

Formação Java



TriadWorks

Lógica de Programação

Versão 1.0.0

www.triadworks.com.br

Conheça nossos cursos:



Acesse:
cursos.triadworks.com.br

Índice

1	Introdução à Lógica de Programação	1
1.1	Introdução	1
1.2	O que iremos precisar?	1
1.3	Entendendo o ambiente	1
1.4	Criando o HTML	3
1.5	Exercícios: Criando nossa HTML	4
1.6	Olá Mundo em JavaScript	4
1.7	Exercícios: Meu primeiro programa em JavaScript	5
1.8	Algumas ferramentas que ajudam	5
1.9	Exercícios: Exercitando a inspeção de elementos	7
2	Melhorando nossa interação com o usuário	9
2.1	Melhorando a comunicação	9
2.2	Exercícios	10
2.3	Utilizando números	11
2.4	Exercícios	12
2.5	Operações matemáticas	12
2.6	Exercícios: Operações matemáticas	13
3	Organizando melhor nosso código	16
3.1	Variáveis	16
3.2	Exercícios: Variáveis	18
3.3	Reaproveitando nosso código com funções	20
3.4	Exercícios: Funções	22
3.5	Passando informações para as funções	25

3.6	Exercícios	27
3.7	Recebendo informações de uma função	29
3.8	Exercícios: Funções com retorno	30
4	Exercitando nossa lógica	31
4.1	Alguns problemas do dia a dia	31
4.2	Exercícios	33
4.3	Para saber mais: Compartilhando código	38
4.4	Usuário interagindo com o seu programa	40
4.5	Exercícios: Interagindo com o seu programa	43
4.6	Melhorando nossos programas, interagindo mais	43
4.7	Exercícios: Melhorando nossos programas	44
5	Tomada de decisão	48
5.1	Condição: se/if	48
5.2	Exercícios: Condição se/if	51
5.3	Outras formas de condições	53
5.4	Exercícios: Outras formas de se fazer uma condição	56
6	Estruturas de repetição	59
6.1	Entendendo a estrutura com while	59
6.2	Exercícios: Fazendo loops com WHILE	60
6.3	Outro tipo de loop com a estrutura do FOR	61
6.4	Exercícios: Loop com FOR	63
6.5	Juntando os conceitos...	63
6.6	Exercícios: Juntando os conceitos...	64
6.7	Garanta que seu programa não vá da erros	70
6.8	Exercícios: Evitando erros	71
6.9	Tem como melhorar?	71
6.10	Exercícios: Melhorias	72
7	Estrutura de dados com Array	75
7.1	Melhorando a interatividade com o HTML	75
7.2	Exercícios: Melhorando a interatividade com o HTML	79
7.3	Estrutura de dados Array	80

7.4	Exercícios: Arrays	82
7.5	Jogo da Adivinhação	83
7.6	Exercícios: Adivinhação	85
7.7	Melhorando nosso jogueirinho de Adivinhação	88
7.8	Exercícios: Melhorando nosso jogueirinho de Adivinhação	90
8	Boas práticas e Orientação a Objetos	93
8.1	Organização	93
8.2	Exercícios: Organizando nosso JavaScript	94
8.3	Orientação a Objetos	94
9	Apêndice - Aprendendo a desenhar e animar!	96
9.1	Criando gráficos com o CANVAS do HTML 5	96
9.2	Exercícios: Melhorando a interatividade com o HTML	98
9.3	Mais desenhos...	102
9.4	Exercícios: Mais desenhos com canvas	107
9.5	Mais uso de funções	113
9.6	Exercícios: Mais uso de funções	115
9.7	Animação simples	117
9.8	Exercícios: Simples movimentos	120
10	Respostas dos Exercícios	122

Versão: 18.7.19

Introdução à Lógica de Programação

1.1 - Introdução

Hoje em dia tudo está ligado diretamente com o desenvolvimento de programas de computadores. Por este motivo saber programar será um requisito básico para qualquer pessoa. Programar é divertido e mais do que divertido é um constante desafio e exercício da mente. Quando você entra neste mundo você já vai começar a sentir seus benefícios, raciocínio mais rápido, raciocínio lógico e uma incrível vontade de entender como tudo funciona.

Ao aprender a programar, você poderá desenvolver vários tipos de aplicativos, sejam eles para sites, celulares, jogos, ferramentas próprias etc. É um mundo com grandes possibilidades.

Para isso se realizar nada melhor que a prática e aprofundamento em cada tipo de ferramenta e linguagem que você deseja utilizar.

Para isso não existe nenhuma mágica, apenas muita prática e estudo. Neste curso iremos iniciar uma caminhada que será bem divertida e cheia de desafios.

Siga os passos dos exercícios, faça todas as nossas sugestões e dicas e veja por si próprio os resultados, sinta como é fazer um texto “virar” algo “palpável”. Se mesmo assim a sua vontade ainda não for saciada, invente, troque os dados, modifique as rotinas o mais importante é experimentar, criar e claro se divertir.

1.2 - O que iremos precisar?

Antes de querer aprender a programar precisamos exercitar a nossa capacidade de raciocinar, nossa capacidade de pensar, resolver problemas, escolher caminhos, isso é o raciocínio lógico.

A lógica de programação vem normalmente antes de tentarmos estudar alguma linguagem, ou seja, ela é um pre-requisito para aprender a programar em qualquer linguagem. Porém, essa forma deixa o conteúdo um pouco chato, na verdade bem chato. Pois não usamos nenhuma rotina ou forma de se executar o que estamos fazendo, apenas o que chamamos de pseudos códigos, ou seja, no final temos apenas “papel”.

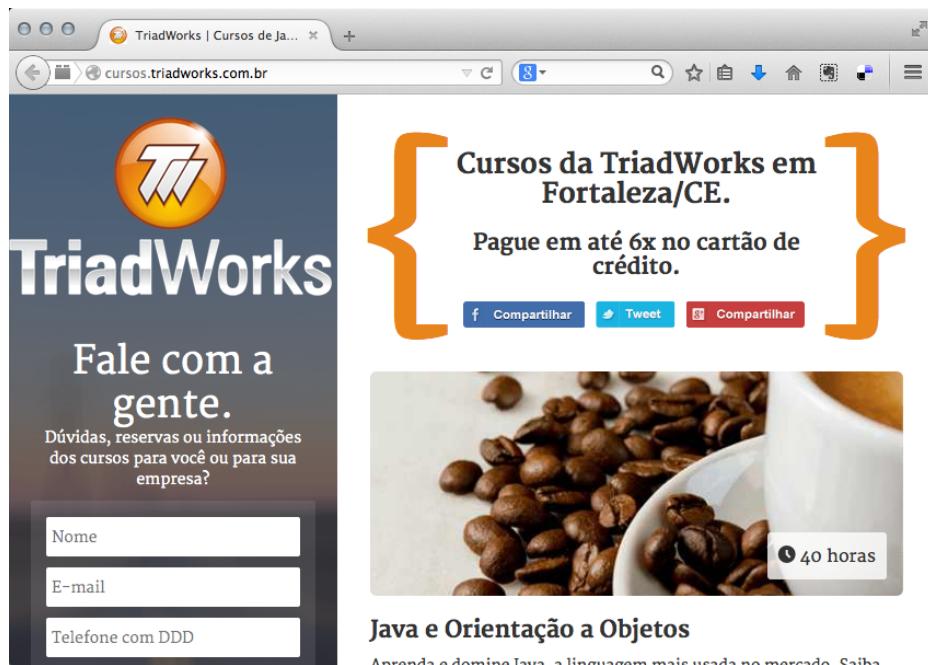
O interessante seria se pudessemos utilizar uma linguagem para aprendermos o conteúdo e que ao mesmo tempo nos mostrasse algum resultado, dessa forma podemos deixar o aprendizado mais divertido e “palpável”.

Por este motivo escolhemos uma linguagem para você: o **JavaScript**, que possui vantagens e desvantagens. Mas o principal é que com ele podemos rapidamente escrever programas e **executá-los** de imediato para ver o resultado e sem precisar de muita coisa ou instalar nada, ou seja, iremos focar no aprendizado e raciocínio lógico. Usaremos o que você já tem, o *Navegador(browser)*, seja ele *Firefox*, *Chrome* ou o *Internet Explore*.

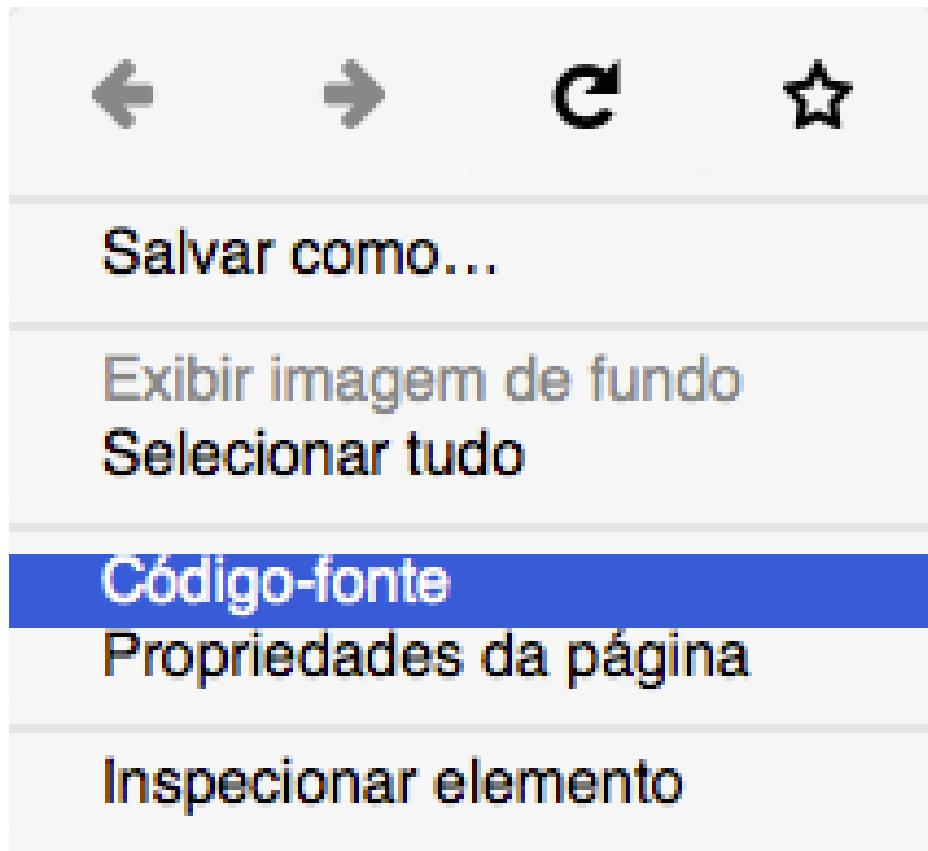
1.3 - Entendendo o ambiente

Como falamos anteriormente iremos utilizar o **JavaScript** como linguagem base para aprender e o ambiente será Web, sim, iremos estudar em um ambiente que todos nos já estamos familiarizados.

A primeira coisa a fazer é abrir o navegador de sua preferência (para o curso iremos utilizar o Firefox). Acesse uma página qualquer, pode ser a página do curso da TriadWorks: <http://cursos.triadworks.com.br>.



Com a página aberta clique com o direito em qualquer parte, como na imagem abaixo:



Cada navegador é diferente, o que estamos utilizando é o do Firefox. Ao selecionar a opção, iremos exibir o

código fonte. Um código é um conjunto de instruções que alguém entende, no nosso caso o navegador. Esse código que aparece é o HTML, que não é bem uma linguagem de programação e sim uma linguagem de marcação de texto (HyperText Markup Language), porém é com ele que o navegador entende o que deve ser exibido e onde ele irá ser exibido.

Não iremos nos aprofundar no HTML, iremos utilizá-lo apenas para exibir certas informações em conjunto com o JavaScript. Se quiser saber mais sobre HTML indicamos um excelente site de estudos na tecnologia: <http://www.w3schools.com/html/>.

1.4 - Criando o HTML

Para começarmos iremos precisar inicialmente de algum editor de texto, pode ser o *Bloco de Notas*(*Notepad*) ou algum de sua preferência, tente o Sublime(<http://www.sublimetext.com/>) ou Notepad++(<http://notepad-plus-plus.org/>).

Um arquivo HTML(HyperText Markup Language), possui a extenção como *.html*, pensando assim já podemos imaginar que editores como o *Word* não irá nos ajudar, pois ele salva com outra extenção *.doc*.

Com o nosso editor aberto, iremos iniciar nossa primeira página *HTML* e iremos escrever uma simples mensagem de boas vindas:

Seja bem vindo ao curso!
<h3>Já estou programando?</h3>

Salvando o arquivo como *minha_pagina.html* em uma pasta qualquer, iremos abri-lo no nosso navegador, o Firefox. Ao abrir o arquivo no browser você perceberá que a mensagem foi exibida.

Seja bem vindo ao curso!

Já estou programando?

Erros de acentuação!

Se você abriu seu arquivo e ele mostrou um código maluco onde era para possuir acentuação, não se desespere. Isso acontece porque há diferentes formatos de armazenamentos de caracteres em *bytes*. O Chrome ou Firefox por exemplo, por padrão, iram lê-lo em uma codificação chamada de *latin1*.

Javascript e HTML.â€ iBooks. Seja bem vindo ao curso!

JÃ¡ estou programando?

Para evitar esse errinho chato, basta adicionar a seguinte tag no **íncio** do seu arquivo *HTML*:

```
<meta charset="UTF-8">
```

Dessa forma estamos informando ao navegador em qual formato de caracteres ele deve ler aquele arquivo, no nosso caso no formato *UTF-8*.

O HTML trabalha com tags(<h3>), que são utilizadas para mudar a forma com que os dados são apresentados. Vamos completar nosso código. Com o código aberto, altere adicionando a mensagem abaixo e salve.

Seja bem vindo ao curso!

<h3>Já estou programando?</h3>

Ainda não, isso é apenas HTML.

Agora basta dar um *reload* (F5) na página.

Seja bem vindo ao curso!

Já estou programando?

Ainda não, isso é apenas HTML.

Observe que o HTML é estático, ou seja, ele serve apenas para **exibir** informações é apenas um conteúdo **fixo** sem nenhum interação com o usuário.

Iremos utilizar o HTML apenas como base para programar com o JavaScript.

1.5 - Exercícios: Criando nossa HTML

1) Iremos agora criar nossa primeira página HTML.

a) Na pasta de sua preferência, crie um arquivo chamado `minha_pagina.html`.

b) Agora abra no seu editor preferido (Sublime ou Notepad++).

c) Com o arquivo aberto, adicione o código abaixo:

Eba, minha primeira página HTML.

Falta pouco para eu aprender a programar de verdade!

Salve o arquivo.

d) Abra o arquivo no seu navegador (Firefox, Chrome etc).

1.6 - Olá Mundo em JavaScript

Iremos agora utilizar a linguagem JavaScript para criarmos programas que interagem com o usuário.

Com o arquivo `.html` aberto, adicione no início dele o trecho abaixo:

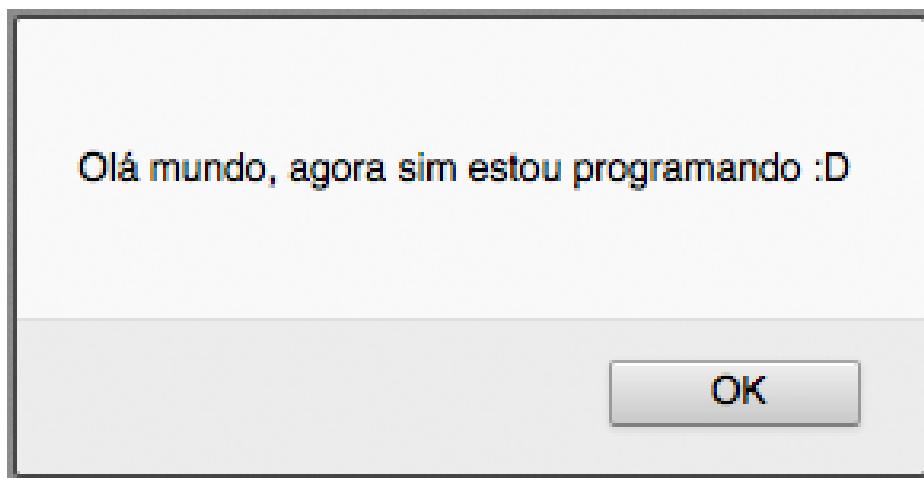
```
<script>
</script>
```

Essa tag “diz” ao seu navegador onde irá iniciar o código JavaScript e onde ele acaba.

Queremos ao abrir a página que o usuário receba uma mensagem de boas vindas:

```
<script>
  alert("Olá mundo, agora sim estou programando :D ");
</script>
```

Ao salvar o arquivo basta agora dar um reload na página que a mensagem será exibida.



1.7 - Exercícios: Meu primeiro programa em JavaScript

1) Abra o arquivo `minha_pagina.html`.

a) Adicione no inicio do arquivo a tag para colocarmos o JavaScript:

```
<script>  
</script>
```

b) Agora adicione o código para mostrar um alerta na página.

```
alert("Olá mundo, agora sim estou programando :D");
```

c) Salve o arquivo e dê reload(F5) na página.

1.8 - Algumas ferramentas que ajudam

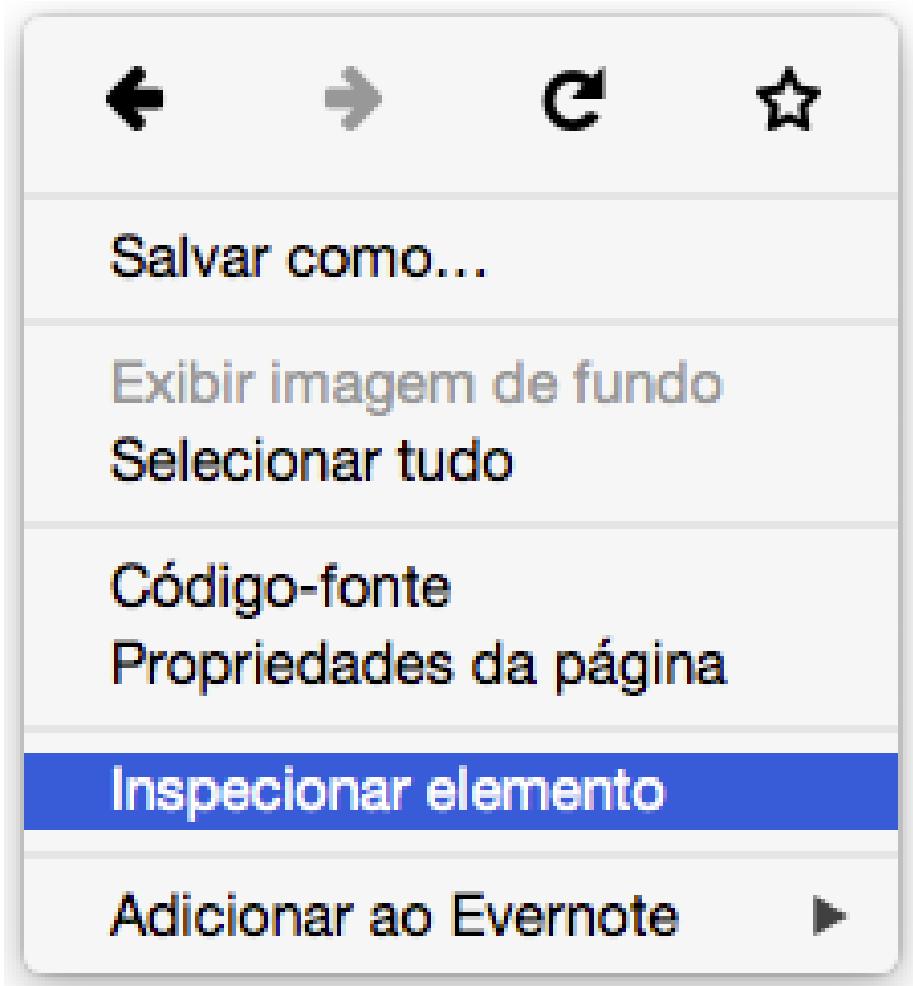
Algumas vezes cometemos alguns erros ao digitarmos algum código e precisamos de ferramentas que nos ajudem a encontrar nosso erro ou as vezes até mostrar se erramos o código.

Com JavaScript não seria diferente. Para isso podemos utilizar o próprio navegador. No nosso caso utilizamos o Firefox que assim como a maioria dos navegadores possuem ferramentas que ajudam exatamente nisso.

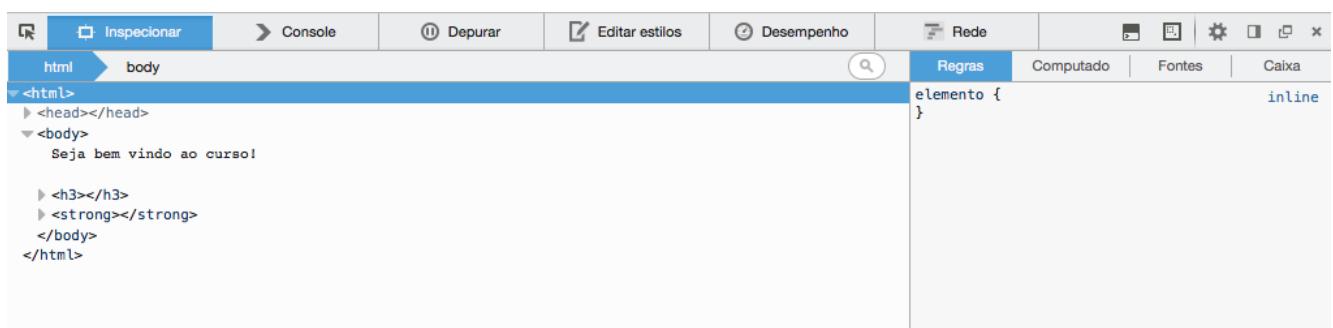
Por exemplo, se colocarmos o seguinte código na nossa página, ele não irá funcionar.

```
<script>  
alert("Olá mundo, agora sim estou programando :D ");  
  
alerta "mensagem com problema :"  
</script>
```

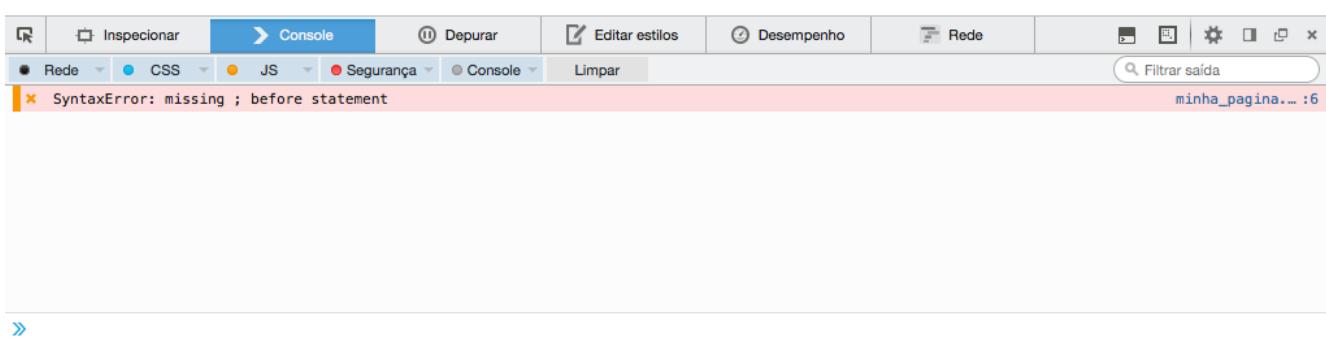
Clicando com o direito na página aberta no navegador, você irá selecionar a opção *Inspecionar Elemento* ou *Element Inspect*.



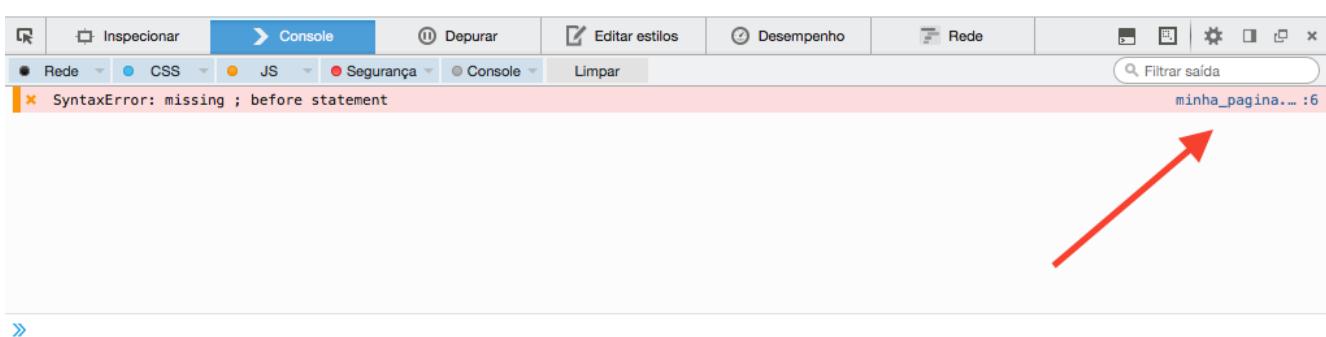
Observe que uma barra com várias abas é aberta logo abaixo:



Selecionando a aba *Console* você verá se deu algum erro no seu código. As vezes você precisa após selecionar a aba, dar um F5(reload):



Aqui ele mostra qual o erro ocorreu e clicando no nome da página, ele irá abrir o arquivo e mostrar exatamente onde o erro ocorreu.



Aqui o editor apontou em qual linha existe o código com problemas. Dessa forma ficando bem mais fácil encontrarmos algum erro no nosso código.

```

1 <meta charset="UTF-8">
2 <script>
3     alert("Olá mundo, agora sim estou programando :D ");
4
5     alerta "mensagem com problema :("
6 </script>
7 Seja bem vindo ao curso!
8     <h3>Já estou programando?</h3>
9     <strong>Ainda não, isso é apenas HTML.</strong>
10

```

1.9 - Exercícios: Exercitando a inspeção de elementos

1) Abra o arquivo `minha_pagina.html`.

a) Adicione no inicio do arquivo a tag para colocarmos o JavaScript:

```

<script>
    alert("Olá mundo, agora sim estou programando :D ");

    alerta "mensagem com problema :("
</script>

```

b) Após salvar o arquivo, atualize a página.

- c) Claro que a página não irá funcionar corretamente. Clique com o direito em cima dela e selecione a opção *Inspecionar Elemento* ou *Element Inspect*.
- d) Clique na aba *Console* e corrija o erro.

Melhorando nossa interação com o usuário

2.1 - Melhorando a comunicação

Já criamos nosso primeiro programa e fizemos ele enviar uma mensagem (alerta) para nosso usuário. Utilizamos o alert para enviar essas mensagens, dessa forma ao entrar na página nosso usuário irá receber um alerta com a nossa mensagem.

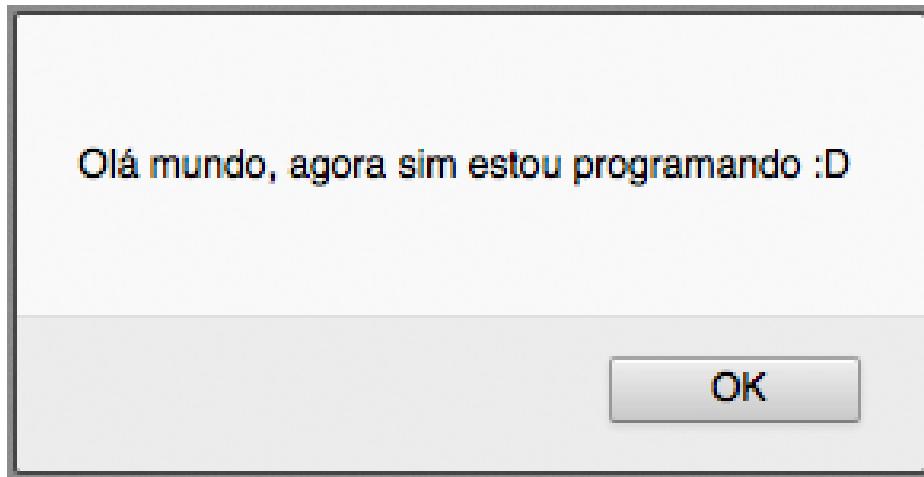
Porém, imagine se tivermos mais de uma mensagem, teremos que clicar no OK várias vezes correto? Isso é bem chato e nada legal para nosso usuário.

Que tal melhorar um pouco mais isso? No lugar de mostrar um alerta, vamos escrever no HTML a nossa mensagem, ou seja, vamos fazer com que a nossa mensagem seja exibida diretamente na página.

No código antigo feito com o alert temos:

```
alert("Olá mundo, agora sim estou programando :D");
```

Ele irá exibir como abaixo:



Mas, como vimos isso ficará bem chato se for mais de uma mensagem. Então para resolver isso iremos utilizar outro comando do JavaScript, o document.write:

```
document.write("Olá mundo, agora sim estou programando :D");
```

A screenshot of a browser window. The page contains only the text "Olá mundo, agora sim estou programando :D" centered in the middle of the white page area.

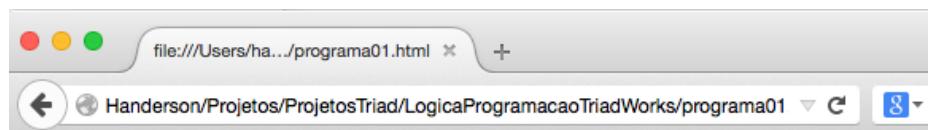
Observe que agora ele “imprimiu” a mensagem diretamente na página. Bem legal neh?

E tem mais, você pode misturar código HTML. Por exemplo:

Temos duas mensagens:

```
document.write("Alterando a forma de exibir uma mensagem para o usuário");
document.write("Bem melhor que um alert");
```

Dessa forma ao abrir o arquivo em um navegador as mensagens irão aparecer na mesma linha, como na imagem abaixo:

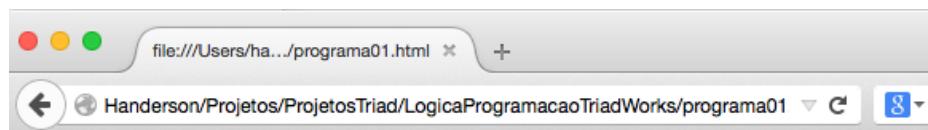


Alterando a forma de exibir uma mensagem para o usuário
Bem melhor que um alert

Para evitar isso basta adicionar a tag
 que serve como um “Enter”:

```
document.write("Alterando a forma de exibir uma mensagem para o usuário<br>");
document.write("Bem melhor que um alert");
```

Veja o resultado:



Alterando a forma de exibir uma mensagem para o usuário
Bem melhor que um alert

Melhor, não?

2.2 - Exercícios

1) Vamos melhorar nosso código:

- Crie um novo arquivo chamado: programa01.html
- Agora abra o arquivo no seu editor.

2) Criando nosso código:

- Com o arquivo aberto, adicione duas mensagens utilizando o document.write:

```
document.write("Alterando a forma de exibir uma mensagem para o usuário");
document.write("Bem melhor que um alert");
```
- Salve o arquivo e abra no seu navegador.

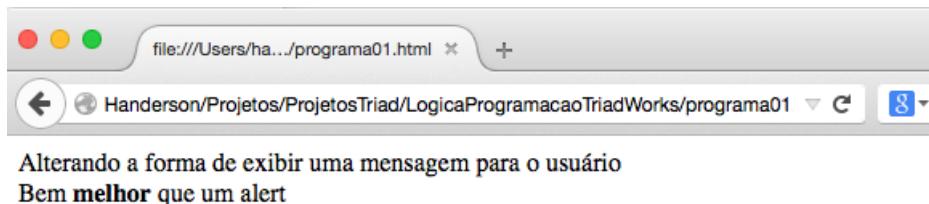
3) a) Adicione uma linha após a primeira mensagem utilizando a tag
:

```
document.write("Alterando a forma de exibir uma mensagem para o usuário<br>");  
document.write("Bem melhor que um alert");
```

b) Salve o arquivo e teste.

c) Agora, queremos deixar em **negrito** a palavra *melhor*, para isso iremos utilizar a tag :

```
document.write("Alterando a forma de exibir uma mensagem para o usuário<br>");  
document.write("Bem <strong>melhor</strong> que um alert");
```



2.3 - Utilizando números

Quando colocamos algum valor entre as aspas "", estamos informando que estamos trabalhando com uma sequência de caracteres, um texto por assim dizer.

Já fizemos várias sequências de caracteres até aqui, "Olá mundo, agora sim estou programando :D", "Alterando a forma de exibir uma mensagem para o usuário
" e "Bem melhor que um alert". Porém, são apenas mensagens fixas, estáticas.

Que tal começarmos a evoluir um pouco mais nosso código?

Se quisermos exibir quantos actions figures(bonecos colecionáveis de personagens de filmes, animes, HQs etc) eu posso?

```
document.write("Eu tenho 10 actions figures do Dragon Ball e sou feliz!");
```

Observe que o numeral "10" não exatamente é um número, pois ele está entre as aspas, logo ele é uma sequência de caracteres, ou seja, apenas um texto.

Vamos a outro exemplo:

```
document.write("Quantos actions figures eu e meu amigo possuímos?");  
document.write("10" + "5");
```

Qual será o resultado?

Bem o resultado será **"105"**. Isso porque ainda estamos trabalhando com texto, sequência de caracteres. Para trabalhar com números efetivamente, basta não passá-los como texto, ou seja, sem as aspas:

```
document.write("Quantos actions figures eu e meu amigo possuímos?");  
document.write(10 + 5);
```

E o resultado deverá ser: **15**.

Quando usamos entre uma sequência de caracteres o sinal +, estamos **concatenando** o texto, ou seja, estamos **juntando** uma sequência de caracteres com outra.

Vamos a outro exemplo:

```
document.write("Olá, meu nome é " + "Handerson Frota");
```

O resultado será a junção (concatenação) das duas sequências: *"Olá, meu nome é Handerson Frota"*.

Podemos ir mais além, o que acontece se concatenarmos uma sequência de caracteres e um número?

```
document.write("Olá, meu nome é " + "Handerson Frota e tenho " + 33 + " anos de idade.");
```

Observe o resultado: *Olá, meu nome é Handerson Frota e tenho 33 anos de idade.*

2.4 - Exercícios

1) Crie um arquivo chamado *programa02.html* e salve na pasta da sua preferência.

a) Imprima uma mensagem com o seu nome usando a concatenação:

```
document.write("Olá, meu nome é " + "Handerson Frota");
```

b) Agora adicione a sua idade a mensagem. O resultado deverá ser: *"Olá, meu nome é Fulano, e tenho 10 anos de idade."*

2) Vamos adicionar mais informações, no mesmo *programa02.html* faça:

a) Abaixo do seu nome, exiba as suas informações, o resultado deverá ser algo semelhante:

"Olá, meu nome é Fulano, e tenho 10 anos de idade."

"Cargo: Estagiário"

"Ano de nascimento: 1998"

"Solteiro"

"Gosto de: Animes, skate etc..."

b) Algo semelhante `document.write("Olá, meu nome é "+ "Handerson Frota e tenho "+ 33 + "anos de idade.");`

2.5 - Operações matemáticas

Quando usamos algum valor dentro das aspas, estamos trabalhando com uma sequência de caracteres e mesmo colocando números, eles continuam como uma sequência de caracteres, por exemplo:

```
document.write("Nasci na data: " + "26" + "1998");
```

O resultado será: *"Nasci na data: 261998"*.

Isso acontece pois estamos ainda trabalhando com a sequência de caracteres. E se quisermos efetuar alguma operação matemática? Basta não passar entre aspas.

```
document.write("Nasci na data: " + 26 + 1998);
```

Resultado será: *"Nasci na data: 2024"*.

Isso porque agora ele somou os números.

Por exemplo, para descobrir nossa idade:

```
document.write(2015 - 1981);
```

Resultado: 34.

Podemos fazer qualquer operação matemática e ela segue quase as mesmas regras de uma operação normal.

Vamos exibir uma mensagem com a operação que descobre nossa idade pelo ano de nascimento.

```
document.write("Minha idade é:" + 2015 - 1981);
```

Resultado: *NaN*

Opá, tivemos uma problema. Isso porque estamos somando a string com uma subtração, então devemos ter cuidado com os parênteses, da mesma forma como é na matemática:

```
document.write("Minha idade é:" + (2015 - 1981));
```

Observe que usamos os parênteses, dessa forma ele irá priorizar a operação que estão no parênteses e logo depois, ele irá concatenar o **resultado** ao texto.

Resultado: *Minha idade é: 34*.

Que tal sabermos a média de idade de nossos amigos?

A operação é: *a soma de todas as idades, dividido pelo total de idades*.

Temos 4 amigos com as respectivas idades: 23, 34, 43, 32.

Então a operação matemática será: $(23+34+43+32) / 4$.

Passando isso para o JavaScript:

```
document.write("Nossa média de idade é:" + (23+34+43+32) / 4);
```

Resultado: *Nossa média de idade é: 33*

2.6 - Exercícios: Operações matemáticas

- 1) Crie um arquivo chamado `programa03.html` e salve na pasta da sua preferência. Faça um programa que imprima o nome de 5 amigos seu. Tendo um resultado semelhante abaixo:

Nome: Guilherme Frota
Nome: Nahana Frota
Nome: William Frota
Nome: Rafael Ponte
Nome: Lina Tarcia

- 2) Vamos alterar o mesmo `programa03.html`:

- a) Adicione para cada amigo a sua idade respectivamente:

Nome: Guilherme Frota - idade: 8
Nome: Nahana Frota - idade: 8
Nome: William Frota - idade: 3
Nome: Rafael Ponte - idade: 31
Nome: Lina Tarcia - idade: 32

b) Imprima mais uma linha que mostre a soma de todas as idades:

```
Nome: Guilherme Frota - idade: 8  
Nome: Nahan Frota - idade: 8  
Nome: William Frota - idade: 3  
Nome: Rafael Ponte - idade: 31  
Nome: Lina Tarcia - idade: 32  
Soma das idades: 74 <-----
```

c) Agora faça com que o programa calcule a média das idades:

```
Nome: Guilherme Frota - idade: 8  
Nome: Nahan Frota - idade: 8  
Nome: William Frota - idade: 3  
Nome: Rafael Ponte - idade: 31  
Nome: Lina Tarcia - idade: 32  
Soma das idades: 74  
A média de idades: 14.8 <-----
```

3) Vamos agora criar um novo programa. Criaremos o programa04.html, onde o mesmo deve descobrir a partir da idade, qual ano a pessoa nasceu.

Descubra o ano de nascimento das pessoas abaixo:

- Nome: Guilherme Frota - idade: 8
- Nome: Nahan Frota - idade: 8
- Nome: William Frota - idade: 3
- Nome: Rafael Ponte - idade: 31
- Nome: Lina Tarcia - idade: 32

O resultado deverá ser algo semelhante:

```
Nome: Gilherme Frota - idade: 8  
Gilherme nasceu em: 2007  
Nome: Nahan Frota - idade: 8  
Nahan nasceu em: 2007  
Nome: William Frota - idade: 3  
William nasceu em: 2012  
Nome: Rafael Ponte - idade: 31  
Rafael nasceu em: 1984  
Nome: Lina Tarcia - idade: 32  
Lina nasceu em: 1983
```

4) Crie um novo programa, chamado de programa05.html. Queremos que esse programa calcule quantos dias cada uma das pessoas abaixo já viveu até hoje.

- Nome: Gilherme Frota - idade: 8 - Gilherme nasceu em: 2007
- Nome: Nahan Frota - idade: 8 - Nahan nasceu em: 2007
- Nome: William Frota - idade: 3 - William nasceu em: 2012
- Nome: Rafael Ponte - idade: 31 - Rafael nasceu em: 1984
- Nome: Lina Tarcia - idade: 32 - Lina nasceu em: 1983

a) O programa deve exibir os dias que essa pessoa já viveu.

Resultado:

Nome: Gilherme Frota - idade: 8, nasceu em: 2007 e já viveu 2920 dias.
Nome: Nahan Frota - idade: 8 ,nasceu em: 2007 e já viveu 2920 dias.
Nome: William Frota - idade: 3 ,nasceu em: 2012 e já viveu 1095 dias.
Nome: Rafael Ponte - idade: 31 ,nasceu em: 1984 e já viveu 11315 dias.
Nome: Lina Tarcia - idade: 32 ,nasceu em: 1983 e já viveu 11680 dias.

- b) Agora faça com que o programa calcule o total de dias que todas essas pessoas já viveram.
- c) Agora calcule a média de dias que seus amigos já viveram e imprima o total.

Resultado:

Nome: Gilherme Frota - idade: 8, nasceu em: 2007 e já viveu 2920 dias.
Nome: Nahan Frota - idade: 8 ,nasceu em: 2007 e já viveu 2920 dias.
Nome: William Frota - idade: 3 ,nasceu em: 2012 e já viveu 1095 dias.
Nome: Rafael Ponte - idade: 31 ,nasceu em: 1984 e já viveu 11315 dias.
Nome: Lina Tarcia - idade: 32 ,nasceu em: 1983 e já viveu 11680 dias.
Total de dias: 29930
Média de dias: 5986

Organizando melhor nosso código

3.1 - Variáveis

No nosso programa anterior exibimos várias informações para o usuário tais como: Nome, Idade, Quando nasceu e quantos dias ele já viveu.

Acho que deu para perceber que estamos repetindo várias informações, vejamos o código:

```
document.write("Nome: Gilherme Frota - idade: " + 8 + ", nasceu em: " + (2015 - 8));
document.write(" e já viveu " + (8*365) + " dias.<br>") ;
document.write("Nome: Nahan Frota - idade: " + 8 + " ,nasceu em: " + (2015 - 8));
document.write(" e já viveu " + (8*365) + " dias.<br>") ;
document.write("Nome: William Frota - idade: " + 3 + " ,nasceu em: " + (2015 - 3));
document.write(" e já viveu " + (3*365) + " dias.<br>") ;
document.write("Nome: Rafael Ponte - idade: " + 31 + " ,nasceu em: " + (2015 - 31));
document.write(" e já viveu " + (31*365) + " dias.<br>") ;
document.write("Nome: Lina Tarcia - idade: " + 32 + " ,nasceu em: " + (2015 - 32));
document.write(" e já viveu " + (32*365) + " dias.<br>") ;
document.write("Total de dias: " + ((8*365) + (8*365) + (3*365) + (31*365) + (32*365)));
document.write("<br>Média de dias: " + ((8*365) + (8*365)
+ (3*365) + (31*365) + (32*365)) / 5);
```

Observe que sempre repetimos o ano corrente, ou seja, o valor de 2015. Assim como repetimos os dias do ano 365. E se por algum motivo alterarmos a data? Por exemplo, quero aqui calcular o total de dias que meus amigos já viveram colocando logo o ano de 2016.

Vou ter que alterar um a um correto? E isso é bem chato, sem falar que podemos esquecer de alterar algum ano, fazendo que o cálculo fique inconsistente.

E mais, quando você vê o valor de 2015 para você que criou o programa é até fácil saber o que ele significa (o ano), mas será que outro programador vai saber que esse valor é referente a um ano? E o valor de 365, será que ele vai saber que isso é o total de dias de um ano?

Para resolver esse problema, precisamos de uma forma de dizer que esse valor é o ano sem perder o próprio valor e deixá-lo mais legível para quem quer que seja.

Dessa forma podemos **atribuir** o valor que queremos ao **ano**. Veja:

```
ano = 2015;
```

O que fizemos aqui foi criar uma **variável** chamada de `ano` e **atribuimos** a ela o valor que ela irá representar dentro do nosso código.

Utilizamos o símbolo `=` para dizer que o valor que está a *direita* será atribuído na *esquerda*.

Deixando a declaração mais legível

Outra forma de criar uma variável é utilizar a palavra chave **var** **antes** do nome da variável:

```
var ano;
```

No JavaScript não é obrigatório você utilizar essa palavra chave, porém já é interessante você ir se acostumando com esse tipo de sintaxe, pois a maioria das linguagens precisam que você defina essa informação antes de declarar qualquer variável.

Vejamos como iremos utilizar essa nova forma:

```
var ano = 2015;

document.write("Nome: Gilherme Frota - idade: " + 8 + ", nasceu em: " + (ano - 8));
document.write(" e já viveu " + (8*365) + " dias.<br>") ;
document.write("Nome: Nahan Frota - idade: " + 8 + ",nasceu em: " + (ano - 8));
document.write(" e já viveu " + (8*365) + " dias.<br>") ;
document.write("Nome: William Frota - idade: " + 3 + " ,nasceu em: " + (ano - 3));
document.write(" e já viveu " + (3*365) + " dias.<br>") ;
document.write("Nome: Rafael Ponte - idade: " + 31 + " ,nasceu em: " + (ano - 31));
document.write(" e já viveu " + (31*365) + " dias.<br>") ;
document.write("Nome: Lina Tarcia - idade: " + 32 + " ,nasceu em: " + (ano - 32));
document.write(" e já viveu " + (32*365) + " dias.<br>") ;
```

Observe que agora se quisermos alterar o ano, não precisamos mais alterar em todo local que ele é utilizado e sim apenas no valor que atribuído a variável **ano**. Sem falar que agora nosso código está ficando bem mais legível.

Podemos ainda melhorar nosso código criando uma variável que represente os dias do ano:

```
var ano = 2015;
var diasDoAno = 365;

document.write("Nome: Gilherme Frota - idade: " + 8 + ", nasceu em: " + (ano - 8));
document.write(" e já viveu " + (8*diasDoAno) + " dias.<br>") ;
document.write("Nome: Nahan Frota - idade: " + 8 + ",nasceu em: " + (ano - 8));
document.write(" e já viveu " + (8*diasDoAno) + " dias.<br>") ;
document.write("Nome: William Frota - idade: " + 3 + " ,nasceu em: " + (ano - 3));
document.write(" e já viveu " + (3*diasDoAno) + " dias.<br>") ;
document.write("Nome: Rafael Ponte - idade: " + 31 + " ,nasceu em: " + (ano - 31));
document.write(" e já viveu " + (31*diasDoAno) + " dias.<br>") ;
document.write("Nome: Lina Tarcia - idade: " + 32 + " ,nasceu em: " + (ano - 32));
document.write(" e já viveu " + (32*diasDoAno) + " dias.<br>") ;

document.write("Total de dias: " + ((8*diasDoAno) + (8*diasDoAno) + (3*diasDoAno)
+ (31*diasDoAno) + (32*diasDoAno)));
document.write("<br>Média de dias: " + ((8*diasDoAno) + (8*diasDoAno) + (3*diasDoAno)
+ (31*diasDoAno) + (32*diasDoAno)) / 5);
```

Tipos de valores nas variáveis

Uma variável poderá guardar o que você quiser: string(texto), número, outro código e expressões matemáticas etc.

Veja que temos duas expressões matemáticas, uma que exibe o total de dias e a outra a média de dias.

```
var totalDias = ((8*diasDoAno) + (8*diasDoAno) + (3*diasDoAno)
+ (31*diasDoAno) + (32*diasDoAno));
var mediaDias = ((8*diasDoAno) + (8*diasDoAno) + (3*diasDoAno)
+ (31*diasDoAno) + (32*diasDoAno)) / 5;
```

Dessa forma seu código vai ficar mais legível.

Estamos utilizando a variável `ano` sem as aspas, isso porque ela contém um valor atribuído para ela. Por exemplo, veja a diferença quando tento imprimir uma variável de duas maneiras:

```
var ano = 2015;
document.write(ano);
document.write("ano");
```

Resultado:

```
2015
ano
```

Quando colocamos uma variável entre as aspas (""), o JavaScript deixa de tratá-la como uma variável e a trata como uma sequência de caractéres. Ou seja, é apenas um outro texto que você deseja imprimir.

Case-sensitive

Case-sensitive é um termo que se refere a um tipo de análise tipográfica da informática. É quando queremos dizer que o “texto” é “*sensível ao tamanho das letras*” ou “*sensível a maiúsculas e minúsculas*”.

Diz-se que um software é case-sensitive ou possui “case sensitivity” quando ele é capaz de analisar uma cadeia de caracteres, avaliar a existência de caixa alta(**MAIÚSCULAS**) e caixa baixa(**MINÚSCULAS**).

O palavra **triadworks** é DIFERENTE de **TRIADWORKS** ou **TriadWorks**.

3.2 - Exercícios: Variáveis

1) Vamos exercitar um pouco a declaração de variáveis.

a) Crie um arquivo chamado `programa06.html` e faça a sua estrutura básica:

```
<meta charset="UTF-8">
<script>
</script>
```

b) Agora copie e cole o código do `programa05.html` que você fez no último exercício e cole entre as tags `<script>`:

```
document.write("Nome: Gilherme Frota - idade: " + 8 + ", nasceu em: " + (2015 - 8));
document.write(" e já viveu " + (8*365) + " dias.<br>") ;
document.write("Nome: Nahana Frota - idade: " + 8 + " ,nasceu em: " + (2015 - 8));
document.write(" e já viveu " + (8*365) + " dias.<br>") ;
document.write("Nome: William Frota - idade: " + 3 + " ,nasceu em: " + (2015 - 3));
document.write(" e já viveu " + (3*365) + " dias.<br>") ;
document.write("Nome: Rafael Ponte - idade: " + 31 + " ,nasceu em: " + (2015 - 31));
document.write(" e já viveu " + (31*365) + " dias.<br>") ;
```

```

document.write("Nome: Lina Tarcia - idade: " + 32 + " ,nasceu em: " + (2015 - 32));
document.write(" e já viveu " + (32*365) + " dias.<br>") ;
document.write("Total de dias: " + ((8*365) + (8*365) + (3*365)
+ (31*365) + (32*365)));
document.write("<br>Média de dias: " + ((8*365) + (8*365)
+ (3*365) + (31*365) + (32*365)) / 5);

```

- c) Agora declare as variáveis necessárias ao nosso código. Queremos representar os seguintes valores: ano, dias do ano, idade de cada amigo, total de dias e a média de dias. E altere onde for necessário no seu código.

```

<meta charset="UTF-8">
<script>
    var ano = 2015;
    var diasDoAno = 365;
    var idadeGuilherme = 8;
    var idadeNahan = 8;
    var idadeWilliam = 3;
    var idadeRafael = 31;
    var idadeLina = 32;

    var totalDias =
        (idadeGuilherme*diasDoAno)
        + (idadeNahan*diasDoAno)
        + (idadeWilliam*diasDoAno)
        + (idadeRafael*diasDoAno)
        + (idadeLina*diasDoAno);

    var mediaDias = totalDias / 5;
</script>

```

- d) Observe que ainda possuímos código sendo repetido (`idadeGuilherme*diasDoAno;`) que na verdade é o calculo para saber os dias de vida de cada amigo. Crie variáveis que condizem com código:

```

var diasGuilherme = idadeGuilherme*diasDoAno;
var diasNahan = idadeNahan*diasDoAno;
var diasWilliam = idadeWilliam*diasDoAno;
var diasRafael = idadeRafael*diasDoAno;
var diasLina = idadeLina*diasDoAno;

```

- e) Veja como nosso programa irá ficar:

```

var ano = 2015;
var diasDoAno = 365;

var idadeGuilherme = 8;
var idadeNahan = 8;
var idadeWilliam = 3;
var idadeRafael = 31;
var idadeLina = 32;

var diasGuilherme = idadeGuilherme*diasDoAno;
var diasNahan = idadeNahan*diasDoAno;
var diasWilliam = idadeWilliam*diasDoAno;
var diasRafael = idadeRafael*diasDoAno;
var diasLina = idadeLina*diasDoAno;

var totalDias =
    diasGuilherme + diasNahan
    + diasWilliam
    + diasRafael

```

```

+ diasLina;

var mediaDias = totalDias / 5;

document.write("Nome: Gilherme Frota - idade: "
    + idadeGuilherme + ", nasceu em: " + (ano - idadeGuilherme));
document.write(" e já viveu " + diasGuilherme + " dias.<br>");

document.write("Nome: Nahan Frota - idade: "
    + idadeNahan + ",nasceu em: " + (ano - idadeNahan));
document.write(" e já viveu " + diasNahan + " dias.<br>");

document.write("Nome: William Frota - idade: "
    + idadeWilliam + ",nasceu em: " + (ano - idadeWilliam));
document.write(" e já viveu " + diasWilliam + " dias.<br>");

document.write("Nome: Rafael Ponte - idade: "
    + idadeRafael + ",nasceu em: " + (ano - idadeRafael));
document.write(" e já viveu " + diasRafael + " dias.<br>");

document.write("Nome: Lina Tarcia - idade: "
    + idadeLina + ",nasceu em: " + (ano - idadeLina));
document.write(" e já viveu " + diasLina + " dias.<br>");

document.write("Total de dias: " + totalDias);
document.write("<br>Média de dias: " + mediaDias);

```

E o resultado será algo semelhante:

```

Nome: Gilherme Frota - idade: 8, nasceu em: 2007 e já viveu 2920 dias.
Nome: Nahan Frota - idade: 8 ,nasceu em: 2007 e já viveu 2920 dias.
Nome: William Frota - idade: 3 ,nasceu em: 2012 e já viveu 1095 dias.
Nome: Rafael Ponte - idade: 31 ,nasceu em: 1984 e já viveu 11315 dias.
Nome: Lina Tarcia - idade: 32 ,nasceu em: 1983 e já viveu 11680 dias.
Total de dias: 29930
Média de dias: 5986

```

3.3 - Reaproveitando nosso código com funções

Muitas vezes por mais que utilizamos variáveis para organizar nosso código, ainda assim precisamos teremos alguns trechos de códigos repetidos, como no exemplo abaixo:

```

1 document.write("Nome: William Frota - idade: "
2     + idadeWilliam + ",nasceu em: " + (ano - idadeWilliam));
3 document.write(" e já viveu " + diasWilliam + " dias.<br>");

```

Observe que para cada mensagem que irei exibir, vou precisar adicionar no final dela uma tag de uma nova linha (
).

Podemos criar uma variável que possua uma valor atribuido a ela com a tag:

```
var tagBr = "<br>";
```

Porém se você observar não queremos somente uma tag *br*, queremos que ele pule uma linha, ou seja, queremos uma **ação** de pular uma linha.

```
document.write("<br>");
```

E não é só isso. Se quisermos pular duas linhas? E se quisermos que ao pular uma linha ele coloque uma imagem? Para fazer isso em uma variável teremos muito trabalho.

Para se fazer esse tipo de operação, ou seja, de ter em um único lugar um grupo de código ou bloco de códigos, onde podemos executar com apenas uma menção, precisamos criar o que chamamos de **função**.

```
function novaLinha() {  
    document.write("<br>");  
}
```

Usamos a palavra chave **function** para dizer que o que estiver logo depois dessa palavra, será o nome da função e que todo o código deste bloco será uma função.

Não basta apenas dizer com a palavra chave **function** que esse código é uma função. Existe mais outra coisa que devemos fazer para que esse código seja entendido como função. Logo após do nome da função adicione os parênteses () e abra e feche as chaves {}.

Pronto, agora temos uma função e o que estiver dentro do bloco delimitado pelas chaves, será executado.

Mas podemos criar de outra forma. Lembra quando falamos que uma variável no JavaScript pode armazenar tudo? Já que estamos acostumados com variáveis, o JavaScript nos permite que uma variável possa ter também uma função, isso mesmo. No lugar da variável guardar valores como texto, números etc, ela irá guardar uma função.

```
var novaLinha = function (){  
    document.write("<br>");  
}
```

Veja que as principais regras da criação de uma função são respeitadas, como a palavra chave **function**, os parênteses () e as chaves {}.

E para executar essa função independente da forma como ela foi criada, basta chamar usando seu nome e os parênteses:

```
novaLinha();
```

Vejamos nosso código anterior alterado com a nova função:

```
document.write("Nome: William Frota - idade: "  
    + idadeWilliam + " ,nasceu em: " + (ano - idadeWilliam));  
document.write(" e já viveu " + diasWilliam + " dias.");  
  
novaLinha();
```

Dessa forma você poderá adicionar mais códigos a essa função caso deseje.

```
var novaLinha = function (){  
    document.write("<br>");  
    document.write("<br>");  
    document.write("<hr>");  
}
```

Ao executar a função ele irá pular duas linhas e exibir uma barra (<hr>).

Funções dentro de funções

Se você observar temos no final do nosso programa um trecho de código que imprime os totais.

```
document.write("Total de dias: " + totalDias);
document.write("<br>");
document.write("Média de dias: " + mediaDias);
```

Podemos criar uma função, onde ela ficará responsável por exibir esses totais.

```
var exibirTotais = function(){
    document.write("Total de dias: " + totalDias);
    document.write("<br>");
    document.write("Média de dias: " + mediaDias);
}
```

Se você lembrar, temos uma função responsável em pular uma nova linha, para isso não precisamos ficar repetindo código.

```
var exibirTotais = function(){
    document.write("Total de dias: " + totalDias);
    novaLinha();
    document.write("Média de dias: " + mediaDias);
}
```

Dessa forma estamos executando uma função, que internamente chama outra função. É dessa forma que você pode montar seu código, como se fosse um quebra-cabeça onde você irá pegando esses pedaços (funções) e montando como você deseja.

3.4 - Exercícios: Funções

1) Crie uma nova página o programa07.html e copie todo o conteúdo do programa06.html para dentro dele.

a) Crie uma função para imprimir uma nova linha:

```
var novaLinha = function(){
    document.write("<br>");
}
```

E não esqueça de retirar do código qualquer menção ao
.

b) Agora adicione a chamada da função novaLinha() onde você deseja.

```
var novaLinha = function(){
    document.write("<br>");
}

document.write("Nome: Gilherme Frota - idade: " + idadeGuilherme
    + ", nasceu em: " + (ano - idadeGuilherme));
document.write(" e já viveu " + diasGuilherme + " dias.");
novaLinha();

document.write("Nome: Nahana Frota - idade: " + idadeNahan
    + ", nasceu em: " + (ano - idadeNahan));
document.write(" e já viveu " + diasNahan + " dias.");
novaLinha();
```

```

document.write("Nome: William Frota - idade: " + idadeWilliam
    + " ,nasceu em: " + (ano - idadeWilliam));
document.write(" e já viveu " + diasWilliam + " dias.");
novaLinha();

document.write("Nome: Rafael Ponte - idade: " + idadeRafael
    + " ,nasceu em: " + (ano - idadeRafael));
document.write(" e já viveu " + diasRafael + " dias.");
novaLinha();

document.write("Nome: Lina Tarcia - idade: " + idadeLina
    + " ,nasceu em: " + (ano - idadeLina));
document.write(" e já viveu " + diasLina + " dias.");
novaLinha();

document.write("Total de dias: " + totalDias);
novaLinha();
document.write("Média de dias: " + mediaDias);

```

2) Agora vamos criar uma função para exibir os totais.

a) Crie logo após a função novaLinha() a função que será responsável em imprimir os totais:

```

var exibirTotais = function(){
    document.write("Total de dias: " + totalDias);
    document.write("Média de dias: " + mediaDias);
}

```

b) Agora adicione a chamada para a função novaLinha():

```

var exibirTotais = function(){
    document.write("Total de dias: " + totalDias);
    novaLinha();
    document.write("Média de dias: " + mediaDias);
}

```

c) Agora utilizando da tag <hr>, vamos adicioná-la no início e fim do código:

```

var exibirTotais = function(){
    document.write("<hr>");
    document.write("Total de dias: " + totalDias);
    novaLinha();
    document.write("Média de dias: " + mediaDias);
    document.write("<hr>");
}

```

d) Agora substitua o código anterior pela chamada da nossa nova função:

```

....codigo anterior....
document.write("Nome: Lina Tarcia - idade: " + idadeLina
    + " ,nasceu em: " + (ano - idadeLina));
document.write(" e já viveu " + diasLina + " dias.");
novaLinha();

exibirTotais();

```

3) Estamos repetindo muito o código que imprime uma linha no HTML. Crie uma função que imprime a linha e vamos adicionar na função exibirTotais():

a) Crie a nova função:

```
var linha = function(){
    document.write("<hr>");
}
```

- b) Agora altere a função onde o código está sendo repetido adicionando a chamada para `linha()`:

```
var exibirTotais = function(){
    linha();
    document.write("Total de dias: " + totalDias);
    novaLinha();
    document.write("Média de dias: " + mediaDias);
    linha();
}
```

Nosso código final deverá ser parecido com esse:

```
<meta charset="UTF-8">
<script>

    var ano = 2015;
    var diasDoAno = 365;
    var idadeGuilherme = 8;
    var idadeNahan = 8;
    var idadeWilliam = 3;
    var idadeRafael = 31;
    var idadeLina = 32;

    var diasGuilherme = idadeGuilherme*diasDoAno;
    var diasNahan = idadeNahan*diasDoAno;
    var diasWilliam = idadeWilliam*diasDoAno;
    var diasRafael = idadeRafael*diasDoAno;
    var diasLina = idadeLina*diasDoAno;

    var totalDias =
        diasGuilherme + diasNahan
        + diasWilliam
        + diasRafael
        + diasLina;

    var mediaDias = totalDias / 5;

    var novaLinha = function(){
        document.write("<br>");
    }

    var linha = function(){
        document.write("<hr>");
    }

    var exibirTotais = function(){
        linha();
        document.write("Total de dias: " + totalDias);
        novaLinha();
        document.write("Média de dias: " + mediaDias);
        linha();
    }

    document.write("Nome: Gilherme Frota - idade: "
        + idadeGuilherme + ", nasceu em: " + (ano - idadeGuilherme));
    document.write(" e já viveu " + diasGuilherme + " dias.");


```

```

novaLinha();

document.write("Nome: Nahan Frota - idade: "
    + idadeNahan + " ,nasceu em: " + (ano - idadeNahan));
document.write(" e já viveu " + diasNahan + " dias.") ;
novaLinha();

document.write("Nome: William Frota - idade: "
    + idadeWilliam + " ,nasceu em: " + (ano - idadeWilliam));
document.write(" e já viveu " + diasWilliam + " dias.") ;
novaLinha();

document.write("Nome: Rafael Ponte - idade: "
    + idadeRafael + " ,nasceu em: " + (ano - idadeRafael));
document.write(" e já viveu " + diasRafael + " dias.") ;
novaLinha();

document.write("Nome: Lina Tarcia - idade: "
    + idadeLina + " ,nasceu em: " + (ano - idadeLina));
document.write(" e já viveu " + diasLina + " dias.");
novaLinha();

exibirTotais();

</script>

```

3.5 - Passando informações para as funções

Desde a primeira versão do nosso código já estamos percebendo que ele vem mudado muito e para melhor. Pois agora podemos mudar a qualquer momento o comportamento do nosso programa, seja as variáveis ou as nossas funções.

Porém ainda possuímos códigos que são repetitivos que no final irão mostrar o mesmo texto, alterando apenas alguns valores como nome, idade etc.

```

Nome: Gilherme Frota - idade: 8, nasceu em: 2007 e já viveu 2920 dias.
Nome: Nahan Frota - idade: 8 ,nasceu em: 2007 e já viveu 2920 dias.
Nome: William Frota - idade: 3 ,nasceu em: 2012 e já viveu 1095 dias.
Nome: Rafael Ponte - idade: 31 ,nasceu em: 1984 e já viveu 11315 dias.
Nome: Lina Tarcia - idade: 32 ,nasceu em: 1983 e já viveu 11680 dias.

```

Nós já sabemos que podemos criar uma função para isso:

```

var exibirGuilherme = function(){
    document.write("Nome: Gilherme Frota - idade: "
        + idadeGuilherme + " ,nasceu em: " + (ano - idadeGuilherme));
    document.write(" e já viveu " + diasGuilherme + " dias.") ;
    novaLinha();
}

```

Apesar disso deixar nosso código mais legível, ainda temos um problema, teremos que repetir todo o bloco de código da mesma forma, só que agora em cada função.

Sem falar que iremos ter várias funções, uma para cada amigo. E se eu quiser adicionar um novo amigo? Vou ter que criar uma nova função e repetir todo o código novamente. Acho que isso não é legal, você concorda?

Vamos criar uma função que não fique “presa” ao nome do amigo:

```
var exibir = function(){
    document.write("Nome: Gilherme Frota - idade: "
        + idadeGuilherme + ", nasceu em: " + (ano - idadeGuilherme));
    document.write(" e já viveu " + diasGuilherme + " dias.");
    novaLinha();
}
```

Esta função ainda não é muito útil, pois ela irá exibir sempre os mesmos valores. O que precisamos fazer é ao criarmos uma função **dizer** a ela que ela irá receber algum valor que podemos passar.

Para *dizer* que a função recebe algum valor, basta **declarar dentro dos parênteses** uma variável, dessa forma ela sabe que irá receber um valor, que será atribuído a essa variável. Vejamos:

```
var exibir = function(valor){
}
```

Dessa forma você poderá utilizar a variável `valor` dentro do bloco da sua função. Essa variável também é conhecida como **argumentos** ou **parâmetros** da função.

```
var exibir = function(valor){
    document.write("Nome: " + valor + " - idade: "
        + idadeGuilherme + ", nasceu em: " + (ano - idadeGuilherme));
    document.write(" e já viveu " + diasGuilherme + " dias.");
    novaLinha();
}
```

Para deixar mais legível nosso código vamos dar um nome a variável que condiz com o que ela representa, que é o nome do nosso amigo.

```
var exibir = function(nome){
    document.write("Nome: " + nome + " - idade: "
        + idadeGuilherme + ", nasceu em: " + (ano - idadeGuilherme));
    document.write(" e já viveu " + diasGuilherme + " dias.");
    novaLinha();
}
```

E agora basta executar essa função, agora observe que anteriormente para executar alguma função usávamos apenas os parênteses vazios:

```
novaLinha();
```

Porém agora a nossa função **necessita** de uma informação para que ela possa ser executada corretamente e na sua declaração dizemos que ela recebe um valor `function(nome)`, então para executarmos ela precisamos passar esse valor:

```
exibir("Handerson Frota");
```

O resultado será algo semelhante:

```
Nome: Handerson - idade: 8, nasceu em: 2007 e já viveu 2920 dias.
```

Agora observe que existem outras informações que o código precisa, como nome por exemplo. Se eu desejar posso passar essas informações, basta separar cada argumento por “vírgulas”.

```
var exibir = function(nome, idade){
    document.write("Nome: " + nome + " - idade: "
        + idade + ", nasceu em: " + (ano - idade));
    document.write(" e já viveu " + diasGuilherme + " dias.");
    novaLinha();
}
```

E para executar segue a mesma regra das vírgulas:

```
exibir("Handerson", 33);
```

Com esses recusos já podemos melhorar e organizar nosso código de uma maneira que ele tenha possibilidade de mudança de comportamentos.

3.6 - Exercícios

1) Vamos criar um novo arquivo programa08.html com o mesmo código do programa07.html, para dessa forma podermos comparar as alterações e evolução do nosso código.

a) Vamos criar a função que irá ser mais genérica e irá exibir todas as informações necessárias. Essa função irá precisar das seguintes informações: nome do amigo, idade, o ano e os dias que ele viveu:

```
var exibirAmigo = function(nome, idade, ano, diasDeVida){

}
```

b) Agora vamos criar o corpo da função, que irá continuar imprimindo o mesmo texto que já temos: "Nome: Handerson - idade: 33, nasceu em: 1982 e já viveu 2920 dias.".

```
document.write("Nome: William Frota - idade: "
    + idadeWilliam + ",nasceu em: " + (ano - idadeWilliam));
document.write(" e já viveu " + diasWilliam + " dias.");
```

c) Retire o que não for necessário, melhorando o código. Queremos para facilitar apenas o texto sem as informações, apenas para deixar o código mais legível:

```
document.write("Nome: - idade: ,nasceu em: e já viveu dias.");
```

d) Agora iremos adicionar esse trecho na nossa função:

```
var exibirAmigo = function(nome, idade, ano, diasDeVida){
    document.write("Nome: - idade: ,nasceu em: e já viveu dias.");
}
```

e) Com o código pronto, agora iremos utilizar as variáveis da função que contém as informações corretas:

```
var exibirAmigo = function(nome, idade, ano, diasDeVida){
    document.write("Nome: " + nome +
        " - idade: " + idade + ", nasceu em: " + (ano - idade) +
        " e já viveu " + diasDeVida + " dias.");
    novaLinha();
}
```

f) Agora retire os trechos de código que serão substituídos pela função:

```
document.write("Nome: Gilherme Frota - idade: "
    + idadeGuilherme + ", nasceu em: "
    + (ano - idadeGuilherme));
document.write(" e já viveu " + diasGuilherme + " dias.");
novaLinha();
```

Alterando pela função:

```
exibirAmigo("Guilherme Frota", idadeGuilherme, ano, diasGuilherme);
```

g) Veja como o código irá ficar no final:

```
var ano = 2015;
var diasDoAno = 365;
var idadeGuilherme = 8;
var idadeNahan = 8;
var idadeWilliam = 3;
var idadeRafael = 31;
var idadeLina = 32;

var diasGuilherme = idadeGuilherme*diasDoAno;
var diasNahan = idadeNahan*diasDoAno;
var diasWilliam = idadeWilliam*diasDoAno;
var diasRafael = idadeRafael*diasDoAno;
var diasLina = idadeLina*diasDoAno;

var totalDias =
    diasGuilherme + diasNahan
    + diasWilliam
    + diasRafael
    + diasLina;

var mediaDias = totalDias / 5;

var novaLinha = function(){
    document.write("<br>");
}

var linha = function(){
    document.write("<hr>");
}

var exibirTotais = function(){
    linha();
    document.write("Total de dias: " + totalDias);
    novaLinha();
    document.write("Média de dias: " + mediaDias);
    linha();
}

var exibirAmigo = function(nome, idade, ano, diasDeVida){
    document.write("Nome: " + nome +
        " - idade: " + idade + ", nasceu em: " + (ano - idade) +
        " e já viveu " + diasDeVida + " dias.");
    novaLinha();
}

exibirAmigo("Guilherme Frota", idadeGuilherme, ano, diasGuilherme);

exibirAmigo("Nahan Frota", idadeNahan, ano, diasNahan);

exibirAmigo("William Frota", idadeWilliam, ano, diasWilliam);

exibirAmigo("Rafael Ponte", idadeRafael, ano, diasRafael);

exibirAmigo("Lina Tarcia", idadeLina, ano, diasLina);
```

```
exibirTotais();
```

3.7 - Recebendo informações de uma função

Já entendemos como utilizar nossas funções e dizer para elas o que queremos (passagem de parâmetros). Mas se quisermos perguntar algo a função?

Por exemplo, queremos saber qual o resultado se somarmos dois números: $4890 + 3409$? A função terá que executar essa operação e nos dizer (devolver) o resultado desta operação.

```
var soma = function(){
    4890 + 3409;
}
```

Em uma situação dessas, colocamos o resultado dentro de uma variável:

```
var soma = function(){
    var resultado = 4890 + 3409;
}
```

Mas não queremos guardar esse valor dentro da função, queremos que ela nos diga o resultado.

Chegou a vez de deixar a função nos dizer o que ela fez, ou seja, queremos que ela nos dê uma resposta, um **retorno** no caso do JavaScript usamos o `return`. Vejamos como fica:

```
var soma = function(){
    var resultado = 4890 + 3409;

    return resultado;
}
```

Ou podemos fazer direto:

```
var soma = function(){
    return 4890 + 3409;
}
```

Dessa forma, a função irá nos retornar o resultado:

```
var valor = soma();
```

Podemos passar os valores para a função que na verdade é o mais correto, assim podemos somar qualquer valor:

```
var soma = function(valor1, valor2){
    return valor1 + valor2;
}
```

E usaríamos dessa forma:

```
var valor = soma(300, 400);
```

3.8 - Exercícios: Funções com retorno

1) Crie o arquivo chamado programa09.html e salve.

- a) Agora vamos declarar as variáveis que irão representar o valor da gasolina, a capacidade do tanque e a quilometragem que o veículo percorreu. Lembre-se de sempre colocar seu código JavaScript dentro das tags `<script>...</script>`.

```
var valorGasolina = 3.50;
var capacidadeTanque = 13;
var percorreu = 360;
```

- b) Agora iremos criar uma função que irá calcular e **retornar** quanto custa encher o tanque, para isso basta multiplicar o **valor da gasolina** pela **capacidade do tanque**:

```
var custoEncherTanque = function(){
    return valorGasolina * capacidadeTanque;
}
```

- c) Agora iremos criar a função que irá calcular qual a quilometragem por litro e retornar o valor do cálculo. Lembrando que para fazer esse cálculo basta dividir a **quilometragem percorrida** pela **capacidade do tanque**.

```
var quilometroPorLitro = function(){
    return percorreu / capacidadeTanque;
}
```

2) Vamos testar nosso programa:

- a) Vamos exibir os seguintes dados:

```
Valor da Gasolina: R$ 3.5
Capacidade do Tanque: 13 litros
Total percorrido: 360 km
Quilômetro por Litro: 27.692307692307693 por litro
```

- b) E para isso precisamos criar as funções que irão imprimir o resultado no HTML:

```
document.write("Valor da Gasolina: R$ " + valorGasolina);
novaLinha();

document.write("Capacidade do Tanque: " + capacidadeTanque + " litros");
novaLinha();

document.write("Total percorrido: " + percorreu + " km");
novaLinha();

document.write("Quilômetro por Litro: " + quilometroPorLitro() + " por litro");
novaLinha();
```

- c) Observe que estamos usando a função `novaLinha`, você poderá criar novamente ou copiar de outro programa que você já fez.

```
var novaLinha = function(){
    document.write("<br>");
}
```

- d) O resultado deverá ser algo semelhante:

```
Valor da Gasolina: R$ 3.5
Capacidade do Tanque: 13 litros
Total percorrido: 360 km
Quilômetro por Litro: 27.692307692307693 por litro
```

Exercitando nossa lógica

4.1 - Alguns problemas do dia a dia

As vezes nosso código não faz o esperado, esquecemos alguma coisa, estamos usando a variável errada, atribuimos o valor errado. Existem várias possibilidades, esses problemas são conhecidos como **bug**.

Como o JavaScript é uma linguagem muito flexível que nos permite quase tudo, isso pode ser um ponto negativo quando erramos alguma coisa, pois dificulta encontrar o que exatamente nos erramos.

Para isso vamos utilizar algumas ferramentas que nos ajudam a trabalhar com essa linguagem e dessa forma facilitar nosso aprendizado.

Ferramentas de debug

Como comentamos qualquer erro no código é conhecido como **bug** e o ato de encontrar e verificar esses bugs, chamamos de **debuggar** ou **depurar**.

Na maioria dos browsers hoje já possuem nativos uma ferramenta para debugar o código/página. Ao acionar a opção *Inspecionar Elemento* do browser como já fizemos para verificar o console no começo do curso, vamos agora selecionar nesta ferramenta a aba **Depurar**.

```

<meta charset="UTF-8">
<script>
var ano = 2015;
var diasDoAno = 365;
var idadeGuilherme = 8;
var idadeNahan = 8;
var idadeWilliam = 3;
var idadeRafael = 31;
var idadelina = 32;
var diasGuilherme = idadeGuilherme*diasDoAno;
var diasNahan = idadeNahan*diasDoAno;
var diasWilliam = idadeWilliam*diasDoAno;
var diasRafael = idadeRafael*diasDoAno;

```

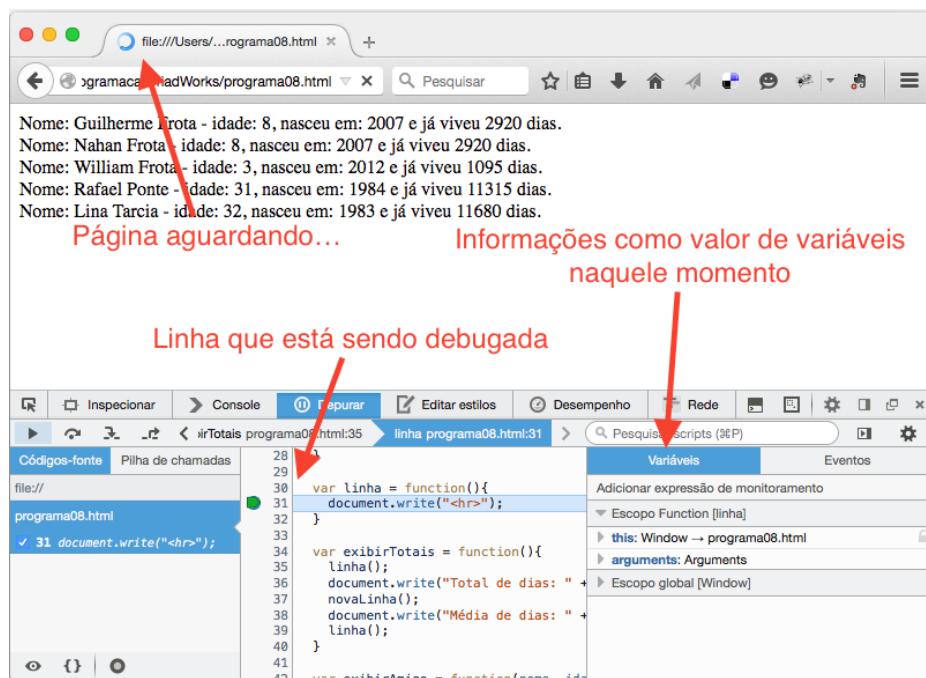
Para testar seu código e verificar o que está acontecendo você precisa adicionar o que achamos de ponto de parada, ou **break point**. Dando dois ou um click (dependendo da ferramenta) ao lado da linha que você deseja verificar.

```

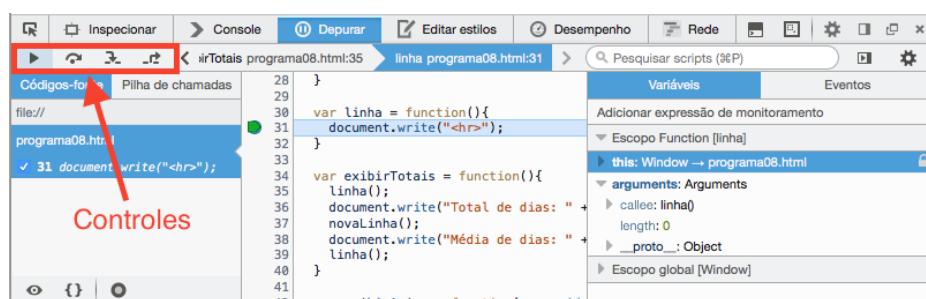
document.write("<br>");
}
var linha = function(){
document.write("<hr>");
}
var exibirTotais = function(){
linha();
document.write("Total de dias: " + totalDias);
novaLinha();
document.write("Média de dias: " + mediaDias);
linha();
}

```

Após adicionar o break point, basta atualizar a página com o F5. Observe que a ferramenta mostra a linha exata onde você está debugando e as informações sobre o seu código e variáveis naquele momento. Veja também que o browser está “parado” exatamente naquele momento.



Para “caminhar” nas linhas do código basta utilizar os controles de debug.



Essa é uma forma bem simples de verificar o que está acontecendo com seu código porém é por muito considerado uma prática não muito legal. Já que se você precisa ficar debugando o seu código é porque nem você entendeu o que fez. Iremos falar mais sobre isso no capítulo de Boas Práticas.

Informações somente para programadores

Além do debug as vezes precisamos exibir certas informações que não queremos exibir ao usuário, ou seja informações que queremos exibir somente no desenvolvimento. Dessa forma fica chato ficar utilizando alerts ou meio intrusivo utilizando `document.write`, já que ele irá alterar o HTML.

Nós só queremos exibir algumas informações apenas para nos certificar que aquela variável está realmente com aquele valor ou se uma função Y foi realmente executada.

Para isso podemos utilizar o comando `console.log` que tem exatamente essa finalidade. Vejamos o exemplo abaixo:

```
document.write("Mensagem que é exibida no HTML");
console.log("Essa mensagem é somente para desenvolvimento e
só irá aparecer no console do inspecionar elemento.");
```

Mensagem que é exibida no HTML



4.2 - Exercícios

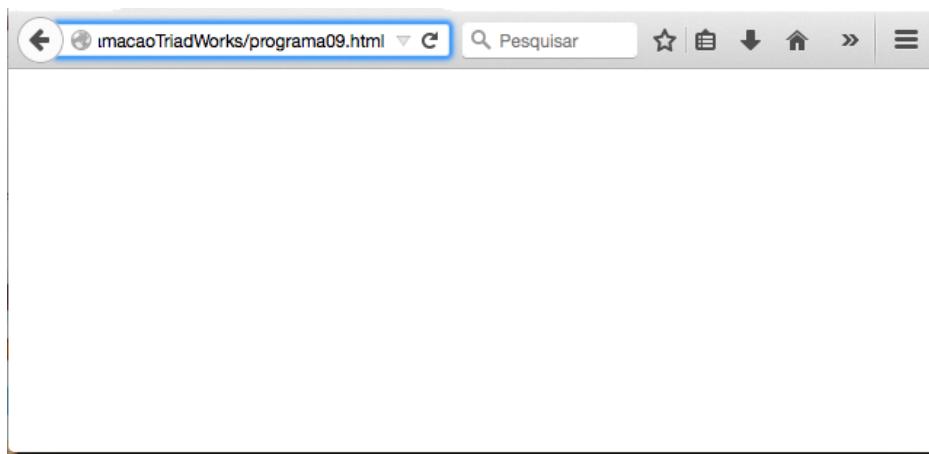
1) Crie o arquivo programa10.html.

a) Agora crie uma função e ao invés de imprimir a mensagem no HTML, imprima no console do browser:

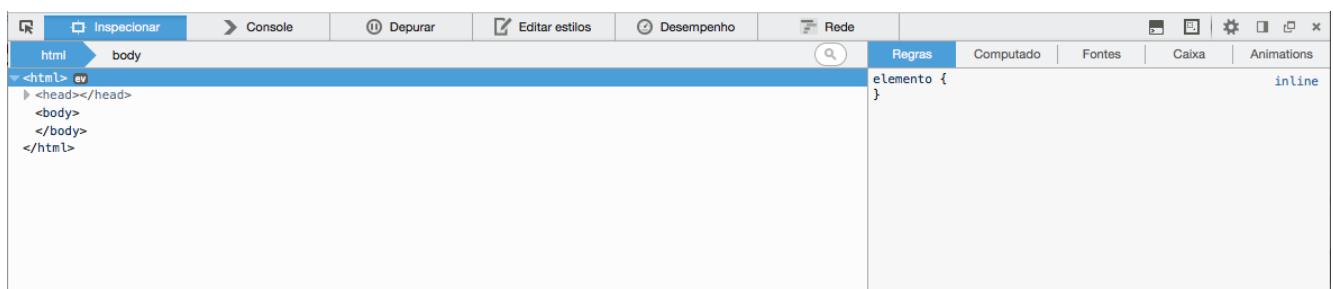
```
<meta charset="UTF-8">
<script>
    var testaConsole = function(){
        console.log("Testando o console!");
    }

    testaConsole();
</script>
```

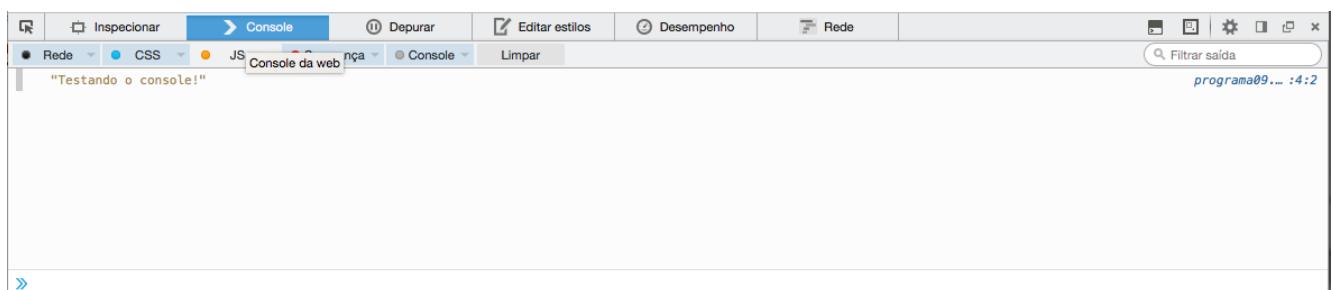
b) Abra o programa no browser:



c) Acesse com o botão direito na página o **Inspecionar Elemento**:



d) Clique na aba *Console* para verificar a mensagem, veja que no HTML ela não é exibida.

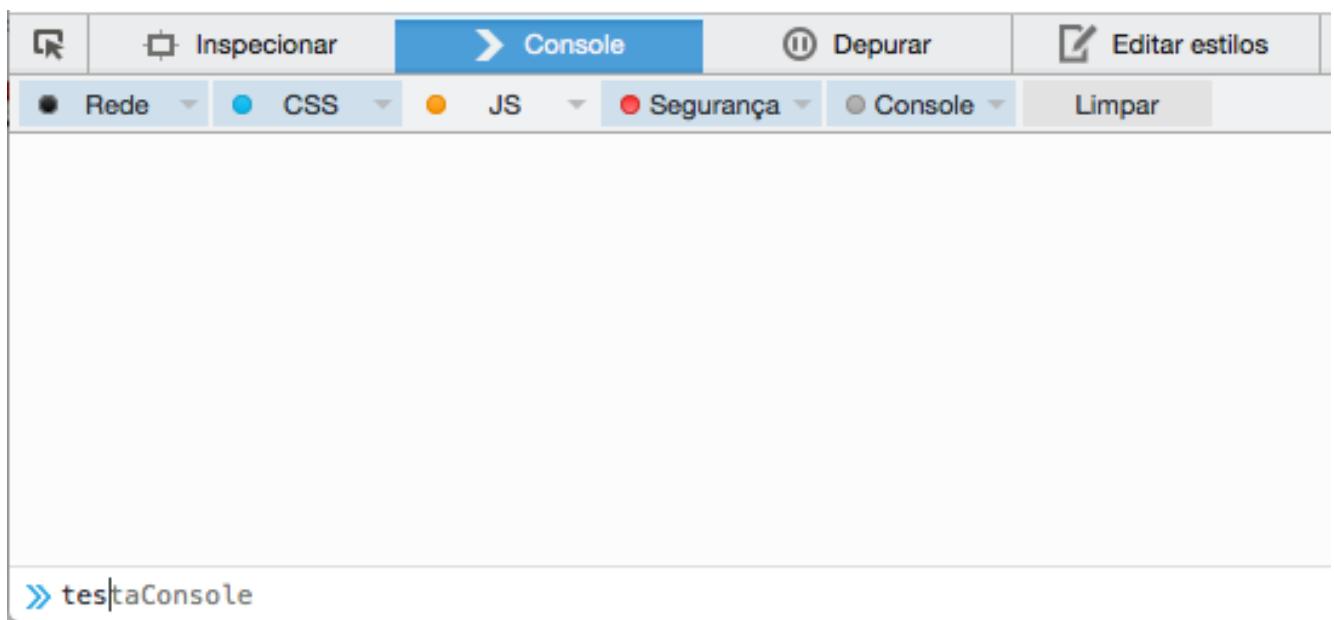


2) Vamos agora brincar mais um pouco com o console.

a) Retire a chamada para a função testaConsole(); evitando assim que ao abrir a página o código seja executado.

```
<meta charset="UTF-8">
<script>
    var testaConsole = function(){
        console.log("Testando o console!");
    }
</script>
```

b) Agora na aba *Console* acesse o espaço logo abaixo para digitar o comando, no nosso caso queremos executar a função testaConsole() diretamente no browser.

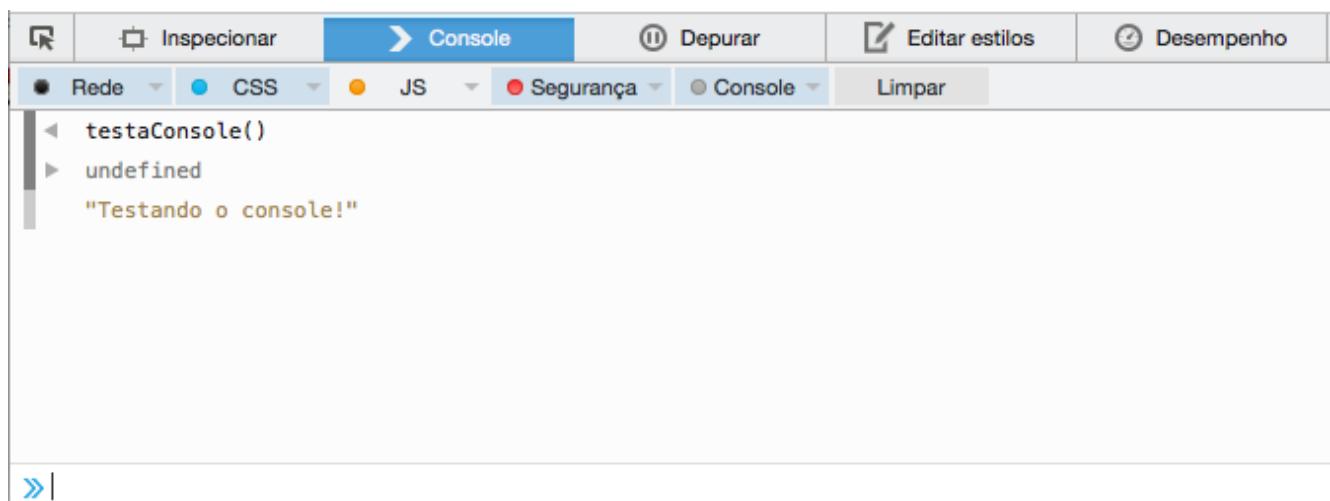


Veja que ao iniciar a digitação do comando ele irá exibir as funções e/ou variáveis que essa página possui.

- c) Ao finalizar a digitação do comando, pressione **Enter**:

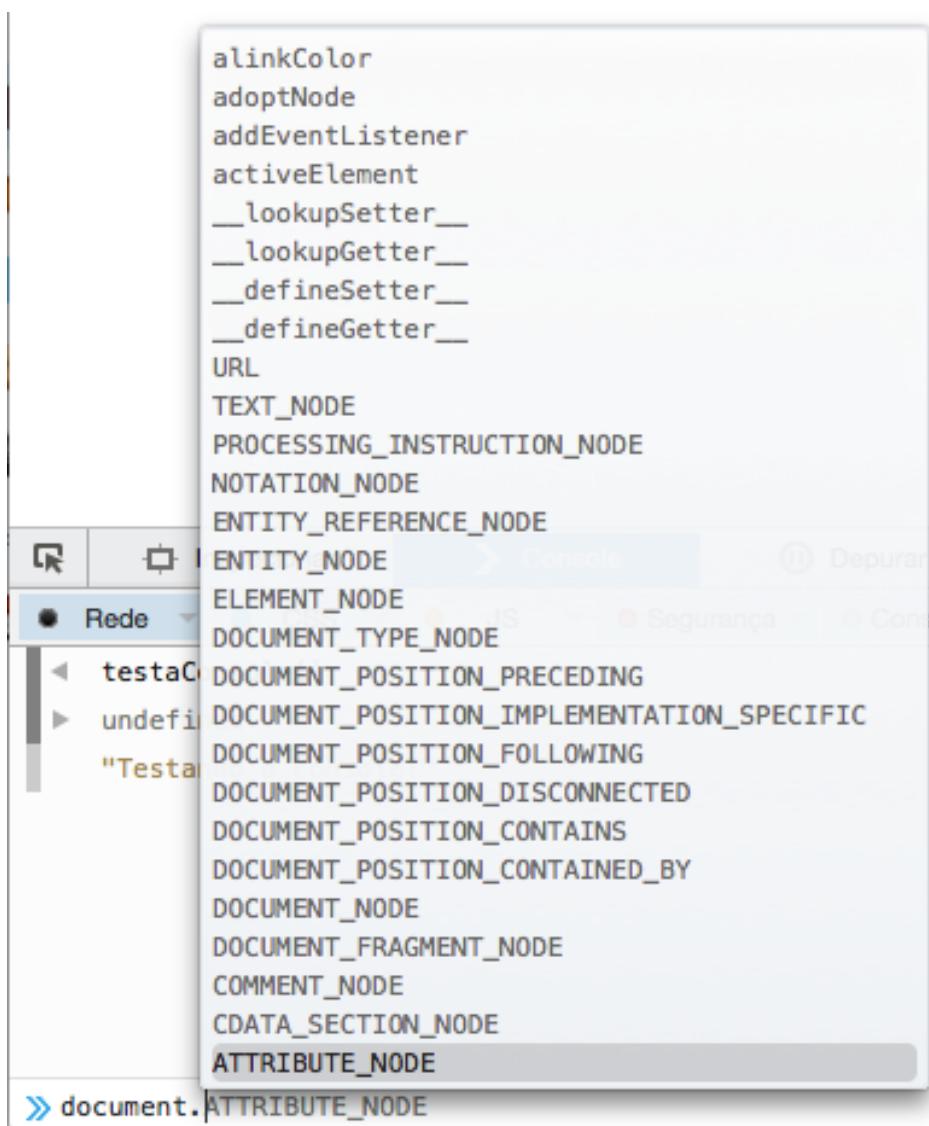


Você executou a função diretamente pelo browser, veja o console o que ele exibe:



3) Digitando outros comandos diretamente no console.

- a) Inicie a digitação com o comando `document.`, e ao pressionar o `" "` você verá que o console irá te sugerir vários comandos que pertencem ao `document`:



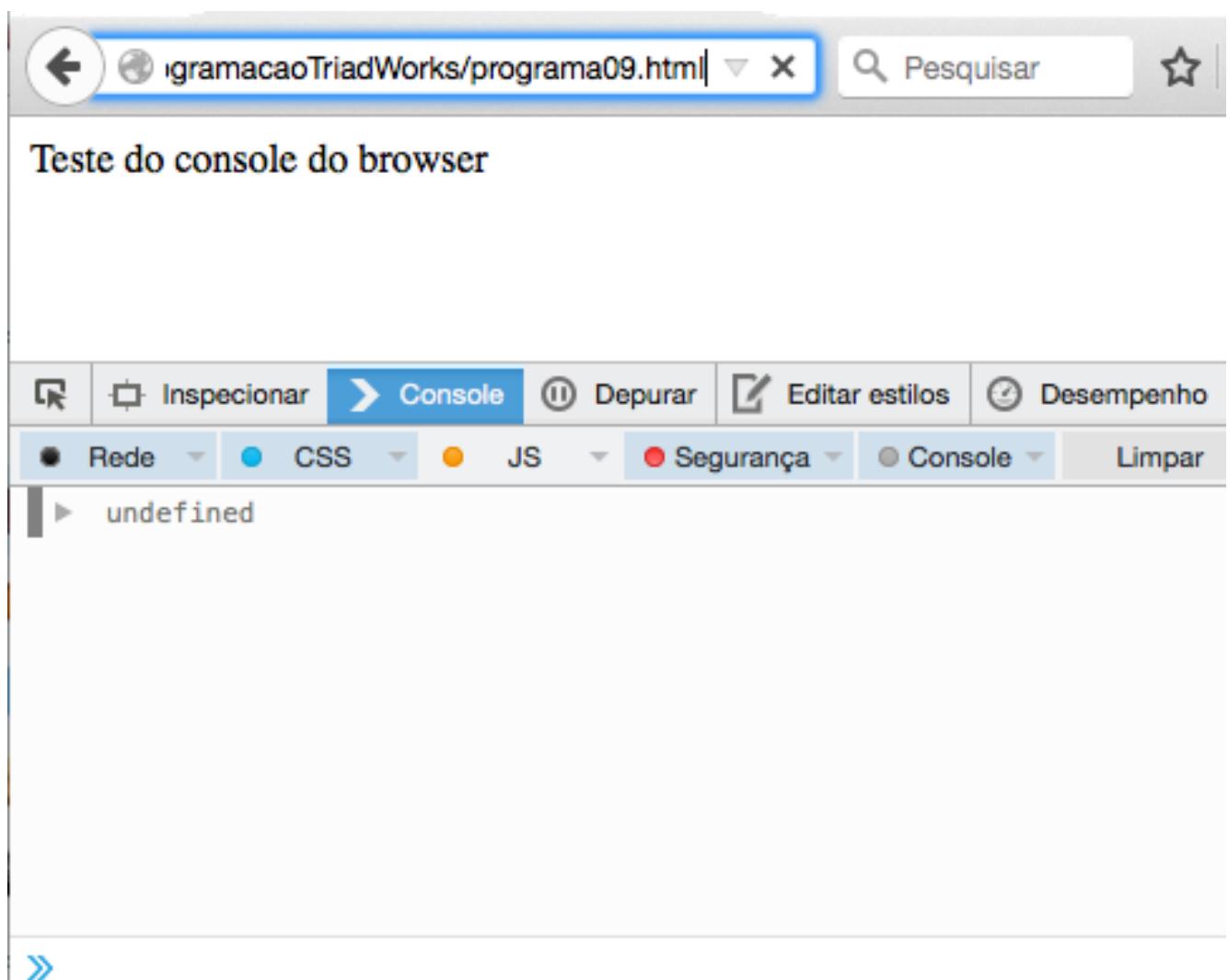
b) Após o “” inicie a digitação do write e selecione ele.

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. In the bottom-left corner of the console area, there is a dropdown menu with two options: 'writeln' and 'write'. The 'write' option is highlighted with a gray background. Below this menu, the command '» document.write' is visible in the console history.

c) Adicione uma mensagem qualquer dentro do document.write():

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The bottom part of the screen displays the command '» document.write("Teste do console do browser")' entered into the console. The text 'Teste do console do browser' is displayed in orange, indicating it was output by the script.

d) Ao finalizar pressione **Enter** para que ele seja executado.



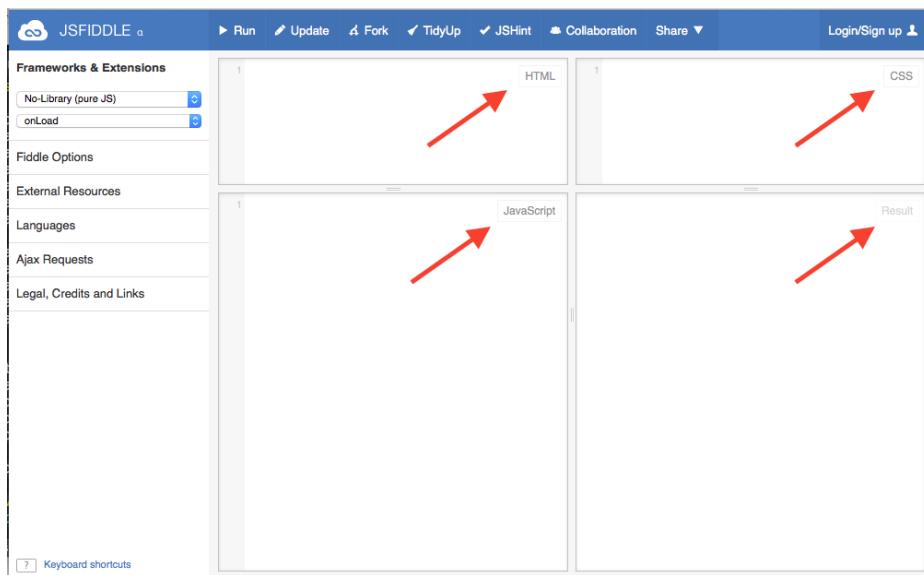
Observe que ele vai “adicionar” a mensagem no seu HTML, igualmente como você vem fazendo até agora, porém, executando diretamente do console do browser.

4.3 - Para saber mais: Compartilhando código

Todo aprendizado requer além de muito estudo, colaboração. Que tal compartilhar seu código com algum amigo para que juntos possam trocar mais conhecimento?

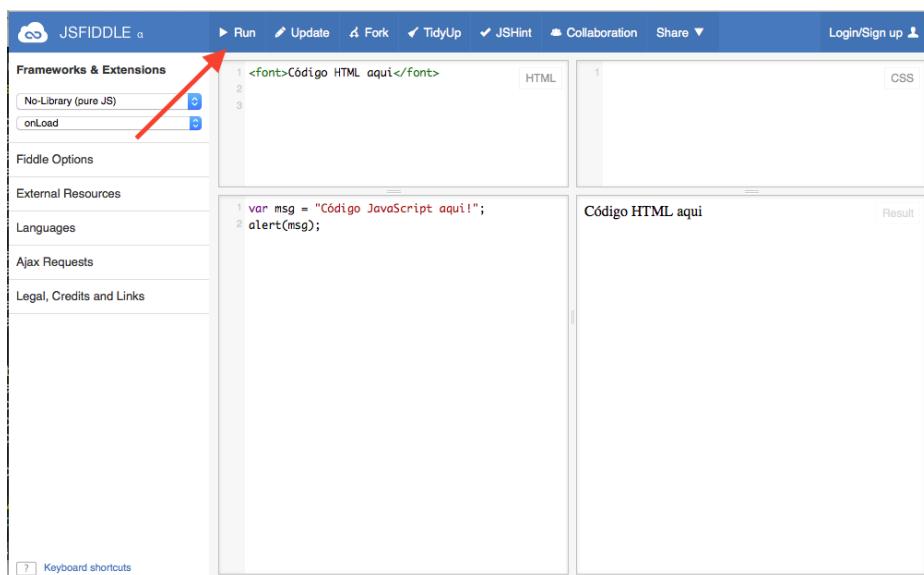
Existe uma ferramenta perfeita para isso e muito simples de utilizar.

O JSFIDDLE (<http://jsfiddle.net/xarw4hLv/>) . Com uma interface bem fácil e intuitiva você poderá criar seu código JavaScript e compartilhar com qualquer amigo.

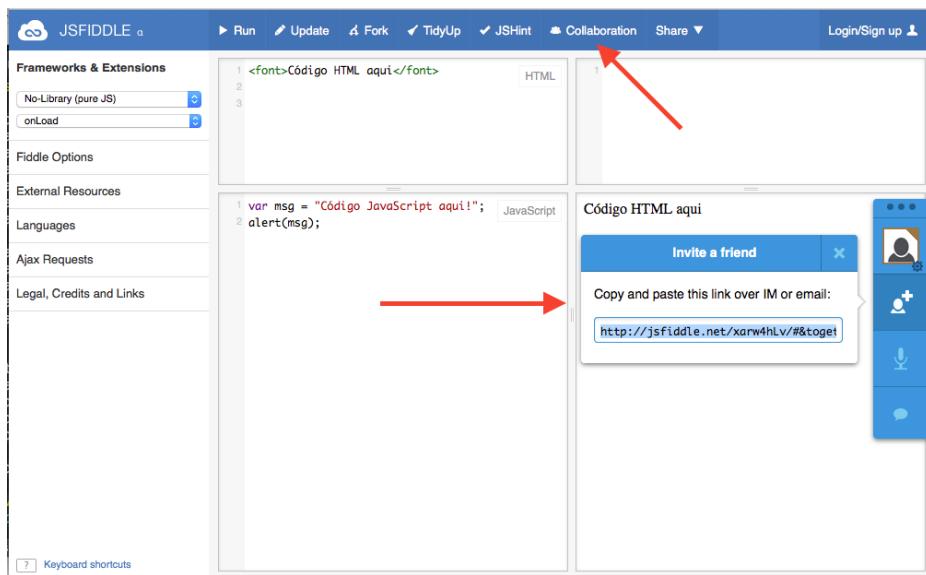


Ela terá 4 espaços: HTML, CSS, JavaScript e o Resultado. Essa ferramenta foi feita para utilizar JavaScript, logo você **não precisa e nem deve** utilizar a tag <script>.

Você também não irá poder utilizar o document.write() por motivos obvios, já que se você escrever no document você irá “retirar” a ferramenta da página, por isso ele nem permite que você utilize. Depois do código feito basta clicar em **RUN**.



E se você quiser não somente exibir o código para um amigo, mas também que vocês possam alterar o código e que cada um veja o que está fazendo, assim como podem também conversar durante esse processo, basta utilizar a opção **Collaboration** e convidar seu amigo.



Crie uma conta e se quiser salve seu código JavaScript na ferramenta :).

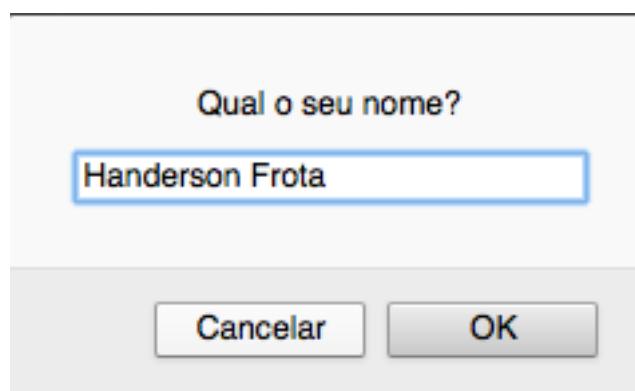
4.4 - Usuário interagindo com o seu programa

Quase todo programa(se não todos) desenvolvido precisa de alguma interação com o usuário ou seja com quem está ou vai usar o programa. Em algum momento o programa deve solicitar alguma informação. Nem sempre podemos deixar os valores fixos no próprio código como estamos fazendo.

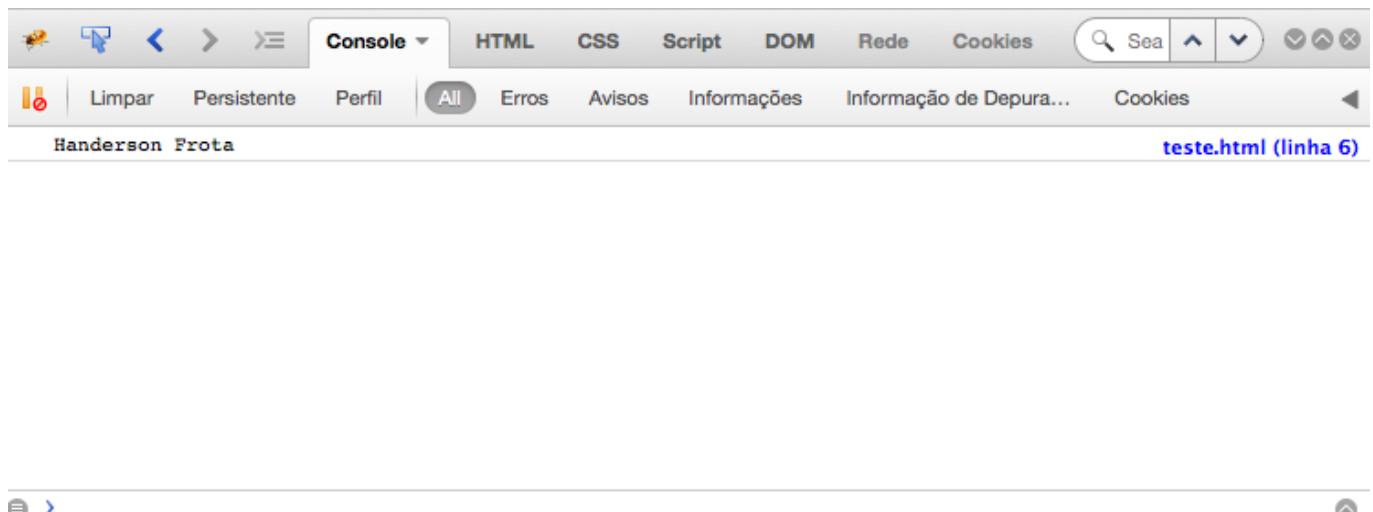
E para estes casos podemos utilizar a função chamada `prompt` do JavaScript para que ele **peça** ao usuário alguma informação, como o nome dele por exemplo:

```
var nome = prompt("Qual o seu nome?");
console.log(nome);
```

Ao executar esse código o navegador irá abrir a tela abaixo com um espaço para que o usuário digite o valor solicitado:



E logo exibindo o resultado:



O valor recebido pode ser qualquer coisa, texto, números etc. Podemos por exemplo solicitar ao usuário que nos informe quais números ele deseja efetuar a soma. Lembra da função que soma que fizemos no capítulo anterior?

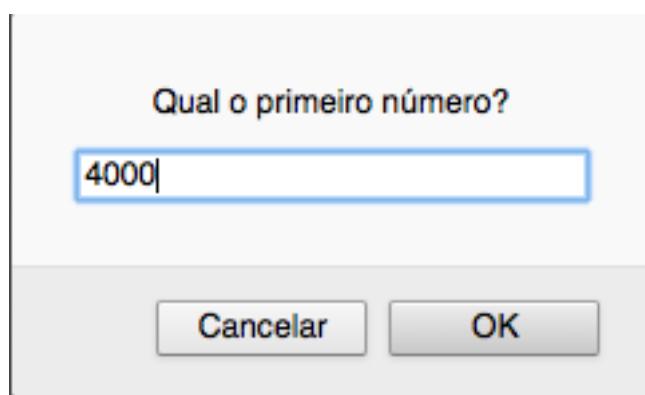
```
var soma = function(valor1, valor2){  
    return valor1 + valor2;  
}
```

Podemos passar para essa função os valores informados pelo usuário:

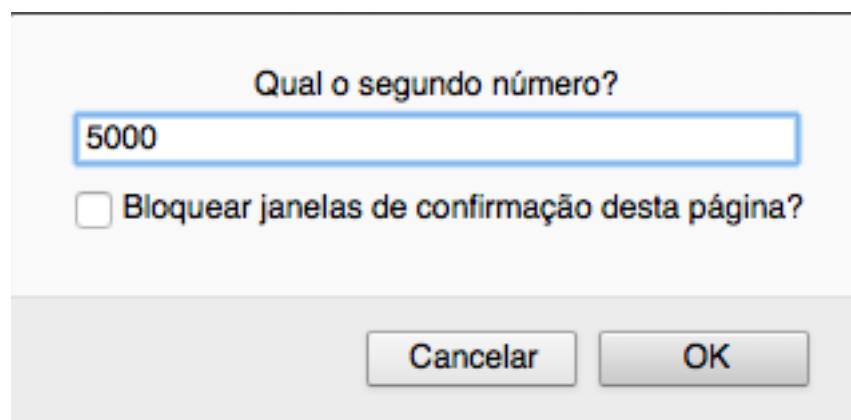
```
var soma = function(valor1, valor2){  
    return valor1 + valor2;  
}  
  
var primeiroValor = prompt("Qual o primeiro número?");  
var segundoValor = prompt("Qual o segundo número?");  
  
var resultado = soma(primeiroValor, segundoValor);  
console.log(resultado);
```

Dessa forma ele exibe **duas** janelas uma de cada vez solicitando os valores respectivos:

Solicitando o primeiro número:



Solicitando o segundo número:



Veja o resultado:

40005000

Opa, tivemos um problema? Sim, perceba que ele **não** efetuou uma operação matemática com o símbolo(+) e sim uma concatenação. Isso porque os valores recebidos são textos. O navegador não sabe que tipo de valor você irá colocar por isso ele *diz* que todos são textos.

Precisamos de alguma forma de “transformar” esses valores em números reais.

Garantindo que será um número

Quando executamos a função soma e passamos os valores os mesmos **não** são somados e sim concatenados e não queremos isso. Queremos que o código seja entendido como **número** e não como texto(string).

```
var soma = function(valor1, valor2){
    return valor1 + valor2;
}
```

O resultado foi:

40005000

Para isso vamos utilizar a função parseInt() que irá receber o valor/variável que será um número e dessa forma ele irá **retornar** um número do valor passado:

```
var valor = parseInt("222");
```

Perceba que passamos uma string ou texto, porém isso é um número, logo ele irá devolver ele como número e assim podemos efetuar a operação. Vamos alterar nossa função soma para que logo após ele receber os valores ele efetue a operação matemática com números:

```
var soma = function(valor1, valor2){
    var valorNumero1 = parseInt(valor1);
    var valorNumero2 = parseInt(valor2);
    return valorNumero1 + valorNumero2;
}
```

Ou pode melhorar o código sendo mais direto. Com o retorno da função `parseInt()` ele já efetue a operação e retorne seu resultado:

```
var soma = function(valor1, valor2){  
    return parseInt(valor1) + parseInt(valor2);  
}
```

Agora executando novamente o código, temos o resultado esperado:

9000

E se o usuário digitar um valor inválido, ou seja no lugar de um número ele digitar uma letra? Bem, caso ele faça isso a função `parseInt` irá retornar um resultado diferente que será o símbolo **NaN**, que nada mais é que um acrônimo em inglês para **Not a Number**. Caso isso aconteça a operação não será efetuada.

4.5 - Exercícios: Interagindo com o seu programa

1) Crie o arquivo chamado `programa11.html` e salve.

a) Utilize o `prompt()` para perguntar o nome ao seu usuário e salve esse valor em uma variável para que possamos utilizá-la mais a frente:

```
var nome = prompt("Qual seu nome?");
```

b) Utilize o `prompt()` para perguntar a idade do seu usuário e salve esse valor em uma variável para utilizá-la mais a frente:

```
var idade = prompt("Qual a sua idade?");
```

c) Agora vamos imprimir a mensagem:

```
document.write("Olá " + nome + " tudo bem? Você tem "  
+ idade + " anos e ainda não sabe programar?");
```

4.6 - Melhorando nossos programas, interagindo mais

Para aplicar tudo que vimos até agora vamos melhorar nossa interação com alguns programas que já fizemos. O primeiro será o programa que calcula os gastos da gasolina que fizemos no arquivo `programa11.html`.

Observe que estamos adicionando os valores diretamente no código:

```
var valorGasolina = 3.50;  
var capacidadeTanque = 13;  
var percorreu = 360;
```

No lugar de adicionar os valores diretamente no código vamos pedir que o usuário entre com os valores dele e para isso usaremos a função `prompt`:

```
var valorGasolina = prompt("Qual o valor da Gasolina?");  
var capacidadeTanque = prompt("Qual a capacidade do seu tanque?");  
var percorreu = prompt("Quanto você já percorreu?");
```

Veja que agora estamos “pegando” o retorno da função `prompt` e salvando na variável. Esse retorno é exatamente o que o usuário irá digitar.

Como estamos utilizando variáveis não iremos alterar mais nada no nosso programa, essa é a magia da programação.

Essa é uma base simples que iremos utilizar para qualquer programa. Observe que tudo que iremos fazer daqui para frente irá seguir o mesmo conceito e tudo que já vimos.

4.7 - Exercícios: Melhorando nossos programas

1) Vamos melhorar nosso sistema que calcula o gasto e quilômetragem da gasolina. Abra o `programa09.html` para que possamos fazer as alterações.

- a) No lugar de darmos os valores as variáveis vamos pedir que o usuário nos informe os dados. Basta utilizar a função `prompt()` e fazer com que cada variável receba o valor digitado:

```
var valorGasolina = prompt("Qual o valor da Gasolina?");
var capacidadeTanque = prompt("Qual a capacidade do seu tanque?");
var percorreu = prompt("Quanto você já percorreu?");
```

Pronto, essa é a única alteração que precisamos fazer e seu sistema estará funcionando, basta agora atualizar no browser e testar.

2) Agora vamos atualizar o `programa08.html`. Neste programa estamos esperando dados de **5 amigos** então vamos atualizar nosso sistema para que ele pergunte as informações **de cada amigo**.

- a) Primeiro iremos criar duas funções: uma que irá retornar o **calculo total de dias** e a outra que irá retornar o **calculo médio de dias**:

O calculo total de dias:

```
var calculoTotalDeDias = function(){
    return diasGuilherme + diasNahan
    + diasWilliam
    + diasRafael
    + diasLina;
}
```

O calculo médio de dias:

```
var calculoMediaDeDias = function(){
    return calculoTotalDeDias() / 5;
}
```

- b) Agora apague o código que foi substituído pelas funções anteriores:

```
var totalDias =
diasGuilherme + diasNahan
+ diasWilliam
+ diasRafael
+ diasLina;
```

```
var mediaDias = totalDias / 5;
```

- c) Não esqueça de alterar a função `exibirTotais()` para que no lugar de utilizar as variáveis `totalDias` e `mediaDias`, utilizem as duas novas funções que criamos.

```
var exibirTotais = function(){
    linha();
    document.write("Total de dias: " + totalDias);
    novaLinha();
```

```

        document.write("Média de dias: " + mediaDias);
        linha();
    }

```

Alterar para:

```

var exibirTotais = function(){
    linha();
    document.write("Total de dias: " + calculoTotalDeDias());
    novaLinha();
    document.write("Média de dias: " + calculoMediaDeDias());
    linha();
}

```

- 3) Agora vamos alterar as funcionalidades que irão pegar os valores de cada amigo. Iremos como exemplo fazer uma, você deve repetir o processo com **atenção** para cada amigo que você tem no código.

- a) Vamos criar inicialmente uma função para representar os dados do primeiro amigo, no nosso exemplo vamos alterar o do *Guilherme Frota*. Crie a função para ele:

```

var entradaGuilherme = function(){
}

```

- b) Agora vamos adicionar na função os comandos para solicitar ao usuário que digite as informações que precisamos:

```

var entradaGuilherme = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
}

```

Com isso não iremos mais precisar das variáveis, `idadeGuilherme`, `diasGuilherme` e `ano`, pode apagá-las:

```

var ano = 2015; <-- pode apagar voce nao precisa mais.
var idadeGuilherme = 8; <-- pode apagar voce nao precisa mais.
var diasGuilherme = idadeGuilherme * diasDoAno; <-- pode apagar voce nao precisa mais.

```

- c) Agora vamos adicionar na função o `calculo` para os dias de vida do amigo:

```

diasGuilherme = idade*diasDoAno;

```

Adicione na função:

```

var entradaGuilherme = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasGuilherme = idade*diasDoAno;
}

```

- d) Agora vamos exibir as informações, basta você copiar a função `exibirAmigo()` para dentro da função `entradaGuilherme`:

Função:

```

var entradaGuilherme = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasGuilherme = idade*diasDoAno;

    exibirAmigo(nome, idade, ano, diasGuilherme);
}

```

- e) No lugar da chamada da função `exibirAmigo()`, agora chame a função `entradaGuilherme`.
Altere:

```
exibirAmigo("Nahan Frota", idadeNahan, ano, diasNahan);
Para:
entradaGuilherme();
f) O código do programa completo deverá ser semelhante:
var diasDoAno = 365;

var calculoTotalDeDias = function(){
    return diasGuilherme + diasNahan
    + diasWilliam
    + diasRafael
    + diasLina;
}

var calculoMediaDeDias = function(){
    return calculoTotalDeDias() / 5;
}

var novaLinha = function(){
    document.write("<br>");
}

var linha = function(){
    document.write("<hr>");
}

var exibirTotais = function(){
    linha();
    document.write("Total de dias: " + calculoTotalDeDias());
    novaLinha();
    document.write("Média de dias: " + calculoMediaDeDias());
    linha();
}

var exibirAmigo = function(nome, idade, ano, diasDeVida){
    document.write("Nome: " + nome +
        " - idade: " + idade + ", nasceu em: " + (ano - idade) +
        " e já viveu " + diasDeVida + " dias.");
    novaLinha();
}

var entradaGuilherme = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasGuilherme = idade*diasDoAno;

    exibirAmigo(nome, idade, ano, diasGuilherme);
}

var entradaNahan = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasNahan = idade*diasDoAno;

    exibirAmigo(nome, idade, ano, diasNahan);
}
```

```
var entradaWilliam = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasWilliam = idade*diasDoAno;

    exibirAmigo(nome, idade, ano, diasWilliam);
}

var entradaRafael = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasRafael = idade*diasDoAno;

    exibirAmigo(nome, idade, ano, diasRafael);
}

var entradaLina = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasLina = idade*diasDoAno;

    exibirAmigo(nome, idade, ano, diasLina);
}

entradGuilherme();
entradNahan();
entradWilliam();
entradRafael();
entradLina();
exibirTotais();
```

Tomada de decisão

5.1 - Condição: se/if

Muitas coisas na nossa vida dependem de escolhas e uma simples pergunta pode gerar várias ações ou consequências.

Assim também funciona na programação. Por exemplo, se fizessemos a seguinte pergunta: *-Você gosta de correr?*

Temos de imediato duas possíveis respostas: SIM ou NÃO. Porém **dependendo** de qual for a resposta escolhida eu vou ter que tomar uma ação ou ter alguma consequências dela. Vejamos.

Caso eu responda SIM, então pegue o sapado e vá correr.

Caso eu escolha NÃO, deite na cama e durma.

Veja que dependendo da minha resposta posso tomar uma ação que está ligada aquela resposta. Passando isso para o português teremos algo semelhante a isso:

```
se "eu responder SIM"  
    "Pegar o sapado e vá correr"  
  
se "eu responder NÃO"  
    "Vá deitar na cama e durma"
```

Chamamos isso de **controle de fluxo** ou **estruturas de decisão** que é um recurso onde o computador irá executar um trecho de código baseado em alguma decisão que o usuário irá tomar ou o próprio computador.

Porém não vamos trabalhar com português e sim com o inglês, logo no lugar do **se** iremos utilizar a palavra **if** que significa a mesma coisa. Depois iremos delimitar o trecho de código que queremos executar com as chaves({}):

```
if("eu responder SIM"){  
    console.log("Pegar o sapado e vá correr");  
}  
  
if("eu responder NÃO"){  
    console.log("Vá deitar na cama e durma");  
}
```

Porém ainda não é tão simples não. A condição **if** só aceita um tipo de resposta, SIM ou NÃO que estão em português, então vamos usar a língua correta, o inglês: **true** e **false**. Chamamos esse tipo de valor de valor booleano ou **boolean**.

```
if(true){  
    console.log("Pegar o sapado e vá correr");
```

```
}
```

```
if(false){  
    console.log("Vá deitar na cama e durma");  
}
```

Porém o **if** só deixa executar o código que está dentro do bloco dele **caso ele seja true**. Ou seja, ele **só irá entrar no if** se a condição for VERDADEIRA e nada mais.

Então como ficaria nosso segundo **if**?

Primeiro vamos colocar essa resposta dentro de uma variável:

```
var resposta = true;
```

Para a primeira condição não teremos problemas, basta substituir pelo valor:

```
var resposta = true;  
  
if(resposta){  
    console.log("Pegar o sapado e vá correr");  
}
```

O nosso problema é na segunda condição, onde caso a resposta seja **false**. Na primeira condição ele simplesmente não vai fazer nada, mas na segunda era para ele exibir a mensagem "*Vá deitar na cama e durma*". Porém o **if** só permite executar o comando dele se ela for **true**.

```
var resposta = false;  
  
if(resposta){  
    console.log("Pegar o sapado e vá correr");  
}  
  
if(resposta){  
    console.log("Vá deitar na cama e durma");  
}
```

Para resolver isso basta **testar** essa condição, verificando se o valor que está na variável é **false**, **se** o valor for **false** então essa **expressão** irá retornar **true**.

Essa expressão irá fazer uma pergunta: *A resposta foi igual a false?*. Esse **igual** é representado pelo símbolo do igual(=) porém como já usamos esse símbolo para atribuição então teremos que usá-lo duas vezes: ==.

```
var resposta = false;  
var respostaExpressao = (resposta == false);
```

Perceba que a expressão é `resposta == false`, para o programa você está **perguntando** se o valor da variável `resposta` é **igual** ao valor `false`. Se a resposta for SIM (`true`) então a expressão irá **retornar** `true`. Segue a mesma ideia da expressão `valor1 + valor2` que retorna o resultado dessa operação.

PS: Neste caso se não quiser usar os parênteses não tem problema.

```
var resposta = false;
var respostaExpressao = resposta == false;

if(resposta){
    console.log("Pegar o sapado e vá correr");
}

if(respostaExpressao){
    console.log("Vá deitar na cama e durma");
}
```

Agora quando a resposta foi SIM(true) ele irá imprimir: “Pegar o sapado e vá correr”. E caso a resposta seja NÃO(false) ele irá executar a pergunta para saber se essa resposta foi mesmo FALSE e irá salvar seu resultado na variável respostaExpressao e caso seja realmente FALSE então o resultado será TRUE e o trecho do código do segundo if irá ser executado.

Podemos diminuir esse código, adicionando a expressão diretamente dentro do if:

```
var resposta = false;

if(resposta){
    console.log("Pegar o sapado e vá correr");
}

if(resposta == false){
    console.log("Vá deitar na cama e durma");
}
```

Se para o dizer se algo é igual a outro usamos o simbolo `==` então o que usamos para dizer se algo é diferente ?

Basta usar o simbolo `!=`, veja um exemplo:

```
var texto1 = "Handerson";
var texto2 = "OutroNome";

if(texto1 != texto2){
    console.log("Sim, os textos são diferentes");
}
```

A pergunta que fizemos na condição foi: “Os textos são diferentes?”.

Quando testar se algo é menor ou maior?

As vezes não queremos saber se a decisão tomada é igual ou diferente de outra, queremos apenas saber se ela é maior ou menor. Por exemplo se quisermos saber quem é o mais velho devemos testar se a idade de um é maior que a idade do outro:

```
var idade1 = prompt("Idade da primeira pessoa?");
var idade2 = prompt("Idade da segunda pessoa?");

if(parseInt(idade1) < parseInt(idade2)){
    console.log("A primeira pessoa é mais nova;");
}
```

```
if(parseInt(idade1) > parseInt(idade2)){
    console.log("A primeira pessoa é mais velha;");
}
```

5.2 - Exercícios: Condição se/if

1) Crie o programa13.html.

a) Cria a função chamada qualMaior:

```
var qualMaior = function(){
}
```

b) Agora declare duas variáveis: x e y e faça com que elas recebam seus respectivos valores utilizando a função prompt:

```
var qualMaior = function(){
    var x = prompt("Digite o primeiro valor");
    var y = prompt("Digite o segundo valor");
}
```

c) Agora faça as condições, que devem ser: Se o primeiro valor é maior que o segundo, e se o segundo deve ser maior que o primeiro:

```
var qualMaior = function(){
    var x = prompt("Digite o primeiro valor");
    var y = prompt("Digite o segundo valor");

    if(x > y){
        document.write("O primeiro valor é MAIOR: " + x);
    }

    if(x < y){
        document.write("O segundo valor é MAIOR: " + y);
    }
}

qualMaior();
```

Execute seu programa. Para testar utilize a seguinte ordem e verifique o resultado:

- X = 9, Y = 7. Resultado: O primeiro valor é MAIOR: 9
- X = 3, Y = 8. Resultado: O segundo valor é MAIOR: 8
- X = 9, Y = 12. Resultado: O primeiro valor é MAIOR: 9
- X = 8, Y = 400. Resultado: O primeiro valor é MAIOR: 8

Você pode observar que mesmo eu passando números altos o JavaScript só irá entender como número o primeiro dígito. Isso porque sabemos que ele não é bem um número e sim apenas um valor qualquer. Vamos resolver esse problema utilizando a função parseInt.

d) Vamos dizer ao JavaScript que os valores que estaremos recebendo nas variáveis são realmente **números inteiros**, utilizando o parseInt:

```
var qualMaior = function(){
    var x = parseInt(prompt("Digite o primeiro valor"));
    var y = parseInt(prompt("Digite o segundo valor"));

    if(x > y){
        document.write("O primeiro valor é MAIOR: " + x);
    }
```

```

if(x < y){
    document.write("O segundo valor é MAIOR: " + y);
}
}

qualMaior();

```

e) Executando o programa confira os resultados:

- X = 9, Y = 7. Resultado: *O primeiro valor é MAIOR: 9*
- X = 3, Y = 8. Resultado: *O segundo valor é MAIOR: 8*
- X = 9, Y = 12. Resultado: *O segundo valor é MAIOR: 12*
- X = 8, Y = 400. Resultado: *O segundo valor é MAIOR: 400*

2) Agora crie um novo programa: programa14.html. Esse programa irá solicitar dois textos (nome ou qualquer coisa) e iremos comparar seus valores.

a) Crie a função que irá receber os valores:

```

var nomesIguais = function(){
    var texto1 = prompt("Digite o primeiro nome");
    var texto2 = prompt("Digite o segundo nome");
}

```

b) Agora faça as condições:

```

var nomesIguais = function(){
    var texto1 = prompt("Digite o primeiro texto");
    var texto2 = prompt("Digite o segundo texto");

    if(texto1 == texto2){
        document.write("Os textos são iguais: " + texto1);
    }

    if(texto1 != texto2){
        document.write("Os textos são diferentes: " + texto1 + " - " + texto2);
    }
}

nomesIguais();

```

3) Agora iremos adicionar uma nova cláusula ao nosso programa, que agora irá verificar se o texto1 é MAIOR que o texto2:

```

var nomesIguais = function(){
    var texto1 = prompt("Digite o primeiro texto");
    var texto2 = prompt("Digite o segundo texto");

    if(texto1 == texto2){
        document.write("Os textos são iguais: " + texto1);
    }

    if(texto1 != texto2){
        document.write("Os textos são diferentes: " + texto1 + " - " + texto2);
    }

    if(texto1.length > texto2.length){
        document.write("O texto: " + texto1 + " é maior que o segundo: " + texto2);
    }
}

```

```
}

nomesIguais();
```

5.3 - Outras formas de condições

Pegando nosso exemplo anterior fizemos uma pergunta simples: *-Você gosta de correr?*. Isso no código ficou semelhante:

```
if(resposta){
    console.log("Pegar o sapado e vá correr");
}

if(resposta == false){
    console.log("Vá deitar na cama e durma");
}
```

Porém se você observar para essa pergunta só temos duas respostas: SIM ou NÃO. E dependendo da resposta vou tomar uma ação, caso a resposta não seja positiva eu tomo outro. Só que acabamos que fazendo a mesma pergunta **duas** vezes, pois temos **dois ifs**.

Bem, se caso a resposta for positiva devemos pegar o sapato, então qualquer coisa diferente disso podemos deitar na cama? Se o pensamento for esse então podemos evitar mais código neste caso, acrescentando o else e retirando um dos if:

```
if(resposta){
    console.log("Pegar o sapado e vá correr");
} else {
    console.log("Vá deitar na cama e durma");
}
```

O else **faz parte** do if e nos iremos ler esse código da seguinte forma:

Se a resposta for verdadeira “Pegue o sapato e vá correr”, se não “vá deitar na cama e durma”

Dessa forma não se faz necessário perguntar novamente, ou seja, utilizar o segundo if.

Como saber o tamanho de um texto?

Agora podemos fazer outro teste, que tal saber se seu nome é maior ou menor que o nome do seu amigo?

Para isso podemos usar a propriedade chamada `length` que quer dizer tamanho/comprimento.

Por exemplo, temos uma variável com o seu nome:

```
var nome = "Handerson Frota";
```

E queremos saber o seu tamanho, então usamos o `length` que é uma propriedade dessa variável. Algumas variáveis possuem algumas propriedades, ações que o JavaScript nos deu.

```
console.log("Tamanho do nome: " + nome.length);
```

Esse código irá retornar: Tamanho do nome: 15.

Como posso saber quem é maior e/ou igual?

O que temos até agora são os seguintes testes: `==(igual)`, `!=(diferente)`, `<(menor que)`, `>(maior que)`.

Mas e se quisermos saber se algo é por exemplo igual e/ou menor? A pergunta seria: O texto é menor e/ou igual ao texto 2? Para fazer essa pergunta corretamente podemos usar dois símbolos combinados: `<= (menor igual)` e `>= (maior igual)`:

Menor igual:

```
if(texto1.length <= texto2.length){  
    //sua ação aqui  
}
```

Maior igual:

```
if(texto1.length >= texto2.length){  
    //sua ação aqui  
}
```

Duas ou mais condições na mesma pergunta?

Parece estranho, mas imagine o seguinte problema: *Preciso verificar se o texto que estou digitando possui o tamanho que eu desejo, mas quero saber se ele é a frase que quero.* Deu para entender?

São duas perguntas:

- O tamanho é igual a x?
- O texto é igual a esse?

Vamos ver isso no código:

```
var texto1 = "Handerson Frota";  
var texto2 = "Handerson B. Frota";  
  
if(texto1.length == texto2.length){  
    console.log("Os textos possuem o mesmo tamanho.");  
}  
  
if(texto1 == texto2){  
    console.log("Os textos são iguais");  
}
```

Veja que preciso fazer dois testes e neste caso o `else` não vai resolver, já que são dois testes **distintos** e a ação de um **não** depende da ação do outro.

Temos o que chamamos de **Operadores Lógicos** que são:

- E / `&&`
- OU / `||`

Agora é muito importante, como estamos combinando duas expressões (testes) então primeiro devemos definir o que nos queremos e isso irá influenciar na escolha de qual operador eu vou precisar utilizar.

Vejamos as perguntas:

* *O texto precisa ser do mesmo tamanho do outro E devem ser iguais;*

Neste caso queremos que as **duas** condições sejam **VERDADEIRAS** ou seja, **true**.

```
var texto1 = "Handerson Frota";
var texto2 = "Handerson B. Frota";

if(texto1.length == texto2.length && texto1 == texto2){
    console.log("Os textos são exatamente iguais.");
}
```

Observe que a ação que ele irá tomar, também será a mesma.

Podemos melhorar um pouco esse código deixando ele um pouco mais legível, utilizando os parenteses para delimitar e separar as expressões:

```
var texto1 = "Handerson Frota";
var texto2 = "Handerson B. Frota";

if( (texto1.length == texto2.length) && (texto1 == texto2) ){
    console.log("Os textos são exatamente iguais.");
}
```

Para esta ação ser executada **AMBAS** as condições **DEVEM** ser verdadeiras. Caso apenas uma delas seja falsa, a ação do **if** não será executada.

Agora vamos mudar um pouco a pergunta:

* *O texto precisa ser do mesmo tamanho do outro OU podem ser iguais;*

Neste caso queremos que apenas **UMA** das seja **VERDADEIRA** não importa qual. Quando utilizo o **OU** se apenas uma delas for verdadeira então ele entra no **if**.

```
var texto1 = "Handerson Frota";
var texto2 = "Handerson B. Frota";

if( (texto1.length == texto2.length) || (texto1 == texto2) ){
    console.log("Os textos são exatamente iguais.");
}
```

Caso a primeira **ou** a segunda condição sejam **VERDADEIRAS** a ação do **if** será executada.

Essas combinações podem ser visualizadas e são conhecidas como a **Tabela Verdade**, veja abaixo essas combinações:

A	B	A E B	A OU B
VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO
FALSO	VERDADEIRO	FALSO	VERDADEIRO
VERDADEIRO	FALSO	FALSO	VERDADEIRO
FALSO	FALSO	FALSO	FALSO
		A && B	A B

Veja um exemplo utilizando expressões:

A	B	A E B	A OU B
$2 > 1$	$4 > 1$	VERDADEIRO	VERDADEIRO
$3 == 2$	$3 == 3$	FALSO	VERDADEIRO
$4 < 6$	$4 < 2$	FALSO	VERDADEIRO
$4 > 10$	$10 == 20$	FALSO	FALSO
		A && B	A B

5.4 - Exercícios: Outras formas de se fazer uma condição

1) Vamos atualizar nosso programa14.html, melhorando assim seu código.

a) Adicione a cláusula else ao seu código:

```
var nomesIguais = function(){
    var texto1 = prompt("Digite o primeiro texto");
    var texto2 = prompt("Digite o segundo texto");

    if(texto1 == texto2){
        document.write("Os textos são iguais: " + texto1);
    }

    if(texto1 != texto2){
        document.write("Os textos são diferentes: " + texto1 + " - " + texto2);
    }

    if(texto1.length > texto2.length){
        document.write("O texto: " + texto1 + " é maior que o segundo: " + texto2);
    }
}

nomesIguais();
```

Ficando dessa forma:

```
var nomesIguais = function(){
    var texto1 = prompt("Digite o primeiro texto");
```

```

var texto2 = prompt("Digite o segundo texto");

if(texto1 == texto2){
    document.write("Os textos são iguais: " + texto1);
}else {
    document.write("Os textos são diferentes: " + texto1 + " - " + texto2);
}

if(texto1.length > texto2.length){
    document.write("O texto: "+ texto1 + " é maior que o segundo: "+ texto2);
}
}

nomesIguais();

```

- 2) Agora vamos melhorar nossa condição para saber se o texto1 é maior que o texto2, deixando nosso código mais legível:

```

var nomesIguais = function(){
    var texto1 = prompt("Digite o primeiro texto");
    var texto2 = prompt("Digite o segundo texto");

    if(texto1 == texto2){
        document.write("Os textos são iguais: " + texto1);
    } else if(texto1.length > texto2.length){
        document.write("O texto: "+ texto1 + " é maior que o segundo: "+ texto2);
    } else {
        document.write("Os textos são diferentes: " + texto1 + " - " + texto2);
    }
}

nomesIguais();

```

- 3) Agora queremos que a condição para saber se um texto é maior que o outro faça parte da primeira verificação, ou seja, se o texto for igual ao outro ele também deve ter o mesmo tamanho.

```

if(texto1 == texto2 && texto1.length > texto2.length){
    document.write("Os textos são iguais: " + texto1
    + "<br> e possuem o mesmo tamanho.");
}

```

Confira o resultado.

- 4) Continuando a nossa melhora do código deixando ele mais legível iremos criar uma função que verifique se um determinado texto é maior que outro, para que possámos utilizá-lo dentro da nossa condição IF e reproveitá-lo quando necessário.

- a) Crie a função possuemOMesmoTamanho que receba os textos que você deseja comparar:

```

var possuemOMesmoTamanho = function(textoUm, textoDOIS){
    if(textoUm.length > textoDOIS.length){
        return true;
    }else {
        return false;
    }
}

```

Observe que agora ele recebe parâmetros.

- b) Agora faça a chamada do método na condição:

```
var possuem = possuemOMesmoTamanho(texto1, texto2);
if(texto1 == texto2 || possuem){
    document.write("Os textos são iguais: " + texto1 + "<br> e possuem o mesmo tamanho.");
} else {
    document.write("Os textos são diferentes: " + texto1 + " - " + texto2);
}
```

c) Ou podemos fazer melhor para evitar códigos desnecessários:

```
if(texto1 == texto2 || possuemOMesmoTamanho(texto1, texto2)){
    document.write("Os textos são iguais: " + texto1 + "<br> e possuem o mesmo tamanho.");
} else {
    document.write("Os textos são diferentes: " + texto1 + " - " + texto2);
}
```

5) Podemos melhorar ainda mais a função possuemOMesmoTamanho retornando diretamente seu valor assim como já fizemos com expressões matemáticas:

```
var possuemOMesmoTamanho = function(textoUm, textoDOIS){
    return textoUm.length > textoDOIS.length;
}
```

Estruturas de repetição

6.1 - Entendendo a estrutura com while

Já estamos executando algumas linhas de códigos de acordo com uma condição. Utilizando o `if` e o `else` conseguimos definir qual trecho deve ser executado. Mas e se quisermos exercutar esse trecho mais de uma vez?

Imagine o seguinte problema: Temos que imprimir os números pares e para isso teremos uma variável `numero` que irá inicializar com o valor zero(0) e iremos somando logo depois de exibir o valor **2**. Assim temos somente números pares.

```
var numero = 0;
console.log("Número par: " + numero);

numero = numero + 2;
console.log("Número par: " + numero);

numero = numero + 2;
console.log("Número par: " + numero);

numero = numero + 2;
console.log("Número par: " + numero);

numero = numero + 2;
console.log("Número par: " + numero);
```

Observe que tivemos que repetir várias vezes e neste caso iremos exibir somente os valores: 0,2,4,6,8.

E se quisermos exibir os números pares até o valor de 100? Poxa vai dar trabalho não?

A ideia é mais ou menos assim: **Enquanto** a variável não chegar no valor de 100 iremos continuar somando 2.

Para isso vamos utilizar o `while` (enquanto) para fazer essa operação:

```
var numero = 0;

while(true){
    console.log("Número par: " + numero);
    numero = numero + 2;
}
```

Vejamos o que esse código diz. Ele fala que enquanto a condição for **verdadeira** (aqui se respeita a mesma regra do `IF`) repita esse trecho de código. Porém **você deve tomar muito cuidado** com essa condição, pois ela está **sempre** dizendo que é verdadeira e **nunca** irá parar de executar o seu código e isso irá com certeza dar um crash(quebrar, parar) o seu programa e no nosso caso o navegador.

Precisamos adicionar uma condição que terá um fim e já temos ela. Observe que eu falei: Enquanto a variável não **chegar no valor de 100** continue somando.

Então vamos alterar:

```
var numero = 0;

while(numero < 100){
    console.log("Número par: " + numero);
    numero = numero + 2;
}
```

O resultado irá imprimir todos os valores **pares** de 0 a **98**. Exato a 98, isso porque a condição foi clara: enquanto o número for **menor** que 100. Para imprimir o valor de 100 devemos dizer: enquanto o número for **menor igual** a 100.

```
var numero = 0;

while(numero <= 100){
    console.log("Número par: " + numero);
    numero = numero + 2;
}
```

Vamos a outro exemplo mais simples. Quero **percorrer** todos os valores de 100 e exibi-los um a um.

```
var contador = 0;

while(contador <= 100){
    console.log("Número: " + contador);
    contador = contador + 1;
}
```

A variável `contador` está iniciando em zero(0) e para cada vez que o código imprime o número ele soma mais um a variável (`contador = contador + 1;`);

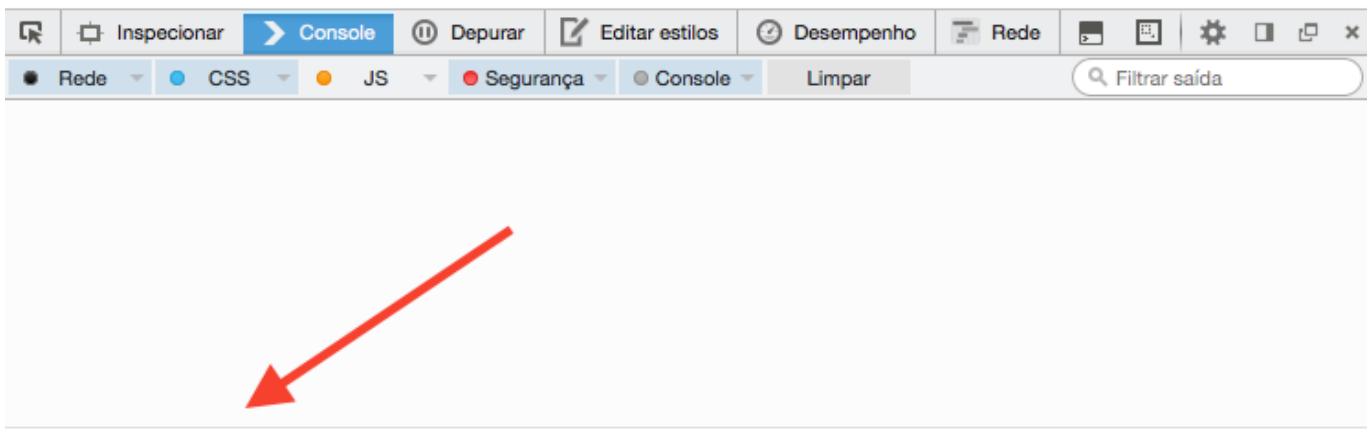
6.2 - Exercícios: Fazendo loops com WHILE

1) Crie o programa15.html.

a) Cria agora a função chamada `primeiroWhile`:

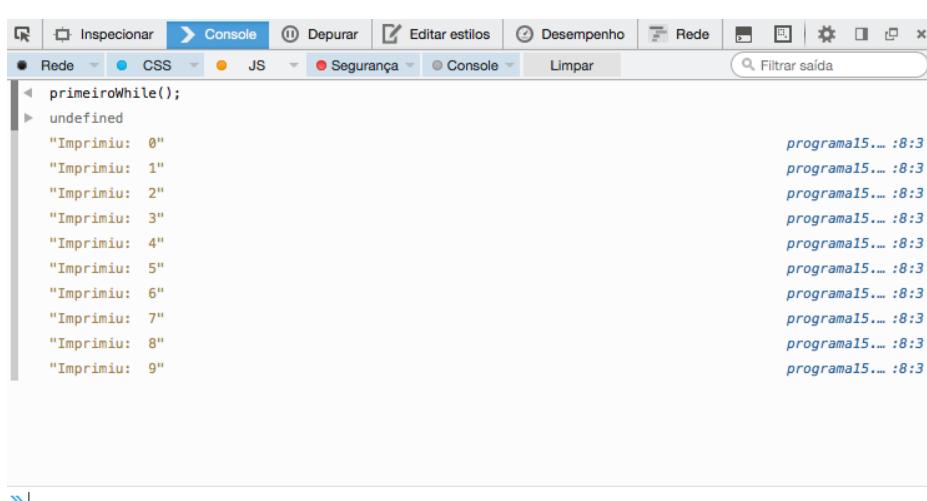
```
var primeiroWhile = function(){
    var x = 0;
    while(x < 10){
        console.log("Imprimiu: " + x);
        x = x + 1;
    }
}
```

b) Agora execute a função (`primeiroWhile()`) utilizando o console do browser como mostrado na imagem abaixo e tecle “ENTER” para executar o programa:



```
» primeiroWhile();
```

c) O resultado será algo semelhante a imagem abaixo:



```

»| 
< primeiroWhile();
> undefined
"Imprimiu: 0"                                     programa15... :8:3
"Imprimiu: 1"                                     programa15... :8:3
"Imprimiu: 2"                                     programa15... :8:3
"Imprimiu: 3"                                     programa15... :8:3
"Imprimiu: 4"                                     programa15... :8:3
"Imprimiu: 5"                                     programa15... :8:3
"Imprimiu: 6"                                     programa15... :8:3
"Imprimiu: 7"                                     programa15... :8:3
"Imprimiu: 8"                                     programa15... :8:3
"Imprimiu: 9"                                     programa15... :8:3

```

6.3 - Outro tipo de loop com a estrutura do FOR

Quando utilizamos o `while` precisamos de alguma forma de dizer que em algum momento ele deve parar. Perceba que se eu não adicionar uma condição e uma parada para essa condição esse loop ficará executando para sempre ou até o programa “quebrar”. Mas a estrutura do `while` é muito simples e ela não nos ajuda a *lembra*r que devemos colocar essas condições, lembra do `while(true)`? Precisamos de um outro tipo de loop que nos ajude a deixar isso bem definido desde o começo.

Essa estrutura é o `for`. Antes de entendermos como ele funciona, vamos recapitular o `while`.

```
inicialização da variável
while(condicao){
    incremento
}
```

Pegando um exemplo temos:

```
var contador = 0;
while(contador <= 100){
    console.log("Número: " + contador);
    contador++;
}
```

Vamos ver a estrutura básica do `for`:

```
for( ; ; ){
    console.log("Número: ");
}
```

Observe que se eu executar esse trecho ele irá rodar infinitamente, pois em momento nenhum definimos quando ele deverá parar ou quando ele deverá iniciar. Observe que temos alguns espaços entre os `;`. Está faltando algo e é nestes espaços onde iremos resolver nosso problema.

```
for( espaco1;espaco2 ;espaco3 ){
    console.log("Número: ");
}
```

Cada espaço desse espera uma informação:

- a) `espaco1`: é a inicialização da variável
- b) `espaco2`: é a condição que quero testar
- c) `espaco3`: é o incremento

Vejamos:

```
for([inicializacao]; [condicao]; [incremento]){
    console.log("Número: ");
}
```

Lembra alguém? Exato o `while` a diferença é que com o `for` isso fica mais legível e não “solto” como no `while`:

`while`:

```
var contador = 0;
while(contador <= 100){
    console.log("Número: " + contador);
    contador++;
}
```

Fazendo o mesmo código utilizando `for`:

```
for(var contador = 0; contador <= 100; contador++){
    console.log("Número: " + contador);
}
```

Bem melhor não acha? Da uma ideia de mais “organizado” e irá se comportar da mesma maneira do `while`.

Incremento com o Pós-incremento ++

Estamos sempre utilizando aqui a seguinte expressão: `x = x + 1`, mas podemos substituir esse código utilizando o que chamamos de **pós incremento**, que é identificado pelo operador “`++`”, basta adicioná-lo depois da variável onde você deseja somar mais **1**.

Esse código:

```
x = x + 1;
```

É a mesma coisa desse:

```
x++;
```

6.4 - Exercícios: Loop com FOR

1) Crie o programa16.html.

- a) Cria agora a função chamada `primeiroFOR`, muito semelhante com a que você fez com o `while` porém agora utilizando a estrutura `for`:

```
var primeiroFOR = function(){
    for(var x = 0; x < 10; x = x + 1){
        console.log("Imprimiu: " + x);
    }
}

primeiroFOR();
```

2) Assim como no exercício anterior, vamos alterar a **incrementação** da variável `x`:

- a) Troque o código abaixo:

```
x = x + 1;
```

- b) Por este bem mais sucinto:

```
for(var x = 0; x < 10; x++){}  
}
```

6.5 - Juntando os conceitos...

Você lembra que fizemos um programa que calcula a idades dos nossos amigos? Mas qual o problema desse programa?

O programa calcula uma certa quantidade fixa de amigos, um número limitado e sempre os mesmos amigos com os mesmos valores.

Para isso precisamos ir aplicando tudo que aprendemos até agora. Já que não precisamos especificar nossos amigos, ou seja, queremos que no momento de execução do programa o usuário possa escolher a quantidade e os valores(idade, nome etc) já sabemos que não iremos precisar por exemplo das variáveis dias de cada amigo.

Porém o principal problema são as funções que solicitam os dados para **cada** amigo, ou seja, temos uma função dessa abaixo para **cada** amigo:

```
var entradaGuilherme = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
```

```

var ano = prompt("Em que ano estamos?");
diasGuilherme = idade*diasDoAno;

exibirAmigo(nome, idade, ano, diasGuilherme);
}

```

Não iremos precisar também das chamadas dessas funções, já que não iremos mais precisar delas.

```

entradaGuilherme();

entradaNahan();

entradaWilliam();

entradaRafael();

entradaLina();

```

Mas como iremos iniciar isso? Como iremos deixar o usuário escolher quantos amigos e entrar com esses dados?

Bem a primeira resposta você já sabe, iremos continuar utilizando o `prompt` para pegar essas informações. Mas como iremos baseado nessa informação passada pelo usuário solicitar os dados de cada um deles?

Você esqueceu do loop? Iremos executar uma função que será mais genérica, ou seja, não é específica a nenhum valor para cada interação do loop:

```

var executarPrograma = function(){
    var qtdAmigos = prompt("Quantos amigos você quer calcular?");
    for(var x = 0 ;x < qtdAmigos;x++){
        var nome = prompt("Qual o nome do seu amigo?");
        var idade = prompt("Qual a idade dele?");
        var ano = prompt("Em que ano estamos?");

        diasAmigo = idade*diasDoAno;
        exibirAmigo(nome, idade, ano, diasAmigo);

    }
}

```

Observe que a função `executarPrograma` inicialmente solicita que o usuário entre com a **quantidade** de amigos que ele deseja calcular e baseado nessa informação iremos com o `for` executar quantas vezes necessário o código que irá calcular e exibir os dados de cada amigo que ele escolheu.

O grande ponto dessa primeira parte é juntas vários conceitos que aprendemos separadamente e fazer com que eles nos ajudem a deixar nosso código mais sucinto, legível e simples.

6.6 - Exercícios: Juntando os conceitos...

1) Abra o `programa08.html` e copie todo o seu código.

a) Após copiar todo o código do `programa08.html`, crie o arquivo `programaCalculoIdades.html` e cole o código dentro. Lembre-se que nosso `programa08.html` possui um código semelhante ao abaixo:

```
var diasDoAno = 365;

var calculoTotalDeDias = function(){
    return diasGuilherme + diasNahan
    + diasWilliam
    + diasRafael
    + diasLina;
}

var calculoMediaDeDias = function(){
    return calculoTotalDeDias() / 5;
}

var novaLinha = function(){
    document.write("<br>");
}

var linha = function(){
    document.write("<hr>");
}

var exibirTotais = function(){
    linha();
    document.write("Total de dias: " + calculoTotalDeDias());
    novaLinha();
    document.write("Média de dias: " + calculoMediaDeDias());
    linha();
}

var exibirAmigo = function(nome, idade, ano, diasDeVida){
    document.write("Nome: " + nome +
        " - idade: " + idade + ", nasceu em: " + (ano - idade) +
        " e já viveu " + diasDeVida + " dias.");
    novaLinha();
}

var entradaGuilherme = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasGuilherme = idade*diasDoAno;

    exibirAmigo(nome, idade, ano, diasGuilherme);
}

var entradaNahan = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasNahan = idade*diasDoAno;

    exibirAmigo(nome, idade, ano, diasNahan);
}

var entradaWilliam = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasWilliam = idade*diasDoAno;
```

```

        exibirAmigo(nome, idade, ano, diasWilliam);
    }

var entradaRafael = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasRafael = idade*diasDoAno;

    exibirAmigo(nome, idade, ano, diasRafael);
}

var entradaLina = function(){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
    diasLina = idade*diasDoAno;

    exibirAmigo(nome, idade, ano, diasLina);
}

entradGuilherme();

entradNahan();

entradWilliam();

entradRafael();

entradLina();

exibirTotais();

```

- b) Feito isso salve e rode o programaCalcidoIdades.html para se certificar que está tudo ok.
- 2) Vamos agora deixar nosso programa mais dinâmico e que possa efetuar o calculo de **quantos** amigos necessitarmos e para **qualquer** um.

- a) Na função calcidoTotalDeDias altere o código para que ele não utilize as variáveis específicas de cada amigo, afinal agora queremos que ele trabalhe com valores dinâmicos:

Altere esse código:

```

var calcidoTotalDeDias = function(){
    return diasGuilherme + diasNahan
    + diasWilliam
    + diasRafael
    + diasLina;
}

```

Deixando ele assim:

```

var diasAmigosSomatorio = 0;
var diasAmigo = 0;

var calcidoTotalDeDias = function(){
    diasAmigosSomatorio = diasAmigosSomatorio + diasAmigo;
    return diasAmigosSomatorio;
}

```

Observe que você retirou todas as variáveis específicas de cada amigo e criou a variável diasAmigo e está sempre somando o valor anterior com o novo e guardando esse valor na nova variável diasAmigosSomatorio e retornando esse total.

- b) Agora **apague** todas as funções de entrada dos dados de cada amigo: entradaGuilherme, entradaWilliam etc. Perceba que não queremos deixar o programa *preso* a somente esses amigos.

- c) Agora **apague** todas as chamadas das funções:

```
entradaGuilherme();
```

```
entradaNahan();
```

```
entradaWilliam();
```

```
entradaRafael();
```

```
entradaLina();
```

```
exibirTotais();
```

- d) Você irá ficar com um código semelhante a este:

```
var diasDoAno = 365;
var diasAmigosSomatorio = 0;
var diasAmigo = 0;

var calculoTotalDeDias = function(){
    diasAmigosSomatorio = diasAmigosSomatorio + diasAmigo;
    return diasAmigosSomatorio;
}

var calculoMediaDeDias = function(){
    return calculoTotalDeDias() / 5;
}

var novaLinha = function(){
    document.write("<br>");
}

var linha = function(){
    document.write("<hr>");
}

var exibirTotais = function(){
    linha();
    document.write("Total de dias: " + calculoTotalDeDias());
    novaLinha();
    document.write("Média de dias: " + calculoMediaDeDias());
    linha();
}

var exibirAmigo = function(nome, idade, ano, diasDeVida){
    document.write("Nome: " + nome +
        " - idade: " + idade + ", nasceu em: " + (ano - idade) +
        " e já viveu " + diasDeVida + " dias.");
    novaLinha();
}
```

- 3) Agora precisamos fazer o função que irá fazer com que nosso programa seja inteligente o suficiente para exibir esses valores a partir de uma informação que o usuário desejar e não somente de amigos **fixos** no código como estava anteriormente.

a) Crie a função executarPrograma:

```
var executarPrograma = function(){
}
```

b) Agora vamos pedir ao usuário que escolha **quantos** amigos ele deseja calcular a idade:

```
var executarProgramma = function(){
    var qtdAmigos = prompt("Quantos amigos você quer calcular?");
}
```

c) Agora utilizando o **for** iremos **baseado** na informação que o usuário irá passar, fazer com que o programa pergunte quantas vezes necessários as informações que desejamos:

```
var executarProgramma = function(){
    var qtdAmigos = prompt("Quantos amigos você quer calcular?");
    for(var x = 0 ;x < qtdAmigos;x++){
}
```

d) Adicione no **for** as perguntas para que o usuário entre com as informações no programa:

```
var executarProgramma = function(){
    var qtdAmigos = prompt("Quantos amigos você quer calcular?");
    for(var x = 0 ;x < qtdAmigos;x++){
        var nome = prompt("Qual o nome do seu amigo?");
        var idade = prompt("Qual a idade dele?");
        var ano = prompt("Em que ano estamos?");
    }
}
```

e) Pronto, já estamos recebendo as informações, agora iremos fazer o mesmo calculo para descobrir os dias de um amigo:

Código anterior para você lembrar:

```
diasGuilherme = idade*diasDoAno;
```

Então teremos o mesmo algorítimo porém agora utilizando uma única variável: **diasAmigo** ao invés da variável mais específica do amigo.

```
var executarProgramma = function(){
    var qtdAmigos = prompt("Quantos amigos você quer calcular?");
    for(var x = 0 ;x < qtdAmigos;x++){
        var nome = prompt("Qual o nome do seu amigo?");
        var idade = prompt("Qual a idade dele?");
        var ano = prompt("Em que ano estamos?");

        diasAmigo = idade*diasDoAno;
    }
}
```

f) Com esse código pronto, já podemos exibir as informações, lembre-se que já temos essa função: **exibirAmigo**:

```
var executarProgramma = function(){
    var qtdAmigos = prompt("Quantos amigos você quer calcular?");
    for(var x = 0 ;x < qtdAmigos;x++){
        var nome = prompt("Qual o nome do seu amigo?");
        var idade = prompt("Qual a idade dele?");
        var ano = prompt("Em que ano estamos?");

        diasAmigo = idade*diasDoAno;
        exibirAmigo(nome, idade, ano, diasAmigo);
    }
}
```

```

        }
    }
}
```

- g) Agora basta adicionar no final do código, a chamada de execução do programa, que no nosso caso é o programaCalculoIdades:

```
executarPrograma();
```

Lembre-se, esse código é adicionado no **final** do código e **não dentro** da função `executarPrograma`. Execute o programa.

- 4) Vamos melhorar um pouco mais. Observe que não estamos exibindo os totais e quem tem essa responsabilidade é a função `exibirTotais`. Vamos executar essa função dentro da função `executarPrograma`:

```

var executarPrograma = function(){
    var qtdAmigos = prompt("Quantos amigos você quer calcular?");
    for(var x = 0 ;x < qtdAmigos;x++){
        var nome = prompt("Qual o nome do seu amigo?");
        var idade = prompt("Qual a idade dele?");
        var ano = prompt("Em que ano estamos?");

        diasAmigo = idade*diasDoAno;
        exibirAmigo(nome, idade, ano, diasAmigo);

        exibirTotais();
    }
}

```

Salve e execute novamente o programa.

- a) Veja que estamos imprimindo várias vezes o total. Isso porque você deve ter percebido que o `exibirTotais` está **dentro do bloco** do `for` e queremos exibir o **total** de tudo, ou seja, devo **esperar** o `for` ser finalizado.

```

var executarPrograma = function(){
    var qtdAmigos = prompt("Quantos amigos você quer calcular?");
    for(var x = 0 ;x < qtdAmigos;x++){
        var nome = prompt("Qual o nome do seu amigo?");
        var idade = prompt("Qual a idade dele?");
        var ano = prompt("Em que ano estamos?");

        diasAmigo = idade*diasDoAno;
        exibirAmigo(nome, idade, ano, diasAmigo);
    }
    exibirTotais();
}

```

Agora sim, iremos imprimir o total apenas no **final** da execução do `for`.

- b) O código final irá ser semelhante a este:

```

var diasDoAno = 365;
var diasAmigosSomatorio = 0;
var diasAmigo = 0;

var calculoTotalDeDias = function(){
    diasAmigosSomatorio = diasAmigosSomatorio + diasAmigo;
    return diasAmigosSomatorio;
}

var calculoMediaDeDias = function(){

```

```

        return calculoTotalDeDias() / 5;
    }

    var novaLinha = function(){
        document.write("<br>");
    }

    var linha = function(){
        document.write("<hr>");
    }

    var exibirTotais = function(){
        linha();
        document.write("Total de dias: " + calculoTotalDeDias());
        novaLinha();
        document.write("Média de dias: " + calculoMediaDeDias());
        linha();
    }

    var exibirAmigo = function(nome, idade, ano, diasDeVida){
        document.write("Nome: " + nome +
            " - idade: " + idade + ", nasceu em: " + (ano - idade) +
            " e já viveu " + diasDeVida + " dias.");
        novaLinha();
    }

    var executarPrograma = function(){
        var qtdAmigos = prompt("Quantos amigos você quer calcular?");
        for(var x = 0 ;x < qtdAmigos;x++){
            var nome = prompt("Qual o nome do seu amigo?");
            var idade = prompt("Qual a idade dele?");
            var ano = prompt("Em que ano estamos?");

            diasAmigo = idade*diasDoAno;
            exibirAmigo(nome, idade, ano, diasAmigo);

        }
        exibirTotais();
    }
}

executarProgramma();

```

6.7 - Garanta que seu programa não vá da erros

Nosso programa que calcula a idade de nossos amigos já evoluiu bastante, porém ainda temos alguns problemas.

Observando a função executarProgramma podemos perceber algumas falhas na lógica. Veja que o código solicita ao usuário que entre com a quantidade de amigos.

```

var executarProgramma = function(){
var qtdAmigos = prompt("Quantos amigos você quer calcular?");
for(var x = 0 ;x < qtdAmigos;x++){
    var nome = prompt("Qual o nome do seu amigo?");
    var idade = prompt("Qual a idade dele?");
    var ano = prompt("Em que ano estamos?");
}

```

```
diasAmigo = idade*diasDoAno;  
exibirAmigo(nome, idade, ano, diasAmigo);  
}  
  
exibirTotais();  
}
```

Mas e se o usuário digitar um caractére? A lestra “x” por exemplo ou a palavra “dez” ao invés do numeral? Ou se ele digitar “0” ou até se ele não digitar **nada**?

Nosso programa não está pronto para tratar esses tipos de problemas e são problemas que temos como **evitar**.

Sempre quando precisamos do usuário para passar alguma informação devemos ter muito cuidado, pois não sabemos o que o usuário poderá digitar. Por este mesmo motivo precisamos garantir do lado do programa que iremos tratar o máximo possível erros comuns.

Como fazemos? Você já sabe como. Basta utilizar o `if` e a tabela verdade para resolver esse problema, vamos aos exercícios.

Mas não fique preocupado muito com isso, pois será impossível você identificar todos os erros possíveis ou não de serem dados em sua aplicação. O importante aqui é que você tenha uma noção que **deve** sempre validar as informações que você irá trabalhar, **principalmente** se essas informações vierem do usuário. Dessa forma você irá sempre evitar os tão temidos bugs/erros.

6.8 - Exercícios: Evitando erros

1) Que tal validar as informações passadas pelo usuário, evitando assim que o programa dê erro?

a) Primeiro vamos verificar se o usuário **não** colocou a quantidade de amigos ou digitou **0**:

```
if (qtdAmigos == "" || qtdAmigos == 0){  
    qtdAmigos = prompt("Não entendi. Quantos amigos mesmo?");  
}
```

Rode o programa e teste.

b) Agora iremos verificar se o valor digitado foi um **número** ou não, utilizando para isso a função do JavaScript chamada `isNaN` que retorna **false** caso ele não seja um número e **true** caso seja. Vamos passar para ela a variável que representa a quantidade, no nosso exemplo a variável: `qtdAmigos`:

```
if (qtdAmigos == "" || qtdAmigos == 0  
    || isNaN(qtdAmigos)){  
    qtdAmigos = prompt("Não entendi. Quantos amigos mesmo?");  
}
```

c) Execute o teste.

6.9 - Tem como melhorar?

Ainda estamos com alguns problemas, no nosso caso estamos validando apenas uma única vez o valor que o usuário nos informa. Utilizando mais uma vez o `for` em conjunto com o `if` para validarmos mais de uma vez e quantas vezes for necessário essas informações poderemos deixar nosso programa ainda melhor e mais “inteligente”.

6.10 - Exercícios: Melhorias

1) Vamos melhorar nossa validação, afinal da maneira como está ela só faz a validação apenas uma vez, na primeira entrada de dados. O que queremos é que ele **sempre** valide os dados de entrada, até que o usuário coloque os dados corretos.

a) Como já sabemos queremos garantir que **sempre** ele efetue a validação, nada melhor que usar um `while` para isso, já que queremos é repetir o trecho do `if` que faz a validação:

```
while(){
    if (qtdAmigos == "" || qtdAmigos == 0){
        qtdAmigos = prompt("Não entendi. Quantos amigos mesmo?");
    }
}
```

Não se preocupe ainda com a condição do `while`.

b) Agora iremos precisar que o `while` seja executado pela primeira vez e para isso precisamos que a condição seja verdadeira (`true`):

```
var testaNumero = true;
while(testaNumero){
    if (qtdAmigos == "" || qtdAmigos == 0
        || isNaN(qtdAmigos)){
        qtdAmigos = prompt("Não entendi. Quantos amigos mesmo?");
    }
}
```

CUIDADO! Não execute ainda esse programa, afinal o `while` será **sempre** verdadeiro e ele não irá parar.

c) Agora vem a seguinte lógica. “Se ele entrar no `if` é porque ele **não** entrou com a informação correta!”. Quando ele entra no `if` o usuário vai digitar novamente o valor, então **preciso** que ele execute a validação **novamente** ou seja, que ele execute o `while` outra vez. E para isso basta adicionar na variável `testaNumero` como `true`, afinal você **não tem** como saber agora se o usuário digitou agora o valor correto.

```
var testaNumero = true;
while(testaNumero){
    if (qtdAmigos == "" || qtdAmigos == 0
        || isNaN(qtdAmigos)){
        qtdAmigos = prompt("Não entendi. Quantos amigos mesmo?");
        testaNumero = true;
    }
}
```

CUIDADO! Não execute ainda esse programa, afinal o `while` será **sempre** verdadeiro e ele não irá parar.

d) Agora precisamos ter uma condição de parada, vamos lembrar da condição inicial: “Se ele entrar no `if` é porque ele **não** entrou com a informação correta!”. Então se ele **não** cair na validação do `if` é porque o usuário digitou o **valor correto**. Agora fica mais fácil, basta adicionar o `else`:

```
var testaNumero = true;
while(testaNumero){
    if (qtdAmigos == "" || qtdAmigos == 0
        || isNaN(qtdAmigos)){
        qtdAmigos = prompt("Não entendi. Quantos amigos mesmo?");
        testaNumero = true;
    }else{
        testaNumero = false;
    }
}
```

Agora sim pode testar seu programa ;).

e) Veja como nosso programa `programaCalculoIdades` irá ficar no final:

```

var diasDoAno = 365;
var diasAmigosSomatorio = 0;
var diasAmigo = 0;

var calculoTotalDeDias = function(){
    diasAmigosSomatorio = diasAmigosSomatorio + diasAmigo;
    return diasAmigosSomatorio;
}

var calculoMediaDeDias = function(){
    return calculoTotalDeDias() / 5;
}

var novaLinha = function(){
    document.write("<br>");
}

var linha = function(){
    document.write("<hr>");
}

var exibirTotais = function(){
    linha();
    document.write("Total de dias: " + calculoTotalDeDias());
    novaLinha();
    document.write("Média de dias: " + calculoMediaDeDias());
    linha();
}

var exibirAmigo = function(nome, idade, ano, diasDeVida){
    document.write("Nome: " + nome +
        " - idade: " + idade + ", nasceu em: " + (ano - idade) +
        " e já viveu " + diasDeVida + " dias.");
    novaLinha();
}

var executarPrograma = function(){
    var qtdAmigos = prompt("Quantos amigos você quer calcular?");
    var testaNumero = true;
    while(testaNumero){
        if (qtdAmigos == "" || qtdAmigos == 0
            || isNaN(qtdAmigos)){
            qtdAmigos = prompt("Não entendi. Quantos amigos mesmo?");
            testaNumero = true;
        }else{
            testaNumero = false;
        }
    }

    for(var x = 0 ;x < qtdAmigos;x++){
        var nome = prompt("Qual o nome do seu amigo?");
        var idade = prompt("Qual a idade dele?");
        var ano = prompt("Em que ano estamos?");

        diasAmigo = idade*diasDoAno;
        exibirAmigo(nome, idade, ano, diasAmigo);
    }
}

```

```
}

exibirTotais();

}

executarPrograma();
```

Estrutura de dados com Array

7.1 - Melhorando a interatividade com o HTML

Até o momento estamos utilizando o HTML apenas para executar nosso código JavaScript, não existe nenhuma interação entre o HTML e o JavaScript. Um exemplo simples é quando esperamos que o usuário entre com alguma informação. Para isso estamos utilizando o próprio JavaScript para receber essa informação com a função `prompt`.

Vamos mudar um pouco isso, vamos fazer com que o JavaScript acesse esses dados que iremos passar via HTML, ou seja, vamos integrar essas duas tecnologias.

O `prompt` é uma função que solicita alguma informação ao usuário como já sabemos, a maneira como ele funciona é enviando uma tela com um espaço para que o usuário digite essa informação e um botão para que quando o usuário termine de digitar ele pressione e a função devolva esse valor para a variável.

```
var valor = prompt("Entre com a informação aqui");
```

No HTML temos algo semelhante, com a mesma ideia do `prompt`.

Primeiro vamos criar o campo onde queremos que o usuário digite a informação. Esse campo no HTML chamamos de `input` e como é HTML então ele é uma tag.

```
<input>
```

O `input` possui algumas propriedades, assim como algumas variáveis do JavaScript. Lembra da propriedade `length`?

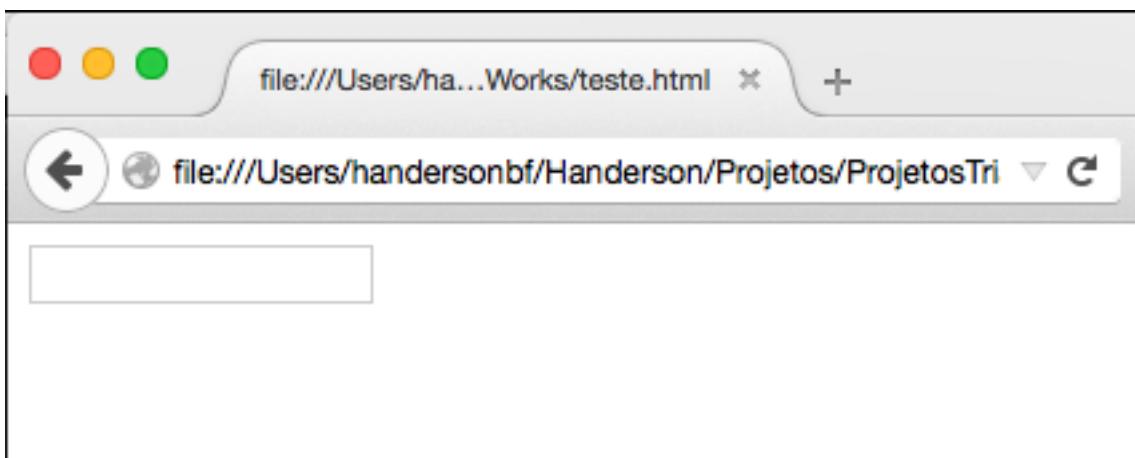
Basicamente iremos utilizar 3 propriedades:

- a) `id`: identificador do campo
- b) `type`: onde definimos qual o TIPO desse campo. Ex: Texto, Senha etc.
- c) `value`: o valor. Quando queremos carregar o `input` já exibindo algum valor de imediato usamos essa propriedade

Pronto, sabendo disso precisamos criar um `input` que irá solicitar alguma informação. No início do HTML adicione o `input`:

```
<input type="text" id="campoTexto" />
```

Quando você carregar a página HTML ele irá exibir:

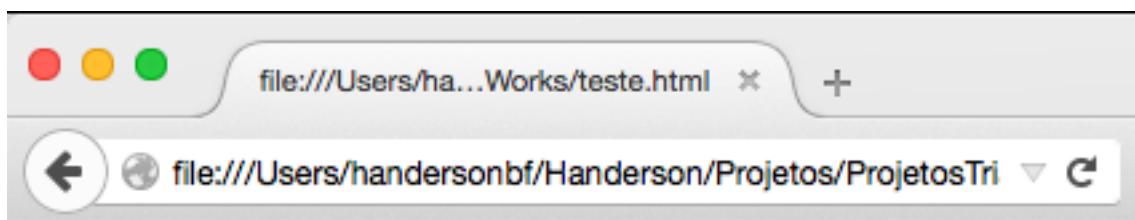


Lembra que temos que pedir ao usuário alguma informação? Precisamos dizer a ele o que queremos que ele digite neste campo.

Entre com a informação:

```
<input type="text" id="campoTexto" />
```

Veja como irá exibir:



Pronto já temos o campo que o usuário irá entrar com a informação, agora precisamos criar um código que pegue o valor que digitamos.

Para pegarmos esse valor precisamos entender que esse `input` faz parte do **documento** da página e temos no JavaScript um cara que representa esse documento, que é o `document`.

Lembra que algumas variáveis do JavaScript possuem propriedades? Eles também possuem funções e é com ela que iremos pegar o valor que está no HTML.

A função será a `getElementById("id")` que se formos traduzir ela está dizendo: “Pegue o Elemento com o ID”.

```
var input = document.getElementById("campoTexto");
console.log(input);
```

Essa função irá vasculhar o documento, que é toda a minha página, procurando por um elemento(tag) com o ID chamado “*campoTexto*”. Exibindo esse valor temos algo:

```
<input id="campoTexto" type="text">
```

Sim, ele irá retornar a própria tag e não o valor dela, veja que estamos pedindo para pegar o ELEMENTO e não o valor. Porém como sabemos algumas variáveis do JavaScript possuem propriedades e ela tem uma propriedade que irá retornar o valor deste elemento. Usaremos o `value`:

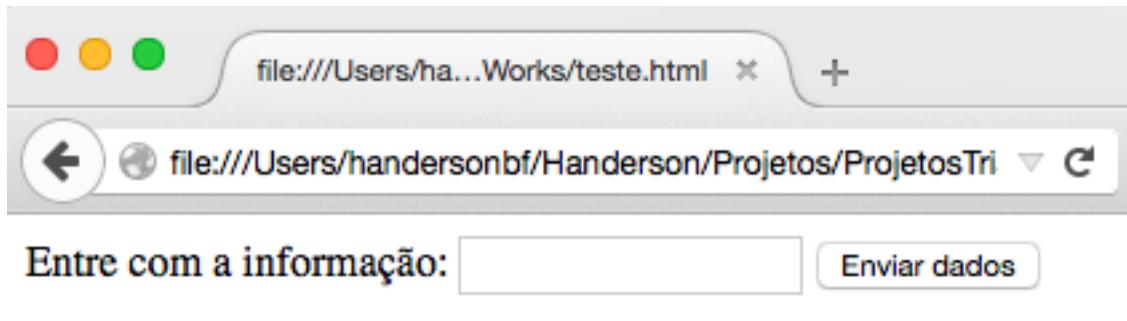
```
var input = document.getElementById("campoTexto");
console.log(input.value);
```

Neste exemplo iremos retornar o **valor** do elemento.

Pegar somente o valor não é o suficiente, precisamos acionar a ação. Para isso iremos utilizar também um `input`, porém iremos passar um `type` diferente. O tipo desse `input` será um `submit`. Quando usamos esse tipo o HTML entende que isso é um botão que irá executar alguma ação.

```
<input type="submit" id="botao" />
```

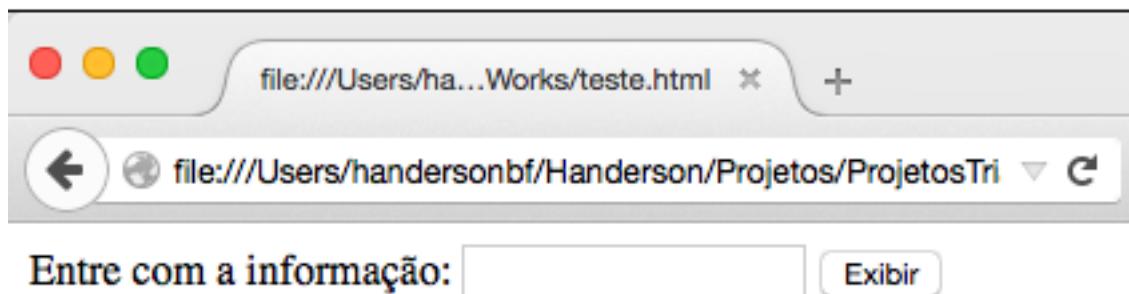
E teremos o seguinte:



E se quisermos alterar o valor do botão, ou seja, o texto dele? Hoje como não falamos nada para o navegador ele irá colocar um valor padrão dele e que pode variar de navegador para navegador. Então vamos definir o nosso valor e claro usaremos a propriedade `value`

```
<input type="submit" id="botao" value="Exibir" />
```

Veja o resultado:



Mas mesmo criando isso tudo, se clicarmos no botão nada irá acontecer. Isso porque não falamos qual ação ele deve tomar. Mais uma vez utilizando o `document.getElementById()` iremos pegar o elemento “botão”.

```
var botao = document.getElementById("botao");
```

Mas como sabemos, a variável `botao` é uma representação do elemento `input` do tipo `submit`. E sabemos também que ela possui uma propriedade. O que devemos fazer é adicionar uma ação neste elemento quando que será executada quando o usuário **clicar** no botão. Essa ação **clicar** é chamada de **onclick** que é a nossa propriedade.

```
var botao = document.getElementById("botao");
botao.onclick = ???
```

Mas qual ação quero executar ao clicar? Vamos criar uma função que irá pegar o valor digitado e exibi-lo no console.

```
var exibirValorDigitado = function(){
  var input = document.getElementById("campoTexto");
  console.log(input.value);
}
```

Agora precisamos passar essa função para o `onclick`:

```
var exibirValorDigitado = function(){
  var input = document.getElementById("campoTexto");
  console.log(input.value);
}

var botao = document.getElementById("botao");
botao.onclick = exibirValorDigitado;
```

Observe que estamos passando a variável que possui uma função, veja que ela **não possui os parênteses**, está apenas como uma **variável**. Se você fizer como o código abaixo, você estará **executando** a função **antes** de clicarmos.

```
botao.onclick = exibirValorDigitado(); <-- executando a função.
```

Pronto, ao clicarmos no botão ele irá pegar o valor que o usuário digitou no campo `input` e exibir no console.

7.2 - Exercícios: Melhorando a interatividade com o HTML

1) Iremos criar o `programa17.html` e nele vamos exercitar como pegar um valor do HTML.

- a) No início do arquivo, logo depois da tag `<meta charset="UTF-8">`, adicione o campo `input` onde iremos digitar o valor:

```
<input type="text" id="campoTexto" />
```

- b) Agora adicione o botão que irá pegar esse valor:

```
<input type="submit" id="botao" value="Exibir o valor digitado"/>
```

- c) Agora iremos criar o nosso JavaScript:

```
<script>
  ...
</script>
```

- d) Precisamos **pegar** o **elemento** `campoTexto`:

```
<script>
  var campoTexto = document.getElementById("campoTexto");
</script>
```

- e) Com o elemento “salvo” na variável `campoTexto`, vamos criar uma função que irá pegar o **valor** deste elemento, no caso o valor digitado:

```
<script>
  var campoTexto = document.getElementById("campoTexto");

  var exibirCampo = function(){
    alert(campoTexto.value);
  }
</script>
```

- f) Agora iremos pegar o elemento que representa o botão da página:

```
<script>
  var campoTexto = document.getElementById("campoTexto");

  var exibirCampo = function(){
    alert(campoTexto.value);
  }
```

```
  var botaoSelecionado = document.getElementById("botao");
</script>
```

- g) Agora adicione um **evento** ou seja, uma ação ao botão, utilizando o `onclick`:

```
<script>
  var campoTexto = document.getElementById("campoTexto");

  var exibirCampo = function(){
    alert(campoTexto.value);
```

```

        }

var botaoSelecionado = document.getElementById("botao");

botaoSelecionado.onclick = exibirCampo;
</script>

```

h) O programa17.html irá ficar semelhante a isso:

```

<meta charset="UTF-8">

<input type="text" id="campoTexto" />
<input type="submit" id="botao" value="Exibir o valor digitado"/>

<script>
var campoTexto = document.getElementById("campoTexto");

var exibirCampo = function(){
    alert(campoTexto.value);
}

var botaoSelecionado = document.getElementById("botao");
botaoSelecionado.onclick = exibirCampo;

</script>

```

7.3 - Estrutura de dados Array

Muitas vezes precisamos trabalhar com vários dados e para isso podemos utilizar várias variáveis para armazenar cada uma dessas informações, como por exemplo quando queremos pegar a idade de todos os alunos da sala de aula e exibi-las:

```

var idadeAluno01 = 20;
var idadeAluno02 = 28;
var idadeAluno03 = 27;
var idadeAluno04 = 30;

```

Para exibir teremos que fazer algo semelhante a isso:

```

console.log("Idade do aluno: " + idadeAluno01);
console.log("Idade do aluno: " + idadeAluno02);
console.log("Idade do aluno: " + idadeAluno03);
console.log("Idade do aluno: " + idadeAluno04);

```

Se você sentiu algo estranho nesse código você está indo no caminho certo. Temos vários problemas aqui. Estamos repetindo código nos `console.log` sem falar que se eu precisar adicionar mais alunos? Ou retirar alunos? Bem complicado correto?

E se tivermos uma estrutura, como um armário, onde podemos *agrupar* esses valores de uma forma mais organizada?

É para isso que existe o array, que é um estrutura de dados onde podemos armazenar de uma forma organizada o que quisermos como um armário.

E para criar esse array é bem simples, o array no JavaScript é identificado pelo colchete `[]`. Ou seja, precisamos apenas criar uma variável e o valor que será atribuído nela deverá estar entre colchetes:

```
var arrayIdades = [20, 28, 27, 30];
```

Observe que cada valor antes da vírgula representa um valor na posição do array:

```
0   1   2   3 <-- posições/ índices  
var arrayIdades = [20, 28, 27, 30];
```

Essa posição é o que chamamos de *índice* e a contagem do índice é a partir do 0 (zero) no JavaScript assim como em muitas linguagens.

Para acessármos qualquer valor do array precisamos utilizar o índice do array e basta adicionar novamente os colchetes para representar o acesso ao array e dentro deles adicionar a posição que você deseja acessar:

```
console.log(array[2]);
```

Resultado será:

27

Já conseguimos ter com o array uma estrutura que representa várias idades, agora precisamos exibir essas idades. No código anterior temos:

```
console.log("Idade do aluno: " + idadeAluno01);  
console.log("Idade do aluno: " + idadeAluno02);  
console.log("Idade do aluno: " + idadeAluno03);  
console.log("Idade do aluno: " + idadeAluno04);
```

Atualizando para array ficaria:

```
console.log("Idade do aluno: " + array[0]);  
console.log("Idade do aluno: " + array[1]);  
console.log("Idade do aluno: " + array[2]);  
console.log("Idade do aluno: " + array[3]);
```

Mas ainda não ficou legal correto? Estamos repetindo código do mesmo jeito, então vamos utilizar uma estrutura de repetição para resolver isso e vamos **percorrer** o array:

```
var arrayIdades = [20, 28, 27, 30];  
  
for(var i=0; i<4; i++){  
    console.log("Idade do aluno" + i + : " " + arrayIdades[i]);  
}
```

Temos 4 posições e iremos contar a partir do **zero** até ele ser menor que **4** no caso a posição **3**.

O resultado será:

```
Idade do aluno0: 20  
Idade do aluno1: 28  
Idade do aluno2: 27  
Idade do aluno3: 30
```

Mas é muito chato ficar alterando o tamanho do array toda vida que ele aumentar e/ou diminuir, na verdade estamos dizendo no `for` que o tamanho do array é fixo e isso não é legal. Então basta substituir o tamanho fixo pelo atributo `.length` que **pertence** ao array no nosso caso será `arrayIdades.length`.

```
var arrayIdades = [20, 28, 27, 30];

for(var i=0; i< arrayIdades.length; i++){
    console.log("Idade do aluno" + i + : " " + arrayIdades[i]);
}
```

Podemos adicionar o que quisermos no nosso array, números, textos, variáveis etc. Vejamos um exemplo de um array de textos com alguns nomes:

```
var arrayNomes = ["Guilherme", "Nahan", "William" , "Handerson"];

for(var i=0; i<4; i++){
    console.log("Nome: " + arrayNomes[i]);
}
```

Vejamos o resultado:

```
Nome: Guilherme
Nome: Nahan
Nome: William
Nome: Handerson
```

7.4 - Exercícios: Arrays

1) Iremos criar o programa18.html e nele vamos exercitar alguns tipos de arrays.

a) Iremos criar uma array de números:

```
var array = [10,40,20,30];
```

b) Agora imprima o valor do array com o `console.log` e mostre a posição 2:

```
console.log(array);
console.log("Valor da posição 2: " + array[1]);
```

c) Adicione na posição 2 um novo valor: **100**:

```
array[1] = 100;
```

d) Imprima novamente:

```
console.log(array);
console.log("Valor da posição 2: " + array[1]);
```

2) Vamos agora utilizar o `for` para percorrer esse mesmo array e exibir também o tamanho dele:

a) Imprima o tamanho do array utilizando o `length`:

```
document.write("Tamanho do Array: " + array.length + "<br>");
```

b) Utilizando agora o `for` vamos percorrer e exibir todos os valores do array:

```
for (var i = 0; i < array.length; i++) {
    document.write("Posição " + i + " - valor: " + array[i]+ "<br>");
}
```

3) Que tal trabalharmos com outros tipos de valores? Vamos agora ao invés de adicionarmos número, vamos adicionar os nomes de nossos amigos. Continue utilizando o mesmo arquivo: programa18.html:

a) Crie uma array com 4 amigos logo abaixo do for do exercício anterior:

```
var arrayTexto = ["Handerson", "Guilherme", "Nahan", "William"];
```

b) Imprima o array:

```
console.log(arrayTexto);
```

c) Agora vamos adicionar mais um amigo na posição que ainda não foi ocupada, a posição 4:

```
arrayTexto[4] = "Thayna";
console.log(arrayTexto);
```

d) Agora vamos imprimir o total de array e percorrer seus valores:

```
document.write("Tamanho do Array: " + arrayTexto.length + "<br>");
for (var x = 0; x < arrayTexto.length; x++) {
    document.write("Nome: " + arrayTexto[x] + "<br>");
}
```

7.5 - Jogo da Adivinhação

Vamos colocar nosso aprendizado em prática. Que tal fazermos um jogo bem simples onde faremos o computador adivinhar um determinado número.

Primeiro iremos pegar os valores que o usuário deverá digitar, utilizando o HTML para isso.

```
<meta charset="UTF-8">
Digite seu chute: <input type="text" id="numero" />
<input type="submit" id="btAdivinar" value="Tente a sorte!" />
<script>

</script>
```

Dessa forma iremos usar o JavaScript para pegar o valor digitado no input.

```
<script>
    var mostra = function(valor){
        document.write(valor + "<br>");
    }

    var sorteio = Math.round(Math.random() * 100);
</script>
```

Veja que estamos usando uma nova função a `Math.round` que é utilizando para *arredondar* um valor e a função `Math.random` que retorna um número aleatório.

A `Math.random` irá selecionar um número e multiplicar por 100, o seu resultado será arredondado pela função `Math.round` deixa um número inteiro. E o resultado de tudo isso será salvo na variável `sorteio`.

Agora precisamos pegar o valor que o usuário irá tentar adivinhar.

```
<script>
    var mostra = function(valor){
        document.write(valor + "<br>");
    }
```

```
var sorteio = Math.round(Math.random() * 100);
var inputNumero = document.getElementById("numero");
</script>
```

Com o valor gerado pelo computador mais o valor que o usuário digitou precisamos apenas fazer a função que irá fazer o sorteio. Na verdade o valor que o usuário digitar será o valor que ele está tentando adivinhar que o computador criou. Para não ficar muito complicado iremos dar **4** chances ao usuário:

```
var tentaAdivinhar = function(){
    var chute = inputNumero.value;

    for(var tentativa = 1; tentativa <= 4; tentativa++){
        }

}
```

Devemos adicionar condições de teste para caso o usuário erre o programa permita que ele tente novamente.

```
for(var tentativa = 1; tentativa <= 4; tentativa++){
    if(chute == sorteio){
        mostra("Que sorte eim ; eu pensei no " + chute
        + " e você também.");
    }else{
        mostra("A que pena, o número que pensei foi outro");
        chute = prompt("Vamos tentar novamente."
        + " Tentativa: " + tentativa);
    }
}
```

Com o programa quase finalizado falta apenas tomar a **ação**, ou seja, vamos adicionar uma ação no botão para que ele execute a função `tentaAdivinhar`.

Para isso lembre-se de **pegar** o `input` do botão:

```
var btAdvinhar = document.getElementById("btAdvinhar");
```

E adicionar a ação na variável que ele representa e para isso precisamos acessar a propriedade do `input` chamada `onclick`, pois queremos que **ao clicar** no botão ele execute a função desejada:

```
var btAdvinhar = document.getElementById("btAdvinhar");
btAdvinhar.onclick = tentaAdivinhar;
```

Observe que estamos utilizando condicionais, loops, variáveis e claro funções.

Podemos ainda fazer com que o programa de Adivinhação dê pistas por exemplo se o número que o usuário digitou é maior ou menor que o número que o computador gerou:

```
var numeroMaior = function(){
    return "Que pena o número é maior que o que você escolheu!";
}

var numeroMenor = function(){
    return "Que pena o número é menor que o que você escolheu!";
}
```

E mais uma vez vamos adicionar um novo código no código antigo:

```

if(chute == sorteio){
    mostra("Que sorte eim ;) eu pensei no "
        + chute + " e você também.");
}else{
    if(chute < sorteio){
        mostra(numeroMaior());
    }
    if(chute > sorteio){
        mostra(numeroMenor());
    }

    chute = prompt("Vamos tentar novamente."
        + " Tentativa: " + tentativa);
}

```

Algo bem simples mas que utiliza todos os conceitos que vimos até agora.

7.6 - Exercícios: Adivinhação

1) Crie o arquivo chamado adivinhar.html:

a) Definindo a estrutura inicial, nada de diferente que já estamos fazendo:

```

<meta charset="UTF-8">
<script>

</script>

```

b) Vamos agora criar os inputs do HTML para digitar o valor e executar a ação

```

<meta charset="UTF-8">
Digite seu chute: <input type="text" id="numero" />
<input type="submit" id="btAdivinhar" value="Tente a sorte!" />
<script>

</script>

```

c) Agora vamos fazer nosso código. Vamos criar uma função que irá mostrar um valor no HTML:

```

<meta charset="UTF-8">
Digite seu chute: <input type="text" id="numero" />
<input type="submit" id="btAdivinhar" value="Tente a sorte!" />
<script>
    var mostra = function(valor){
        document.write(valor + "<br>");
    }
</script>

```

d) Iremos agora criar uma variável que irá conter um valor aleatório de 0 a 100, que o computador vai gerar para nós com a função `Math.random()`:

```

<meta charset="UTF-8">
Digite seu chute: <input type="text" id="numero" />
<input type="submit" id="btAdivinhar" value="Tente a sorte!" />
<script>
    var mostra = function(valor){
        document.write(valor + "<br>");
    }

```

```
var sorteio = Math.round(Math.random() * 100);
</script>
```

Com isso teremos na variável sorteio um valor aleatório entre 0 a 100.

- e) Precisamos pegar o valor que o usuário irá tentar adivinhar, o valor que ele digitou no campo input e guardar em outra variável:

```
<meta charset="UTF-8">
Digite seu chute: <input type="text" id="numero" />
<input type="submit" id="btAdivinhar" value="Tente a sorte!" />
<script>
    var mostra = function(valor){
        document.write(valor + "<br>");
    }

    var sorteio = Math.round(Math.random() * 100);
    var inputNumero = document.getElementById("numero");
</script>
```

- f) Pronto, agora precisamos fazer a função que realmente irá fazer o sorteio e tentar *adivinar* o número que o usuário digitou. Queremos que o usuário faça 4 tentativas para acertar o número.

```
var tentaAdivinhar = function(){
    var chute = inputNumero.value;

    for(var tentativa = 1; tentativa <= 4; tentativa++){

    }
}
```

- g) Agora iremos fazer o teste e caso seja igual, devemos exibir uma mensagem que ele acertou:

```
mostra("Que sorte eim ;) eu pensei no " + chute + " e você também.");
```

- h) E caso seja diferente ele avise que o usuário errou e peça que ele tente novamente:

```
mostra("A que pena, o número que pensei foi outro");
chute = prompt("Vamos tentar novamente." + " Tentativa: " + tentativa);
```

- i) Unindo tudo teremos:

```
for(var tentativa = 1; tentativa <= 4; tentativa++){
    if(chute == sorteio){
        mostra("Que sorte eim ;) eu pensei no " + chute
        + " e você também.");
    }else{
        mostra("A que pena, o número que pensei foi outro");
        chute = prompt("Vamos tentar novamente."
        + " Tentativa: " + tentativa);
    }
}
```

- j) Agora iremos adicionar **uma ação** ao input do botão, logo abaixo da função tentaAdivinhar:

```
var btAdivinhar = document.getElementById("btAdivinhar");
```

- k) Após pegarmos o elemento input vamos adicionar uma ação para ele, no caso a variável que possui a função tentaAdivinhar:

```
var btAdivinhar = document.getElementById("btAdivinhar");
btAdivinhar.onclick = tentaAdivinhar;
```

- l) Veja como nosso programa irá ficar no final:

```
<meta charset="UTF-8">
Digite seu chute: <input type="text" id="numero" />
```

```

<input type="submit" id="btAdvinhar" value="Tente a sorte!" />
<script>
    var mostra = function(valor){
        document.write(valor + "<br>");
    }

    var sorteio = Math.round(Math.random() * 100);
    var inputNumero = document.getElementById("numero");

    var tentaAdivinhar = function(){
        var tentativa = 1;
        var chute = inputNumero.value;

        for(var tentativa = 1; tentativa <= 4; tentativa++){
            if(chute == sorteio){
                mostra("Que sorte eim ; eu pensei no " + chute + " e você também.");
            }else{
                mostra("A que pena, o número que pensei foi outro");
                chute = prompt("Vamos tentar novamente." + " Tentativa: " + tentativa);
            }
        }
    }

    var btAdvinhar = document.getElementById("btAdvinhar");
    btAdvinhar.onclick = tentaAdivinhar;

```

</script>

- 2) Vamos melhorar nosso código criando algumas funções para deixar nosso código mais limpo e queremos também dar pistas para o usuário, dizendo se o número foi um número **maior** ou **menor** do que ele pediu. **Logo depois da função** `tentaAdivinhar` iremos criar nossas novas funções:

- a) Vamos criar a função que irá exibir a mensagem se ele for **maior**:

```

var numeroMaior = function(){
    return "Que pena o número é maior que o que você escolheu!";
}

```

- b) Vamos criar a função que irá exibir a mensagem se ele for **menor**:

```

var numeroMenor = function(){
    return "Que pena o número é menor que o que você escolheu!";
}

```

- c) Agora iremos adicionar as condições caso ele erre o número:

```

if(chute == sorteio){
    mostra("Que sorte eim ; eu pensei no "
        + chute + " e você também.");
}else{
    if(chute < sorteio){
        mostra(numeroMaior());
    }
    if(chute > sorteio){
        mostra(numeroMenor());
    }
}

chute = prompt("Vamos tentar novamente."
    + " Tentativa: " + tentativa);
}

```

d) Veja o código com irá ficar:

```

<meta charset="UTF-8">
Digite seu chute: <input type="text" id="numero" />
<input type="submit" id="btAdvinhar" value="Tente a sorte!" />
<script>
    var mostra = function(valor){
        document.write(valor + "<br>");
    }

    var sorteio = Math.round(Math.random() * 100);
    var inputNumero = document.getElementById("numero");

    var tentaAdivinhar = function(){
        var chute = inputNumero.value;

        for(var tentativa = 1; tentativa <= 4; tentativa++){
            if(chute == sorteio){
                mostra("Que sorte eim ; eu pensei no "
                    + chute + " e você também.");
            }else{
                if(chute < sorteio){
                    mostra(numeroMaior());
                }
                if(chute > sorteio){
                    mostra(numeroMenor());
                }
            }

            chute = prompt("Vamos tentar novamente."
                + " Tentativa: " + tentativa);
        }
    }

    var numeroMaior = function(){
        return "Que pena o número é maior que o que você escolheu!";
    }

    var numeroMenor = function(){
        return "Que pena o número é menor que o que você escolheu!";
    }

    var btAdvinhar = document.getElementById("btAdvinhar");
    btAdvinhar.onclick = tentaAdivinhar;

</script>
```

7.7 - Melhorando nosso joguinho de Adivinhação

Hoje nosso joguinho está gerando randomicamente os valores que queremos que o usuário “adivinhe”:

```
var sorteio = Math.round(Math.random() * 100);
```

E as vezes isso pode ficar bem difícil e pode tornar nosso jogo meio chato :). Podemos limitar os números que queremos que o usuário acerte utilizando **arrays**.

Para isso precisamos declarar uma **array** com alguns números:

```
var sorteio = [2, 3, 5, 8];
```

Ao invés de termos um número gerado pelo computador aleatoriamente, teremos números definidos.

Agora precisamos substituir o `for` que define a quantidade de tentativas para que ele defina essas tentativas pela quantidade de números que temos no nosso array:

Nosso `for` anterior estava dessa forma:

```
for(var tentativa = 1; tentativa <= 4; tentativa++){  
}
```

Não é o que queremos, então iremos alterar o `for` para que ele utilize o array:

```
for(var tentativa=0; tentativa < sorteio.length; tentativa++){  
}
```

Após alterarmos o `for` vamos modificar o algoritmo que testa se o valor que o usuário digitou é realmente o valor do array.

Antes nosso código estava assim:

```
if(chute == sorteio){  
    mostra("Que sorte eim ;) eu pensei no "  
          + chute + " e você também.");  
}  
else{  
    if(chute < sorteio){  
        mostra(numeroMaior());  
    }  
    if(chute > sorteio){  
        mostra(numeroMenor());  
    }  
  
    chute = prompt("Vamos tentar novamente."  
                  + " Tentativa: " + tentativa);  
}
```

Agora utilizando array:

```
if(chute == sorteio[tentativa]){  
    mostra("Que sorte eim ;) eu pensei no "  
          + chute + " e você também.");  
}  
else{  
    if(chute < sorteio[tentativa]){  
        mostra(numeroMaior());  
    }  
    if(chute > sorteio[tentativa]){  
        mostra(numeroMenor());  
    }  
  
    chute = prompt("Vamos tentar novamente."  
                  + " Tentativa: " + tentativa);  
}
```

Vejamos como o programa está ficando com nossas alterações:

```

var sorteio = [2, 3, 5, 8];

for(var tentativa=0; tentativa < sorteio.length; tentativa++){
    if(chute == sorteio[tentativa]){
        mostra("Que sorte eim ;) eu pensei no "
            + chute + " e você também.");
    } else{
        if(chute < sorteio[tentativa]){
            mostra(numeroMaior());
        }
        if(chute > sorteio[tentativa]){
            mostra(numeroMenor());
        }
    }

    chute = prompt("Vamos tentar novamente."
        + " Tentativa: " + tentativa);
}
}

```

Correção do código

Se executarmos nosso programa ele irá ter um “probleminha”. Observe que se o usuário acertar a tentativa ele irá continuar pergutando até finalizar o tamanho do array. Bem, isso não seria justo para nosso usuário já que ao acertar devemos **parar** a execução do programa.

O que precisamos fazer é alterar nossa condição para caso ele consiga acertar o programa pare de executar e para isso devemos utilizar uma palavra chave chamada **break** que é **parar**:

```

if(chute == sorteio[tentativa]){
    mostra("Que sorte eim ;) eu pensei no "
        + chute + " e você também.");
    break;
}

```

Dessa forma, ao acertar o valor do array ele irá mostrar uma mensagem parabenizando e **parar** o programa.

7.8 - Exercícios: Melhorando nosso joguinho de Adivinhação

1) Vamos tentar deixar o “joguinho” mais fácil, vamos criar uma **array** que alguns número que iremos escolher e vamos utilizá-los, assim iremos facilitar mais o jogo:

a) No lugar da variável sorteio ter apenas um único valor, vamos fazer com que a variável sorteio receba uma **array**:

```
var sorteio = [2, 3, 5, 8];
```

b) Agora precisamos utilizar o array dentro do nosso for:

```

for(var tentativa=0; tentativa < sorteio.length; tentativa++){
    if(chute == sorteio[tentativa]){
        mostra("Que sorte eim ;) eu pensei no "
            + chute + " e você também.");
    } else{
        if(chute < sorteio[tentativa]){
            mostra(numeroMaior());
        }
    }
}

```

```

        }
        if(chute > sorteio[tentativa]){
            mostra(numeroMenor());
        }

        chute = prompt("Vamos tentar novamente."
                    + " Tentativa: " + tentativa);
    }
}

```

- c) Observe que mesmo acertando o valor, ele irá pedir para tentar novamente, então isso não está legal, afinal ele já acertou o valor. O que devemos fazer agora é utilizar a palavra chave **break** para **parar** a execução naquele momento:

```

for(var tentativa=0; tentativa < sorteio.length; tentativa++){
    if(chute == sorteio[tentativa]){
        mostra("Que sorte eim ; eu pensei no "
                + chute + " e você também.");
        break; <--palavra chave aqui
    }else{
        if(chute < sorteio[tentativa]){
            mostra(numeroMaior());
        }
        if(chute > sorteio[tentativa]){
            mostra(numeroMenor());
        }
    }

    chute = prompt("Vamos tentar novamente."
                    + " Tentativa: " + tentativa);
}
}

```

- d) Veja como deverá ficar nosso programa agora usando uma array:

```

<meta charset="UTF-8">
Digite seu chute: <input type="text" id="numero" />
<input type="submit" id="btAdivinhar" value="Tente a sorte!" />
<script>
    var mostra = function(valor){
        document.write(valor + "<br>");
    }

    var sorteio = [2, 3, 5, 8];
    var inputNumero = document.getElementById("numero");

    var tentaAdivinhar = function(){
        var chute = inputNumero.value;
        for(var tentativa=0; tentativa < sorteio.length; tentativa++){
            if(chute == sorteio[tentativa]){
                mostra("Que sorte eim ; eu pensei no "
                        + chute + " e você também.");
                break;
            }else{
                if(chute < sorteio[tentativa]){
                    mostra(numeroMaior());
                }
                if(chute > sorteio[tentativa]){
                    mostra(numeroMenor());
                }
            }

            chute = prompt("Vamos tentar novamente.");
        }
    }

```

```
        + " Tentativa: " + tentativa);
    }
}

var numeroMaior = function(){
    return "Que pena o número é maior que o que você escolheu!";
}

var numeroMenor = function(){
    return "Que pena o número é menor que o que você escolheu!";
}

var btAdvinhar = document.getElementById("btAdvinhar");
btAdvinhar.onclick = tentaAdivinhar;

</script>
```

- 2) OPICIONAL: Quer deixar o jogo mais complicado? Que tal adicionar para cada posição do array que o computador **escolha** o número?

```
var sorteio = [Math.round(Math.random() * 100),
    Math.round(Math.random() * 100),
    Math.round(Math.random() * 100),
    Math.round(Math.random() * 100)
];
```

Boas práticas e Orientação a Objetos

8.1 - Organização

Em qualquer linguagem devemos ter sempre em mente a ideia de **responsabilidade**, ou seja, cada tipo de arquivo ou tecnologia deverá ter a sua responsabilidade e portanto deverá estar **separada ao máximo**.

Durante o curso você viu que criamos todo nosso código JavaScript dentro do HTML e isso não é uma boa prática, afinal o arquivo HTML é para HTML.

Para resolver esse problema é bem simples e não muda muita coisa. Basta primeiro criar um **novo arquivo** porém desta vez com a extensão **js**, dessa forma: `arquivo.js`.

Esse tipo de extensão indica que tudo que está dentro dele **deve** ser apenas JavaScript.

Depois de feito essa mudança, como iremos dizer ao código HTML que estamos usando um JavaScript?

Para isso temos a ideia de **importar** o arquivo na página HTML. E para isso usaremos a tag `<script>` de uma maneira mais simples e direta.

Observe a declaração:

```
<script>
    //código javascript aqui
</script>
```

O que devemos fazer é apagar todo o seu conteúdo e **dentro** juntamente com o nome **script** adicionar o que chamamos de **propriedade** da tag chamada `src` ou `source` que quer dizer `fonte`, a fonte onde devemos “pegar” nosso arquivo JS.

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

Pronto! Feito e agora basta abrir o HTML no seu navegador. Lembre-se que ele deve estar **no mesmo local** da sua página HTMI ou você deverá colocar todo o endereço do arquivo.

Outras maneiras de criar funções e Debugs

Para facilitar o entendimento de funções e variáveis utilizamos uma forma bem peculiar de criar nossas funções, peculiar porque poucas são as linguagens que aceitam essa maneira que fizemos no curso.

Utilizamos um raciocínio bem simples: *“Criamos uma variável e ela irá receber como valor uma função inteira.”*

```
var variavel = function(){}
```

Outro exemplo:

```
var novaLinha = function(){
    document.write("<br>");
}
```

Porém essa não é uma maneira mais comum em muitas linguagens, como disse utilizamos dessa forma para facilitar o entendimento.

A outra forma de se criar essa função é declarando inicialmente que ela é uma função e logo depois dar o seu nome, que antes era o nome de uma variável.

```
function funcao(){
}
```

Com essa maneira ficaria:

```
function novaLinha(){
    document.write("<br>");
}
```

8.2 - Exercícios: Organizando nosso JavaScript

1) Crie um arquivo chamado de adivinhacao.js no mesmo local onde se encontra o seu programa chamado adivinhacao.html.

a) Cópie todo o JavaScript **apenas** o JavaScript, **não precisa** copiar a tag <script> e cole dentro do arquivo adivinhacao.js:

b) Agora apague TODO o JavaScript do arquivo adivinhacao.html.

c) Adicione o source ou seja o arquivo js que você deseja importar:

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

d) Ficaremos com um código semelhante a esse:

```
<meta charset="UTF-8">
Digite seu chute: <input type="text" id="numero" />
<input type="submit" id="btAdvinhar" value="Tente a sorte!" />
<script src="sorteio.js"></script>
```

e) Você poderá fazer isso para todos os programas que você criou até agora.

8.3 - Orientação a Objetos

Depois de termos visto os principais recursos e exercitamos a programação utilizando o JavaScript já estamos prontos para ir além, fazer algo mais profissional.

Pensando nisso podemos ver alguns conceitos básicos que você terá que aprender e estudar.

Um desses conceitos que já estamos utilizando é o de **Objeto**.

Entendendo um Objeto

Não iremos ter tempo hábil para nem sequer dizer o significado de o que é e para que serve um objeto, porém podemos já adiantar uma ideia básica e simples.

Você durante todo o curso utilizou de certos objetos, como o `document` que é um objeto que representa o documento da página tem o `console`, `btAdvinhar` que é um objeto que representa o INPUT no caso o botão.

A ideia mais básica de um objeto é a organização e claro a **responsabilidade** de guardar algumas informações. Por exemplo:

No nosso primeiro programa temos informações como: nome do amigo, idade do amigo:

```
var nome = "William Frota";
var idade = 3;
```

Observe que os dados estão representando algo, neste caso um amigo. Só que eles ficam meio “soltos” ou seja, seria interessante termos algo que os representasse como um **objeto**.

Para criar um objeto no JavaScript é bem simples:

```
var objeto = {
  nome : "valor",
  idade : 0
}
```

Ou seja:

```
var amigo = {
  nome : "William Frota",
  idade: 3
}
```

E para acessar não é diferente do que você já fez com o `document` e/ou `console`:

```
console.log(amigo.nome);
console.log(amigo.idade);
```

Quando se usa o `(.)` ponto, estamos dizendo que queremos acessar **algo** do objeto, nesta caso uma variável. Você poderá acessar variáveis como também e já fizemos isso, funções.

Entender objetos é um assunto um pouco complexo e extenso que não cabe agora. Porém é um pre-requisito **importante** para o desenvolvedor. Por isso temos o curso de Java e Orientação a Objetos onde ensinamos os principais conceitos desse paradigma e que poderá ser utilizado para **qualquer** linguagem orientada a objetos, como o Java, Ruby etc.

Apêndice - Aprendendo a desenhar e animar!

9.1 - Criando gráficos com o CANVAS do HTML 5

As vezes fica mais interessante ter algo visual para mostrarmos, não somente textos ou *alertas*. Para ter algo mais visual podemos utilizar o JavaScript juntamente com o elemento `canvas` do HTML 5 para desenhar formas e animá-las também.

O real objetivo deste capítulo é além de impressionar seus amigos com desenhos e animações queremos que você tenha um entendimento melhor como as funções, variáveis, ifs, arrays e tudo que vimos aqui trabalhando juntas.

O elemento CANVAS

Pela especificação o elemento `canvas` é basicamente:

"Uma tela de bitmap dependente de resolução que pode ser usada para a renderização de elementos gráficos ou outras imagens visuais rapidamente."
– W3C

Resumindo a história o CANVAS é um elemento novo no HTML 5 que permite que você desenhe elementos gráficos usando o JavaScript. Ele pode ser usado para renderizar texto, imagens, gráficos, linhas gradientes e outros efeitos dinamicamente.

Para saber mais: http://www.w3schools.com/html/html5_canvas.asp

Criando o primeiro CANVAS

Para criar um canvas, precisamos utilizar a tag html `<canvas>`:

```
<canvas></canvas>
```

E definirmos 3 propriedades:

- id**: identificador do canvas.
- width**: a largura do canvas.
- height**: altura do canvas.

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
```

API 2D

A API de tela 2D é um objeto que permite desenhar e manipular imagens e elementos gráficos em um elemento canvas. Para fazer referência ao contexto da tela, você chama o `getContext`, que é uma função no elemento canvas. Iremos ver mais detalhes mais a frente.

O que é importante agora saber é que cada tela tem seu próprio contexto, então, se você quiser que sua página tenha vários elementos canvas, você deve ter uma referência a cada contexto individual com que quiser trabalhar.

Existem muitas funções no elemento CANVAS, acesse o link para saber mais: http://www.w3schools.com/tags/ref_canvas.asp

Usando o JavaScript no CANVAS

Após definir o elemento canvas na sua página HTML ao tentar abrir no navegador nada será mostrado, pois como já comentamos precisamos do JavaScript para desenhar e interagir com o canvas.

Então devemos ter uma representação do elemento canvas no JavaScript, algo que já fizemos antes com o elemento `input` utilizando a função `getElementById`:

```
<canvas id="meuCanvas" width="600" height="400"> </[canva]s>
<script>
  var tela = document.getElementById("meuCanvas");
</scr[ipt]>
```

Temos agora a representação do CANVAS como variável: `tela`. Essa variável irá representar exatamente uma tela em branco onde iremos precisar de um *pincel* para pintar o que queremos.

Para criar esse pincel vamos pegar da tela a variável que irá representar isso:

```
var pincel = tela.getContext("2d");
```

Veja que a partir da minha tela eu consigo um pincel e assim poderei desenhar o que eu quiser.

Para saber mais...

Temos várias funções no pincel que podemos utilizar para pintar, desenhar e animar o canvas, iremos utilizar alguns aqui, mas não se preocupe, acesse o link abaixo para saber mais sobre eles. Tudo sobre o `pincel(getContext())`: http://www.w3schools.com/tags/ref_canvas.asp

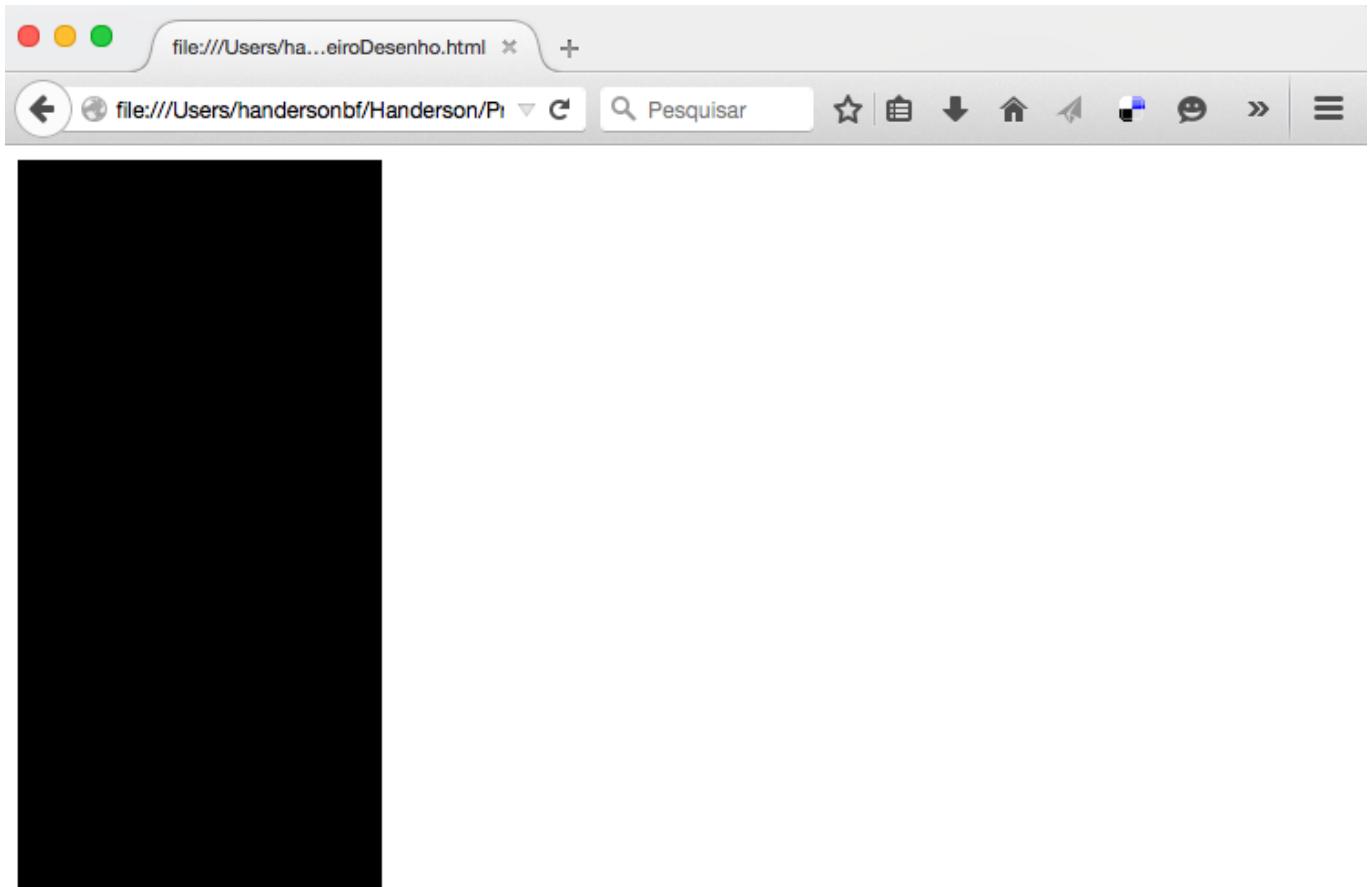
Antes de começar a pintar precisamos escolher a cor do nosso pincel. E usaremos a função `fillStyle` para isso, vamos escolher a cor preta:

```
pincel.fillStyle = "black";
```

Vamos desenhar um retângulo utilizando a função `fillRect`:

```
pincel.fillRect(0, 0, 200, 400);
```

Ao abrir no navegador teremos algo semelhante:



Bem simples não?

Para desenhar outro retângulo basta repetir as funções:

```
pinzel.fillStyle = "orange";
pinzel.fillRect(200, 0, 200, 400);
```

9.2 - Exercícios: Melhorando a interatividade com o HTML

1) Vamos fazer nosso primeiro gráfico. Crie o arquivo `meuPrimeiroDesenho.html`.

a) Agora crie a tela onde você deseja desenhar, no nosso caso o `canvas`:

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
```

b) Agora vamos pegar o elemento que irá representar no JavaScript nossa tela:

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
<script>
    var tela = document.getElementById("meuCanvas");
```

```
</script>
```

c) Vamos criar nosso “`pinzel`”:

```
<canvas id="meuCanvas" width="600" height="400"> </[canvas]>
<script>
    var tela = document.getElementById("meuCanvas");
    var pincel = tela.getContext("2d");

</script>
```

d) Usando a função `fillStyle` vamos definir a cor do nosso pincel, no exemplo sendo preta:

```
<canvas id="meuCanvas" width="600" height="400"> </[canvas]>
<script>
    var tela = document.getElementById("meuCanvas");
    var pincel = tela.getContext("2d");

    pincel.fillStyle = "black";

</script>
```

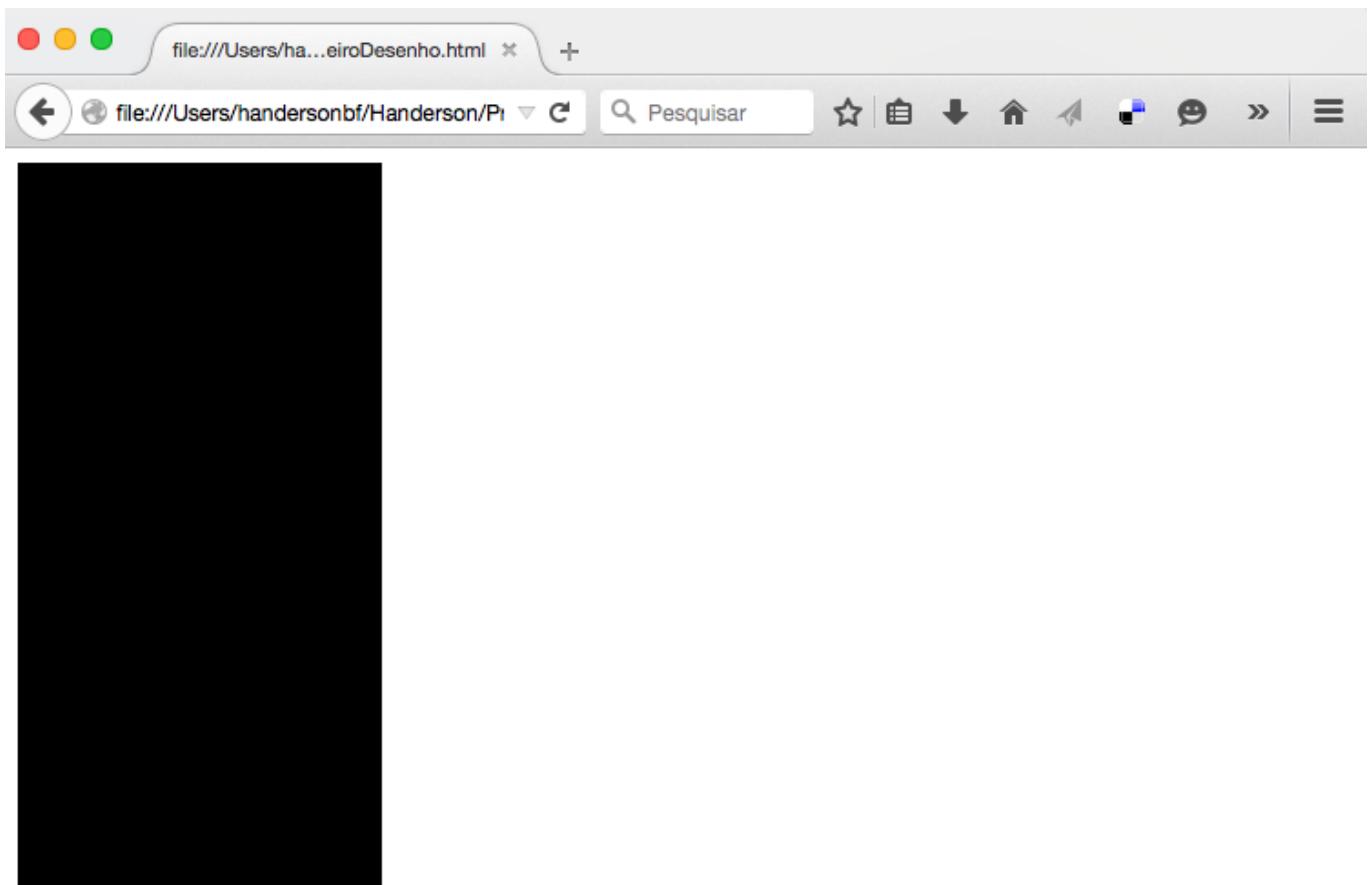
e) Agora com a função `fillRect` vamos desenhar nosso retângulo:

```
<canvas id="meuCanvas" width="600" height="400"> </[canvas]>
<script>
    var tela = document.getElementById("meuCanvas");
    var pincel = tela.getContext("2d");

    pincel.fillStyle = "black";
    pincel.fillRect(0, 0, 200, 400);

</script>
```

Veja como ele irá ficar:



f) Vamos agora desenhar outro retângulo, laranja:

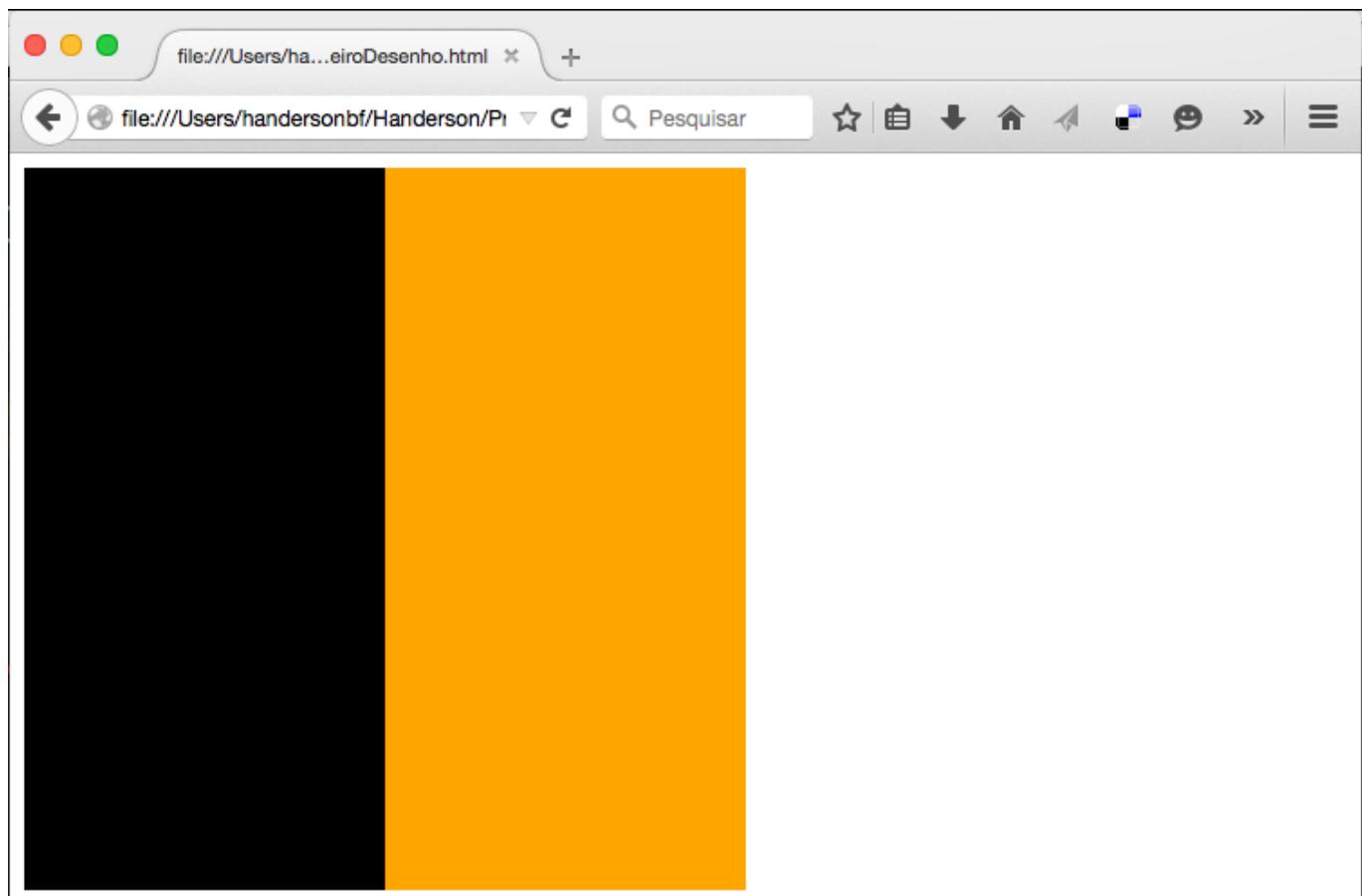
```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
<script>
    var tela = document.getElementById("meuCanvas");
    var pincel = tela.getContext("2d");

    pincel.fillStyle = "black";
    pincel.fillRect(0, 0, 200, 400);

    pincel.fillStyle = "orange";
    pincel.fillRect(200, 0, 200, 400);

</script>
```

Veja como ele irá ficar:



g) Outro retângulo, agora vermelho:

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
<script>
    var tela = document.getElementById("meuCanvas");
    var pincel = tela.getContext("2d");

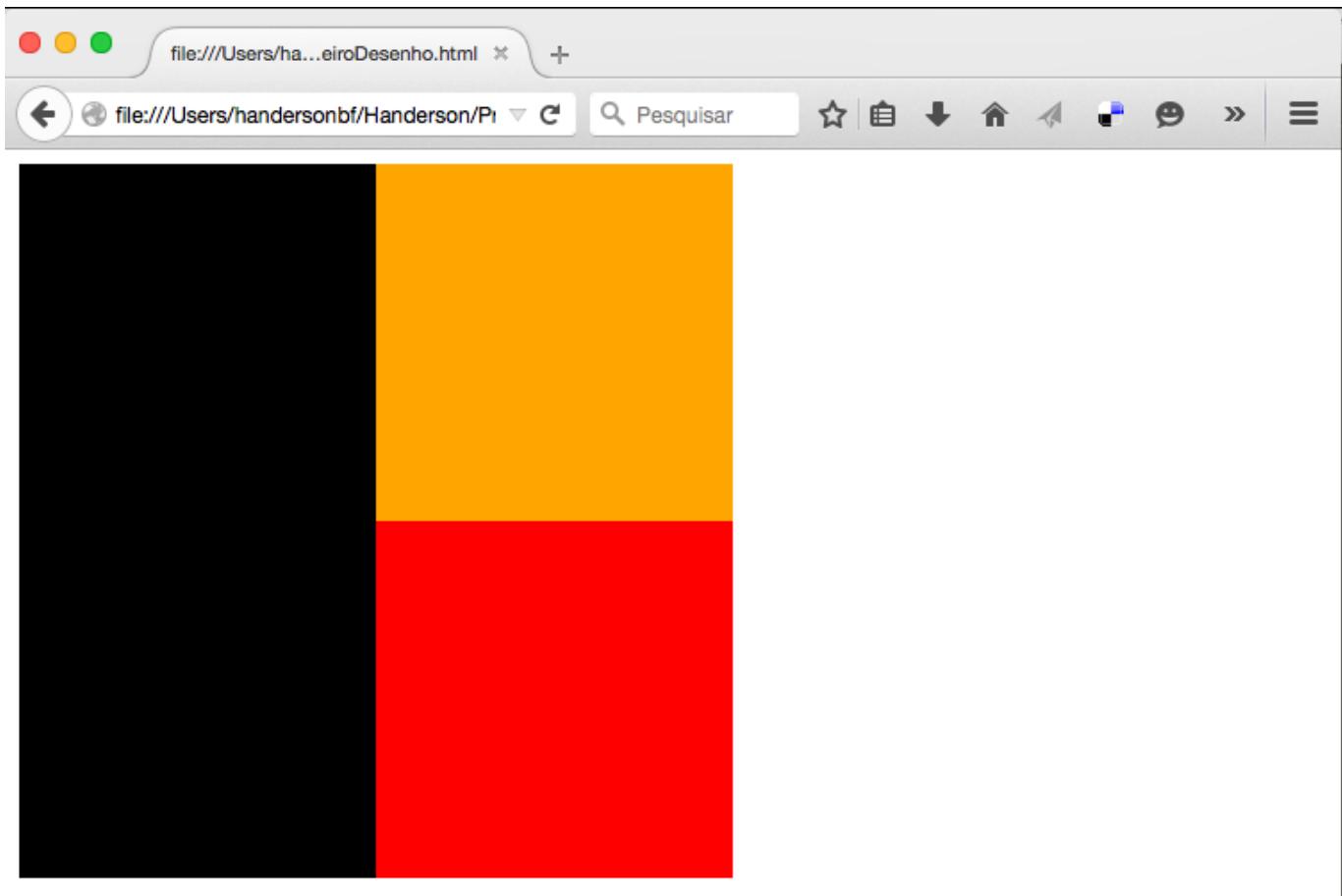
    pincel.fillStyle = "black";
    pincel.fillRect(0, 0, 200, 400);

    pincel.fillStyle = "orange";
    pincel.fillRect(200, 0, 200, 400);

    pincel.fillStyle = "red";
```

```
pinzel.fillRect(200, 200, 200, 400);  
</script>
```

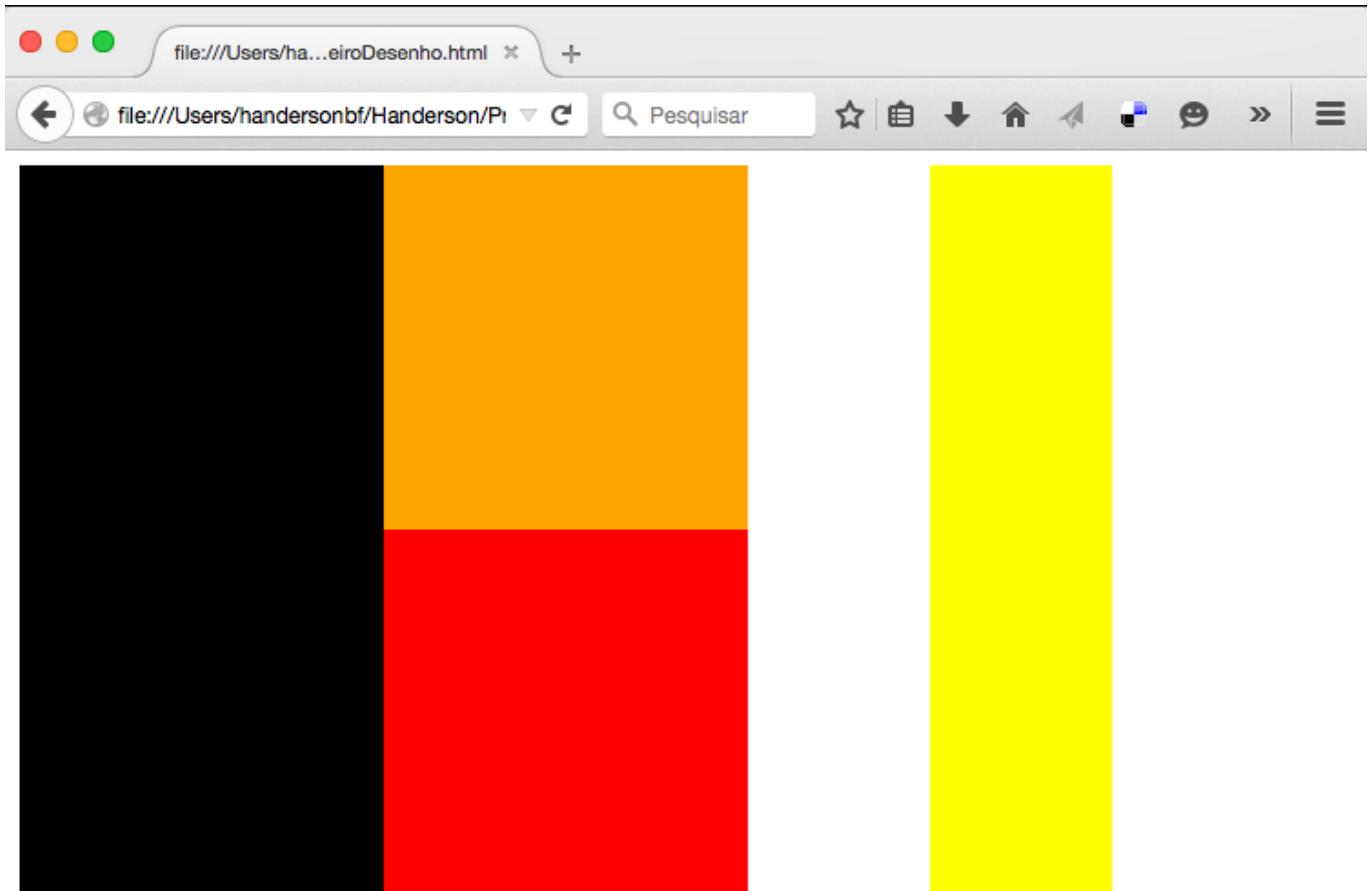
Veja como ele irá ficar:



h) E para finalizar, um amarelo:

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>  
<script>  
    var tela = document.getElementById("meuCanvas");  
    var pinzel = tela.getContext("2d");  
  
    pinzel.fillStyle = "black";  
    pinzel.fillRect(0, 0, 200, 400);  
  
    pinzel.fillStyle = "orange";  
    pinzel.fillRect(200, 0, 200, 400);  
  
    pinzel.fillStyle = "red";  
    pinzel.fillRect(200, 200, 200, 400);  
  
    pinzel.fillStyle = "yellow";  
    pinzel.fillRect(500, 0, 200, 400);  
</script>
```

Veja como ele irá ficar:



9.3 - Mais desenhos...

Para entender melhor o CANVAS vamos tentar exemplificar como ele é desenhado. Vamos criar as 4 bordas do canvas.

Primeiro precisamos criar o CANVAS no HTML:

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
```

Agora vamos pegar o elemento e o pincel a partir dele, como fizemos no capítulo anterior:

```
var tela = document.getElementById("meuCanvas");
var pincel = tela.getContext("2d");
```

Para visualizarmos melhor que tal colocarmos um texto no canvas? Para isso vamos utilizar a função `strokeText` que será responsável em **pintar** um texto, sim pois não será bem um "*texto de verdade*".

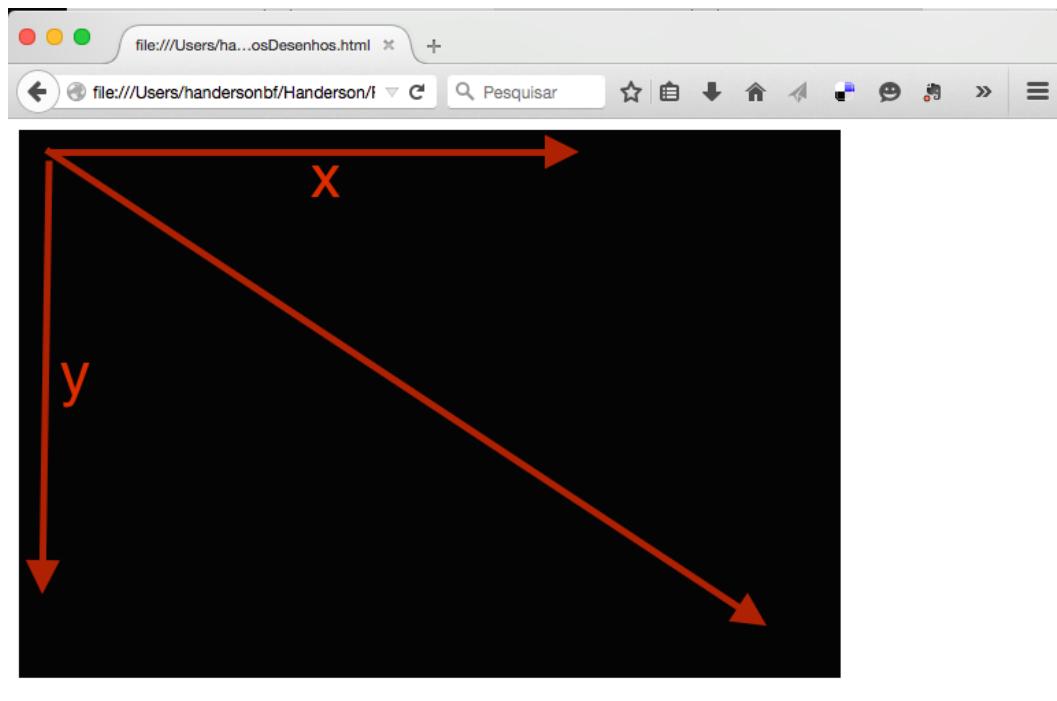
```
pincel.strokeText(" <-- Borda Esquerda", 10, 200);
```

Entenda que para cada borda você terá que pegar as coordenadas do canvas. Para isso teremos que entender as posições na tela.

A função `fillRect(x, y, width, height)` espera as seguintes informações em ordem:

- a) `x`: A posição em que será iniciado o desenho da esqueda para direita.

- b) **y**: A posição em que será iniciado o desenho de cima para baixo.
- c) **width**: Largura do desenho.
- d) **height**: Altura do desenho.



Então podemos desenhar as 4 bordas dessa forma.

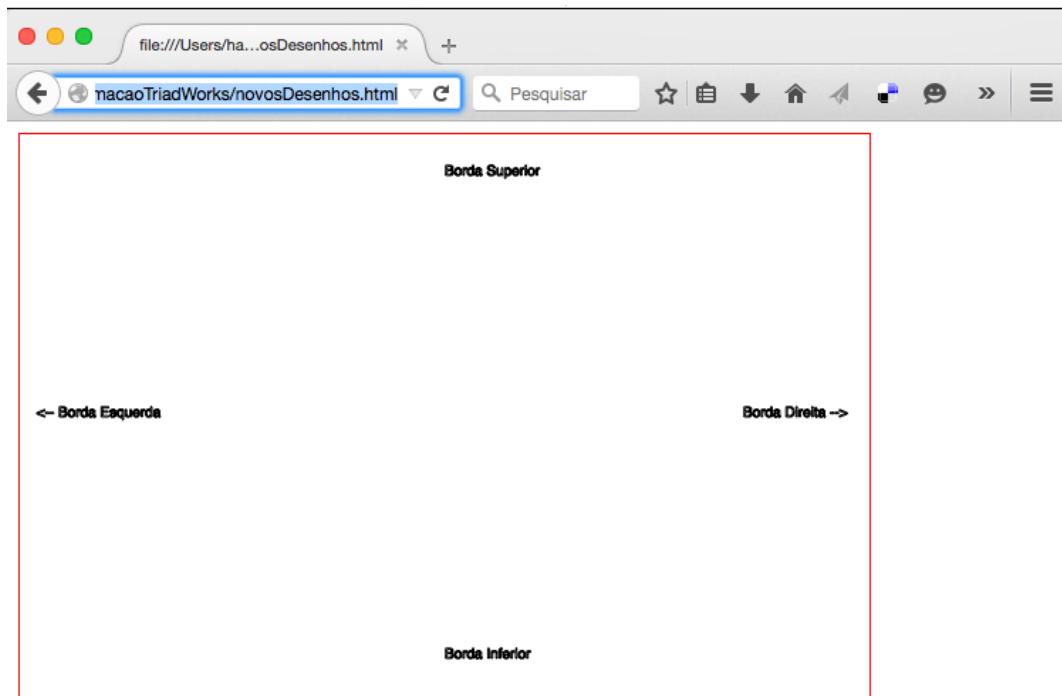
```
//borda esquerda
pincel.strokeText(" <-- Borda Esquerda",10,200);
pincel.fillRect(0, 0, 1, 400);

//borda direta
pincel.strokeText("Borda Direita -->",510,200);
pincel.fillRect(599, 0, 600, 400);

// //borda superior
pincel.strokeText("Borda Superior",300,30);
pincel.fillRect(0, 0, 600, 1);

// //borda inferior
pincel.strokeText("Borda Inferior",300,370);
pincel.fillRect(0, 399, 600, 400);
```

Ficando com essa cara:

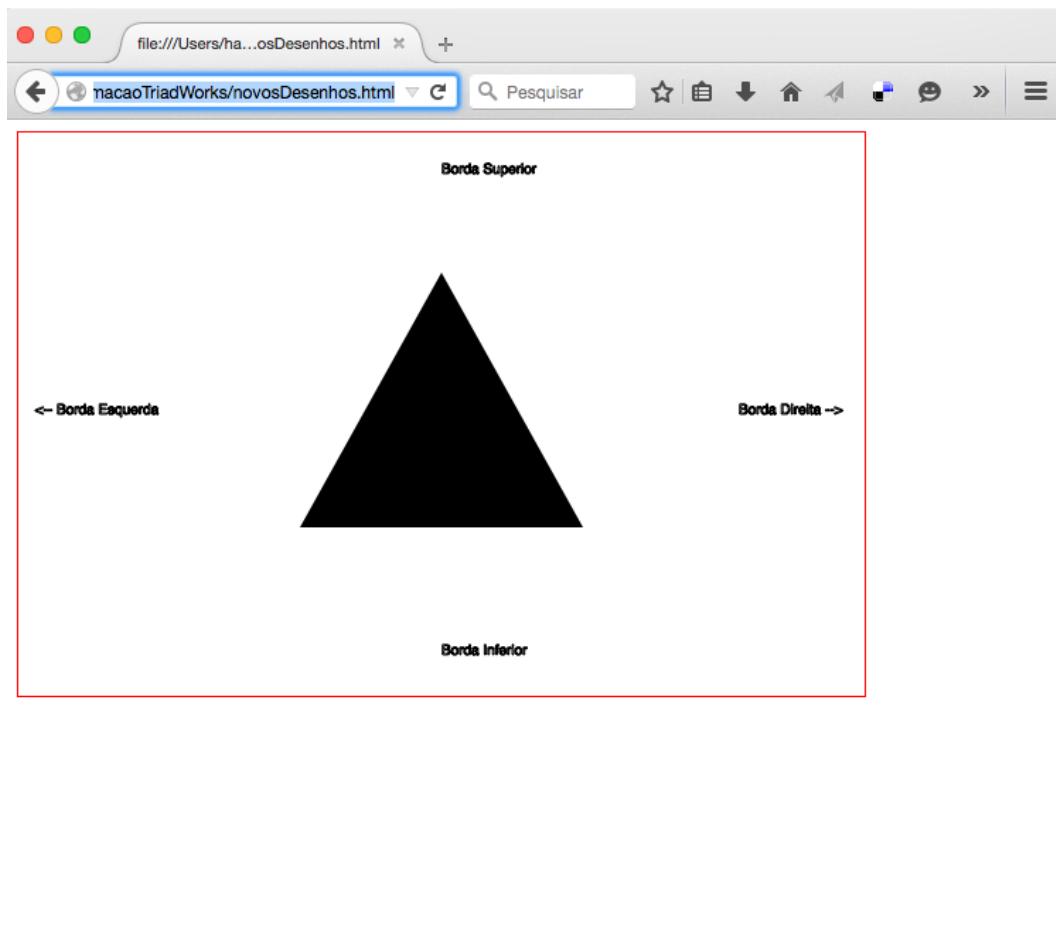


Temos também a função para desenhar linhas e para isso devemos utilizar algumas funções em conjunto.

Para iniciar um novo desenho ou redefinir um desenho já feito utilizamos a função `beginPath()`. É como se tivéssemos dizendo para o browser para ele iniciar um novo desenho e preencher os subcaminhos para que a forma de desenho seja preenchida. Isso é necessário por exemplo para desenhar um triângulo, onde teremos mais de uma linha.

```
pincel.fillStyle = "black";
pincel.beginPath();
pincel.moveTo(300,100);
pincel.lineTo(200,280);
pincel.lineTo(400,280);
pincel.fill();
```

Observe que após o `beginPath` estamos desenhando duas linhas com as funções `lineTo` e estamos utilizando a função `moveTo` para mover o pincel para a posição `x->300 e y->100` do canvas.



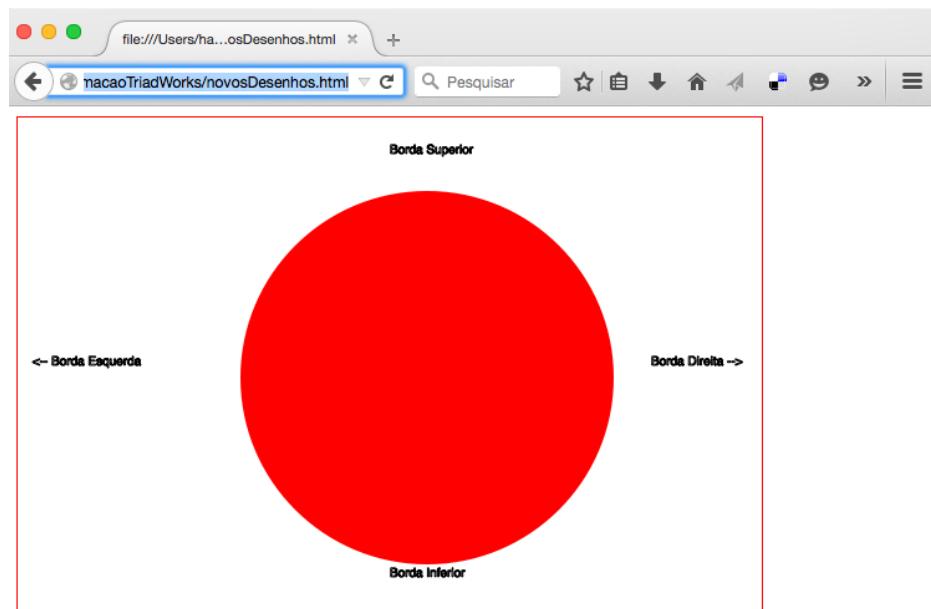
Observe que se tentarmos desenhar algo depois e usarmos o `beginPath` iremos praticamente “pintar” por cima do desenho anterior caso seja na mesma posição:

```
pincel.fillStyle = "black";
pincel.beginPath();
pincel.moveTo(300,100);
pincel.lineTo(200,280);
pincel.lineTo(400,280);
pincel.fill();

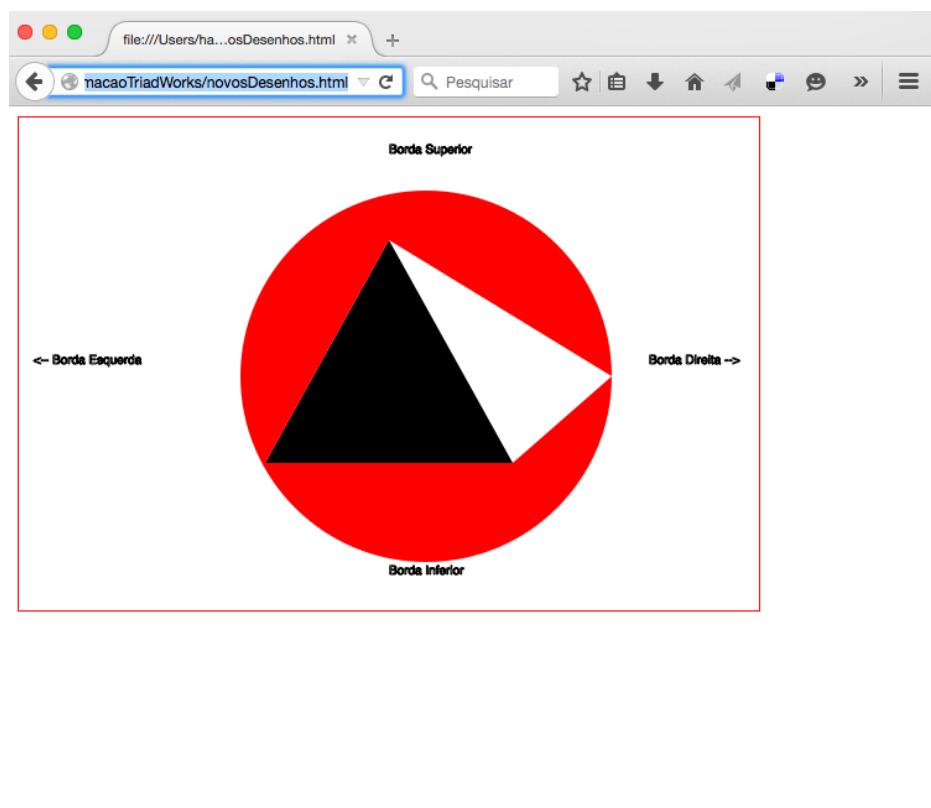
pincel.fillStyle = "red";
pincel.beginPath();
pincel.arc(330,210,150, 0, 2*Math.PI);
pincel.fill();
```

Aqui desenhamos um triângulo e logo em seguida um círculo que está na mesma posição do triângulo, logo ele desenha por cima, pois você deixou claro que é um novo desenho e não faz parte do anterior.

Novo desenho, com o `beginPath`:



Novo desenho, **SEM** o beginPath:



9.4 - Exercícios: Mais desenhos com canvas

1) Crie o arquivo chamado novosDesenhos.html. E vamos exercitar mais os conceitos.

a) Crie um canvas de 600x400.

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
```

b) Agora vamos criar a "tela" e o "pincel":

```
var tela = document.getElementById("meuCanvas");
var pincel = tela.getContext("2d");
```

c) Vamos agora criar nossa borda e a queremos **vermelha** (red):

```
pincel.fillStyle = "red";
```

d) Pronto, agora vamos criar o texto informando que borda iremos fazer. Para isso use a função `strokeText`:

```
pincel.strokeText(" <-- Borda Esquerda",10,200);
```

e) E agora desenhe a borda, ele vai desenhar da posição X sendo ZERO, até o final do canvas que é a posição 400.

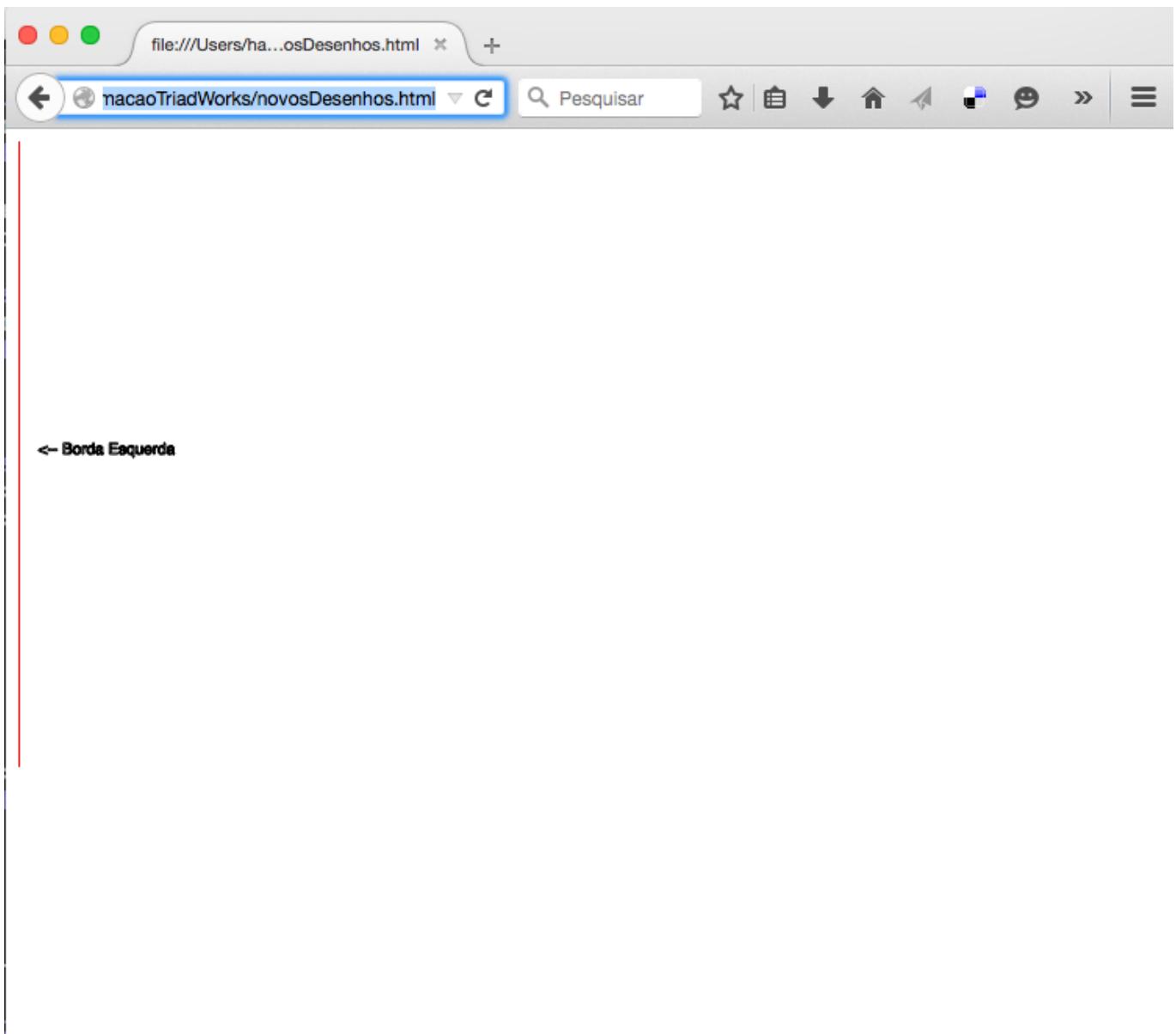
```
pincel.fillRect(0, 0, 1, 400);
```

Veja como fica:

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
<script>
    var tela = document.getElementById("meuCanvas");
    var pincel = tela.getContext("2d");

    pincel.fillStyle = "red";

    //borda esquerda
    pincel.strokeText(" <-- Borda Esquerda",10,200);
    pincel.fillRect(0, 0, 1, 400);
</script>
```



2) Vamos dar continuidade aos nossos desenhos finalizando as bordas:

a) Vamos fazer a borda da direta e vamos manter a cor vermelha.

```
pincel.strokeText("Borda Direita -->",510,200);  
pincel.fillRect(599, 0, 600, 400);
```

b) Borda superior:

```
pincel.strokeText("Borda Superior",300,30);  
pincel.fillRect(0, 0, 600, 1);
```

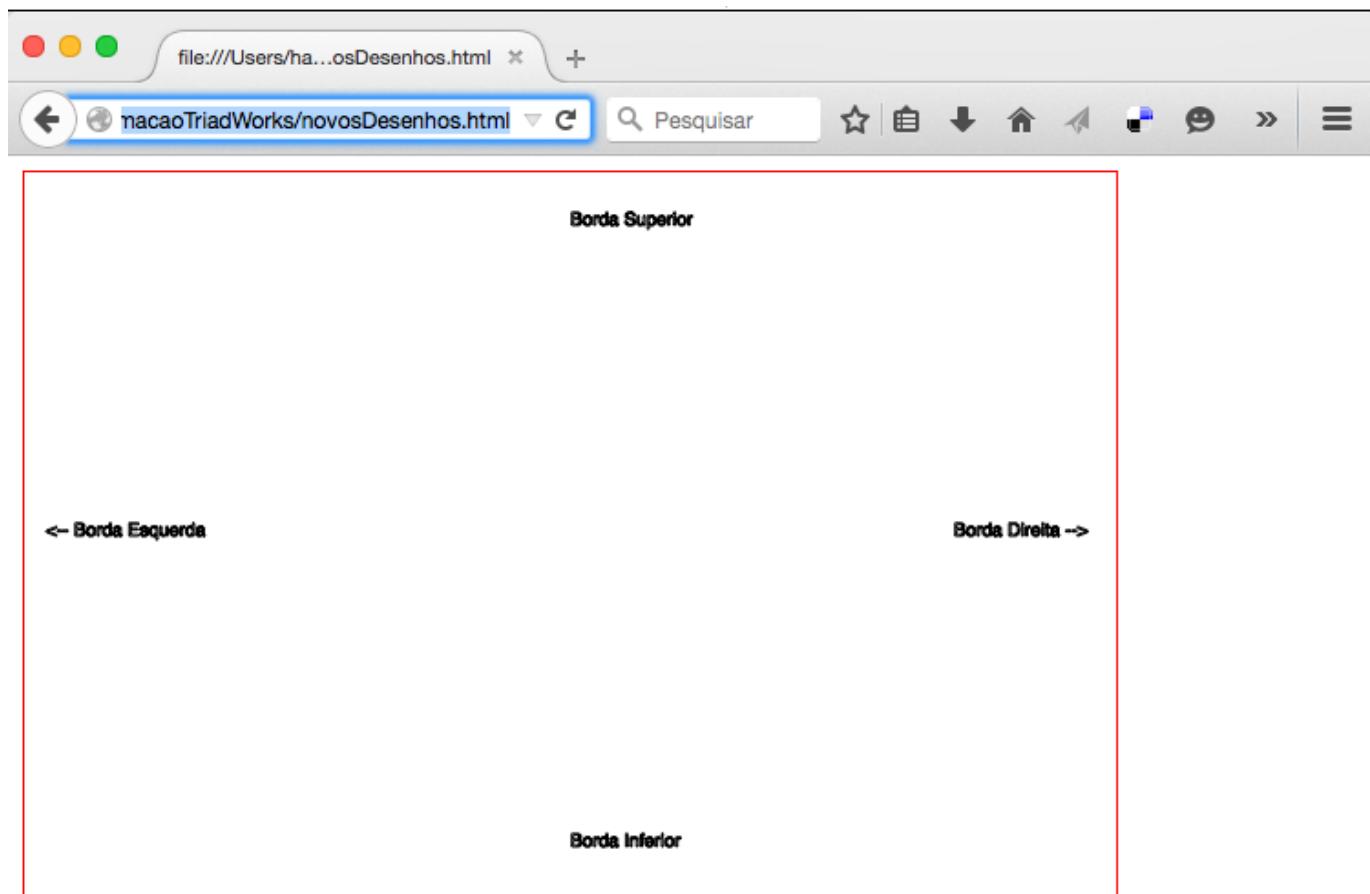
c) E finalmente a borda inferior:

```
pincel.strokeText("Borda Inferior",300,370);  
pincel.fillRect(0, 399, 600, 400);
```

Veja como irá ficar:

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>  
<script>  
    var tela = document.getElementById("meuCanvas");  
    var pincel = tela.getContext("2d");
```

```
pincel.fillStyle = "red";  
  
//borda esquerda  
pincel.strokeText(" <-- Borda Esquerda",10,200);  
pincel.fillRect(0, 0, 1, 400);  
  
//borda direta  
pincel.strokeText("Borda Direita -->",510,200);  
pincel.fillRect(599, 0, 600, 400);  
  
// //borda superior  
pincel.strokeText("Borda Superior",300,30);  
pincel.fillRect(0, 0, 600, 1);  
  
// //borda inferior  
pincel.strokeText("Borda Inferior",300,370);  
pincel.fillRect(0, 399, 600, 400);  
</script>
```



3) Vamos agora desenhar um triângulo. E queremos desenhá-lo no centro.

a) A cor do triângulo será preta:

```
pincel.fillStyle = "black";
```

b) Vamos iniciar a trajetória do desenho:

```
pincel.beginPath();
pincel.moveTo(300,100);
pincel.lineTo(200,280);
pincel.lineTo(400,280);
```

c) E por fim, pintamos:

```
pincel.fillStyle = "black";
pincel.beginPath();
pincel.moveTo(300,100);
pincel.lineTo(200,280);
pincel.lineTo(400,280);
pincel.fill();
```

d) Vamos conferir o resultado final:

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
<script>
    var tela = document.getElementById("meuCanvas");
    var pincel = tela.getContext("2d");

    pincel.fillStyle = "red";

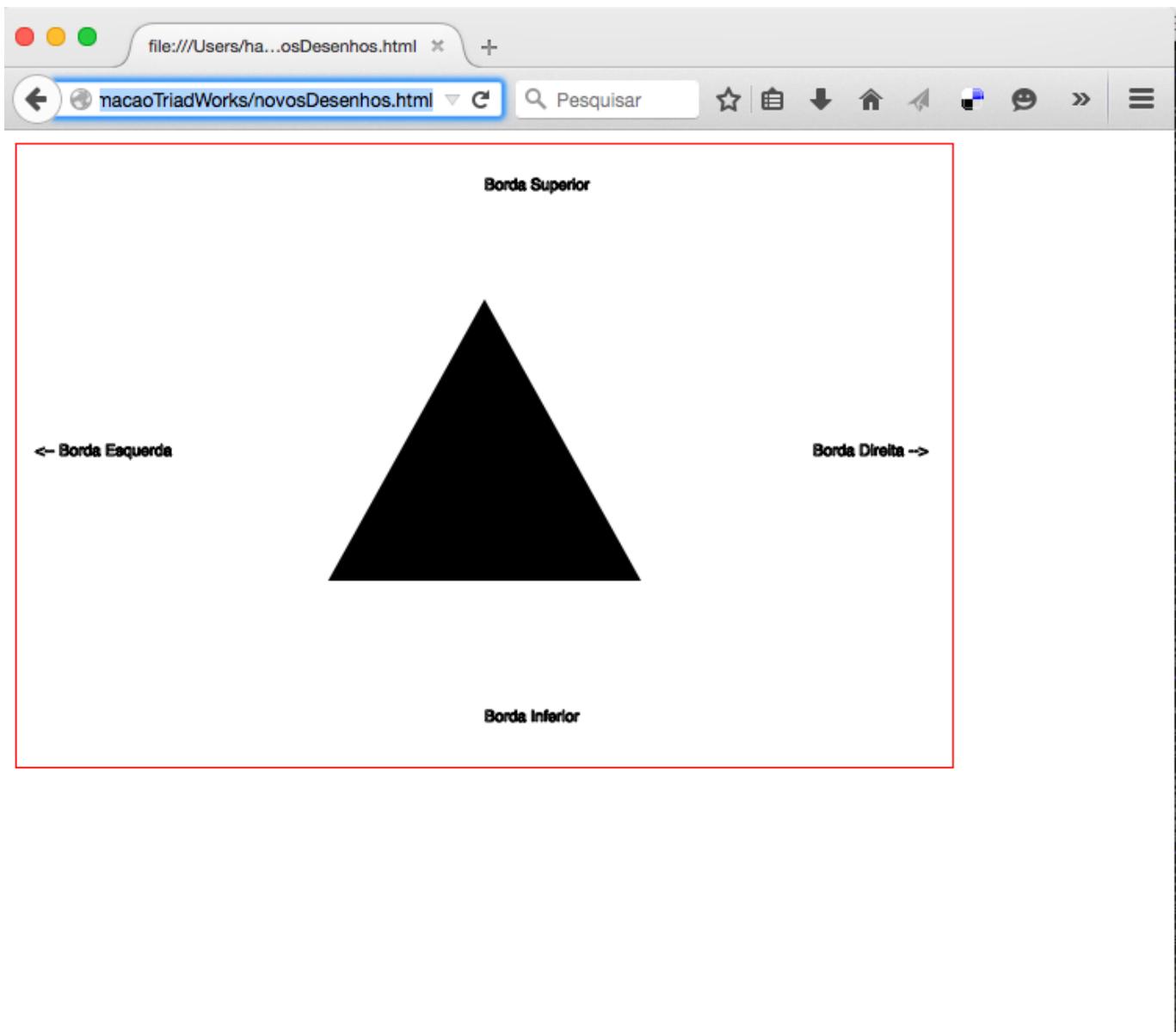
    //borda esquerda
    pincel.strokeText(" <-- Borda Esquerda",10,200);
    pincel.fillRect(0, 0, 1, 400);

    //borda direta
    pincel.strokeText("Borda Direita -->",510,200);
    pincel.fillRect(599, 0, 600, 400);

    // //borda superior
    pincel.strokeText("Borda Superior",300,30);
    pincel.fillRect(0, 0, 600, 1);

    // //borda inferior
    pincel.strokeText("Borda Inferior",300,370);
    pincel.fillRect(0, 399, 600, 400);

    pincel.fillStyle = "black";
    pincel.beginPath();
    pincel.moveTo(300,100);
    pincel.lineTo(200,280);
    pincel.lineTo(400,280);
    pincel.fill();
</script>
```



4) Vamos agora desenhar um círculo no centro.

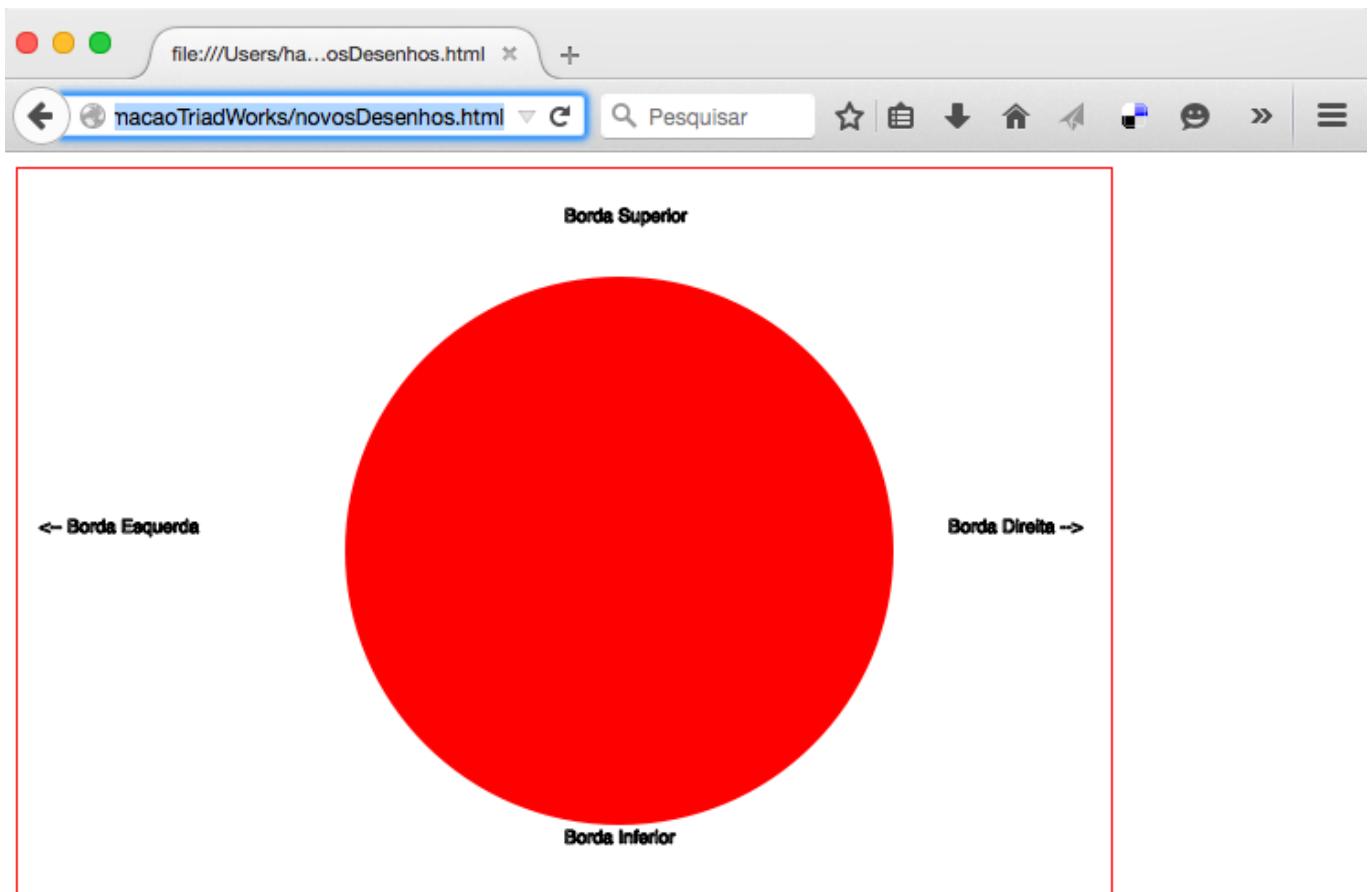
a) No final do código que desenha um triângulo, adicione o círculo. Ele será vermelho:

```
pincel.fillStyle = "red";
```

b) Agora vamos desenhá-lo:

```
pincel.beginPath();
pincel.arc(330,210,150, 0, 2*Math.PI);
pincel.fill();
```

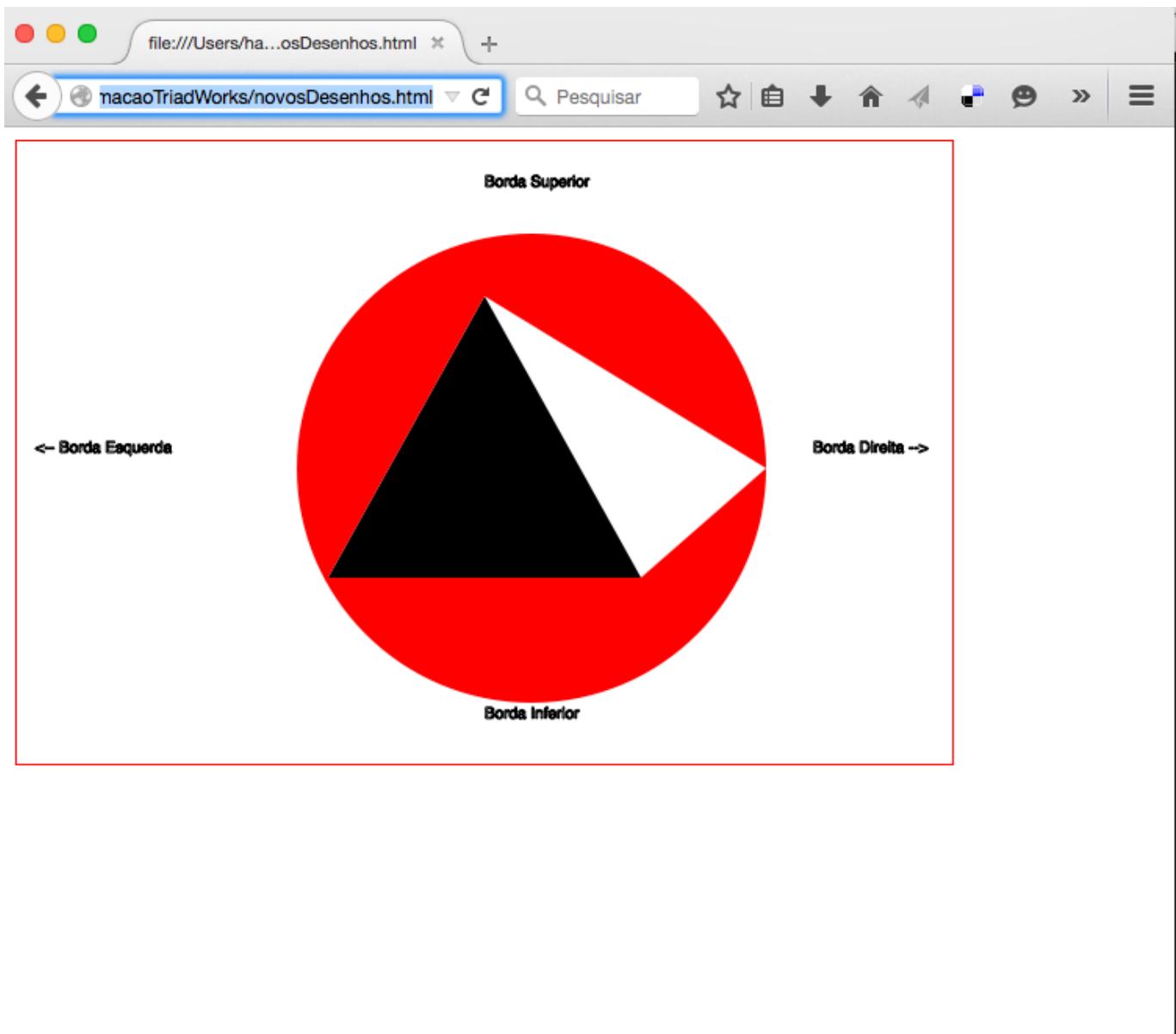
c) Veja como ele irá ficar:



- d) Agora vamos “misturar” os dois desenhos, ou seja, vamos **retirar** a função que diz que será iniciado um novo desenho: `pincel.beginPath()`.

```
pincel.fillStyle = "red";
//pincel.beginPath(); <-- retire
pincel.arc(330,210,150, 0, 2*Math.PI);
pincel.fill();
```

- e) Confira o efeito final:



9.5 - Mais uso de funções

Veja que estamos repetindo muito nosso código:

```
//borda esquerda
pincel.strokeText(" <- Borda Esquerda",10,200);
pincel.fillRect(0, 0, 1, 400);

//borda direta
pincel.strokeText("Borda Direita -->",510,200);
pincel.fillRect(599, 0, 600, 400);

// //borda superior
pincel.strokeText("Borda Superior",300,30);
pincel.fillRect(0, 0, 600, 1);

// //borda inferior
pincel.strokeText("Borda Inferior",300,370);
pincel.fillRect(0, 399, 600, 400);
```

Que tal criarmos uma função que evite ficar repetindo todo esse processo?

Perceba que temos duas ações:

- a) Desenhar um texto;
- b) Desenhar o retângulo;

Vamos fazer a função para desenhar o texto. Para isso precisamos passar como parâmetro da função os dados necessários que são: texto, posição X e posição Y:

```
var desenhaTexto = function(texto, x, y){}
```

Agora basta adicionar o algoritmo utilizando os parâmetros:

```
var desenhaTexto = function(texto, x, y){  
    pincel.strokeText(texto,x,y);  
}
```

E para desenhar o retângulo temos 4 parâmetros, que são: posição X, posição Y, largura e altura.

```
var desenhaRetangulo = function(x, y, largura, altura){}
```

Agora basta adicionar o algoritmo com os parâmetros que iremos receber:

```
var desenhaRetangulo = function(x, y, largura, altura){  
    pincel.fillRect(x, y, largura, altura);  
}
```

Porém temos **duas** ações que devemos tomar(texto e retângulo vide *linha*) então basta criarmos uma função que **junte** as duas:

```
var desenharBorda = function(){  
}
```

Só que temos um probleminha aqui, se você observar as coordenadas de X e Y não são as mesmas no texto nem no retângulo. Então teremos que especificar para que possamos ter apenas uma única chamada de função:

```
var desenharBorda = function(texto, xTexto, yTexto, xRect, yRect, largura, altura){  
    desenhaTexto(texto, xTexto, yTexto);  
    desenhaRetangulo(xRect, yRect, largura, altura);  
}
```

Agora basta alterar seu código:

```
//borda esquerda  
desenharBorda(" <-- Borda Esquerda", 10,200, 0, 0, 1, 400);  
  
//borda direta  
desenharBorda("Borda Direita -->", 510,200, 599, 0, 600, 400);
```

```
// //borda superior
desenharBorda("Borda Superior", 300,30, 0, 0, 600, 1);

// //borda inferior
desenharBorda("Borda Inferior", 300,370, 0, 399, 600, 400);
```

Pronto! Um exemplo de função dentro de função.

9.6 - Exercícios: Mais uso de funções

1) Crie um novo arquivo novoDesenhoComFuncoes.html e para facilitar **copie e cole** do arquivo novosDesenhos e cole neste novo arquivo.

a) Agora crie e função para desenhar um texto:

```
var desenhaTexto = function(texto, x, y){
    pincel.strokeText(texto,x,y);
}
```

b) Agora a função que desenha um retângulo:

```
var desenhaRetangulo = function(x, y, largura, altura){
    pincel.fillRect(x, y, largura, altura);
}
```

c) Vamos criar agora a função que será **responsável** em desenhar uma borda:

```
var desenharBorda = function(texto, xTexto, yTexto,
                               xRect, yRect, largura, altura){
    desenhaTexto(texto, xTexto, yTexto);
    desenhaRetangulo(xRect, yRect, largura, altura);
}
```

2) Agora vamos a alteração principal.

a) Vamos substituir as repetições de códigos:

```
//borda esquerda
pincel.strokeText(" <-- Borda Esquerda",10,200);
pincel.fillRect(0, 0, 1, 400);

//borda direta
pincel.strokeText("Borda Direita -->",510,200);
pincel.fillRect(599, 0, 600, 400);

// //borda superior
pincel.strokeText("Borda Superior",300,30);
pincel.fillRect(0, 0, 600, 1);

// //borda inferior
pincel.strokeText("Borda Inferior",300,370);
pincel.fillRect(0, 399, 600, 400);
```

b) Substitua cada grupo de funções pela função desenharBorda com seus respectivos valores:

```
//borda esquerda
desenharBorda(" <-- Borda Esquerda", 10,200, 0, 0, 1, 400);

//borda direta
desenharBorda("Borda Direita -->", 510,200, 599, 0, 600, 400);
```

```
// //borda superior
desenharBorda("Borda Superior", 300,30, 0, 0, 600, 1);
```

```
// //borda inferior
desenharBorda("Borda Inferior", 300,370, 0, 399, 600, 400);
```

- c) Veja como o código final irá ficar:

```
<canvas id="meuCanvas" width="600" height="400"> </[canva]>
<script>
    var tela = document.getElementById("meuCanvas");
    var pincel = tela.getContext("2d");

    var desenhaTexto = function(texto, x, y){
        pincel.strokeText(texto,x,y);
    }

    var desenhaRetangulo = function(x, y, largura, altura){
        pincel.fillRect(x, y, largura, altura);
    }

    var desenharBorda = function(texto, xTexto, yTexto,
                                xRect, yRect, largura, altura){
        desenhaTexto(texto, xTexto, yTexto);
        desenhaRetangulo(xRect, yRect, largura, altura);
    }

    pincel.fillStyle = "red";

    //borda esquerda
    desenharBorda(" <-- Borda Esquerda", 10,200, 0, 0, 1, 400);

    //borda direta
    desenharBorda("Borda Direita -->", 510,200, 599, 0, 600, 400);

    // //borda superior
    desenharBorda("Borda Superior", 300,30, 0, 0, 600, 1);

    // //borda inferior
    desenharBorda("Borda Inferior", 300,370, 0, 399, 600, 400);

    pincel.fillStyle = "black";
    pincel.beginPath();
    pincel.moveTo(300,100);
    pincel.lineTo(200,280);
    pincel.lineTo(400,280);
    pincel.fill();

    pincel.fillStyle = "red";
    // pincel.beginPath();
    pincel.arc(330,210,150, 0, 2*Math.PI);
    pincel.fill();

</[script]>
```

- d) Você pode criar mais funções, para o desenho do triângulo por exemplo.

9.7 - Animação simples

Com o CANVAS também é possível realizar animações e é isso que vamos entender como fazer agora. Claro iremos fazer uma animação bem simples apenas para entender os conceitos básicos.

Para você entender o básico do que é uma animação é bem simples: imagine um desenho qualquer, onde a cada segundo iremos pintar esse desenho, logo depois vamos reepinta-lo um pouco mais a frente e assim por diante. Resumindo isso é animação.

Primeiro vamos pintar um circulo:

```
var circulo = function(x, y, raio){  
    pincel.fillStyle = "black";  
    pincel.beginPath();  
    pincel.arc(x, y, raio, 0, 2*Math.PI);  
    pincel.fill();  
}  
  
circulo(200, 100, 10);
```

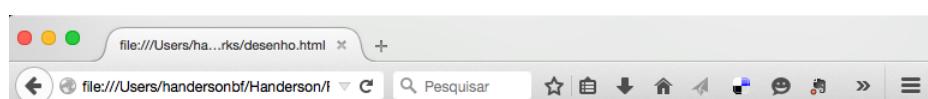
A função `Math.PI` retorna o valor de PI.

Essa função desenha um círculo baseado nos parâmetros que iremos passar.

Vamos agora fazer uma função que pinte o circulo em cada posição diferente do X, basta para isso passar a variável do `for`:

```
var desenhar = function(){  
    for (var i = 0; i < 600; i++) {  
        circulo(i, 100, 10);  
    }  
}  
  
desenhar();
```

Se você rodar esse código **não teremos** o resultado esperado, veja:



Ele fez certo em partes, ele pintou o circulo em cada posição do X, porém, para dar a ideia de animação como foi já foi explicado, precisamos limpar a tela antes dele pintar novamente. E iremos usar a função `clearRect` para isso:

```
var limpaTela = function(){
    pincel.clearRect(0,0,600,400);
}
```

Agora antes dele pintar o próximo circulo devemos limpar a tela primeiro:

```
var desenhar = function(){
    for (var i = 0; i < 600; i++) {
        limpaTela();
        circulo(i, 100, 10);
    }
}

desenhar();
```

Porém agora tivemos um outro problema, está bem rápido, tão rápido que parece que ele só desenhou o circulo no final do canvas. Para isso precisamos definir um tempo de intervalo entre cada desenho.

Para definirmos esse intervalo iremos utilizar uma nova função `setInterval` que recebe uma função que ele irá executar a cada milisegundos.

```
setInterval(funcaoJavaScript, 3000);
```

Neste exemplo acima temos a cada 3 segundo a execução da função `funcaoJavaScript`.

A função `desenhar` é responsável em pintar um circulo em cada posição do CANVAS. Então devemos agora executar essa função a cada 3 segundos por exemplo:

```
setInterval(desenhar, 3000);
```

Como iremos executar a função conforme um intervalo o `for` não será mais necessário para este caso:

```
var desenhar = function(){
    for (var i = 0; i < 600; i++) {
        limpaTela();
        circulo(i, 100, 10);
    }
}

desenhar();
```

E a função `desenhar` ficará bem mais enxuta:

```
var desenhar = function(){
    limpaTela();
    circulo(i, 100, 10);
}

desenhar();
```

Agora basta chamar a função setInterval:

```
var desenhar = function(){
    limpaTela();
    circulo(x,100,10);
}

setInterval(desenhar,30);
```

Algo estranho no código? Se você percebeu que precisamos passar o valor de `x` você acertou. Para isso vamos apenas adicionar um contador:

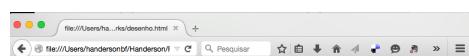
```
var desenhar = function(){
    limpaTela();
    circulo(x,100,10);
    x++;
}

setInterval(desenhar,30);
```

Agora nossa animação está pronta. Um circulo preto será pintado no canto esquerdo do navegador e ele irá “caminhar” até a ponda da direita:



●



●



●

Isso é só uma ideia de como você já pode usar a sua criatividade para criar mais animações e até jogos.

9.8 - Exercícios: Simples movimentos

1) Crie o arquivo chamado pequenaAnimacao.html.

a) Defina um canvas:

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
```

b) Agora defina a tela e o pincel:

```
var tela = document.getElementById("meuCanvas");
var pincel = tela.getContext("2d");
```

c) Vamos criar uma função que desenha um pequeno círculo preto:

```
var circulo = function(x, y, raio){
    pincel.fillStyle = "black";
    pincel.beginPath();
    pincel.arc(x, y, raio, 0, 2*Math.PI);
    pincel.fill();
}
```

d) Agora vamos criar uma função para limpar a tela:

```
var limpaTela = function(){
    pincel.clearRect(0,0,600,400);
}
```

e) Vamos agora fazer um `for` que irá desenhar a bola em cada posição que queremos:

```
for (var i = 0; i < 600; i++) {
    limpaTela();
    circulo(i, 100, 10);
}
```

Rode o programa.

2) Como já sabemos é muito rápido, então vamos definir a velocidade que ele irá executar esse código.

a) Vamos criar uma função chamada desenhar e nela iremos colocar o conteúdo do `for`, esse poderá ser apagado:

```
var x = 1;
var desenhar = function(){
    limpaTela();
    circulo(x,100,10);
    x++;
}
```

b) Agora utilizando a função `setInterval()` vamos passar a função desenhar e dizer que ela deverá ser executada a cada 30 milisegundos.

```
setInterval(desenhar,30);
```

c) Veja como o código final irá ficar:

```
<canvas id="meuCanvas" width="600" height="400"> </canvas>
<script>
    var tela = document.getElementById("meuCanvas");
    var pincel = tela.getContext("2d");

    var circulo = function(x, y, raio){
        pincel.fillStyle = "black";
```

```
    pincel.beginPath();
    pincel.arc(x, y, raio, 0, 2*Math.PI);
    pincel.fill();
}

var limpaTela = function(){
    pincel.clearRect(0,0,600,400);
}

for (var i = 0; i < 600; i++) {
    limpaTela();
    circulo(i, 100, 10);
}
var x = 1;
var desenhar = function(){
    limpaTela();
    circulo(x,100,10);
    x++;
}

setInterval(desenhar,30);
</script>
```

- d) Altere como você desejar o código para ver outras animações, por exemplo, aumentando a velocidade do círculo:

```
var desenha = function(){
    limpaTela();
    circulo(x+x,100,10);
    x++;
}
```

- e) Alterando o valor de Y:

```
var desenha = function(){
    limpaTela();
    circulo(x,100+x,10);
    x++;
}
```

- f) Ou aumentando o tamanho do círculo:

```
var desenha = function(){
    limpaTela();
    circulo(x,100,10+x);
    x++;
}
```

Respostas dos Exercícios

2.6 - Exercícios

1. Código que imprime os nossos amigos:

```
<meta charset="UTF-8">
<script>
    document.write("Nome: Guilherme Frota<br>");
    document.write("Nome: Nahan Frota<br>");
    document.write("Nome: William Frota<br>");
    document.write("Nome: Rafael Ponte<br>");
    document.write("Nome: Lina Tarcia<br>");
</script>
```

2. a) Adicione para cada amigo a sua idade respectivamente:

```
<meta charset="UTF-8">
<script>
    document.write("Nome: Guilherme Frota - idade: " + 8 + "<br>");
    document.write("Nome: Nahan Frota - idade: " + 8 + "<br>");
    document.write("Nome: William Frota - idade: " + 3 + "<br>");
    document.write("Nome: Rafael Ponte - idade: " + 31 + "<br>");
    document.write("Nome: Lina Tarcia - idade: " + 32 + "<br>");
</script>
```

- b) Imprima mais uma linha que mostre a soma de todas as idades:

```
<meta charset="UTF-8">
<script>
    document.write("Nome: Guilherme Frota - idade: " + 8 + "<br>");
    document.write("Nome: Nahan Frota - idade: " + 8 + "<br>");
    document.write("Nome: William Frota - idade: " + 3 + "<br>");
    document.write("Nome: Rafael Ponte - idade: " + 31 + "<br>");
    document.write("Nome: Lina Tarcia - idade: " + 32 + "<br>");
    document.write("Soma das idades: " + (8 + 3 + 31 + 32));
</script>
```

- c) Agora faça com que o programa calcule a média das idades:

```
<meta charset="UTF-8">
<script>
    document.write("Nome: Guilherme Frota - idade: " + 8 + "<br>");
    document.write("Nome: Nahan Frota - idade: " + 8 + "<br>");
    document.write("Nome: William Frota - idade: " + 3 + "<br>");
    document.write("Nome: Rafael Ponte - idade: " + 31 + "<br>");
    document.write("Nome: Lina Tarcia - idade: " + 32 + "<br>");
    document.write("Soma das idades: " + (8 + 3 + 31 + 32));
    document.write("<br>A média de idades: " + (8 + 3 + 31 + 32)
</script>
```

3. <meta charset="UTF-8">
<script>
 document.write("Nome: Gilherme Frota - idade: " + 8 + "
");
 document.write("Guilherme nasceu em: " + (2015 - 8) + "
");
 document.write("Nome: Nahan Frota - idade: " + 8 + "
");
 document.write("Nahan nasceu em: " + (2015 - 8) + "
");
 document.write("Nome: William Frota - idade: " + 3 + "
");
 document.write("William nasceu em: " + (2015 - 3) + "
");
 document.write("Nome: Rafael Ponte - idade: " + 31 + "
");
 document.write("Rafael nasceu em: " + (2015 - 31) + "
");
 document.write("Nome: Lina Tarcia - idade: " + 32 + "
");
 document.write("Lina nasceu em: " + (2015 - 32) + "
");
</script>

4. a) O programa deve exibir os dias que essa pessoa já viveu.

```
document.write("Nome: Gilherme Frota - idade: " + 8 + ", nasceu em: " + (2015 - 8));  
document.write(" e já viveu " + (8*365) + " dias.<br>");  
document.write("Nome: Nahan Frota - idade: " + 8 + ",nasceu em: " + (2015 - 8));  
document.write(" e já viveu " + (8*365) + " dias.<br>");  
document.write("Nome: William Frota - idade: " + 3 + ",nasceu em: " + (2015 - 3));  
document.write(" e já viveu " + (3*365) + " dias.<br>");  
document.write("Nome: Rafael Ponte - idade: " + 31 + ",nasceu em: " + (2015 - 31));  
document.write(" e já viveu " + (31*365) + " dias.<br>");  
document.write("Nome: Lina Tarcia - idade: " + 32 + ",nasceu em: " + (2015 - 32));  
document.write(" e já viveu " + (32*365) + " dias.<br>");
```

b) Cálculo do total de dias.

```
document.write("Total de dias: "  
    + ((8*365) + (8*365) + (3*365) + (31*365) + (32*365)));
```

c) Cálculo da média de dias.

```
document.write("<br>Média de dias: "  
    + ((8*365) + (8*365) + (3*365) + (31*365) + (32*365)) / 5);
```