença Pública Creative Commons  
Atribuição-NãoComercial-Compartilhual 4.0 Internacional

**Professor(es) Autor(es)**

Ewerton Mendonça

**Revisão de Língua Portuguesa**

Letícia Garcia

**Design Educacional**

Deyvid Souza Nascimento

Renata Marques de Otero

**Diagramação (2.ed. 2019)**

Jailson Miranda

**Catálogo e Normalização**

Hugo Cavalcanti (Crb-4 2129)

---

**Elaborado por Hugo Carlos  
Cavalcanti | CRB-4 2129**

**Coordenação de Curso**

Anderson Elias

**Coordenação Executiva**

George Bento Catunda

Terezinha Mônica Sinício Beltrão

**Coordenação Geral**

Paulo Fernando de Vasconcelos Dutra

Conteúdo produzido para os Cursos Técnicos da  
**Secretaria Executiva de Educação Profissional de  
Pernambuco, em convênio com o Ministério da  
Educação  
(Rede e-Tec Brasil).**

**1.ed. | junho, 2017**

**Dados Internacionais de  
Catalogação na Publicação (CIP) de  
acordo com ISDB**

---



## Sumário

Introdução.....	7
1.Competência 01   Conhecer os Fundamentos do Javascript.....	8
1.1 Internet.....	8
1.2 Modelo Cliente/Servidor.....	10
1.3 Padrões Web.....	11
1.4 O que é Programação.....	12
1.5 Javascript.....	13
1.6 Document Object Model.....	14
1.7 IDE.....	15
1.8 Testando.....	16
1.9 Primeiro código.....	17
1.10 O que pode ser feito em Javascript.....	18
1.11 Onde escrever o código Javascript.....	18
1.12 Saída de resultado JavaScript.....	19
2.Competência 02   Conhecer as Técnicas de Programação Procedural com o Javascript.....	21
2.1 Sintaxe.....	21
2.2 Literais e variáveis.....	21
2.3 Declaração de variáveis e atribuição de valores.....	22
2.4 Operadores e expressões.....	22
2.5 Identificadores.....	23
2.6 Operadores comparativos.....	24
2.7 Operadores lógicos.....	25
2.8 Precedência.....	25
2.9 Estruturas condicionais.....	26
2.9.1 If...else.....	26



2.9.2 Switch...case.....	29
2.10 Estruturas de repetição.....	31
2.10.1 For.....	32
2.10.2 While.....	34
2.10.3 Do...while.....	35
2.11 Endentando o código.....	36
3.Competência 03   Conhecer as Técnicas de Programação Orientada a Objetos com o Javascript..	37
3.1 Primitivas Javascript.....	37
3.2 Objetos Javascript.....	38
3.3 Propriedades de um objeto.....	38
3.4 Métodos de um objeto.....	38
3.5 Criando um objeto.....	38
3.5.1 Criando um objeto literal.....	39
3.5.2 Criando com o comando new.....	39
3.5.3 Criando com o construtor do objeto.....	40
3.6 A palavra-chave this.....	40
3.7 Manipulando propriedades de um objeto.....	41
3.8 Manipulando métodos de um objeto.....	41
3.9 Prototype.....	42
4.Competência 04   Conhecer os Fundamentos de JQuery.....	43
4.1 Adicionando a biblioteca em sua página.....	44
4.1.1 Fazendo o download do JQuery.....	44
4.1.2 Utilizando um repositório.....	45
4.2 sintaxe JQuery.....	46
4.3 Evento ready.....	46
4.4 Seletores.....	47



4.4.1 Selecionando Tag.....	47
4.4.2 Selecionando pelo atributo ID.....	48
4.4.3 Selecionando pelo atributo class.....	49
4.4.4 Exemplos de seletores.....	49
4.5 Eventos JQuery.....	50
4.6 Obtendo a tag, o texto e o valor de um campo.....	52
4.7 Obtendo o valor de um atributo.....	53
4.8 Definindo conteúdo.....	53
5.Competência 05   Utilizar os Efeitos do JQuery.....	55
5.1 Esconder (hide) e mostrar (show).....	55
5.1.1 Efeito toggle.....	56
5.2 Efeito de desaparecimento gradual (fade).....	56
5.3 Efeito de deslizar.....	58
5.4 Efeito de animação.....	58
5.5 Encadeamento de métodos.....	59
6.Competência 06   Utilizar o Bootstrap Framework.....	61
Conclusão.....	67
Referências.....	68
Minicurriculo do Professor.....	69
Ewerton Mendonça.....	69



## Introdução

Bem-vindo ao curso de Web Design Avançado. Nele vamos aprender do começo como programar utilizando uma linguagem muito simples chamada JavaScript. Mas, não é porque ela é muito simples que é fácil aprender a programar. Algumas pessoas levam mais tempo que outras e pessoas organizadas, meticolosas e regradas levam vantagens. Caso você tenha se esforçado, mas não foi suficiente para aprender, não se desespere, apenas você precisa de mais tempo para aprender. Então, seja paciente, vá no seu ritmo e nunca desista. Temos milhões de exemplos, tutoriais e videoaulas na internet que podem lhe ajudar. Só tome o cuidado de coletar material novo, dos últimos dois anos pelo menos.

Como falei, aprender programação não é fácil e você precisa compreender um conceito antes de seguir para o próximo. Coloque em mente que o aprendizado é como se fosse uma escada. Para atingir os degraus superiores, você tem que subir no mais próximo. Caso tente pular degraus pode cair e voltar para o chão, de uma forma bem dolorida. Então, suba um degrau de cada vez.

Aprender programação também pode ser comparado com aprender qualquer coisa nesta vida, mas vamos comparar com tocar violão ou dirigir. Primeiro, algumas pessoas aprendem mais rápido que outras, é normal, e não significa que há pessoa que não aprenda. Só vai ter que se esforçar mais. Outro ponto em comum é que não dá para aprender a tocar violão ou dirigir um carro, sem um violão ou sem um carro. Não basta ler este caderno de estudo, você precisa fazer os exemplos e ver o resultado. Se acostumar com a tecnologia. Por último e mais importante, para se aprender a tocar violão você precisa tocar o violão... muito. Tem de praticar. De preferência todos os dias. A linguagem, ou seja, o que significa os comandos, dá para ensinar, mas ensinar a pensar, isso ninguém consegue. Cada um raciocina de uma forma diferente, encontrando soluções de seu jeito, e cada aluno aprende a pensar da sua forma pessoal. Mas como se aprende a raciocinar? Com certeza não é assistindo aula. É fazendo, e fazendo, e fazendo, e fazendo exercícios. Mais uma dica. Você não pode ver a resposta do exercício. Se ver, perdeu o exercício, ele não serve mais. Você precisa descobrir a resposta. Cada questão é como uma charada.

Ufa! Realmente é trabalhoso esse processo, mas é o melhor. Não se engane que assistindo a videoaula na internet você vai aprender. Seria como aprender a tocar violão apenas assistindo o outro tocar violão, ou aprender a dirigir sem ligar um carro, ou ficar forte na academia apenas assistindo alguém fazer exercício. Ridículo né? Com programação é igual.



## 1.Competência 01 | Conhecer os Fundamentos do Javascript

A linguagem JavaScript faz parte do mundo da web. Ela foi criada para que páginas web respondessem a eventos. Este é seu mundo. Nesta competência vamos conhecer este mundo. Além disso, vamos conhecer alguns conceitos iniciais sobre o universo das linguagens de programação, para poder comparar e saber onde o JavaScript se encontra nesta babel da programação.

Como o JavaScript afeta os elementos das páginas HTML, é extremamente necessário saber HTML e CSS. Neste curso focamos apenas no JavaScript, então você não vai encontrar material que te ensine sobre o HTML e CSS. O curso Introdução a Web Design é que tem esse foco e você encontrará nele todo o material necessário para aprender HTML e CSS de forma introdutória.

Avisos dados, vamos começar o trabalho.

### 1.1 Internet

A internet é a rede global de computadores. Ela não possui um centro ou uma empresa que a controla. São bilhões de dispositivos interligados, e quando falo em dispositivos falo de computadores de mesa, smartphones, impressoras, roteadores, carros, relógios, geladeiras, enfim, qualquer coisa que se conecte com a internet.

Ela foi criada a pedido dos militares dos Estados Unidos na época da Guerra Fria como um sistema de comunicação com suporte forte a falhas. A ideia era que, caso uma bomba atômica atingisse o solo americano, o país não ficaria sem comunicação.

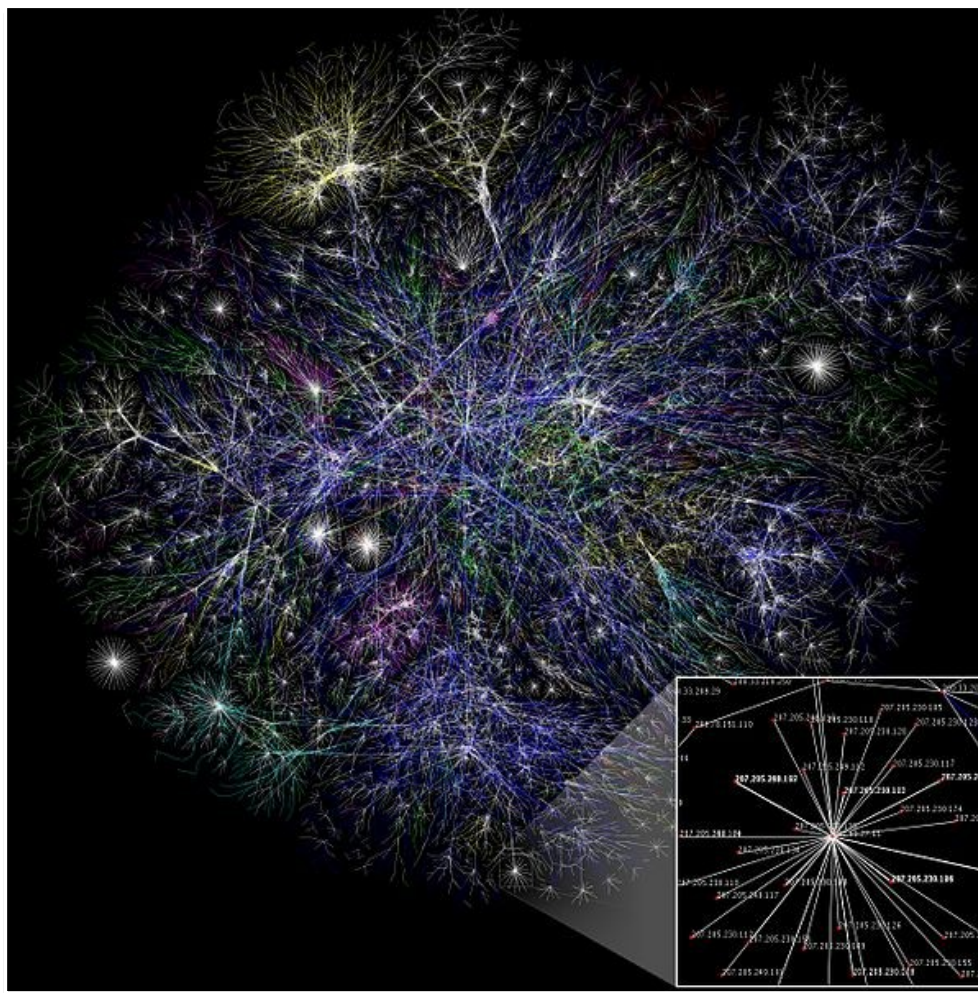


As bombas atômicas são um grande medo da humanidade. Assista ao vídeo no link abaixo e veja a quantidade de bombas atômicas disparadas no mundo desde que elas foram inventadas. Muitas delas são testes, mas, de qualquer forma, foram disparadas no planeta desde 1945.

Das instalações militares a internet ganhou os centros de pesquisa e universidades, depois invadiu o setor privado, as empresas e a casa das pessoas. Hoje está no nosso bolso. Será que no futuro a internet estará dentro de nós?



A internet é uma rede de redes. Ela não é uma unidade, é um conjunto. Temos as redes das universidades e de empresas ligadas umas às outras formando a internet. Quando você compra um roteador para distribuir a internet pela sua casa, você está criando mais uma rede e conectando ela a internet. A junção de todas as redes conectadas forma a internet.



**Figura 1 - Mapa parcial da Internet em 15 de janeiro de 2005, baseado em dados da opte.org.**

**Fonte:** [www.opte.org/maps/](http://www.opte.org/maps/)

**Descrição:** a imagem mostra um gráfico de linhas coloridas conectadas.

O JavaScript é uma linguagem criada neste mundo.

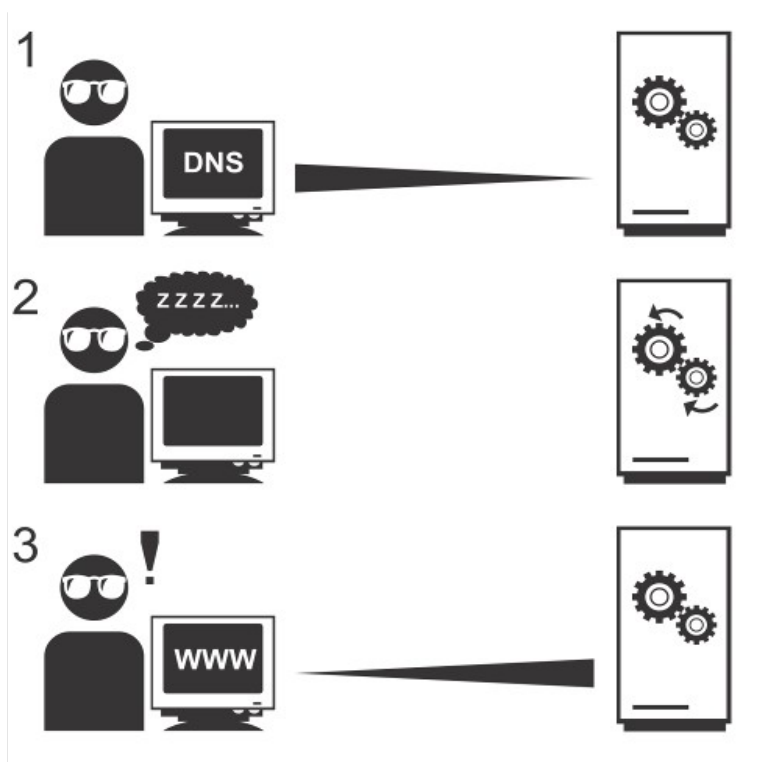


Este é um brevíssimo resumo sobre o assunto. Tem muito mais coisa. Acesse o link abaixo para ter acesso a mais informações.

## 1.2 Modelo Cliente/Servidor

A internet foi construída de uma forma. A esta forma chamamos arquitetura. A arquitetura da internet segue o modelo cliente/servidor. Neste modelo os pontos finais da rede internet são computadores que fornecem algum serviço. Podemos ter tanto um computador muito poderoso, que fornece o serviço de páginas web, como uma impressora que fornece o serviço de impressão em rede. Temos vários programas que fornecem diversos serviços, como o já citado servidor de páginas web. Temos também o servidor de e-mail, o servidor de arquivos, o servidor de impressão e muitos outros. Observe que classificamos esses programas como servidores.

Os programas que solicitam um serviço são chamados de clientes. Assim, você utiliza um navegador web, como o Chrome ou Firefox, para solicitar uma página web. O navegador é um programa cliente. O servidor de páginas web recebe sua solicitação e envia a página para seu navegador. Sempre será assim, com programas clientes solicitando recursos de programas servidores.



**Figura 2 - passos simplificados da solicitação de uma página web para uma aplicação.**

**Fonte:** Ewerton Mendonça.

**Descrição:** a ilustração é dividida em três fases: a requisição do usuário para o servidor, o processamento do servidor e a entrega do resultado do processamento ao usuário.



Vamos aprender como utilizar o JavaScript no lado do cliente. Isso significa que ele vai funcionar nos navegadores web, seja ele Chrome, Firefox, Opera ou Edge. Mas tenha em mente que navegadores antigos podem não executar o JavaScript adequadamente. Você pode escrever seu código perfeito, mas seu navegador, por ser antigo, pode não rodar o programa adequadamente.



Programadores utilizam o termo 'rodar' o programa para dizer que ele trabalhará, ou seja, o sistema operacional executará o programa. Com o advento dos smartphones, os programas passaram a ser chamados de aplicativos ou pela sua abreviação: app.

## 1.3 Padrões Web

Internet e web são duas coisas diferentes. A internet é o meio, é por onde trafegam os dados. Ela dá suporte aos serviços. Em uma comparação simples, seria como a rede de estradas de nosso país. Vários serviços são realizados através das estradas do Brasil, como o turismo e a entrega de mercadorias. A web é um desses serviços, que usa a internet para entregar páginas web.

No começo da web ela era bem simples e os fabricantes de navegadores ampliavam suas características para oferecer um serviço melhor. Para você ter uma ideia, quando a internet surgiu ela só possuía texto preto em um fundo branco, e só. Muita coisa mudou de lá para cá, não foi?

O problema era que, quando um fabricante de navegador modificava o HTML, os outros não faziam, ou faziam de modo diferente. Aí uma página que funcionava no Internet Explorer, não funcionava no Netscape Navigator, e vice-versa. Um pesadelo para desenvolvedores web. Essa situação continuou por um longo tempo, até que algumas das maiores empresas com interesse no crescimento da web se juntaram para estabelecer um consórcio e definir padrões. Com isso nasceu a W3C.

A W3C estuda e fornece um padrão para as tecnologias ligadas à web. Sua última definição foi o HTML 5, que é uma marca que reúne as versões mais novas do HTML, CSS e JavaScript. A W3C não implementa navegador nenhum, ela apenas escreve um documento que diz como os navegadores devem realizar uma determinada ação, cada fabricante é que fica encarregado de fazer seu navegador seguindo o padrão. Como todos seguem o padrão, hoje em dia



temos a garantia de que uma página será exibida de forma igual em todos os navegadores modernos.



Padrões web vão muito além do HTML e do CSS. Descubra mais sobre ele no link abaixo.

## 1.4 O que é Programação

O HTML é uma linguagem de marcação e o CSS é uma configuração de exibição dos elementos. Nem uma nem outra são linguagem de programação, mas o JavaScript é, e por isso ele é mais difícil de ser aprendido, significando que você deve se dedicar mais para aprendê-lo.

Existem diversas linguagens de programação, como o PHP, Java, Python, C, ASP.NET etc. Cada uma é forte em algum aspecto e serve para um contexto. Algumas delas são concorrentes como o PHP, Java para web e o ASP.NET. O JavaScript não possui um concorrente, ela é um padrão web. Foi desenvolvido inicialmente para que o navegador o executasse, ao invés do sistema operacional. Mas o que é uma linguagem de programação?

Pense no computador como seu servo. Ele fará qualquer coisa que você disser. Mas ele tem a linguagem dele, então você tem que aprender a falar no idioma dele para dar-lhe ordens. Ele também é extremamente obediente e não importa o que você acha que ele deveria saber, ele fará exatamente o que você disser. Mesmo que esteja errado. Assim, se você dissesse para ele:

1. Tire o lixo;
2. Varra a casa;
3. Limpe a cozinha.

Ele primeiro tiraria o lixo, depois varreria a casa e, por fim, limparia a cozinha. Seria mais inteligente fazer ao contrário. Mas ele não discutirá com você. Apenas fará, estando certo ou errado. Então, suas ordens devem estar na ordem correta.

Outra coisa é que se você escrever qualquer palavra errada, ou na sintaxe errada, ele vai ignorar o comando, e as coisas vão dar erradas.





Então, é assim que trabalharemos, escrevendo uma lista ordenada de ordens escritas em JavaScript e pedindo para o navegador executar aquelas ordens.

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h2>JavaScript</h2>
6
7 <p id="demo">Texto de um parágrafo.</p>
8
9 <button type="button" onclick="minhaFuncao()">Try it</button>
10
11 <script>
12     function minhaFuncao() {
13         document.getElementById("demo").innerHTML = "Texto modificado.";
14     }
15 </script>
16
17 </body>
18 </html>
```

Figura 3 – código de uma página HTML com o código JavaScript está escrito

Fonte: Ewerton Mendonça.

**Descrição:** exibição do código de uma página HTML onde o código JavaScript está escrito nela. As linhas 12, 13 e 14 são escritas em JavaScript. O restante está em HTML.

## 1.5 Javascript

Brendan Eich trabalhava na empresa Netscape quando desenvolveu o JavaScript. Na época, seu nome era Mocha, o nome de um tipo de café. Posteriormente passou a se chamar LiveScript, antes de, finalmente, ganhar o nome JavaScript. LiveScript foi o nome oficial da linguagem quando foi lançada pela primeira vez na versão beta do navegador Netscape 2.0 em setembro de 1995, mas teve seu nome mudado em um anúncio conjunto com a Sun Microsystems em dezembro de 1995 quando foi implementado no navegador Netscape versão 2.0B3.

Quando a Netscape, que criou o navegador Netscape Navigator, adicionou suporte à tecnologia Java em seu navegador, através dos Applets Java, eles decidiram aproveitar a popularidade da marca e substituir de LiveScript para JavaScript. Isto causou uma grande confusão, dando a impressão de que Java e JavaScript eram a mesma coisa.

JavaScript rapidamente adquiriu ampla aceitação como linguagem de script do lado do cliente de páginas web. Como consequência, a Microsoft desenvolveu um dialeto compatível com a linguagem de nome JScript. JScript adicionou novos métodos para consertar métodos do JavaScript



relacionados à data que apresentavam problemas. JScript foi incluído no Internet Explorer 3.0, liberado em agosto de 1996. JavaScript e JScript são tão similares que os dois termos são comumente usados como sinônimos. A Microsoft, entretanto, declara muitas características nas quais JScript não conforma com a especificação do JavaScript.

Em novembro de 1996 a Netscape anunciou que tinha submetido JavaScript para ECMA Internacional como candidato a padrão industrial e o trabalho subsequente resultou na versão padronizada chamada ECMAScript.

JavaScript tem se transformado na linguagem de programação mais popular da web. Inicialmente, no entanto, muitos profissionais denegriram a linguagem, pois ela tinha como alvo principal o público leigo. Com o advento do Ajax, JavaScript teve sua popularidade de volta e recebeu mais atenção profissional. O resultado foi a proliferação de frameworks e bibliotecas, como o JQuery e o Bootstrap, práticas de programação melhoradas.



Conheça mais sobre o JavaScript no link abaixo:

## 1.6 Document Object Model

Um arquivo HTML é um arquivo de texto simples. Nesse texto, temos as tags e o texto que é o conteúdo do documento. As tags HTML estão umas dentro das outras, como se fossem caixas dentro de caixas, o que chamamos de Modelo de Caixa, ou Box Model. Quando o navegador lê o documento HTML ele cria na memória uma estrutura de dados parecido com uma árvore de cabeça para baixo, contendo todo o conteúdo dos elementos. Essa preparação é necessária para poder criar a representação visual da página. A criação dessa representação visual é chamada de Render Tree. O JavaScript tem o poder de manipular esta árvore alterando como o documento é exibido. O Document Object Model, ou DOM, era criado de formas diferentes pelos diversos navegadores, o que gerava todos os problemas que existiam no JavaScript. Com a definição de um padrão para criação da árvore, este problema foi resolvido e é através do DOM que acessamos todos os elementos da página e até de componentes do navegador.

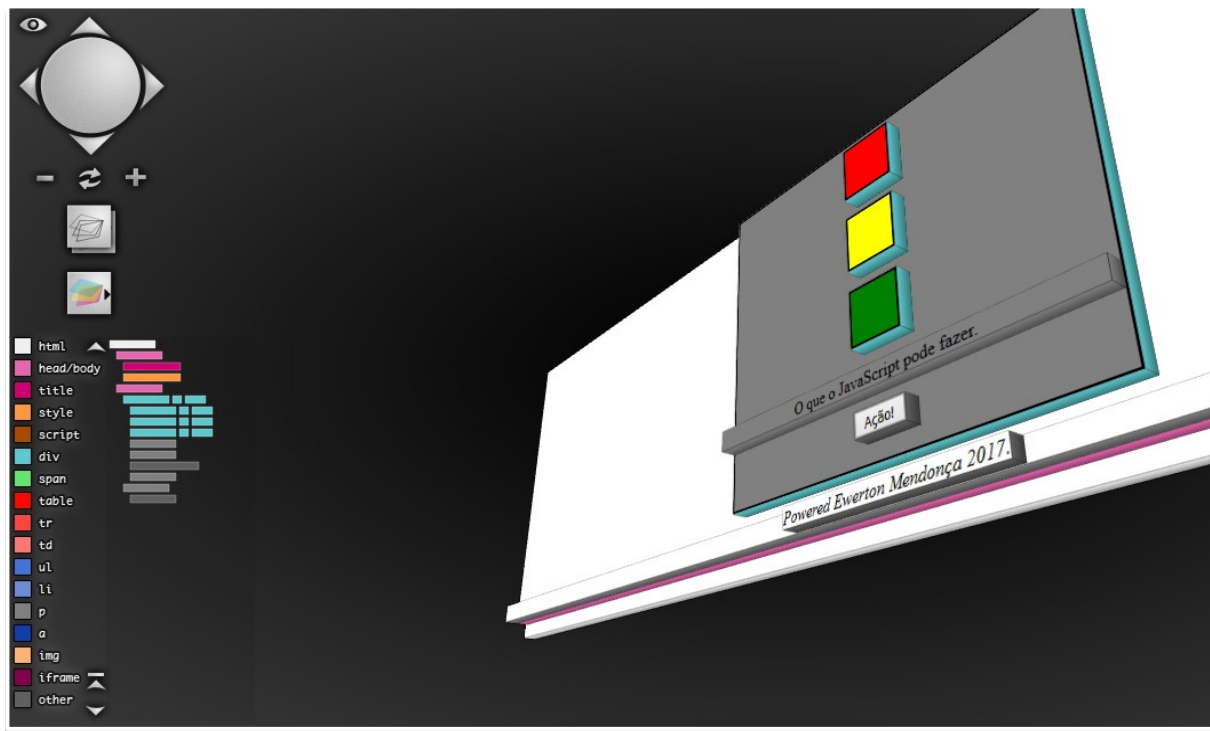


Figura 4 – exibição 3D da árvore DOM do código de exemplo.

Fonte: Ewerton Mendonça.

**Descrição:** O plug-in Tilt do Firefox mostra de forma 3D a árvore DOM através de camadas, onde uma caixa dentro de outra é vista como uma camada acima.

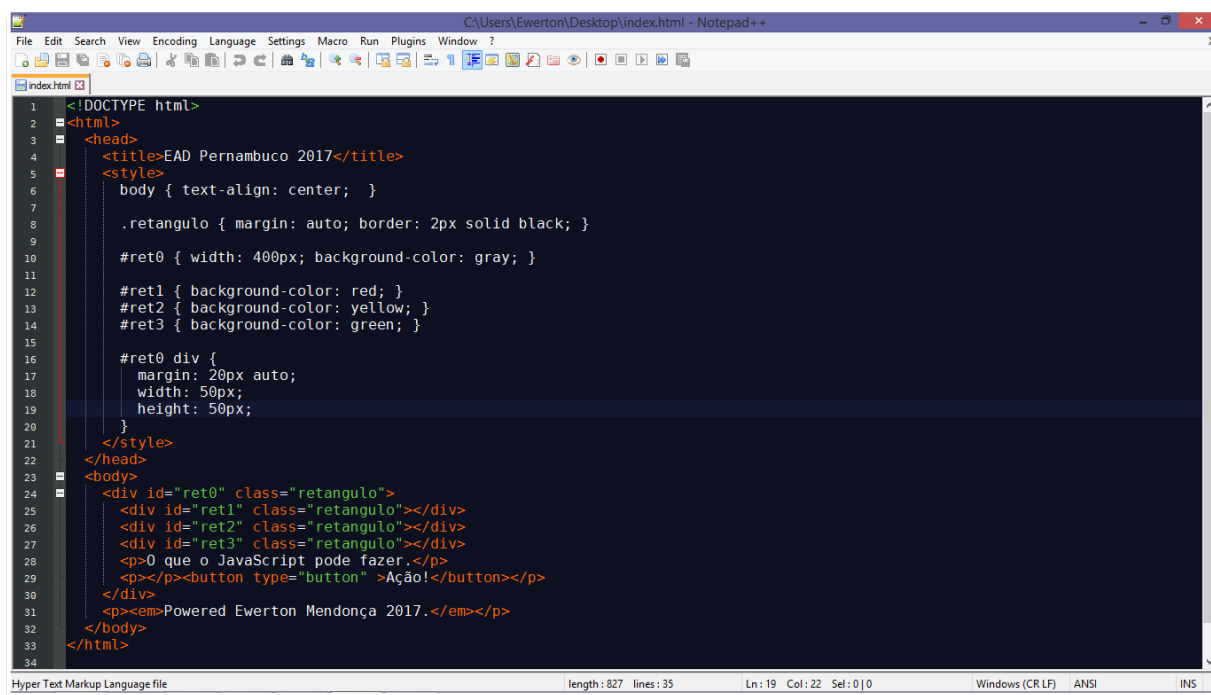


## 1.7 IDE

Agora que você já conhece um pouco sobre a web, vamos colocar a mão na massa. Mas que programa podemos utilizar para construir páginas web?

Você pode fazer um site web completo utilizando apenas o Bloco de Notas, Notepad do Windows ou qualquer editor de texto. Não utilize o Word. Nesse caso, é melhor utilizar o Bloco de Notas por ser um editor mais simples. Como sugestão, mas ele não é necessário, temos o Notepad++.

Um arquivo HTML é um texto que o navegador interpreta e monta para você visualizar. Como uma receita, são as instruções para preparar um bolo, por exemplo.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>EAD Pernambuco 2017</title>
5 <style>
6   body { text-align: center; }
7
8   .retangulo { margin: auto; border: 2px solid black; }
9
10  #ret0 { width: 400px; background-color: gray; }
11
12  #ret1 { background-color: red; }
13  #ret2 { background-color: yellow; }
14  #ret3 { background-color: green; }
15
16  #ret0 div {
17    margin: 20px auto;
18    width: 50px;
19    height: 50px;
20  }
21 </style>
22 </head>
23 <body>
24 <div id="ret0" class="retangulo">
25   <div id="ret1" class="retangulo"></div>
26   <div id="ret2" class="retangulo"></div>
27   <div id="ret3" class="retangulo"></div>
28   <p>0 que o JavaScript pode fazer.</p>
29   <p></p><button type="button" >Ação!</button></p>
30 </div>
31 <p><em>Powered Ewerton Mendonça 2017.</em></p>
32 </body>
33 </html>
```

Figura 5 – Notepad++ com o código HTML e CSS.

Fonte: Ewerton Mendonça.

**Descrição:** a imagem mostra o Notepad++ com o código HTML e CSS que utilizaremos nos exemplos. Você pode copiar esse código para utilizá-lo no Bloco de Notas do Windows. Você não precisa de uma IDE para fazer o código JavaScript.

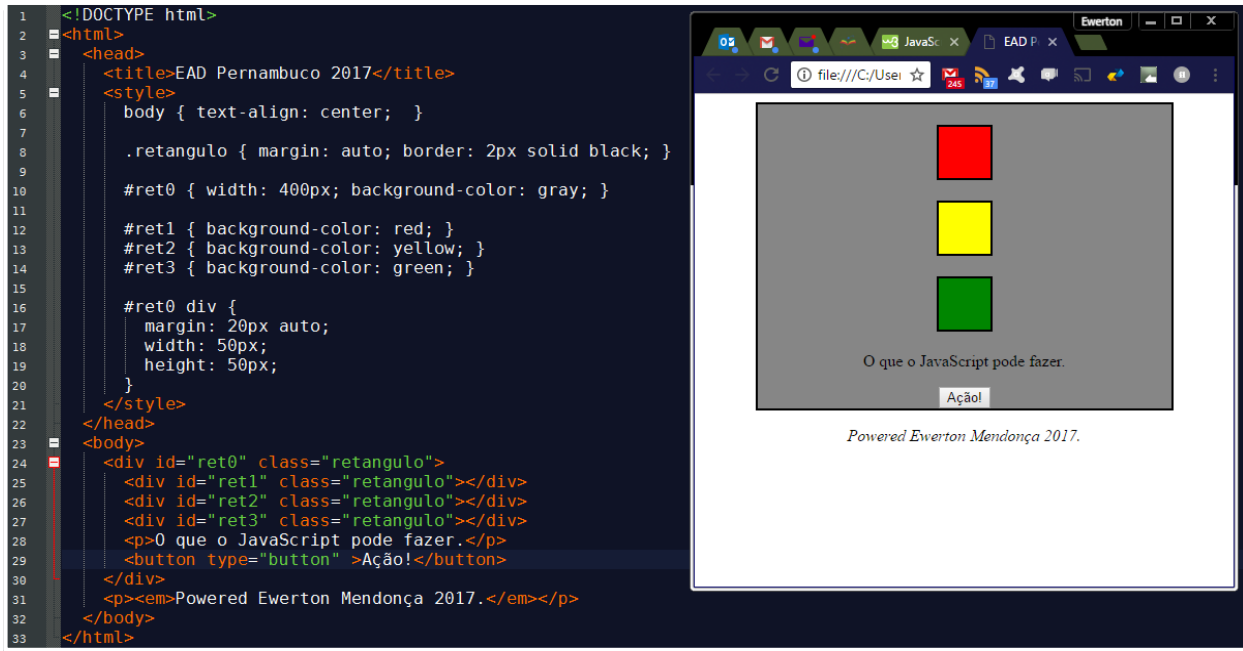
## 1.8 Testando

Para testar o código JavaScript você precisará de um navegador. Sugerimos o Chrome, Firefox ou Edge. Evite utilizar o Internet Explorer da Microsoft.





Para ver a página de exemplo da Figura 5 basta arrastar o arquivo para o navegador. Ele renderizará a página, exibindo o documento. Observe o Resultado na Figura 6.



**Figura 6 – Código HTML com CSS embutido e exibição do resultado no navegador.**

**Fonte:** Ewerton Mendonça.

**Descrição:** exibição do código HTML no Notepad++ e sua renderização no Navegador Chrome.

## 1.9 Primeiro código

Agora que você já sabe onde escrever seu código, que precisa de um HTML para seu JavaScript manipular e onde testar, vamos escrever um código bem simples e verificar se ele está funcionando.

Um aviso! O computador é muito rígido com o que você escreve. Se algo não funcionar, procure algum erro de digitação. Maiúsculas e minúsculas são diferentes, então, respeite. Copie o código da Figura 7, as partes que foram modificadas no código estão destacadas em vermelho. Atualize o navegador com o novo código e clique no botão [Ação!] para fazer o navegador aparecer uma caixa de alerta com seu nome. Para aparecer seu nome, substitua o texto entre aspas duplas no código, onde está escrito “Coloque seu nome aqui.”.

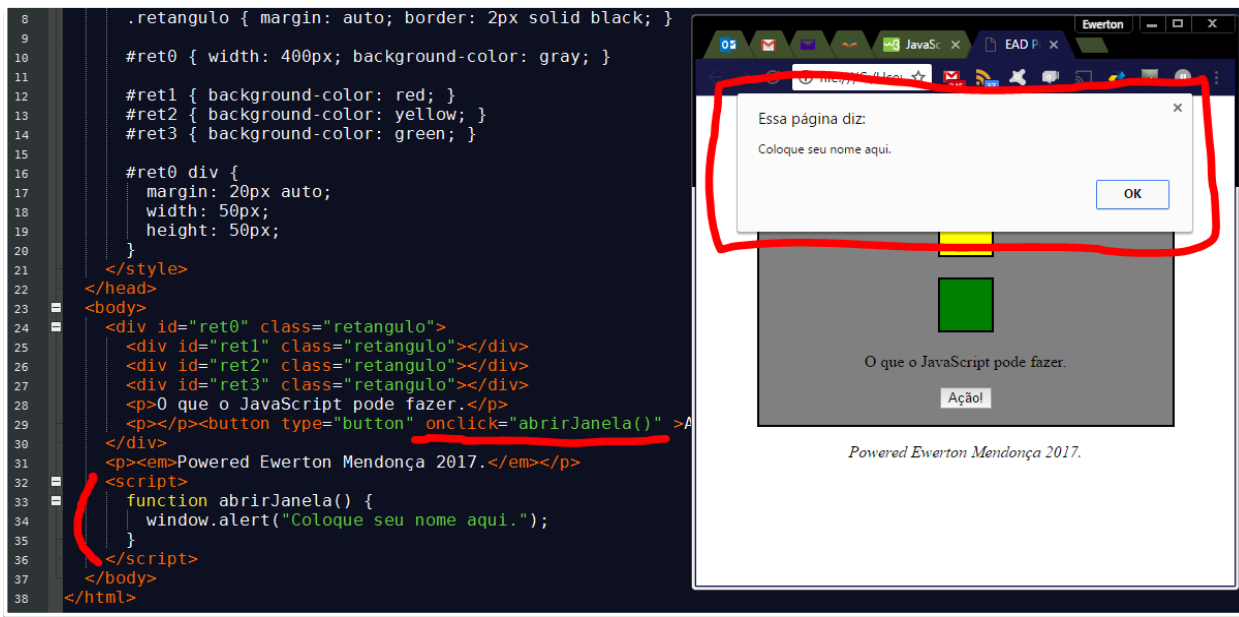


Figura 7 – código de exemplo no Notepad++ e navegador Chrome exibindo o resultado.

Fonte: Ewerton Mendonça.

**Descrição:** Código HTML de exemplo com o código JavaScript destacado em vermelho e exibição do resultado no navegador com a caixa de diálogo circundada em vermelho.

## 1.10 O que pode ser feito em Javascript

O JavaScript manipula o navegador e a página web. Ele também possui algumas características a mais.

- JavaScript pode mudar o conteúdo dos elementos HTML;
- JavaScript pode mudar o valor dos atributos dos elementos HTML;
- JavaScript pode mudar o CSS de um elemento HTML;
- JavaScript pode esconder e mostrar elementos HTML;
- JavaScript pode reagir a eventos;
- JavaScript pode se comunicar com o servidor web.

## 1.11 Onde escrever o código Javascript

Podemos inserir o código JavaScript no documento HTML utilizando a tag `<script>`. Fizemos dessa forma no código da Figura 7. Podemos escrever a tag `<script>` na tag `<head>` ou na `<body>`. Caso você precise utilizar essa forma, prefira escrever no final da tag `<body>`, assim todos



os elementos são carregados antes do JavaScript. Essa não é uma boa estratégia porque o JavaScript fica misturado com o HTML e, quando ele cresce, fica muito confuso.

A melhor opção é escrever seu código JavaScript em um arquivo separado com a extensão .js e fazer seu HTML puxá-lo. Para associar o arquivo JavaScript com o documento HTML utilize a tag `<script>` com o atributo `'src'` com o endereço do arquivo .js onde está seu código JavaScript. Observe um exemplo na Figura 8, temos dois arquivos: o **index.html** e o **script.js**.

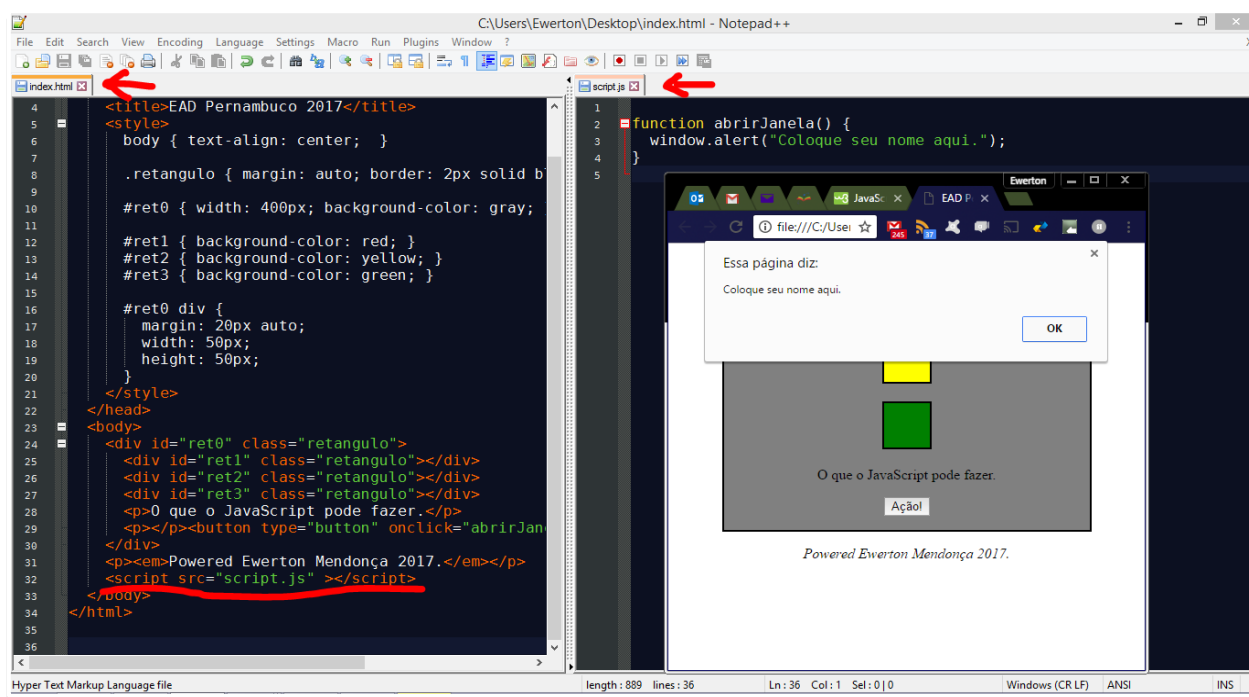


Figura 8 – código de exemplo no Notepad++ e navegador Chrome exibindo o resultado.

Fonte: Ewerton Mendonça.

**Descrição:** arquivo HTML com a tag `<script>` destacada em vermelho e o JavaScript no arquivo `script.js`, com o navegador exibindo o resultado.

Lembre-se de que a melhor forma de fazer é em um arquivo separado, mas, por uma questão de espaço no caderno de estudo, podemos colocar o código JavaScript junto ao HTML.

## 1.12 Saída de resultado JavaScript

As exibições de resultado de computação, como o cálculo de operações matemáticas, por exemplo, são realizadas de quatro maneiras em JavaScript.

- Você pode exibir resultados escrevendo em um elemento HTML com **innerHTML**;
- Você pode utilizar todo o documento HTML com **document.write()**;



- Como fizemos no código da Figura 7, utilizando uma caixa de alerta com **window.alert();**
- Escrevendo no console do navegador com **console.log()**.

Vamos utilizar nos exemplos as formas mais simples.

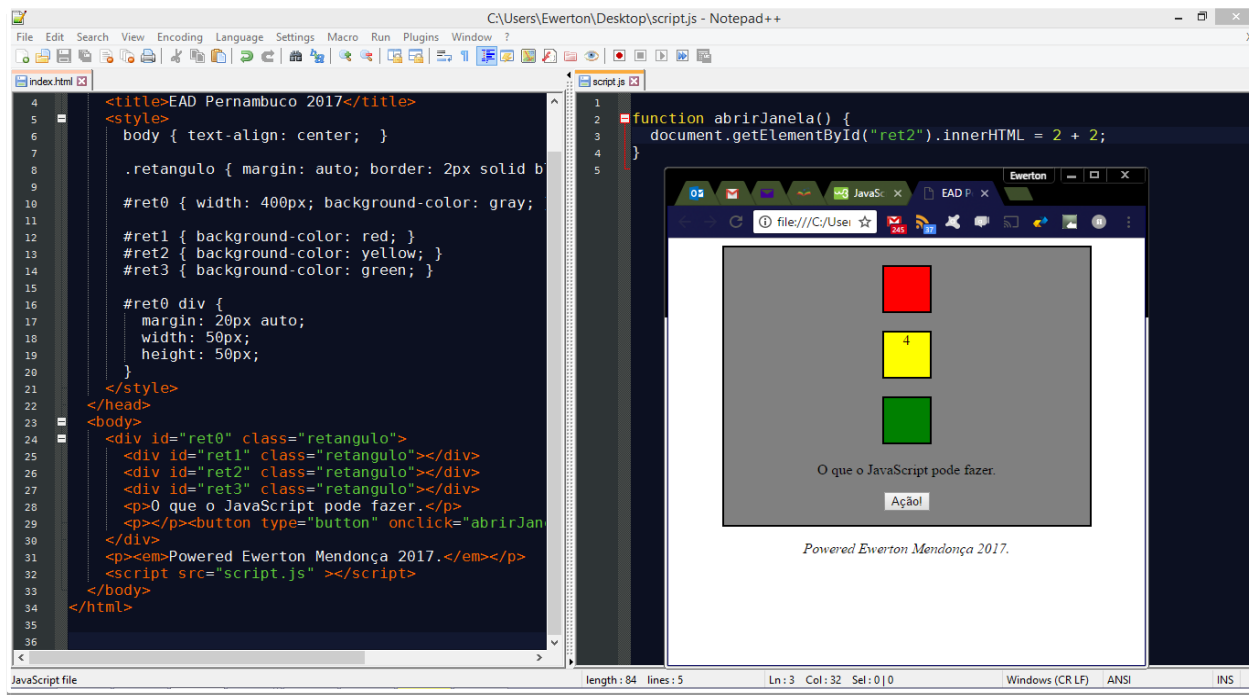


Figura 9 – código HTML e JavaScript no Notepad++ e Exibição do resultado no navegador.

Fonte: Ewerton Mendonça.

**Descrição:** Exemplo de exibição de resultado através da inscrição de texto em um elemento HTML. Ao apertar no botão [Ação!] o resultado da soma de 2 + 2 é mostrado no elemento com a identificação 'ret2', em amarelo. Observe o código JavaScript acima da janela do navegador.

Com isso, finalizamos a primeira competência e você deve entender absolutamente tudo para seguir adiante. Caso ainda não tenha compreendido, faça uma pesquisa na internet. Você encontrará muito material escrito e em videoaula, apenas tenha o cuidado de ler somente o material atualizado. Você ainda pode utilizar o fórum do AVA colocando a sua dúvida no título do post para que os tutores possam identificar mais rápido sua necessidade.

Repita os exemplos e modifique-os para entender melhor. Ainda não explicamos muita coisa. Deixamos para a próxima competência para não o sobrecarregar, então foque apenas no que foi explicado.



## 2.Competência 02 | Conhecer as Técnicas de Programação Procedural com o Javascript

A programação procedural é o estilo usado na maioria das linguagens de programação. Quando se aprende uma linguagem procedural as outras ficam mais fáceis porque a estrutura não muda, o que muda são detalhes de escrita ou alguma especificação, como atalhos.

Nesta competência vamos aprender o básico sobre programação. Este conhecimento será útil no aprendizado de qualquer linguagem, pois as estruturas básicas são praticamente as mesmas em todas as linguagens classificadas como procedurais.

A sintaxe de uma linguagem são as regras utilizadas para organizar as palavras-chaves. Começaremos com as regras básicas da sintaxe JavaScript.

### 2.1 Sintaxe

Um programa de computador é um conjunto - “instruções” que são “executadas” por um computador. Em uma linguagem de programação as instruções são comandos. Os comandos em JavaScript são separados por ponto e vírgula (;). Os comandos podem ser valores, operadores, expressões, palavras-chaves e comentários.

### 2.2 Literais e variáveis

Em JavaScript temos definidos dois tipos de valores: valores constantes e valores variáveis. Os valores constantes são chamados de **literals** e os valores variáveis são chamados de **variáveis**. Os valores literais de números fracionários são escritos utilizando ponto ao invés da vírgula e os valores de texto são escritos entre aspas duplas ou simples. Tanto faz.

```
1
2 5 // valor literal inteiro
3 1000.24 // valor literal fracionário
4 "Texto 1" // valor literal de texto
5 "Texto 1" // valor literal de texto
6
```

Figura 10 – Valores literais em JavaScript.

Fonte: Ewerton Mendonça.

Descrição: exibição de valores literais inteiro, fracionário e texto em JavaScript.



Com os valores literais podemos realizar qualquer operação matemática. Você pode calcular a média de notas, a área de um quadrado, seu imposto de renda etc.

## 2.3 Declaração de variáveis e atribuição de valores

Em uma linguagem de programação, variáveis são usadas para armazenar valores. JavaScript usa a palavra-chave `var` para ordenar a criação de uma variável ao computador. Para criar uma variável utilizamos a palavra-chave `'var'` seguida de um nome que identifique ela. Esse identificador deve ser único em todo o programa. Um sinal de igual (=) é utilizado para definir um valor para aquela variável. Assim, no exemplo da Figura 11, temos duas variáveis, `x` com o valor de 2 e `y` com o valor de 3. Podemos mudar o valor da variável quantas vezes precisarmos com o operador de atribuição (=).

```
1  var x;           // comando para criar uma variável.
2  x = 2;           // comando para atribuir um valor a variável.
3
4  var y = 3;       // comando único para criar e atribuir.
```

Figura 11 – Declaração de variável e atribuição de valores em JavaScript.

Fonte: Ewerton Mendonça.

**Descrição:** código de exemplo que cria a variável `x` e atribui o valor 2 a ela, e cria a variável `y` com o valor de 3 no mesmo comando.

## 2.4 Operadores e expressões

Da mesma forma que matemática, podemos criar expressões com os operadores matemáticos soma (+), subtração (-), multiplicação (\*) e divisão (/). Uma expressão é um conjunto que pode ser composto por valores e operadores que dão um resultado. Por exemplo `2 + 10 / 2`, que vai dar 7. Primeiro são calculados os operadores multiplicativos (\*) e (/) e depois os aditivos (+ e -).

Concatenar é unir dois pedaços. Podemos concatenar dois literais de texto com o operador de soma (+). Por exemplo: `"Fulano" + " de " + "Tal"`, tem o resultado `"Fulano de Tal"`.

A linguagem JavaScript tem um conjunto de palavras-chaves limitado. Uma palavra-chave é um comando. Observe na Figura 11 que `'var'` é uma palavra-chave usada para ordenar a criação de variáveis.

Observe também que na Figura 10 e 11 escrevemos um texto explicativo. Ele é ignorado pelo navegador por ser um comentário. Um texto é entendido como um comentário quando é





iniciado por duas barras ( // ). Caso o comentário seja de várias linhas, utilizamos no começo ( /\* ) e no final ( \*/ ). Veja o exemplo na Figura 12.

```
1  var x = 2;      // Este texto é um comentário de uma linha.
2
3  /*
4  Este é um comentário de várias linhas.
5  A linha de baixo também é ignorada.
6  var x = 3;
7  */
8
9
10 function abrirJanela() {    // Aqui é ignorado
11     document.getElementById("ret2").innerHTML = 2 + 2;
12 }
```

Figura 12 – Exemplo de comentários.

Fonte: Ewerton Mendonça.

Descrição: exemplo de código JavaScript com comentários de uma e várias linhas.

## 2.5 Identificadores

Identificadores são nomes. Utilizamos identificadores para nomear variáveis, funções e rótulos. Mas devemos seguir algumas regras para nomear identificadores.

- O primeiro caractere deve ser uma letra, sublinhado ( \_ ) ou o cifrão ( \$ ).
- Os caracteres subsequentes podem ser letras, números, sublinhado e cifrão.

JavaScript é sensível à maiúsculas e minúsculas, então identificadores como amor, Amor, aMoR e AMOR são todos diferentes. A maioria dos erros dos iniciantes são de digitação.



```
1 function abrirJanela() {  
2     var x, y; // Criamos duas variáveis.  
3     x = 5.5; y = 8 // Atribuímos valores.  
4  
5     // pegamos o elemento 'ret2' no HTML.  
6     var d = document.getElementById("ret2")  
7  
8     // calculamos a média e colocamos o valor no elemento.  
9     d.innerHTML = ( x + y ) / 2;  
10 }  
11  
12
```

Figura 13 – código de exemplo no Notepad++ e navegador Chrome exibindo o resultado.

Fonte: Ewerton Mendonça.

**Descrição:** código de exemplo no Notepad++ e navegador Chrome exibindo o resultado. Agora quando o botão é apertado, ele calcula a média dos valores das variáveis x e y.

## 2.6 Operadores comparativos

Os operadores comparativos comparam duas expressões e resultam em **verdadeiro** ou **falso**. Por exemplo:  $3 > 2$  é igual a verdadeiro, e  $50 - 20 < 10 + 2$  é falso. Além de maior que e menor que temos outros operadores.





Operador	Descrição
>	Maior que
<	Menor que
>=	Maior que ou igual
<=	Menor que ou igual
==	Igual
!=	Diferente

**Tabela 1 – Tabela com os operadores comparativos.**

**Fonte:** Ewerton Mendonça.

**Descrição:** tabela que relaciona os operadores comparativos e sua descrição.

## 2.7 Operadores lógicos

Os operadores lógicos comparam dois valores lógicos e resultam em um valor lógico.

Operador	Descrição
&&	Operador lógico 'e'
	Operador lógico 'ou'
!	Operador lógico 'não', inverte o valor lógico.

**Tabela 2 – Tabela com os operadores lógico.**

**Fonte:** Ewerton Mendonça.

**Descrição:** tabela que relaciona os operadores lógicos e sua descrição.

Tabela com os valores possíveis dos operadores lógicos.

&& ('e' lógico)	('ou' lógico)	! ('não' lógico) Baseado na expressão 1
V	V	F
F	V	F
F	V	V
F	F	V

**Tabela 3 – Tabela verdade com os operadores lógico.**

**Fonte:** Ewerton Mendonça.

**Descrição:** tabela verdade com os resultados possíveis em operações lógicas.

## 2.8 Precedência

Podemos ter uma grande sequência de operadores em uma expressão. No entanto, o computador só pode calcular uma operação de cada vez e ainda deve seguir as regras matemáticas. Assim, ele tem uma ordem de precedência a seguir. A tabela 4 lista os operadores em JavaScript e informa quem é realizado primeiro do que o outro.



Descrição	Exemplo
As expressões em parênteses internos são resolvidas primeiro.	$(1 + (4 / 2)) - 3$
Multiplicativos	$2 * 3 + 1$
Aditivos	$2 - 1 > 3$
Comparativos	$3 > 2 == 1$
Igualdades	$3 == 2 \&\& \text{true}$
Lógicos	$X = \text{true}    \text{false}$
Atribuição	$X = 3$

**Tabela 4 – Tabela de precedência de operadores.**

**Fonte:** Ewerton Mendonça.

**Descrição:** Tabela de precedência de operadores com os respectivos itens

## 2.9 Estruturas condicionais

O fluxo normal de leitura dos comandos é de cima para baixo, da mesma forma que lemos um texto aqui no ocidente. No entanto, por vezes, na programação, precisamos desviar o fluxo, seja para evitá-lo em certas condições, como para repeti-lo.

São certos momentos da programação que, se for de um jeito faça algo, senão for faça outro. Esse tipo de fluxo é muito comum e depende de um teste. JavaScript é como a maioria das linguagens de programação procedurais, temos dois tipos de estruturas com esta característica:

- IF...ELSE;
- SWITCH...CASE.

Vamos conhecer cada uma delas com exemplos.

### 2.9.1 If...else

Uma das estruturas procedurais mais antigas e úteis é a estrutura condicional IF. Em português ela quer dizer 'se'. Ela tem o poder de desviar o fluxo normal de leitura das linhas na dependência de um teste. Se o teste resultar em verdadeiro, a linha processa, senão, o bloco de código é evitado. Observe o exemplo da Figura 14.

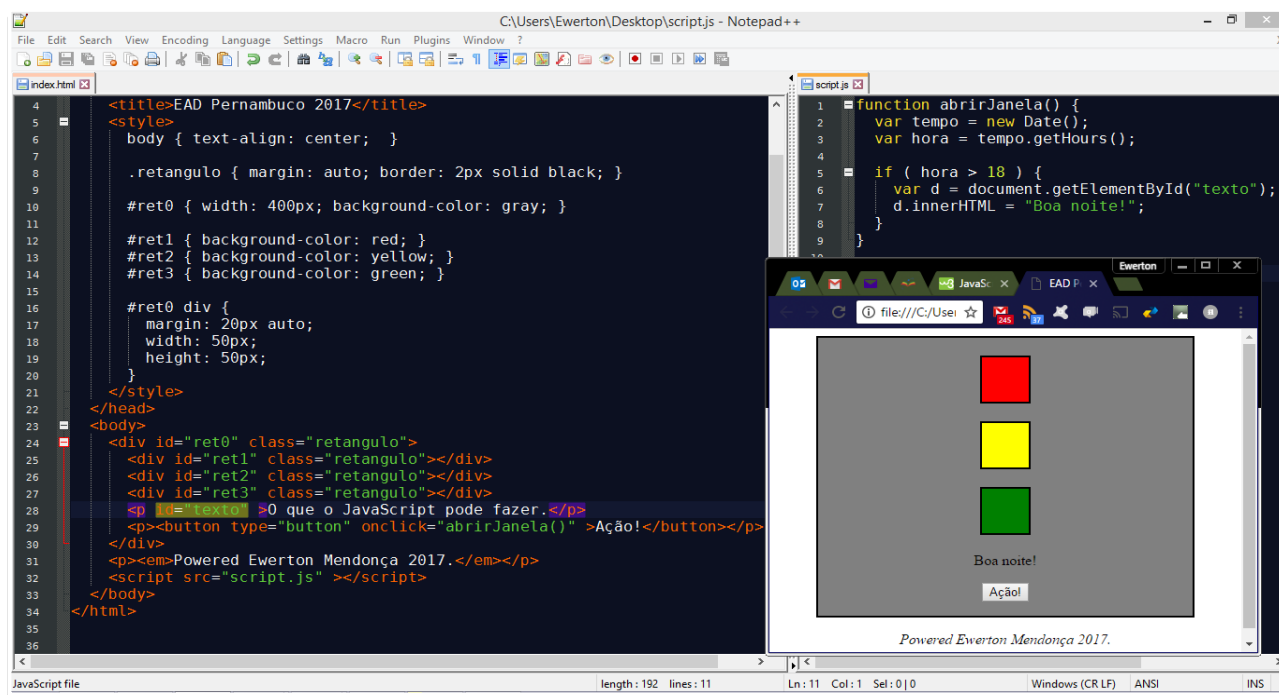


Figura 14 – Código de exemplo da estrutura if.

Fonte: Ewerton Mendonça.

**Descrição:** código de exemplo no Notepad++ e navegador Chrome exibindo o resultado.

Atente que houve pequenas mudanças no arquivo HTML. No arquivo **script.js** temos as seguintes linhas e suas explicações:

**Linha 1** – A primeira linha cria uma função que é executada quando o botão é pressionado. Veremos o que é função mais adiante.

**Linha 2** – Precisamos de algo que nos dê a hora do sistema. Essa linha pega o tempo do computador e guarda na variável tempo.

**Linha 3** – O tempo do sistema possui data, hora, segundo e outros valores. Nesta linha pegamos apenas o valor da hora. Esse valor vai de 0 até 23 horas.

**Linha 5** – Fazemos um teste “hora > 18”. Dependendo da hora esse teste pode ser verdadeiro ou falso. Se o teste resultar em verdade, o bloco de comandos delimitado por chaves { } é executado, senão, ele será pulado, ou seja, ignorado.

**Linha 6** – Esta e a próxima linha serão executadas apenas se o teste resultar em verdade. Então, pegamos no HTML um elemento com a id de valor “texto”.

**Linha 7** – Nesta linha substituímos seu conteúdo por “Boa noite!”.

**Linha 8** – Final do bloco do if. Se o teste resultar em falso, o fluxo será desviado para esta linha.



É claro que você tem que testar esse código a noite para ver seu resultado. Isso não seria tão conveniente. Gostaríamos que, se o teste resultasse em falso, ele desse um 'Bom dia!', o que é muito possível.

A instrução if possui um complemento opcional, o 'else'. O 'else' significa 'senão'. Se for verdade faça isso, senão faça aquilo. Vou mudar o código para que se o teste resultar em falso passarmos outra mensagem.

```
1  = function abrirJanela() {  
2      var tempo = new Date();  
3      var hora = tempo.getHours();  
4  
5  =  if ( hora > 18 ) {  
6      var d = document.getElementById("texto");  
7      d.innerHTML = "Boa noite!";  
8  } else {  
9      var d = document.getElementById("texto");  
10     d.innerHTML = "Bom dia!";  
11 }  
12 }
```

**Figura 15 – Código da estrutura if com o else.**

**Fonte:** Ewerton Mendonça.

**Descrição:** Código de exemplo da estrutura if com o else exibido no Notepad++.

**Linha 8 –** Marca o início do bloco 'else'.

As outras linhas são cópias e já foram explicadas.

Podemos colocar estruturas dentro de estruturas porque, afinal, todos são comandos. Observe a Figura 16 com o código para 'Bom dia!', 'Boa tarde!' e 'Boa noite!'. Fiz ele diferente para te fazer pensar um pouquinho. Tenha bastante atenção nas linhas e procure entender o que acontece.



```
1 function abrirJanela() {
2     var tempo = new Date();
3     var hora = tempo.getHours();
4
5     if ( hora < 12 ) {
6         var d = document.getElementById("texto");
7         d.innerHTML = "Bom dia!";
8     } else {
9         if ( hora < 18 ) {
10            var d = document.getElementById("texto");
11            d.innerHTML = "Boa tarde!";
12        } else {
13            var d = document.getElementById("texto");
14            d.innerHTML = "Boa noite!";
15        }
16    }
17 }
```

Figura 16 – Código com estrutura if aninhadas.

Fonte: Ewerton Mendonça.

Descrição: código de exemplo com estrutura if aninhadas no Notepad++.

## 2.9.2 Switch...case

Outra estrutura condicional é a switch...case. Ela também funciona desviando o fluxo, mas de uma forma bem diferente. No switch...case temos um bloco de código com linhas marcadas com case. O teste do switch compara um valor com cada um dos cases do bloco de código e quando acha um que bata, ou seja, que seja igual, ele prossegue daquele ponto em diante. Vamos ver um exemplo na Figura 17.

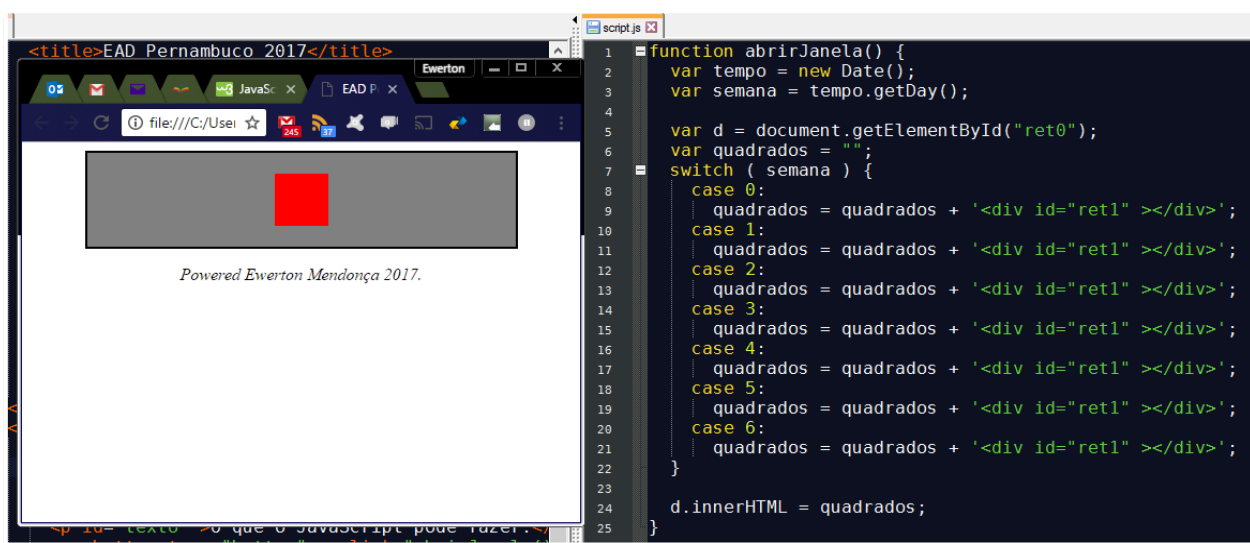


Figura 17 – código da estrutura switch...case no Notepad++ e navegador Chrome exibindo o resultado.

Fonte: Ewerton Mendonça.



**Descrição:** Código de exemplo da estrutura switch...case. Como o teste foi realizado no sábado, então, o número comparado foi 6, apenas um quadrado. Se fosse o domingo, que é o zero, teria sido criado 7 quadrados vermelhos.

Abaixo, temos a explicação das linhas que são diferentes do exemplo anterior:

**Linha 3** – Declaramos a variável semana que guarda o dia da semana que vai de 0 até 6, começando pelo domingo, o zero.

**Linha 6** – Declaramos uma variável quadrado que vai acumulado o texto com as tags <div> para na linha 24 colocar dentro da tag com a id 'ret0'.

**Linha 7** – Início da estrutura switch onde fica a variável que será testada.

**Linha 8** – Esta é a primeira marcação. Se a variável tiver o valor 0, o fluxo começa desta linha e vai até o final. Acontece da mesma forma com as linhas 10, 12, 14, 16, 18 e 20. Se não for nenhum dos valores, o fluxo vai para a linha 22.

Pode parecer estranho que o fluxo comece em uma linha e siga adiante sem parar. Mas caso você queira que ele pare, podemos incluir o comando 'break' que encerra o fluxo, direcionando para o final. O exemplo da Figura 18 funciona da mesma forma, mas o break interrompe o fluxo contínuo quando é lido. Assim, apenas o nome do dia da semana é escrito.

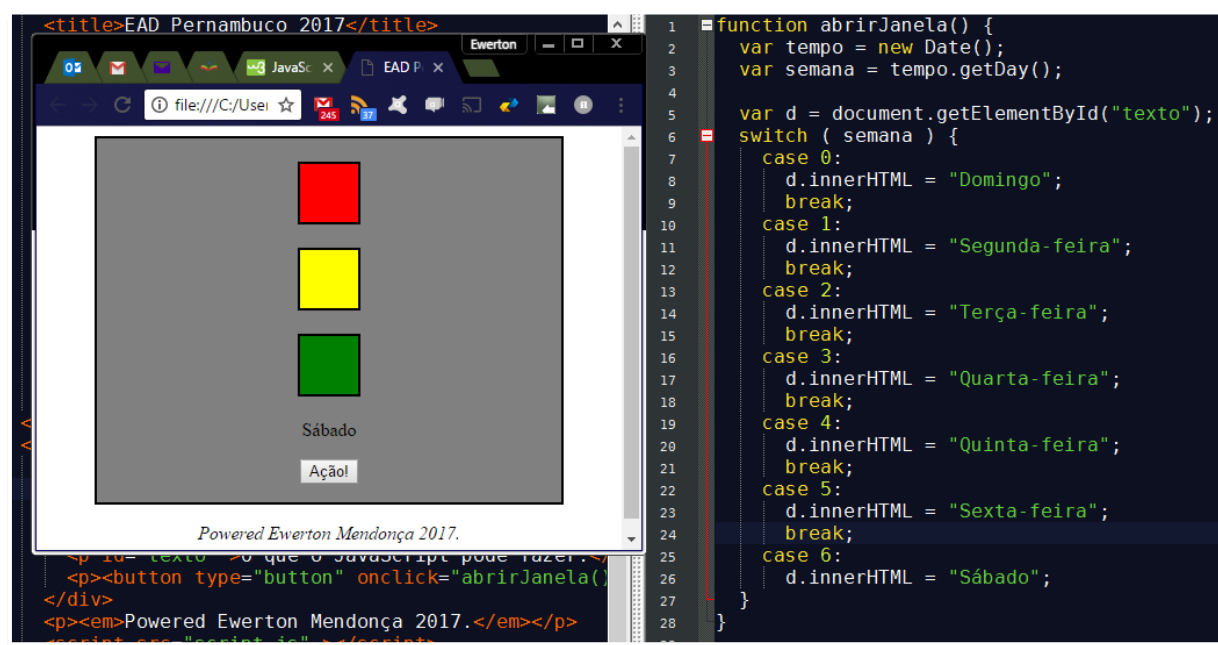


Figura 18 – código estrutura switch...case com o uso do comando break no Notepad++ e navegador Chrome exibindo o resultado.

Fonte: Ewerton Mendonça.

**Descrição:** Código de exemplo da estrutura switch...case com o uso do comando break. O teste foi feito em um sábado. O break desvia o fluxo para o final do switch, na linha 27.





Ainda temos a cláusula opcional 'default'. Caso não bata com nenhum 'case' a cláusula 'default' é executada. No exemplo da Figura 19, se não for sábado nem domingo, será escrito "Dia de trabalho."

```
1  = function abrirJanela() {  
2      var tempo = new Date();  
3      var semana = tempo.getDay();  
4  
5      var d = document.getElementById("texto");  
6  = switch ( semana ) {  
7      case 0:  
8          d.innerHTML = "Domingo";  
9          break;  
10     case 6:  
11         d.innerHTML = "Sábado";  
12         break;  
13     default:  
14         d.innerHTML = "Dia de trabalho.";  
15     }  
16 }
```

Figura 19 – Código da estrutura switch...case com a cláusula default

Fonte: Ewerton Mendonça.

Descrição: Código de exemplo no Notepad++ da estrutura switch...case com a cláusula default.

## 2.10 Estruturas de repetição

As estruturas de repetição são como as estruturas de condicionais, elas servem para desviar o fluxo, mas com outra intenção. No caso delas, o fluxo é desviado para repetir um bloco de comandos. Também é realizado um teste para saber até quando será repetido. Se você errar o momento de parar, pode ser que ele nunca chegue a parar e a sensação é de que tudo está travado, mas o problema é que nunca chega a condição de parada.

Outro nome para esse estilo de fluxo é loop, ou laço. Loop é o termo em inglês para aquelas voltas que o avião dá. Pesquise vídeos na internet sobre loop para ter uma ideia melhor.

Veremos três estruturas de repetição:

- FOR;
- WHILE;
- DO...WHILE.

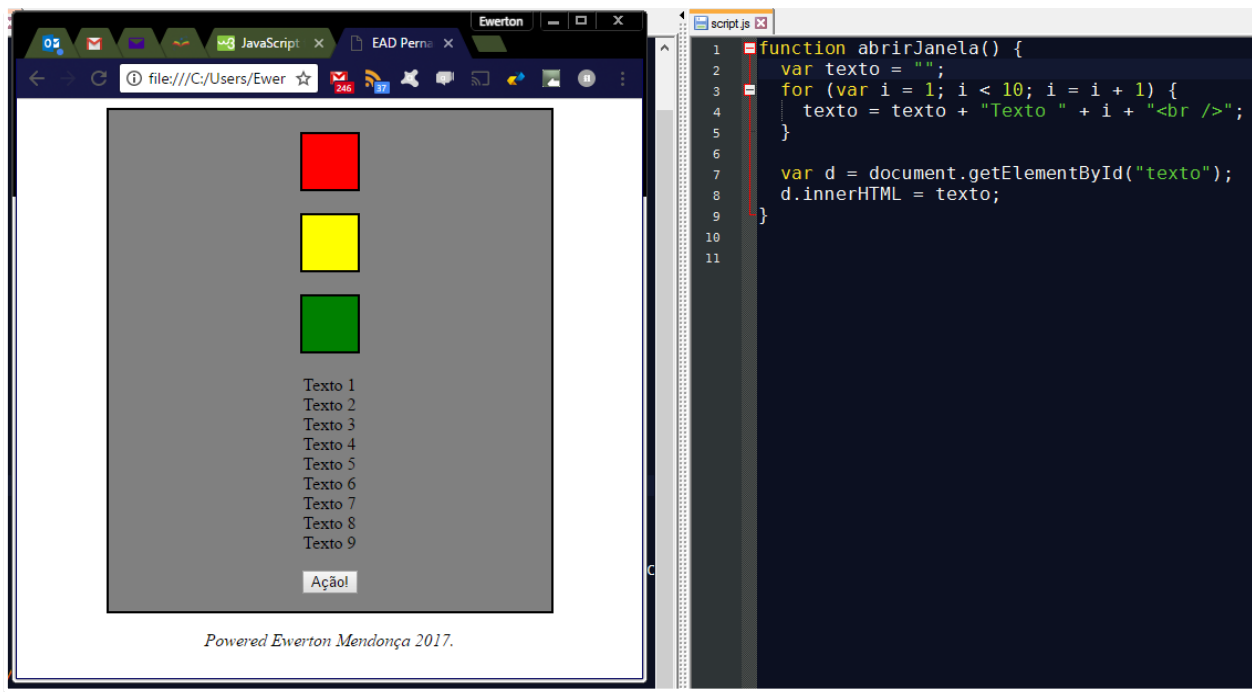
Elas fazem a mesma coisa e podemos utilizar qualquer uma delas em todos os momentos, só que cada uma possui características próprias que facilitam a vida em um



determinado momento. Vamos conhecer essas estruturas e suas características para podermos decidir sabiamente em quais momentos devemos utilizá-las.

## 2.10.1 For

O laço for é a mais antiga das estruturas de repetição. Ela possui três parâmetros que determinam sua condição de parada. Ela também utiliza, em sua composição, uma variável de controle que pode ser criada na estrutura ou podemos utilizar qualquer outra variável numérica criada anteriormente. Vamos ver um exemplo e a explicação das linhas.



**Figura 20 – Código de exemplo da estrutura de repetição for.**

**Fonte:** Ewerton Mendonça.

**Descrição:** código de exemplo no Notepad++ da estrutura de repetição for e navegador Chrome exibindo o resultado.

As estruturas de repetição são excelentes para se fazer um trabalho repetitivo. No caso do for, é quando sabemos exatamente quantas repetições têm de ser feitas. Agora vamos às explicações das linhas.

**Linha 2** – Criamos uma variável ‘texto’ para guardar as linhas que serão adicionadas no HTML.

**Linha 3** – Essa é a estrutura for. Começa pela palavra-chave ‘for’ e depois criamos uma variável ‘i’ para ser nosso contador, que começa a contar de 1. Depois do ponto e vírgula colocamos



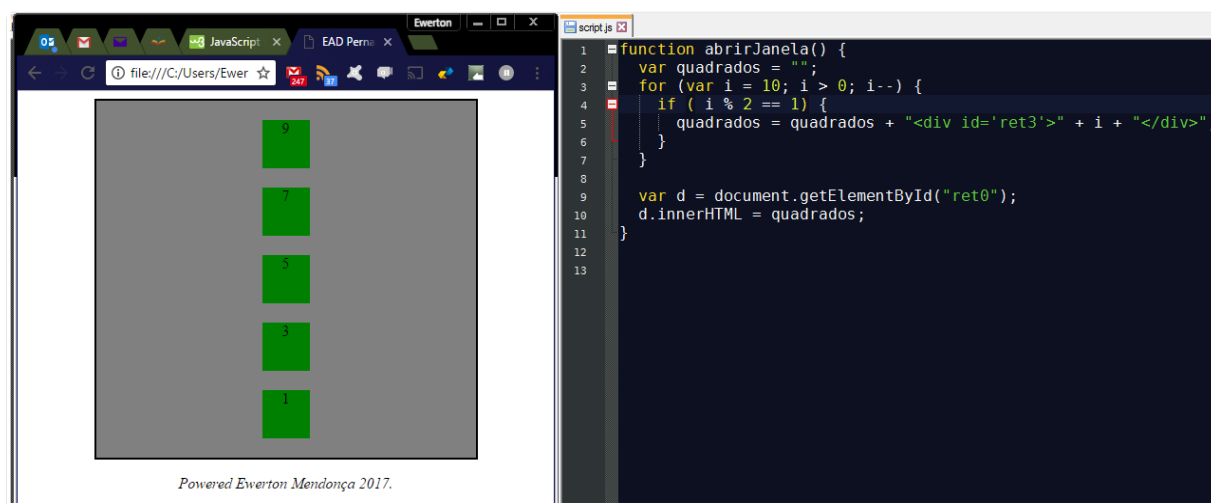


um teste ' $i < 10$ ', enquanto esse teste for verdadeiro, será executado o bloco seguinte. Aí, temos outro ponto e vírgula e como será adicionado os valores na variável de controle, que, nesse caso, é somando 1 ao que já está na variável. Poderíamos ter utilizado o atalho `i++` que adiciona 1 ou `i--` que subtrai 1 da variável, mas essa situação é se começarmos de um valor alto e formos diminuindo.

**Linha 4** – Guarda em 'texto' o que já estiver em 'texto' junto com 'Texto ', o valor de `i` e '`<br />`' para pular a linha no HTML.

O restante do código já explicamos anteriormente.

Podemos colocar estruturas dentro de estruturas para fazer aplicações mais complexas. Observe o exemplo da Figura 21 que utiliza o laço **for** do maior para o menor e um **if** para filtrar o que será exibido.



**Figura 21 – Código de exemplo no Notepad++ e navegador Chrome exibindo o resultado**

**Fonte:** Ewerton Mendonça.

**Descrição:** Código de exemplo com um laço for que conta do maior para o menor, faz um teste para saber se é ímpar o valor de `i`, e, se for, adiciona um quadrado com o valor de `i` no seu interior.

Vamos a explicação das linhas:

**Linha 2** – Criamos uma variável 'quadrados' para guardar a tag do quadrado verde.

**Linha 3** – Temos o laço for que começa de 10, testa se `i` é maior que 0 e vai diminuindo de 1 em 1 através do `i--`.

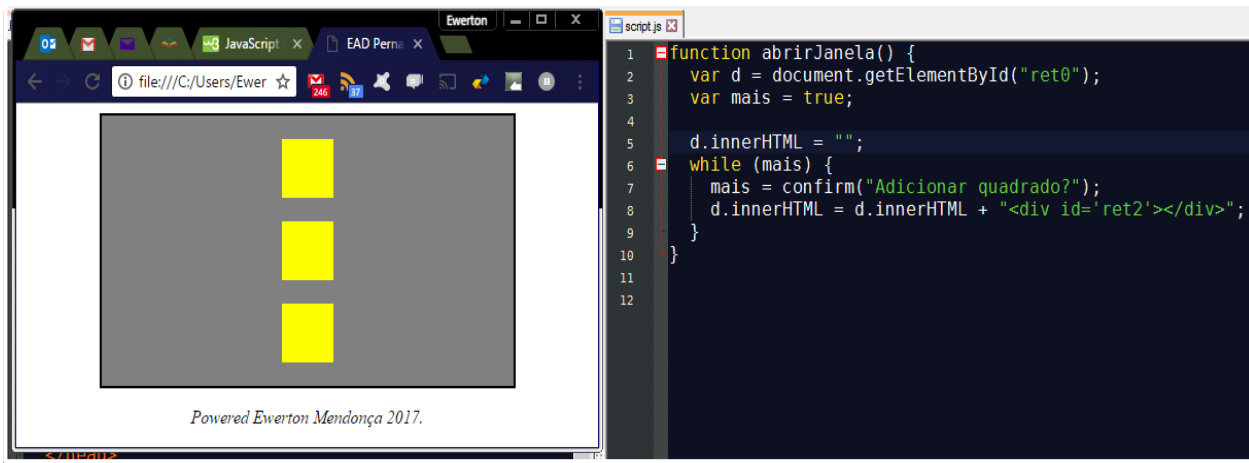
**Linha 4** – Utilizamos um if que fará o bloco, se o resto da divisão de `i` por 2 for 1. Essa expressão diz se o valor é ímpar ou par. Se for ímpar, o bloco é executado.

**Linha 5** – Adicionamos a variável 'quadrado' ao que ela já tem com mais uma tag do quadrado verde.



## 2.10.2 While

A estrutura while também serve para repetir. Ela faz seu teste, mas não possui, em sua estrutura, uma variável de controle. Ela é utilizada quando não sabemos quando a repetição deve parar.



**Figura 22 – Exemplo da estrutura while.**

**Fonte:** Ewerton Mendonça.

**Descrição:** código de exemplo da estrutura while no Notepad++ e navegador Chrome exibindo o resultado.

Vamos à explicação das linhas:

**Linha 2** – Pegamos o elemento HTML com o id ‘ret0’ para colocarmos os quadrados. Na Figura 26 é o retângulo cinza na página.

**Linha 3** – Vamos utilizar a variável ‘mais’ como o controle da repetição. Ela começa verdadeira, então, sempre teremos ao menos um quadrado amarelo. Definimos seu valor para verdadeiro com ‘true’.

**Linha 5** – Apagamos o conteúdo do elemento ‘ret0’.

**Linha 6** – Temos o início da estrutura while que testa a variável ‘mais’ para saber se ela é verdadeira. O laço vai se repetir enquanto ela for verdadeira.

**Linha 7** – O comando confirm abre uma caixa de diálogo que pergunta ‘ok’ ou ‘cancel’. Se o usuário apertar o botão de cancelar, a caixa retorna falso. Depois que o usuário decidir e apertar o botão que quer, colocamos o valor (verdadeiro ou falso) na variável ‘mais’.

**Linha 8** – Adicionamos ao que já estiver como conteúdo do elemento ‘ret0’ uma tag de quadrado amarelo.



**Linha 9** – Retornamos para a Linha 6 e fazemos o teste da variável ‘mais’ novamente. Se for falso, se encerra a repetição.

Gaste algum tempo analisando esse código para entender bem direitinho. Até agora ele é o mais importante para nós.

## 2.10.3 Do...while

Observe no exemplo da Figura 22 que se o valor da variável ‘mais’ for falso, não será repetido nenhuma vez. Isso acontece porque o teste é realizado no começo da estrutura, então, pode não ser realizado nenhuma vez. A estrutura do...while é uma variação do while onde seu teste é no final. Utilizando ela, sabemos que, ao menos uma vez, será realizado o trabalho. É a mesma estrutura utilizada no final de um jogo onde se pergunta ao jogador se ele quer jogar novamente. Veja o exemplo da Figura 23 que é o mesmo da Figura 22 mudando a estrutura. Observe que não precisamos definir um valor inicial para a variável ‘mais’.

```
1 function abrirJanela() {  
2     var d = document.getElementById("ret0");  
3     var mais;  
4  
5     d.innerHTML = "";  
6     do {  
7         d.innerHTML = d.innerHTML + "<div id='ret2'></div>";  
8         mais = confirm("Adicionar quadrado?");  
9     } while (mais);  
10 }
```

Figura 23 – Exemplo da estrutura do...while.

Fonte: Ewerton Mendonça.

**Descrição:** código de exemplo da estrutura do...while no Notepad++ e navegador Chrome exibindo o resultado.

## 2.11 Endentando o código

Observe que as estruturas podem ser utilizadas em conjunto para fazer programas mais complexos, e esse é o caso normal. Ou seja, normalmente, estamos programando estruturas dentro de estruturas, laços dentro de laços com estruturas condicionais dentro de estruturas condicionais e dentro delas outros laços. Como isso pode ficar muito complicado, utilizamos tabulações dentro dos níveis. Observe no exemplo da Figura 24 que as linhas 7 e 8 ficam mais afastadas por estarem em um bloco de código dentro de um bloco de código. Faça dessa forma que vai ajudar todo mundo

## Competência 02



que observar o seu código. Endentar o código é uma boa prática de programação que todos os profissionais utilizam.



## 3.Competência 03 | Conhecer as Técnicas de Programação Orientada a Objetos com o Javascript

Em JavaScript os objetos são os elementos mais poderosos. Se você entender o paradigma orientado a objeto, você entende de JavaScript.

Em JavaScript, quase “tudo” é um objeto:

- Booleanos (verdadeiro ou falso) podem ser objetos (se definidos com o comando **new**)
- Os números podem ser objetos (se definidos com o comando **new**)
- As Strings (texto) podem ser objetos (se definidos com o comando **new**)
- As datas são sempre objetos
- As matemáticas são sempre objetos
- Expressões regulares são sempre objetos
- Matrizes são sempre objetos
- As funções são sempre objetos
- Objetos são sempre objetos

Todos os valores JavaScript, exceto primitivos, são objetos.

### 3.1 Primitivas Javascript

Um valor primitivo é apenas um valor simples, por exemplo, 3 , 4.25 e “Fulano de Tal”. Os tipos primitivos são tipos que guardam um valor primitivo. Em JavaScript temos cinco tipos de dados primitivos:

- **String** – Que são textos literais. Por exemplo, “Fulano de Tal”;
- **Número** – Que são valores numéricos. Por exemplo, 3;
- **Booleano** – Que pode ser verdadeiro ou falso;
- **Nulo** – Que indica que não existe um objeto associado;
- **Indefinido** – Que indica que não existe nada, nem valor nem objeto.



Valores primitivos são literais e não podem ser alterados. O valor 3 sempre será 3. É diferente de uma variável que pode guardar um valor e mudar durante o programa.

## 3.2 Objetos Javascript

Objetos JavaScript são variáveis que guardam variáveis, ou seja, um conjunto de valores identificados. Os valores são escritos como pares nome: valor (nome e valor separados por dois pontos). Observe o exemplo:

```
var aluno = { primeiroNome:"Fulano", ultimoNome:"de Tal", idade:30,  
matriculado:true };
```

## 3.3 Propriedades de um objeto

Propriedades de um objeto são os valores identificados desse objeto. Utilizando o exemplo anterior temos.

Pro priedade	Valor
pri meiroNome	Fulano
ulti moNome	de Tal
ida de	30
ma triculado	true

## 3.4 Métodos de um objeto

Métodos são ações que os objetos podem realizar. As propriedades dos objetos podem guardar valores, objetos e funções. Um método de um objeto JavaScript é uma propriedade cujo valor é uma função.

A propriedade nomeCompleto pode retornar à junção dos nomes. Assim,

```
nomeCompleto = function() { return this.primeiroNome + " " + this.ultimoNome; }
```



## 3.5 Criando um objeto

Com o JavaScript, você pode definir e criar seus próprios objetos. Existem diferentes maneiras de criar objetos:

- Defina e crie um único objeto, usando um objeto literal.
- Defina e crie um único objeto, com o comando **new**.
- Defina um construtor de objeto e crie objetos do tipo utilizando o construtor.

### 3.5.1 Criando um objeto literal

Esta é a maneira mais fácil de criar um objeto JavaScript. Usando um objeto literal, definimos e criamos ao mesmo tempo um objeto em um comando. Um objeto literal é uma lista de pares nome: valor (como idade: 30) dentro de chaves { }. O exemplo a seguir cria um novo objeto JavaScript com quatro propriedades:

```
1 var aluno = {  
2   primeiroNome: "Fulano",  
3   ultimoNome: "de Tal",  
4   idade: 30,  
5   matriculado: true  
6 };
```

Figura 24 – Declaração e criação de um objeto literal.

Fonte: Ewerton Mendonça.

Descrição: código da declaração e criação de um objeto literal no Notepad++.

### 3.5.2 Criando com o comando new

O exemplo a seguir também cria um novo objeto JavaScript com quatro propriedades.

```
1 var aluno = new Object();  
2 aluno.primeiroNome = "Fulano";  
3 aluno.ultimoNome = "de Tal";  
4 aluno.idade = 30;  
5 aluno.matriculado = true;
```

Figura 25 – Utilizando o comando new para criar um objeto e depois adicionar suas propriedades.

Fonte: Ewerton Mendonça.

Descrição: código da declaração e criação de um objeto pelo comando new no Notepad++.





Os objetos criados com o código da Figura 24 e 25 são exatamente iguais, mas o da Figura 24 executa mais rápido.

### 3.5.3 Criando com o construtor do objeto

Os exemplos anteriores são limitados em muitas situações. Eles só criam um único objeto. Mas, às vezes nós precisamos de ter um “tipo de objeto” que pode ser usado para criar muitos objetos daquele mesmo tipo. A maneira padrão de criar um “tipo de objeto” é usando uma função de construção de objeto:

```
1 function aluno(primeiro, ultimo, idade, matriculado) {  
2     this.primeiroNome = primeiro;  
3     this.ultimoNome = ultimo;  
4     this.idade = idade;  
5     this.matriculado = matriculado;  
6 }  
7 var aluno1 = new aluno("Fulano", "de Tal", 30, true);  
8 var aluno2 = new aluno("Beltrano", "de Tal", 40, false);
```

Figura 26 – Criando uma função de construção de objeto e criando dois objetos do tipo aluno.

Fonte: Ewerton Mendonça.

Descrição: código de criação da função de construção e criação de dois objetos pelo construtor.

A função aluno é um construtor de objetos. Uma vez criado um construtor, podemos criar quantos objetos precisarmos daquele tipo.

### 3.6 A palavra-chave this

Em JavaScript a palavra-chave **this** refere-se ao objeto que possui o código. O valor de **this**, quando usado em uma função, é o objeto onde a função foi originalmente escrita. O valor de **this**, quando usado em um objeto, é o próprio objeto. Ela não é uma variável e por isso você não pode definir um valor para ela.

Na Figura 26 a palavra **this** na função se refere ao objeto que é criado no momento. Assim, cada objeto criado com **new** guardará dados diferentes, mesmo que utilize a mesma função de construção.





## 3.7 Manipulando propriedades de um objeto

Para acessar o valor guardado em uma propriedade, coloque o nome da variável com o objeto seguido de um ponto e o nome da propriedade. Onde você colocar essa sequência o navegador substituirá pelo valor. Exemplo na Figura 27.

Para modificar o valor de uma propriedade, ou criar uma propriedade, coloque o nome da variável com o objeto seguido de um ponto, o nome da propriedade, o símbolo de atribuição ( = ) e o valor. Exemplo na Figura 27.

Para remover uma propriedade de um objeto basta utilizar a palavra-chave, delete, seguida pelo nome do objeto, um ponto, e o nome da propriedade. Exemplo na Figura 27.

```
1 function aluno(primeiro, ultimo, idade, matriculado) {  
2     this.primeiroNome = primeiro;  
3     this.ultimoNome = ultimo;  
4     this.idade = idade;  
5     this.matriculado = matriculado;  
6 }  
7 var a = new aluno("Fulano", "de Tal", 30, true);  
8  
9 // Utilizando as propriedades do objeto 'a'  
10 var nome = a.primeiroNome + " " + a.ultimoNome;  
11  
12 // Modificando o valor da propriedade  
13 a.idade = 3;  
14  
15 // Deletando uma propriedade do tipo aluno  
16 delete aluno.matriculado;
```

Figura 27 – Manipulando propriedades de um objeto.

Fonte: Ewerton Mendonça.

**Descrição:** código que utiliza e modifica propriedades de um objeto e apaga uma propriedade de um tipo de objeto.

## 3.8 Manipulando métodos de um objeto

Um método é uma função guardada em uma propriedade. Para solicitar a execução de um método de um objeto, coloque o nome do objeto seguido de um ponto e o nome da propriedade que guarda a função seguido do abre e fecha parênteses e seus argumentos, se houver. Veja o exemplo na Figura 28.



Novos métodos devem ser incluídos na função construtora para serem utilizados. Você ainda pode atribuir uma nova função a uma propriedade para transformá-la em um método ou trocar a função de um método.

```
index.html
4 <title>EAD Pernambuco 2017</title>
5 <style>
6   body { text-align: center; }
7
8   .retangulo { margin: auto; border: 2px solid black; }
9
10  #ret0 { width: 400px; background-color: gray; }
11
12  #ret1 { background-color: red; }
13  #ret2 { background-color: yellow; }
14  #ret3 { background-color: green; }
15
16  #ret0 div {
17    margin: 20px auto;
18    width: 50px;
19    height: 50px;
20  }
21 </style>
22 </head>
23 <body>
24   <div id="ret0" class="retangulo">
25     <div id="ret1" class="retangulo"></div>
26     <div id="ret2" class="retangulo"></div>
27     <div id="ret3" class="retangulo"></div>
28     <p id="texto">>0 que o JavaScript pode fazer</p>
29     <p><button type="button" onclick="nome()">Executar</button></p>
30   </div>
31   <p><em>Powered Ewerton Mendonça 2017.</em></p>
32   <script src="script.js"></script>
33 </body>
34 </html>

script.js
1 function aluno(primeiro, ultimo, idade, matriculado) {
2   this.primeiroNome = primeiro;
3   this.ultimoNome = ultimo;
4   this.idade = idade;
5   this.matriculado = matriculado;
6   this.nomeCompleto = function() {
7     return this.primeiroNome + " " + this.ultimoNome;
8   }
9 }
10
11 function nome() {
12   var a = new aluno("Fulano", "de Tal", 30, true);
13   // Executa a função guardada na propriedade
14   document.getElementById("ret0").innerHTML = a.nomeCompleto();
15 }

Navegador:
file:///C:/Users/Ewer...
Fulano de Tal
Powered Ewerton Mendonça 2017.
```

Figura 28 – Conteúdo dos arquivos index.html e script.js, além da exibição do resultado no navegador.

Fonte: Ewerton Mendonça.

**Descrição:** Código HTML com uma alteração na linha 29 e código do script.js com a utilização do método 'nomeCompleto()' em uma função que 'nome()' que coloca o nome completo em um elemento HTML.

## 3.9 Prototype

Um prototype é o tipo de um objeto e sua função construtora é a forma de definir um prototype. Suas propriedades e métodos são manipulados da mesma forma que mostramos anteriormente. A única solicitação que há é que o identificador de prototypes comece por letra maiúscula. Não é obrigatório, é apenas uma boa prática de programação.



## 4. Competência 04 | Conhecer os Fundamentos de JQuery

JQuery é uma biblioteca de JavaScript, lançada em dezembro de 2006 no BarCamp de Nova York por John Resig. A sintaxe do jQuery foi desenvolvida para tornar mais simples a navegação do documento HTML, a seleção de elementos DOM, criar animações, manipular eventos, desenvolver aplicações AJAX e criação de plug-ins sobre ela. Seu objetivo é o de simplificar a programação de código JavaScript.

Uma biblioteca é uma coleção de funções prontas que foram testadas e pensadas para serem utilizadas em uma grande variedade de situações. Com o jQuery, conseguir fazer efeitos mirabolantes em sua página é muito fácil. A filosofia do jQuery é “escrever menos e fazer mais”.

A biblioteca jQuery contém os seguintes recursos:

- Manipulação HTML/DOM;
- Manipulação CSS;
- Métodos sobre eventos HTML;
- Efeitos e animações;
- AJAX;
- Funções de utilidade comuns.

Além disso, jQuery tem plug-ins para quase qualquer coisa, basta procurar na internet.

Há muitas outras bibliotecas JavaScript na internet, mas jQuery parece ser a mais popular, e também a mais extensível. Muitas das maiores empresas da web usam jQuery, como:

- Google;
- Microsoft;
- IBM;
- Netflix.

No entanto, para aprender jQuery corretamente você precisa saber HTML, CSS e JavaScript. Dê uma revisada nos assuntos passados se você esqueceu ou teve dificuldade em algum conceito.



Leia mais sobre o jQuery no link abaixo:

## 4.1 Adicionando a biblioteca em sua página

Você pode adicionar a biblioteca jQuery em suas páginas de duas formas:

- Fazendo o download da biblioteca;
- Fazendo uma referência a um repositório.
- 

### 4.1.1 Fazendo o download do JQuery

Existem duas versões do jQuery, que são iguais, mas formatadas de formas diferentes:

- **Versão de produção:** é a versão que você usa quando disponibiliza seu site na web. Ela é comprimida para ser carregada mais rápido pelo navegador.
- **Versão de desenvolvimento:** não possui compressão e, por isso, é legível, caso você precise abrir os arquivos por algum motivo.

Ambas as versões podem ser baixadas do site jQuery.com. Figura 29.

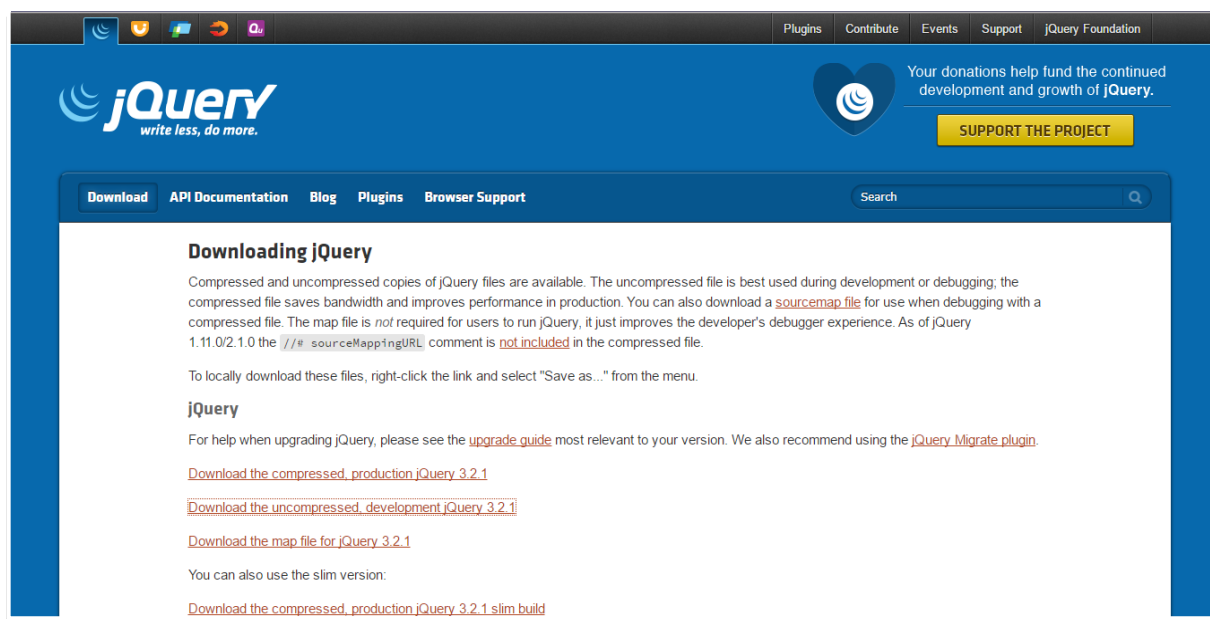


Figura 29 – Página de download do site da jQuery.com.

Fonte: Ewerton Mendonça.



**Descrição:** navegador exibindo a página de download do site da jQuery.com.

A biblioteca jQuery é um único arquivo JavaScript e você faz referência a ele com a tag `<script>` HTML. Observe que a tag `<script>` deve estar dentro da seção `<head>`. Coloque o arquivo baixado no mesmo diretório que as páginas onde você deseja usá-lo.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>EAD Pernambuco 2017</title>
5    <style>
6      body { text-align: center; }
7
8      .retangulo { margin: auto; border: 2px solid black; }
9
10     #ret0 { width: 400px; background-color: gray; }
11
12     #ret1 { background-color: red; }
13     #ret2 { background-color: yellow; }
14     #ret3 { background-color: green; }
15
16     #ret0 div {
17       margin: 20px auto;
18       width: 50px;
19       height: 50px;
20     }
21   </style>
22   <script src="jquery-3.2.1.min.js"></script>
23 </head>
24 <body>
25   <div id="ret0" class="retangulo">
26     <div id="ret1" class="retangulo"></div>
27     <div id="ret2" class="retangulo"></div>
28     <div id="ret3" class="retangulo"></div>
29     <p id="texto" >0 que o JavaScript pode fazer.</p>
30     <p><button type="button" onclick="acao()" >Ação!</button></p>
31   </div>
32   <p><em>Powered Ewerton Mendonça 2017.</em></p>
33   <script src="script.js" ></script>
```

Figura 30 – Página de exemplo com a linha 22 que adiciona a biblioteca à página.

Fonte: Ewerton Mendonça.

Descrição: código que adiciona a biblioteca à página na tag `<head>`.

## 4.1.2 Utilizando um repositório

Se você não quiser fazer o download e hospedar jQuery você mesmo, você pode incluí-la por um CDN (Content Delivery Network), um repositório que guarda a biblioteca. Tanto o Google quanto o Microsoft hospedam a jQuery. Para usar o link da jQuery disponibilizado pelo Google ou pela Microsoft:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```



```
<script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.2.1.min.js"></script>
```

## 4.2 sintaxe JQuery

A utilização do jQuery é muito fácil. Basicamente você seleciona um elemento no HTML e executa ações sobre ele, ou sobre eles, no caso de mais de um elemento.

A sintaxe básica é: **\$(seletor).ação()**

- O sinal de cifrão \$ é usado para acessar a biblioteca;
- **Seletor** é a maneira para encontrar o elemento na página HTML;
- **Ação** é uma função do jQuery que será executada sobre o elemento.

Observe a Figura 31 que possui alguns exemplos de utilização.

```
1  $(this).hide() // Esconde o elemento corrente.
2
3  $("p").hide()  // Esconde todos as tags <p>
4
5  $(".teste").hide() // Esconde todos as tags com a class 'teste'
6
7  $("#teste").hide() // Esconde todos as tags com o id 'teste'
```

Figura 31 – Exemplo de código de acesso a elementos HTML e executando sobre eles a ação de esconder.

Fonte: Ewerton Mendonça.

Descrição: código demonstrando a seleção de elementos no HTML e a execução de ação sobre eles.

## 4.3 Evento ready

O evento ready é um evento especial que espera todo o carregamento da página antes de executar o código do jQuery. Isso serve para evitar que qualquer código jQuery seja executado antes que o documento HTML esteja carregado.

É uma boa prática esperar que o documento seja carregado completamente e esteja pronto antes de trabalharmos com ele. Isso também evita que você coloque o seu código JavaScript antes do corpo do seu documento, na tag <head>.

Eis alguns exemplos de ações que podem falhar se os métodos forem executados antes que o documento esteja carregado completamente:





- Tentando ocultar um elemento que ainda não foi criado;
- Tentando obter o tamanho de uma imagem que ainda não está carregada.

```
1 $(document).ready(function() {  
2     $("p").hide();  
3 }
```

**Figura 32 – código do evento ready().**

**Fonte:** Ewerton Mendonça.

**Descrição:** Exemplo de utilização do evento ready().

## 4.4 Seletores

Os seletores são uma parte importante da maneira como o jQuery é manipulado. Os seletores permitem selecionar e manipular elementos em uma página HTML. Os seletores jQuery são usados para “achar”, ou selecionar, elementos HTML baseados em seu nome, id, classes, tipos, atributos, valores de atributos e muito mais. É baseado nos seletores CSS existentes e, além disso, tem alguns próprios seletores personalizados. Todos os seletores no jQuery começam com o sinal de dólar e parênteses: **\$ ()**.

### 4.4.1 Selecionando Tag

O seletor de elemento jQuery pode selecionar tags com base no nome da tag. Você pode selecionar todos os `<p>` elementos em uma página, veja o exemplo na Figura 33, quando o usuário clica no botão, todos os quadrados desaparecem.

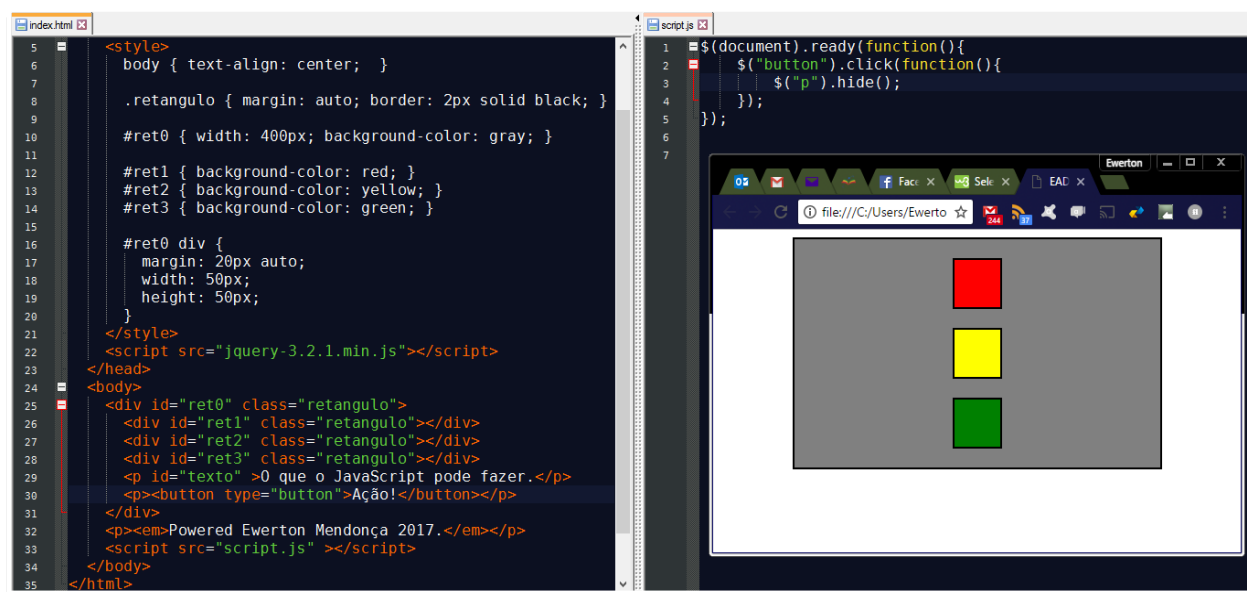


Figura 33 – exibição do código de exemplo em HTML e do código JavaScript

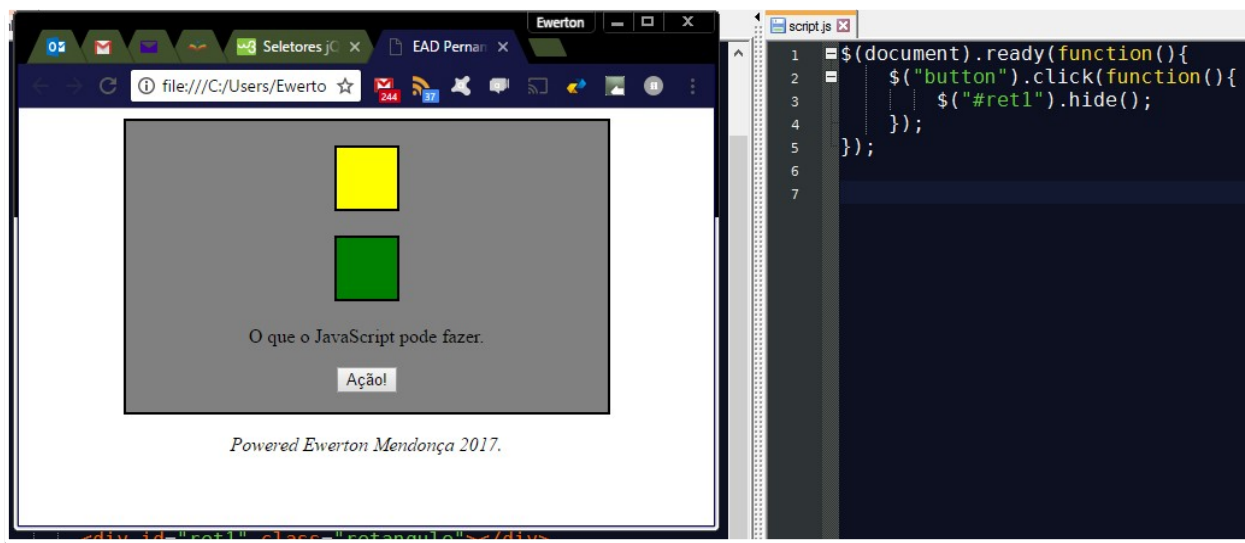
Fonte: Ewerton Mendonça.

**Descrição:** exibição do código de exemplo em HTML e do código JavaScript para selecionar elementos baseados em tag. Houve alterações no HTML, observe na linha 30.

Atente que no HTML não precisamos do atributo 'onClick'. A própria jQuery cuida de atribuir o efeito do botão. Atente, também, que ainda separamos nosso código JavaScript em outro arquivo, para ficar organizado e mais legível. Já em nosso código JavaScript usamos o seletor baseado em tags para selecionar todas as tags <button> na linha 2 e todas as tags <p> na linha 3.

## 4.4.2 Selecionando pelo atributo ID

A forma como selecionamos pelo atributo id é quase igual à seleção de tags, mas como o id deve ser único, o retorno de elementos pelo id deve ser de apenas um elemento. A diferença está que, na seleção pelo atributo id, utilizamos o símbolo de jogo da velha ( # ) antes do nome. Observe no exemplo da Figura 34.



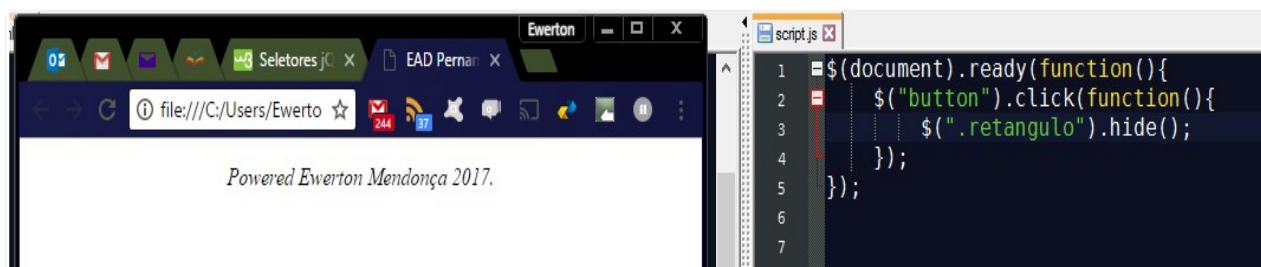
**Figura 34 – Exemplo de seleção pelo id de 'ret1' na linha 3.**

**Fonte:** Ewerton Mendonça.

**Descrição:** código de exemplo de seleção pelo id e resultado no navegador.

## 4.4.3 Selecionando pelo atributo class

A seleção pelo atributo class é realizada colocando na frente do valor um ponto. Observe no exemplo da Figura 35.



**Figura 35 – Exemplo de seleção pelo atributo class de 'retangulo' na linha 3.**

**Fonte:** Ewerton Mendonça.

**Descrição:** código de exemplo de seleção pelo atributo class e resultado no navegador.

## 4.4.4 Exemplos de seletores

Estude estes exemplos mais complexos para compreender o poder do jQuery:

- `$("*")` Seleciona todos os elementos.
- `$(this)` Seleciona o elemento corrente.
- `$("p.intro")` Seleciona todos as tags <p> com o class="intro".
- `$("p:first")` Seleciona a primeira tag <p>.
- `$("ul li:first")` Seleciona a primeira tag <li> da primeira <ul>.



`$("ul li:first-child")` Seleciona a primeira tag `<li>` de cada `<ul>`.

`$("[href]")` Seleciona todas as tags que possuam um atributo `'href'`.

`$("a[target='_blank']")` Seleciona todas as tags `<a>` em que o atributo `target` possua o valor `'_blank'`.

`$("a[target!='_blank']")` Seleciona todas as tags `<a>` que o atributo `target` NÃO possua o valor `'_blank'`.

`$(":button")` Seleciona todas as tags `<button>` e tags `<input>` que possua o atributo `type` com o valor `'button'`.

`$("tr:even")` Seleciona todas as tags até a tag `<tr>`.

`$("tr:odd")` Seleciona todas as linhas ímpares das tags `<tr>`.

Estude esses exemplos de seleção de tags e experimente para entender o que eles realizam. Em caso de dúvida, faça uma pesquisa na web para entender melhor.

## 4.5 Eventos JQuery

Todas as diferentes ações do visitante que uma página da Web pode responder são chamadas de eventos. JQuery é feito sob medida para responder a eventos em uma página HTML. Um evento representa o momento preciso em que algo acontece. Por exemplo:

- Movendo um mouse sobre um elemento;
- Selecionando um botão;
- Clicando em um elemento.

Na Tabela 5 temos alguns dos eventos mais utilizados:

Eventos com o mouse	Eventos com o teclado	Eventos de formulário	Eventos do documento
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

**Tabela 5 – Tabela com os eventos mais utilizados classificados por tipo.**

**Fonte:** Ewerton Mendonça.

**Descrição:** tabela que classifica os eventos mais utilizados.

Para atribuir um evento você faz uma seleção, coloca um ponto e chama a função relacionada ao evento. Depois você precisa definir uma função que será executada quando o



evento ocorrer. Veja o exemplo nas Figuras 33, 34 e 35.

**`$(document).ready();`** Permite-nos executar uma função quando o documento está carregado completamente.

**`$("#p").click();`** A função é executada quando o usuário clica no elemento HTML.

**`$("#p").dblclick();`** A função é executada quando o usuário clica duas vezes no elemento.

**`$("#p").mouseenter();`** A função é executada quando o ponteiro do mouse entra no elemento HTML.

**`$("#p").mouseleave();`** A função é executada quando o ponteiro do mouse deixa o elemento.

**`$("#p").mousedown();`** A função é executada quando qualquer botão é pressionado para baixo, enquanto o mouse está sobre o elemento HTML.

**`$("#p").mouseup();`** A função é executada quando qualquer botão é liberado, enquanto o mouse está sobre o elemento HTML.

**`$("#p").hover();`** A primeira função é executada quando o mouse entra no elemento HTML e a segunda função é executada quando o mouse sai do elemento HTML.

**`$("#input").focus();`** A função é executada quando o campo de formulário obtém foco.

**`$("#input").blur();`** A função é executada quando o campo de formulário perde o foco.

O método `on()` atribui um ou mais manipuladores de eventos para os elementos selecionados. Veja o exemplo na Figura 36.

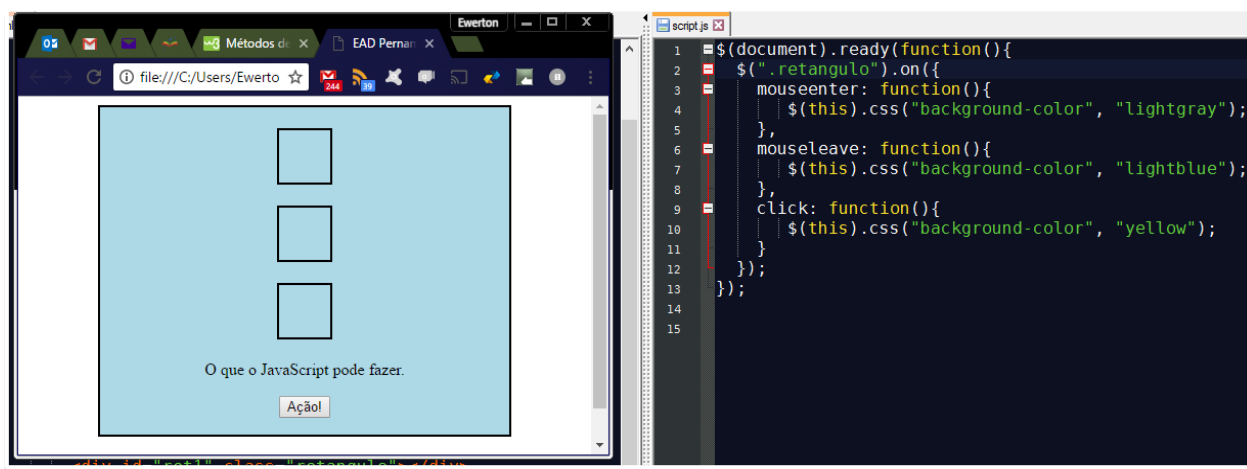


Figura 36 – Exemplo de utilização do método `on()`.

Fonte: Ewerton Mendonça.

Descrição: código de exemplo do método `on()` e exibição do resultado no navegador.



## 4.6 Obtendo a tag, o texto e o valor de um campo

Três métodos simples da jQuery, mas úteis, para manipulação DOM são:

- **text()** - Define ou retorna o conteúdo de texto dos elementos selecionados;
- **html()** - Define ou retorna o conteúdo dos elementos selecionados (incluindo marcação HTML);
- **val()** - Define ou retorna o valor de campos de formulário.

O exemplo da Figura 37 demonstra como obter conteúdo com os métodos jQuery `text()` e `html()`. O método `val()` funciona da mesma forma, mas precisa de um campo de formulário para pegar o valor.

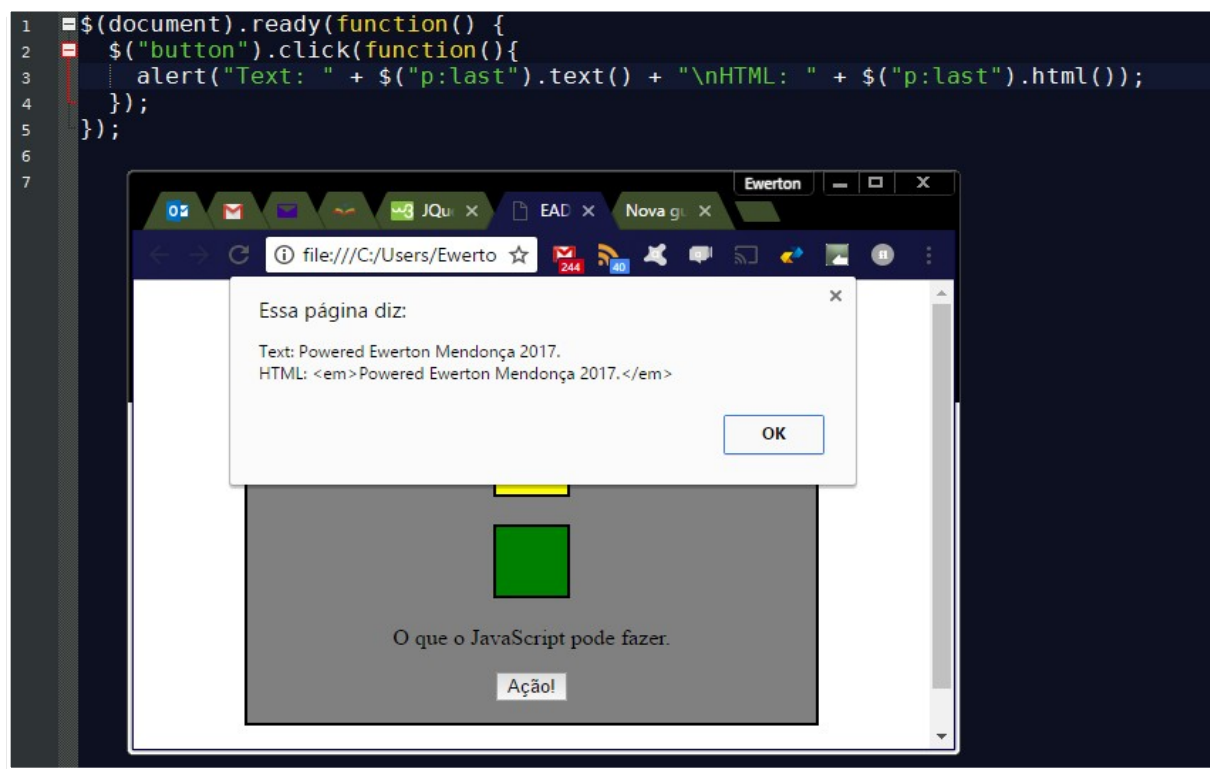


Figura 37 – utilização das funções `text()` e `html()`.

Fonte: Ewerton Mendonça.

Descrição: Exemplo do código no Notepad++ das funções `text()` e `html()` e navegador mostrando o resultado.

Observe na exibição do navegador da Figura 43 que o texto pego com `text()` das tags `<p>` foi modificado com `'p:last'`, ou seja, a última tag `<p>`. Perceba a diferença entre `text()` e `html()`.





Como a última tag <p> possui uma tag <em>, ela é exibida quando usamos html() e não quando usamos text().

## 4.7 Obtendo o valor de um atributo

O método jQuery **attr()** é usado para obter valores de atributos. O exemplo da Figura 38 demonstra como obter o valor do atributo type em uma tag <button>.

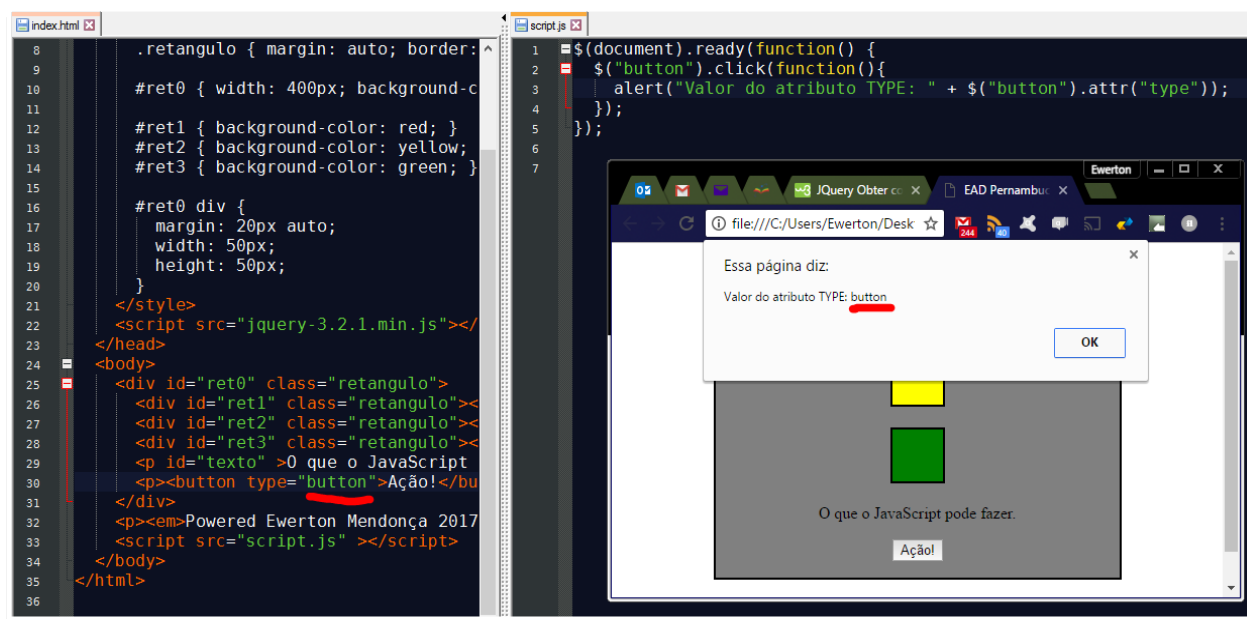


Figura 38 – Exemplo de utilização da função attr().

Fonte: Ewerton Mendonça.

**Descrição:** código HTML e JavaScript no Notepad++ da função attr() e o navegador com o resultado. Em destaque o valor capturado.

## 4.8 Definindo conteúdo

A definição de conteúdo em modo texto com text(), em modo HTML com html() e colocando valores em campos de formulário com val() funciona da mesma forma para pegar os valores, no entanto colocamos o valor que queremos definir entre os parênteses.

A troca de valores de atributo necessita definir qual atributo será modificado.

Analise o exemplo da Figura 39.

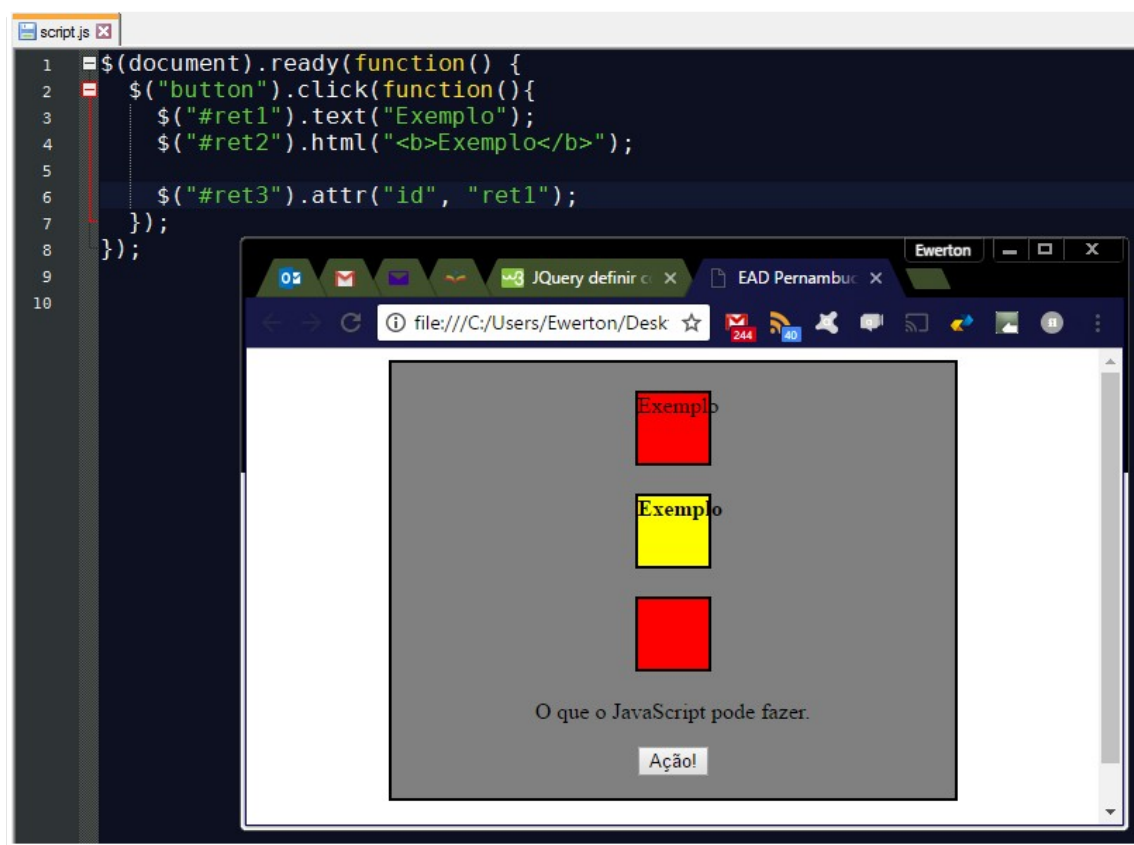


Figura 39 – código text() e html() no Notepad++ e navegador mostrando o resultado.

Fonte: Ewerton Mendonça.

**Descrição:** Exemplo onde utilizamos text() e html() para colocar conteúdo no HTML. No caso do atributo, modificamos o atributo id do terceiro retângulo, por isso ele é renderizado com as configurações de CSS do primeiro quadrado.

Observe que no quadrado amarelo utilizamos html() para inserir a tag <b> que deixa o texto em negrito. A tag <b> não deve ser utilizada, o negrito é colocado com CSS, mas nesse caso foi apenas para demonstrar o exemplo.

No caso do atributo, modificamos o atributo id do terceiro retângulo, por isso ele é renderizado com as configurações de CSS do primeiro quadrado.



## 5.Competência 05 | Utilizar os Efeitos do JQuery

Os efeitos da biblioteca jQuery são um show à parte. Ela contém animações prontas utilizando funções matemáticas. Esses feitos podem ser associados a vários elementos e combinados. Veremos alguns deles:

- `hide();`
- `show();`
- `fade();`
- `slide();`
- `animate();`

O limite é a criatividade, mas tem muitos exemplos prontos na web que podem ser copiados, aplicados, combinados e estudados.

### 5.1 Esconder (hide) e mostrar (show)

Com jQuery, você pode ocultar e mostrar elementos HTML com os métodos `hide ()` e `show ()`.

Sua sintaxe é:

**`$(selector).hide(velocidade , callback);`**

**`$(selector).show(velocidade, callback);`**

O parâmetro de **velocidade** é opcional e especifica a velocidade da ocultação/exibição. Pode ter os seguintes valores: “slow” (devagar), “fast” (rápido) ou a definição do tempo em milissegundos, um segundo equivale a 1000.

O parâmetro de `callback` é opcional e é uma função que será executada após o método `hide()` ou `show()` ser concluído.

Veja o exemplo na Figura 40. Como o efeito é animado, você terá que copiar e executar em seu navegador para ver a experiência de uso. Copie com bastante atenção, uma letrinha ou símbolo errado e nada funcionará.



```
1 = $(document).ready(function() {
2 =   $("#ret1").click(function(){
3     $(this).hide("slow");
4   });
5 =   $("#ret2").click(function(){
6     $(this).hide("fast");
7   });
8 =   $("#ret3").click(function(){
9     $(this).hide(5000);
10  });
11
12 =  $(".button").click(function(){
13    $(".retangulo").show();
14  });
15  });
```

Figura 40 – Exemplo dos efeitos hide() e show().

Fonte: Ewerton Mendonça.

Descrição: código com o exemplo dos efeitos de hide() e show() no Notepad++.

Quando os quadrados coloridos são clicados eles desaparecem. Quando o botão é clicado, todos aparecem. Observe o tempo de desaparecimento configurado para cada um.

## 5.1.1 Efeito toggle

Podemos alternar entre mostrar e esconder com apenas uma função, toggle(). Ela funciona como um liga-desliga. Replique o código da Figura 41 e execute para experimentar.

```
1 = $(document).ready(function() {
2 =   $(".button").click(function(){
3     $("#ret1").toggle();
4   });
5   });
```

Figura 41 – código de exemplo do efeito toggle().

Fonte: Ewerton Mendonça.

Descrição: Exemplo do efeito toggle(). Quando aperta o botão o quadrado vermelho desaparece, se apertar novamente, ele aparece.

## 5.2 Efeito de desaparecimento gradual (fade)

Com jQuery você pode fazer desaparecer ou aparecer um elemento de forma gradual. JQuery tem os seguintes métodos **fade**:

**fadeOut()** - Faz o elemento desaparecer gradativamente;



**fadeOut()** - Faz o elemento aparecer gradativamente;

**fadeToggle()** - Alterna entre desaparecer e aparecer;

**fadeTo()** - Faz o elemento aparecer ou desaparecer até um certo ponto determinado entre 0 e 1, sendo 0 totalmente invisível e 1 totalmente visível.

Assim como o efeito de hide/show, o fade também tem os mesmos parâmetros de velocidade e callback.

O parâmetro de **velocidade** é opcional e especifica a velocidade da ocultação/exibição. Pode ter os seguintes valores: "slow" (devagar), "fast" (rápido) ou a definição do tempo em milissegundos, um segundo equivale a 1000.

O parâmetro de callback é opcional e é uma função que será executada após o método hide() ou show() ser concluído.

```
1  = $(document).ready(function() {
2      $("#ret1").hide(); // Esconde o quadrado vermelho
3
4  = $("#ret2").click(function(){
5      $("#ret1").fadeIn();
6      $("#ret2").fadeOut();
7  });
8  = $("#ret3").click(function(){
9      $(this).fadeTo("slow", 0.5); // Metade invisível
10 });
11
12 = $("button").click(function(){
13     $("em").fadeToggle(3000);
14 });
15 });
```

Figura 42 – Código de exemplo no Notepad++.

Fonte: Ewerton Mendonça.

**Descrição:** Neste exemplo do fade o quadrado vermelho inicia escondido, o quadrado amarelo quando clicado faz o fadeIn() do quadrado vermelho e o fadeOut() do quadrado amarelo. O quadrado verde quando clicado desaparece até metade da transparência, 0.5, e o botão faz a assinatura da página apagar e quando clicado de novo aparece em 3 segundos.

Atente que, para demonstrar o efeito de fadeIn() o elemento deveria estar escondido e para isso foi colocada a linha 2.



## 5.3 Efeito de deslizar

Com jQuery você pode criar um efeito deslizante em elementos. JQuery tem os seguintes métodos de slide:

- `slideDown()` - Faz o elemento deslizar para baixo até aparecer completamente.
- `slideUp()` - Faz o elemento deslizar para cima até desaparecer completamente.
- `SlideToggle()` - Alterna entre o deslizar para cima e para baixo o elemento.

Assim como o efeito de `hide/show`, o `fade` também tem os mesmos parâmetros de velocidade e `callback`.

Veja o exemplo na Figura 43.

```
1  = $(document).ready(function() {
2      $("#ret1").hide(); // Esconde o quadrado vermelho
3
4  = $("#ret1").click(function(){
5      $(this).slideUp();
6  });
7  = $("#ret2").click(function(){
8      $(this).slideUp();
9  });
10 = $("#ret3").click(function(){
11     $("#ret1").slideDown();
12     $("#ret2").slideDown();
13 });
14
15 = $(".em").click(function(){
16     $(".ret0").slideToggle();
17 });
18 });
```

Figura 43 – código de exemplo do efeito de slide.

Fonte: Ewerton Mendonça.

**Descrição:** Os quadrados vermelho e amarelo quando clicados deslizam para cima até desaparecer. O quadrado verde faz os quadrados vermelho e amarelo deslizarem para baixo até aparecer totalmente. A assinatura da página faz o quadrado cinza desaparecer e quando clicado novamente faz aparecer.

## 5.4 Efeito de animação

O método da biblioteca jQuery **`animate()`** é usada para criar animações personalizadas. Sua sintaxe é:

**`$(selector).animate({parâmetros}, velocidade, callback);`**

Sendo:





- **Parâmetros** é a propriedade CSS que será animada;
- **Velocidade** é opcional e dita a duração da animação, igual ao que já foi explicado anteriormente;
- **Callback** é opcional e contém uma função que será executada quando o efeito terminar.

**OBSERVAÇÃO:** por padrão, todos os elementos HTML têm uma posição estática e não podem ser movidos. Para manipular a posição, lembre-se de primeiro definir a propriedade de posição CSS do elemento para *relative*, *fixed* ou *absolute*!

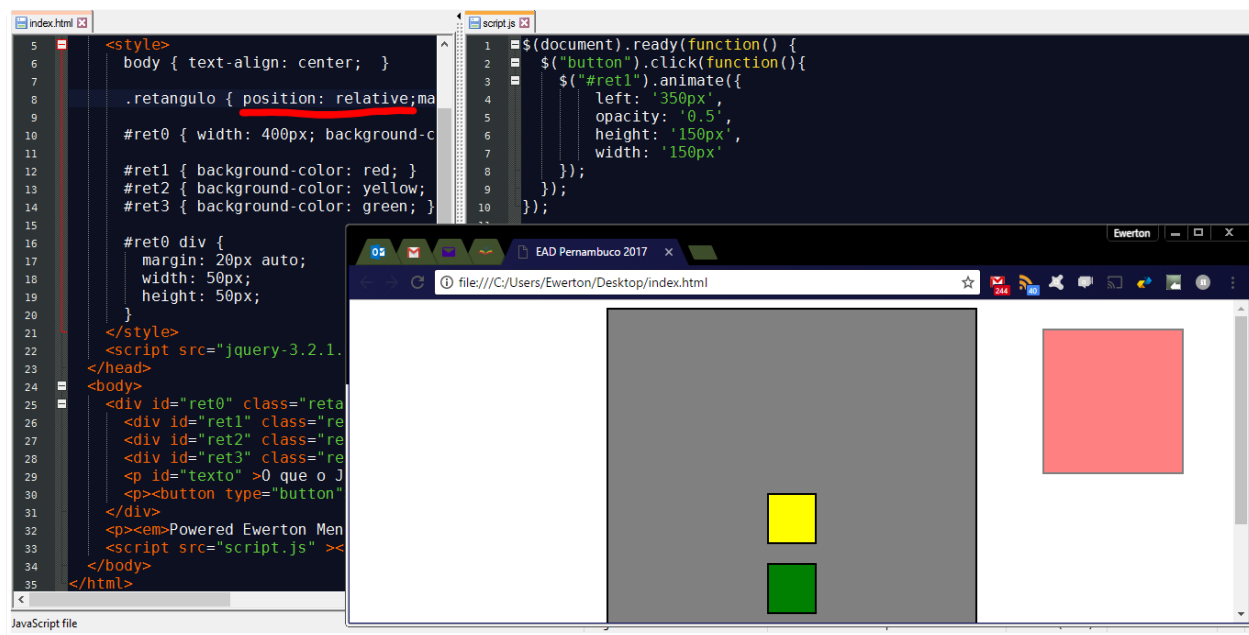


Figura 44 – código HTML e JavaScript com a demonstração do efeito de `animate()`.

Fonte: Ewerton Mendonça.

**Descrição:** Quando o botão é clicado, o quadrado vermelho é deslocado para a esquerda, fica metade transparente com a altura e largura em 150px. Observe que, para poder realizar o efeito de mover, precisamos definir a propriedade `position` para *relative* no CSS, sublinhado em vermelho.

## 5.5 Encadeamento de métodos

Até agora, temos escrito comandos jQuery um de cada vez (um após o outro). No entanto, existe uma técnica chamada **encadeamento**, que nos permite executar vários comandos jQuery, um após o outro, no(s) mesmo(s) elemento(s). Usando encadeamento, os navegadores não precisam encontrar o(s) mesmo(s) elemento(s) mais de uma vez, sendo bem mais rápido.



Para encadear uma ação, basta anexar um efeito ao efeito anterior. O exemplo da Figura 49 encadeia os métodos `css()`, `slideUp()` e `slideDown()`. O elemento `"#ret0"` muda para vermelho, depois desliza para cima e, em seguida, desliza para baixo quando você clica na assinatura da página.

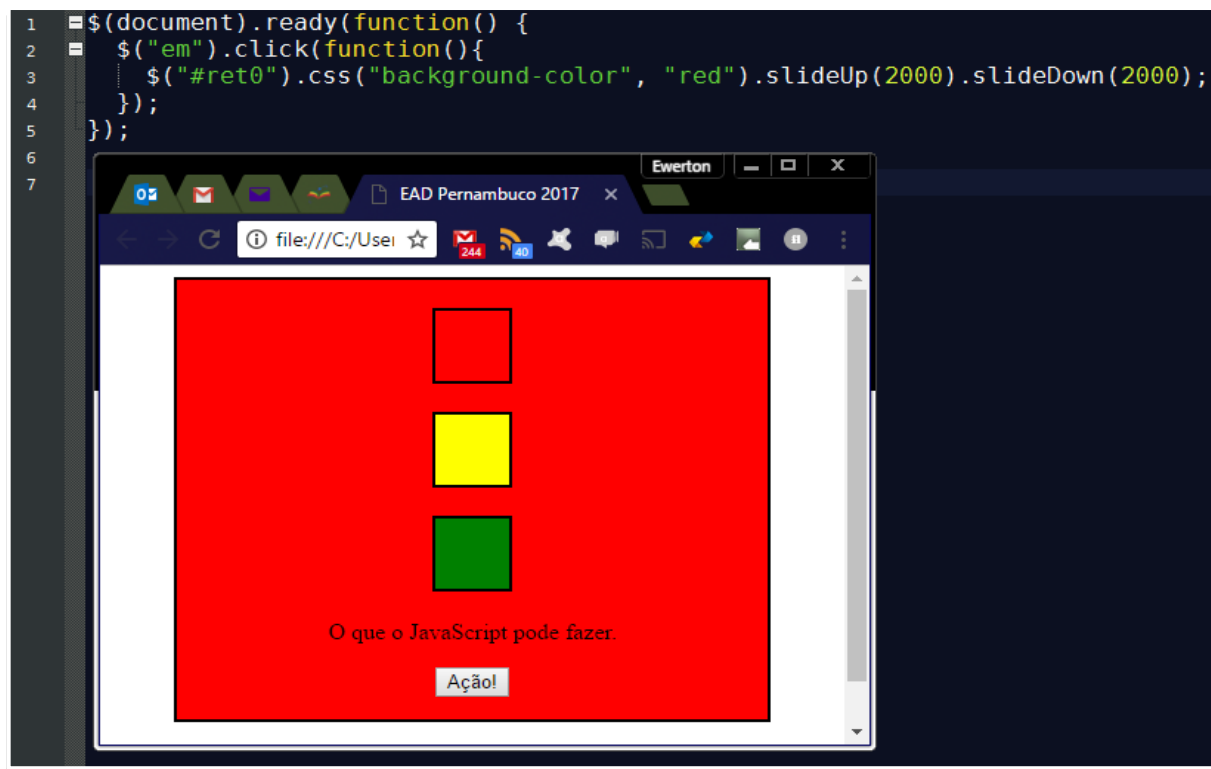


Figura 45 – código de exemplo de métodos encadeados e resultado final no navegador.

Fonte: Ewerton Mendonça.

**Descrição:** Ao clicar na assinatura da página, o quadrado cinza muda para vermelho, desliza para cima e depois para baixo.



## 6. Competência 06 | Utilizar o Bootstrap Framework

Para finalizar, nesta competência vamos conhecer um framework chamado Bootstrap. Um framework é uma ferramenta que ajuda no desenvolvimento de forma rápida de uma camada da aplicação. No caso do Bootstrap, é a camada de front-end. Ela já nos dá algo pronto de HTML, CSS e jQuery, só faltando colocarmos nosso conteúdo.

O Bootstrap foi desenvolvido na empresa Twitter por Mark Otto e Jacob Thornton utilizando os padrões abertos da W3C. Eles queriam padronizar os diversos componentes, tanto em comportamento como estilo nas aplicações da empresa. O projeto foi disponibilizado como código aberto e pode ser copiado e utilizado até em projetos comerciais.

No Bootstrap já temos vários comportamentos bem definidos através da jQuery, então, também precisamos da jQuery para o Bootstrap funcionar. Ele também tem um estilo predefinido de bom gosto, mas que você pode modificar apenas acrescentando seu estilo CSS depois de todos os arquivos.



Vamos fazer um exemplo do zero, mas caso você queira aprender mais sobre este framework pode acessar a página do Bootstrap traduzida pela Globo.com, que utiliza muito em seus sites. O endereço está abaixo e a Figura 50 mostra o site em português do Brasil.

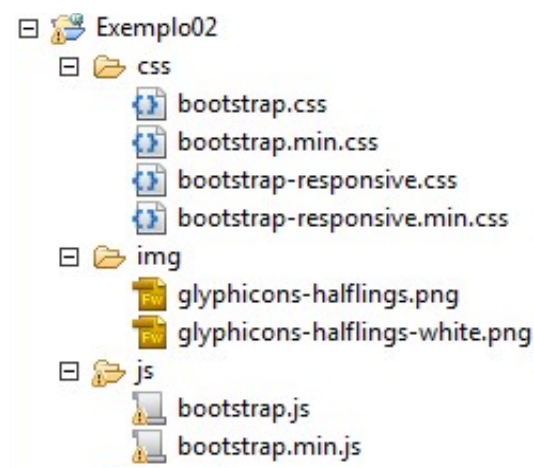


**Figura 46 – Site do Bootstrap em português do Brasil.**

**Fonte:** <http://getbootstrap.com.br/>

**Descrição:** navegador exibindo a página principal do site do Bootstrap do Brasil.

Após baixarmos o framework, temos a árvore de arquivos da Figura 47. Ela contém três pastas: a pasta **css** tem os arquivos de estilo compactado e descompactado, como na jQuery; a pasta **img** contém os ícones utilizados pelo framework; e a pasta **js** contém os arquivos JavaScript que utilizam a jQuery. Observe que a biblioteca jQuery não está entre eles. Devemos fazer o download e colocar. O melhor lugar é na pasta **js**. Assim, faça o download da jQuery, se não a tiver, e coloque na pasta **js**.



**Figura 47 – Árvore de arquivo do framework Bootstrap.**

**Fonte:** Ewerton Mendonça.

**Descrição:** Explorer do Windows exibindo a árvore de arquivos do projeto.

Agora crie um arquivo HTML com a estrutura básica que mostramos em competência anterior. Relacione esta página com a biblioteca jQuery, depois com o CSS e o JavaScript do



Bootstrap. O código para isso está na Figura 48.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>EXEMPLO BOOTSTRAP</title>
5
6     <link href="css/bootstrap.min.css" rel="stylesheet" media="screen" />
7
8     <script src="js/jquery-1.11.1.min.js"></script>
9     <script src="js/bootstrap.js"></script>
10  </head>
11  <body>
```

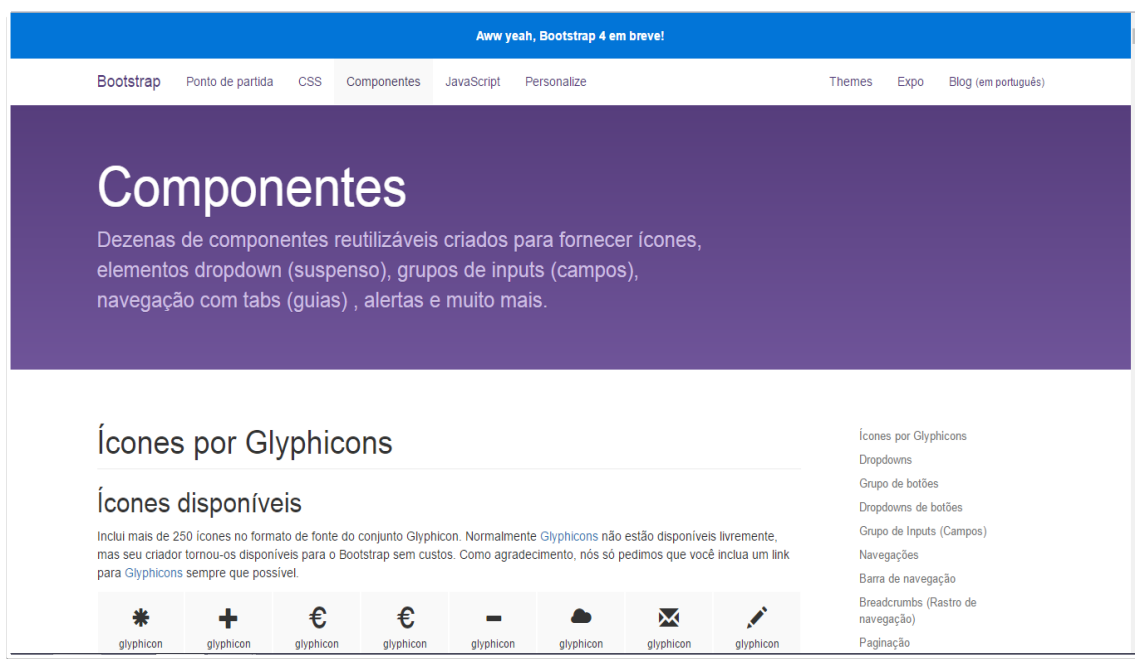
**Figura 48 – conteúdo HTML da página de modelo.**

**Fonte:** Ewerton Mendonça.

**Descrição:** Arquivo index.html com a inclusão do CSS e arquivos JavaScript do Bootstrap e da biblioteca jQuery.

O Bootstrap já possui diversos componentes com CSS e JavaScript prontos. Para utilizá-los, apenas devemos saber suas estruturas HTML e nomes de classes. Toda a documentação do Bootstrap está em português e pode ser consultada no link abaixo. Nele você pode ver como o componente funciona, sua aparência e o código para acrescentá-lo em sua página. Uma vez que relacionamos corretamente o arquivo HTML com os arquivos necessários do framework, podemos copiar e colar em nossa página qualquer componente. Tenha curiosidade e experimente. A Figura 49 possui uma captura de tela da página explicativa dos componentes.

<http://getbootstrap.com.br/components/>



**Figura 49 – navegador exibindo a página de componentes do site do Bootstrap do Brasil.**

**Fonte:** <http://getbootstrap.com.br/components/>

**Descrição:** Página dos componentes do Bootstrap da Globo.com toda em português do Brasil, com explicação dos componentes, exemplos e código.

Vamos utilizar alguns desses componentes. Por exemplo, um formulário de login. A Figura 50 mostra o código HTML do formulário de login, com `<input>` para digitar o e-mail e a senha, um elemento checkbox e um botão para enviar os dados. Seguindo os códigos de exemplo da página explicativa dos componentes, incluímos os atributos class com os valores apropriados. São esses valores de class que adicionam o poder do framework.





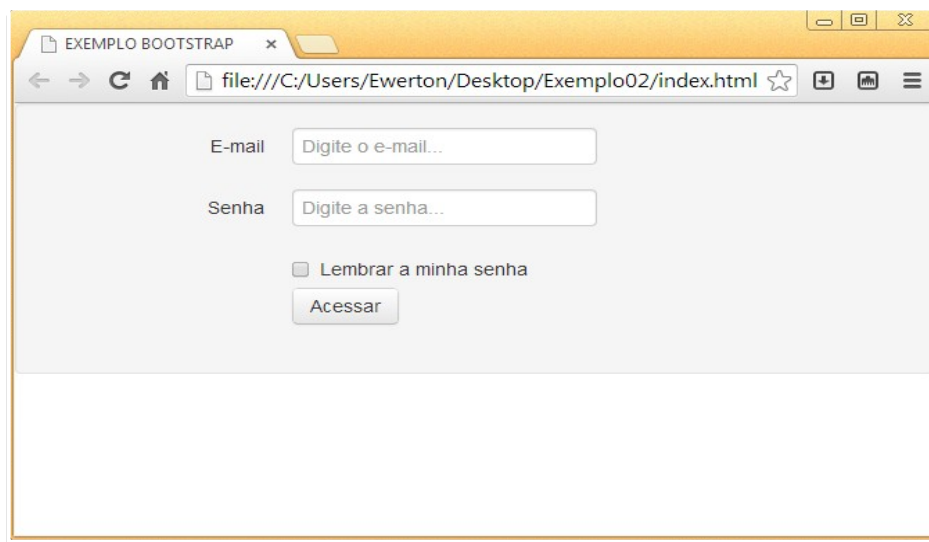
```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>EXEMPLO BOOTSTRAP</title>
5
6     <link href="css/bootstrap.min.css" rel="stylesheet" media="screen" />
7
8     <script src="js/jquery-1.11.1.min.js"></script>
9     <script src="js/bootstrap.js"></script>
10  </head>
11  <body>
12    <form class="form-horizontal well">
13      <div class="control-group">
14        <label class="control-label" for="inputEmail">E-mail</label>
15        <div class="controls">
16          <input id="inputEmail" type="text" placeholder="Digite o e-mail..." />
17        </div>
18      </div>
19      <div class="control-group">
20        <label class="control-label" for="inputSenha">Senha</label>
21        <div class="controls">
22          <input id="inputSenha" type="password" placeholder="Digite a senha..." />
23        </div>
24      </div>
25      <div class="control-group">
26        <div class="checkbox">
27          <input type="checkbox" /> Lembrar a minha senha</div>
28        <button class="btn" type="submit">Acessar</button>
29      </div>
30    </form>
31  </body>
32 </html>
```

**Figura 50 – código HTML com o formulário da página de exemplo.**

**Fonte:** Ewerton Mendonça.

**Descrição:** Código do formulário, com os atributos class com os valores que adicionam o poder do bootstrap.

Na Figura 51 temos a exibição da página no navegador. Observe como a página já aparece estilizada corretamente.



**Figura 51 – Resultado no navegador já formatado pelo Bootstrap.**

**Fonte:** Próprio autor.

**Descrição:** navegador exibindo o resultado da formatação do Bootstrap.

Vimos rapidamente um exemplo de como utilizar o Bootstrap para acelerar a construção de um site. Mas para utilizá-lo adequadamente você deve conhecer bem o framework, então, leia,

# Competência 06



pesquise. Você será um profissional melhor quanto maior for o tempo investido em aprendizado.



## Conclusão

Chegamos ao final da segunda parte de nosso caderno de estudos. Foi um excelente investimento de tempo que você fez ao se dedicar a este curso, mas para se tornar um profissional será necessário mais pesquisa e investimento, afinal esta área de criação para web é gigantesca.

O autor do livro Fora de Série – Outliers, Malcolm Gladwell, cita uma pesquisa sobre pessoas que são consideradas profissionais em suas áreas. Na pesquisa ele explica que para chegar a um nível profissional são necessárias 10.000 horas dedicadas de forma apaixonada naquilo que se faz.

Gostando do que faz, você presta mais atenção e entende bem melhor do que pessoas que estudam por obrigação. Dessa forma, é fácil chegar às 10.000 horas. Pois é. Sei que são muitas horas estudando, testando código, fuçando e pesquisando sobre os assuntos vistos, mas se você se apaixonar pelo tema, não será um martírio, será sim, bem divertido.

Espero que você tenha gostado, tanto quanto eu, de fazer este curso.

Boa sorte ao entrar nesta maravilhosa área da profissão de web designer.



## Referências

ROBBINS, JENINIFER NIEDERST. APRENDENDO WEB DESIGN: GUIA PARA INICIANTES, 3ª EDIÇÃO — PORTO ALEGRE: BOOKMAN, 2010.

KALBACH, JAMES. DESIGN DE NAVEGAÇÃO WEB: OTIMIZANDO A EXPERIÊNCIA DO USUÁRIO, PORTO ALEGRE: BOOKMAN, 2009.

ZEMEL, TÁCIO. WEB DESIGN RESPONSIVO: PÁGINAS ADAPTÁVEIS PARA TODOS OS DISPOSITIVOS, SÃO PAULO: CASA DO CÓDIGO, 2012.

SILVA, MAURICIO SAMY. HTML 5: A LINGUAGEM DE MARCAÇÃO QUE REVOLUCIONOU A WEB, SÃO PAULO: NOVATEC EDITORA, 2011.

SILVA, MAURICIO SAMY. CSS3: DESENVOLVA APLICAÇÕES WEB PROFISSIONAIS COM O USO DOS PODEROSOS RECURSOS DE ESTILIZAÇÃO DAS CSS3, SÃO PAULO: NOVATEC EDITORA, 2012.

SILVA, MAURICIO SAMY. CONSTRUINDO SITES COM CSS E (X)HTML: SITES CONTROLADOS POR FOLHAS DE ESTILO EM CASCATA, SÃO PAULO: NOVATEC EDITORA, 2008.

SILVA, MAURICIO SAMY. CRIANDO SITES COM HTML: SITES DE ALTA QUALIDADE COM HTML E CSS, SÃO PAULO: NOVATEC EDITORA, 2008.

SILVA, MAURICIO SAMY. JQUERY - A BIBLIOTECA DO PROGRAMADOR JAVASCRIPT, 2ª EDIÇÃO — SÃO PAULO: NOVATEC EDITORA, 2010.

GLADWELL, MALCOLM. FORA DE SÉRIE: OUTLIERS. RIO DE JANEIRO: SEXTANTE, 2008.



## Minicurrículo do Professor

### Ewerton Mendonça



Formado em Sistemas de Informação pela UPE e Design pela UFPE, com mestrado em Ciência da Computação pela UFPE. Atualmente, é professor da Faculdade de Ciências e Letras de Caruaru – FAFICA e Unifavip Devry Brasil. Possui experiência na área de desenvolvimento WEB e design gráfico desde 1998.

