# Lógica da Programação

Conteúdo

[Lógica da Programação 1](#_Toc45307867)

[Aula 1: Comece a Programar Hoje 3](#_Toc45307868)

[Episódio 1. 3](#_Toc45307869)

[Episódio 2. 3](#_Toc45307870)

[Episódio 3 7](#_Toc45307871)

[Aula 2: Comunique-se com o Usuário 13](#_Toc45307872)

[Episódio 1. 13](#_Toc45307873)

[Episódio 2. 19](#_Toc45307874)

[Aula 3: Torne seu programa dinâmico com Variaveis 23](#_Toc45307875)

[Episódio 1. 23](#_Toc45307876)

[Episódio 2. 27](#_Toc45307877)

[Aula 4: Crie suas próprias funcionalidades 28](#_Toc45307878)

[Episódio 1. 28](#_Toc45307879)

[Episódio 2. 32](#_Toc45307880)

[Aula 5: Pratique Resolvendo problemas do seu dia-a-dia 40](#_Toc45307881)

[Episódio 1. 40](#_Toc45307882)

[Episódio 2. 46](#_Toc45307883)

[Episódio 3. 49](#_Toc45307884)

[Episódio 4. 51](#_Toc45307885)

[Aula 6: Execute Códigos Diferentes Dependendo da Condição 55](#_Toc45307886)

[Aula 8: Intereja de maneira diferente com o Usuário 61](#_Toc45307887)

[Episódio 1. 61](#_Toc45307888)

[Episódio 2 68](#_Toc45307889)

[Aula 9: Trabalhe com Muitos Dados 71](#_Toc45307890)

[Episódio 1 71](#_Toc45307891)

# Aula 1: Comece a Programar Hoje

### Episódio 1.

Olá! Me chamo **Flávio Almeida**, bem-vindos ao curso de **Lógica de Programação I: Os primeiros programas com JavaScript e HTML**.

Se você iniciou este curso é porque deseja aprender a programar e dar os primeiros passos neste universo da programação. Se sua meta for criar uma aplicação web, um aplicativo para seu smartphone favorito, macros de Excel, ou automatizar tarefas do dia a dia, o seu ponto de partida será o mesmo - a **lógica de programação**.

Assim como em qualquer profissão, o programador precisa de ferramentas para que possa exercer esta função. Uma delas é o próprio navegador, o browser que você costuma utilizar para navegar na web. Podem ser utilizados o Edge, o Firefox, Chrome, no caso, utilizarei o **Google Chrome**. É recomendável ao aluno utilizar o mesmo navegador do instrutor para ter paridade visual com o que é feito ao longo do curso.

Se você tem dúvidas de que o browser é uma ferramenta de desenvolvimento, basta olharmos para qualquer página e percebermos que tudo que está ali é resultado do trabalho de programação de algum indivíduo. No caso, observaremos a página inicial da [Alura](https://www.alura.com.br/). Alguém elaborou um conjunto de instruções, que ao serem interpretadas pelo navegador, exibiu uma página para nós.

Ao clicarmos na página e utilizarmos o atalho "Ctrl + U", veremos o conjunto de instruções, os códigos que foram escritos para gerar tudo que foi feito, que foi pensado por uma equipe de desenvolvedores:

Além do browser, será necessário utilizarmos algum **editor de texto**, pode ser um simples, que já vem no sistema operacional. Entretanto, caso seja usuário do **OS Windows**, é interessante utilizar algum outro, como o [**NotePad++**](https://notepad-plus-plus.org/download/v7.5.5.html), ou o [**Sublime**](https://www.sublimetext.com/3).

Nos [exercícios](https://cursos.alura.com.br/course/logica-programacao-javascript-html/task/17638) desta aula, você encontra os endereços para fazer o download destes editores, bem como instruções para suas instalações.

No caso, utilizo o **Sublime**, se você desejar adotá-lo para ter uma paridade visual com o instrutor ao longo do treinamento, o endereço para baixá-lo estará disponível nos exercícios.

Como já temos o editor de texto e o navegador, vamos dar início ao nosso primeiro programa?

## Episódio 2.

Nesta aula, daremos início à criação de nosso **primeiro programa**.

Para melhor organização, criaremos uma pasta chamada **logica**, onde inseriremos todos os arquivos que vamos construir ao longo do treinamento.

Abriremos o editor de texto, que é o **Sublime**, e tem um arquivo em branco. Como padrão, temos o atalho "Ctrl + S" que serve para salvarmos o arquivo que será o programa.

Clicaremos na pasta logica e salvaremos o novo arquivo como primeiro\_teste.html. Ainda não temos certeza que será um programa, por isto faremos primeiro um teste.

Tendo o arquivo, é importante a inserção do .html, por esta ser a linguagem que o navegador entende e, ao utilizarmos a extensão, estamos orientando-o sobre qual deve ser a interpretação do arquivo por parte do navegador e do sistema operacional.

Em nosso primeiro teste, no Sublime, digitaremos:

Meu primeiro teste!

Seria isso um programa?

Feito isso, salvaremos, escrevemos as mensagens e deixamos o espaço de duas linhas entre a primeira frase e a segunda.

Retornaremos ao navegador, no caso o **Google Chrome**. Solicitaremos que ele abra o nosso arquivo. Selecionaremos, na barra de menu superior, "Arquivo > Abrir arquivo...", ou com a tecla de atalho "CTRL + O" e no diretório selecionaremos o arquivo que acabamos de criar.

Ao abrirmos, é exibido o seguinte:

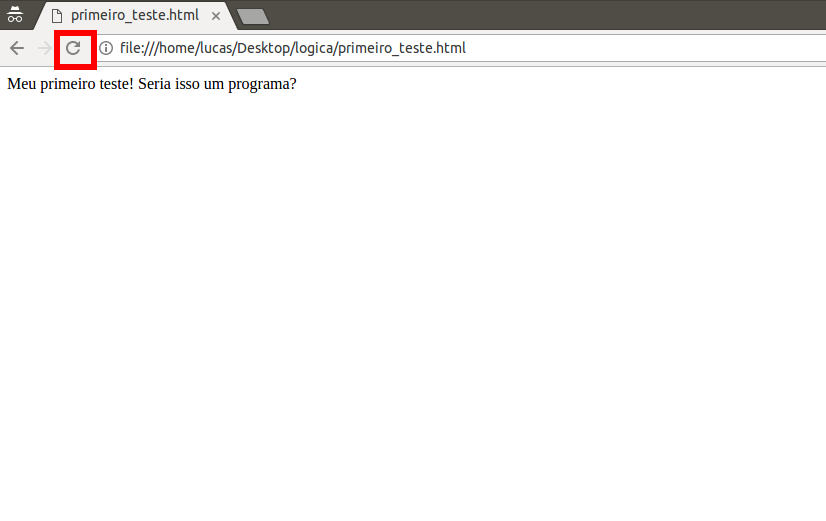
Meu primeiro teste! Seria isso um programa?

Notamos que há uma peculiaridade, será que isto é um programa?

Quando digitamos em nosso editor de texto, no código-fonte ou origem, pulamos duas linhas, entretanto, ao exibir no navegador, isto é ignorado completamente, e as frases são exibidas lado a lado.

Por que isso acontece?

A primeira coisa que precisamos entender é que o navegador está preparado para ler código **HTML**. Ele não sabe que, ao pularmos a linha com a tecla "Enter", ainda que sejam várias vezes, ele deve pular linhas. Tentaremos colocar quantas linhas quisermos entre as duas frases, e recarregar a página utilizando o botão "Recarregar", logo ao lado da barra onde se digita o endereço no navegador:



Com o uso do atalho "Ctrl + R", ou simplesmente pressionando-se a tecla "Enter" com o cursor sobre o endereço, o navegador nunca mostrará as linhas puladas. A ideia é que precisamos utilizar um comando especial, que o HTML tenha preparado, para pular linhas, no caso o <br>:

Meu primeiro teste!

<br>

Seria isso um programa?

Trata-se de uma *tag* HTML, e todas têm esta característica de começar com o símbolo de "menor" (**<**), o nome da tag - "br" é abreviação de *break* -, e fecha com o símbolo de "maior" (**>**). Isto indica ao navegador que ele deve pular uma linha.

Feitas estas alterações, atualizaremos a página e veremos que de fato foi pulada uma linha:

Meu primeiro teste!

Seria isso um programa?

Ou seja, foi passada uma instrução HTML de quebra, que o navegador está preparado para entender. Se quisermos pular mais duas linhas, basta acrescentarmos mais dois comandos <br>:

Meu primeiro teste!

<br>

<br>

<br>

Seria isso um programa?

Após salvarmos, retornaremos ao navegador, e recarregaremos a página. Temos o seguinte resultado:

Meu primeiro teste!

Seria isso um programa?

É esta a primeira tag HTML que aprendemos, para pular linhas. Como estamos criando o programa, geraremos uma área de destaque que exibe a mensagem "Meu primeiro teste!".

O **HTML** conta com a tag <h1>. O "h" se origina da palavra *heading*, ou seja, cabeçalho. Significa que este é o primeiro título, ou título principal é algo que precisa ser destacado em uma página.

Uma particularidade da tag <br> é que, no início, inserimos <> e pronto - já para a tag <h1> e a maioria das tags HTML, precisaremos da mesma tag com uma barra </h1> para fechá-la, conforme vemos abaixo:

<h1> Meu primeiro teste!</h1>

Isso indica que ela começou antes de "Meu" e terminou após "teste!". Ou seja, tudo que está entre <h1> e </h1> está marcado pela tag <h1>. É como se usássemos um marcador de texto e, depois, passássemos sobre a frase "Meu primeiro teste!" e, desta forma, marcaríamos o conteúdo.

Ao recarregarmos a página, veremos que foi dado destaque à frase:



Isto porque trata-se do primeiro título de uma página.

Agora ficamos com um questionamento: o que acabamos de criar é um programa?

## Episódio 3

Dando continuidade às aulas anteriores, ficamos com o seguinte questionamento: seria isto um programa? Para verificarmos, faremos o seguinte:

<h1>Meu primeiro teste!</h1>

<br>

Seria isso um programa? Descubra visitando a Alura aqui!

Utilizaremos uma tag chamada **âncora**, representada pela letra "a". O conteúdo dela será a palavra "aqui":

<h1>Meu primeiro teste!</h1>

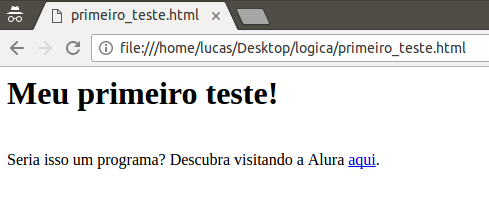
<br>

Seria isso um programa? Descubra visitando a Alura <a>aqui</a>!

Por meio do atributo href, indicaremos o endereço web da Alura:

<a href="http://www.alura.com.br">aqui</a>

Utilizamos uma tag HTML de âncora para criar links na web. Salvaremos e retornaremos ao navegador, feito isso, recarregaremos a página. Vemos que a palavra aqui está em destaque:



Ao clicar no texto somos direcionados para a página inicial da Alura.

O problema do código que escrevemos é que, apesar de interessante, o HTML é estático, isto é, não é dinâmico. E se quisermos que o HTML, em vez de pular poucas linhas, pule dez, quinze, ou vinte? Teremos que abrir o arquivo HTML e inserir diversas tags do tipo <br>, da seguinte forma:

<h1>Meu primeiro teste!</h1>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

Seria isso um programa? Descubra visitando a Alura <a href="http://www.alura.com.br">aqui</a>.

Assim, sempre que quisermos aumentar ou diminuir o número de quebras de linha teremos que remover ou adicionar as tags <br>, sempre modificando o documento. Como ele é estático, não é possível, por exemplo, exibir um contador na tela com a passagem das horas, tampouco será possível capturar as informações do usuário.

Para que possamos de fato programar, precisaremos utilizar uma outra linguagem. O navegador é poliglota, e "fala" algumas linguagens além de HTML, sendo capaz de trabalhar com uma linguagem verdadeiramente de programação, que é o **JavaScript**.

Neste treinamento, aprenderemos a lógica de programação utilizando a linguagem **JavaScript**.

Queremos exibir um alerta para o usuário, um pop-up que surja na tela e contenha a seguinte mensagem: "isso aqui é uma linguagem dinâmica" ou "isso aqui é uma linguagem de programação".

Como fazer isso? Primeiro, definiremos a mensagem em JavaScript. E para criarmos um texto no JavaScript, é necessário que ele esteja entre aspas:

<h1>Meu primeiro teste!</h1>

<br>

Seria isso um programa? Descubra visitando a Alura <a href="http://www.alura.com.br">aqui</a>!

"Isso sim é um programa"

O JavaScript nos permite exibir um pop-up na tela, que será de alerta. Precisaremos utilizar uma instrução da linguagem indicando que um alerta deverá ser exibido:

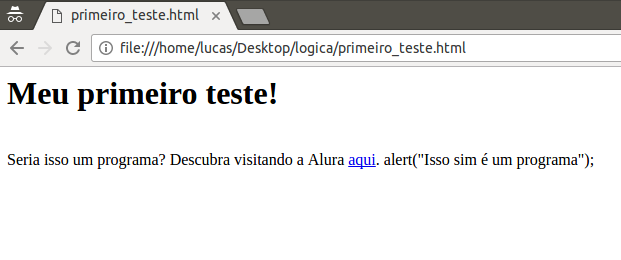
alert"Isso sim é um programa";

Há uma peculiaridade: o texto deve estar entre **parênteses e aspas duplas** **" "**, e com um **;** (ponto e vírgula) no fim:

alert("Isso sim é um programa");

Uso do ponto e vírgula no final de uma instrução JavaScript é opcional e neste treinamento não teremos problemas com sua ausência. Contudo, ao avançar nesta linguagem você verá determinados processamentos de código podem resultar em algo não esperado na ausência do ponto e vírgula. Sendo assim, vamos criar desde já o hábito de terminar uma instrução com o ponto e vírgula.

Adiante veremos mais sobre isso. Em teoria, isto é suficiente para escrevermos nosso primeiro código em **JavaScript**. Salvaremos e atualizaremos a página. Ao abrirmos o arquivo, o resultado é o seguinte:



Porém, o pop up não aparece. O que temos é o texto que gostaríamos que fosse exibido na tela para o usuário aparecendo no corpo do HTML e, ainda, com um problema de acentuação - que resolveremos adiante.

Por que isto acontece? Como o navegador é poliglota, ou seja, entende mais de uma linguagem (por exemplo, HTML e JavaScript), temos que indicar que determinada linha é uma instrução.

Como faremos isto? Com uma tag <script>, que também abre e fecha:

<script>

alert("Isso sim é um programa");

</script>

Assim, o conteúdo do <h1> é: "Meu primeiro teste!"; e o da tag <script> é o alert:

<h1>Meu primeiro teste!</h1>

<br>

Seria isso um programa? Descubra visitando a Alura <a href="http://www.alura.com.br">aqui</a>!

<script>

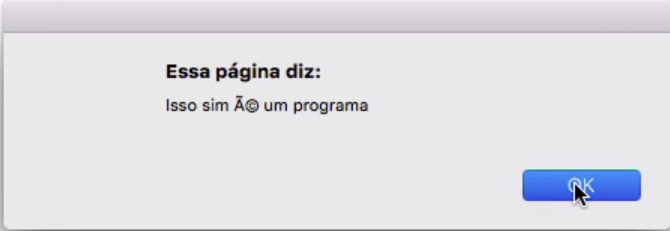
alert("Isso sim é um programa");

</script>

Quando o navegador for processar esta instrução, ele perceberá que ela está inserida em uma tag <script>, e deixará de olhá-la como **HTML**, passando a interpretá-la como JavaScript. Retornaremos ao navegador e recarregaremos a página, e teremos o seguinte alerta:



Podemos pressionar "Ok", e vemos que há um problema de acentuação:



Para resolvermos o problema da acentuação, ao salvarmos o arquivo em nosso disco, o faremos com um conjunto de caracteres, e o padrão é UTF-8. Sendo assim, utilizamos a tag desta maneira:

<meta charset="UTF-8">

Estamos indicando ao navegador como deve ser a interpretação das cadeias de caracteres, ou seja, do texto que está sendo exibido. Portanto, a simples adição desta tag resolverá o problema de acentuação:

<meta charset="UTF-8">

<h1>Meu primeiro teste!</h1>

<br>

Seria isso um programa? Descubra visitando a Alura <a href="http://www.alura.com.br">aqui</a>!

<script>

alert("Isso sim é um programa");

</script>

Salvaremos e retornaremos ao navegador. Podemos utilizar o atalho "Ctrl + S" para salvar. Recarregaremos a página e veremos o pop up, bem como o texto em HTML, com o problema de acentuação resolvido. Será que é necessário todo este texto? O que aconteceria, por exemplo, se removêssemos os parênteses de "Isso sim é um programa"?

<meta charset="UTF-8">

<h1>Meu primeiro teste!</h1>

<br>

Seria isso um programa? Descubra visitando a Alura <a href="http://www.alura.com.br">aqui</a>!

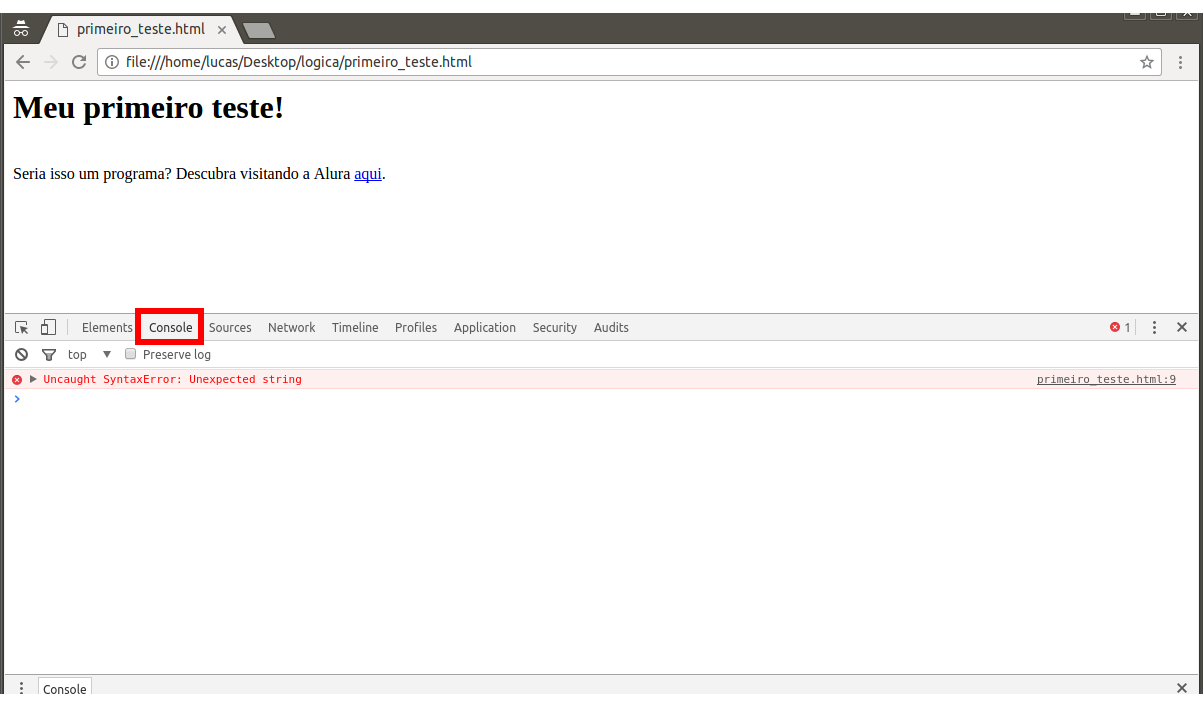
<script>

alert"Isso sim é um programa";

</script>

Salvaremos, retornaremos ao navegador e, ao recarregaremos a página, nenhum pop up será exibido. Com isto, entendemos que é interessante aprendermos que todo navegador conta com um plug-in, um depurador, que indica onde nosso código tem problemas. Ele pode ser acessado por meio da tecla "F12", ou por "Ctrl + Shift + C".

No depurador, há uma aba chamada "Console", localizada na barra de menu superior. É possível o abrirmos acessando o menu "Visualizar" também, na barra de menu superior, selecionando "Desenvolvedor > Console JavaScript".



Há uma mensagem indicando que em nosso teste, na linha 9, há um erro de sintaxe:

Uncaught SyntaxError: Unexpected string

String é um nome bastante comum para texto, dados do tipo "texto", em programação. Portanto, temos um texto não esperado. Retornaremos ao editor de texto no ponto indicado pelo console, isto é, a linha 9, onde se localiza o erro. Toda linguagem de programação de qualidade possui algum mecanismo capaz de indicar onde há erros de sintaxe em seu código.

Feito isso, retornaremos os parênteses de onde os tiramos:

<meta charset="UTF-8">

<h1>Meu primeiro teste!</h1>

<br>

Seria isso um programa? Descubra visitando a Alura <a href="http://www.alura.com.br">aqui</a>!

<script>

alert("Isso sim é um programa");

</script>

Salvaremos, retornaremos ao navegador e recarregaremos a página. Vimos que sumiu a mensagem de erro e foi exibido uma alerta em pop up.

Os elementos que vimos agora - **o browser, o editor de texto, o depurador** - são usados para identificarmos os erros em nosso código e compõem as ferramentas que utilizaremos por todo nosso treinamento.

# Aula 2: Comunique-se com o Usuário

Episódio 1.

Veremos agora como podemos complementar a frase "A idade do Flávio é", exibindo a idade. Podemos inserir na mesma frase, da seguinte forma:

1 <meta charset="UTF-8">

2

3 <script>

4

5 alert("A idade do Flávio é 18");

6

7 </script>

Ao salvarmos o programa e recarregarmos a página, vemos que este método funciona. Alternativamente, podemos criar um segundo alerta para exibir esta informação:

1 <meta charset="UTF-8">

2

3 <script>

4

5 alert("A idade do Flávio é");

6 alert("18");

7

8 </script>

Salvaremos e recarregaremos a página no navegador. Surge um primeiro alerta, com a mensagem "A idade do Flávio é", pressionaremos "OK" e, em seguida, surge um segundo pop up, com a mensagem "18".

Só que, se utilizarmos o alert no mundo JavaScript para exibir um texto para o usuário, e caso tenhamos dez mensagens, quantas vezes o usuário terá de pressionar "OK"? Várias. Isso tornará a experiência do visitante ruim. Se ele quiser ler a última linha, não conseguirá, porque surgirão muitos alertas seguidos.

O que fazer? O mundo JavaScript é representado por tudo que está inserido na tag <script>. Todo o conteúdo desta tag é interpretado pelo navegador como JavaScript. O mundo HTML é compreendido como tudo que está **fora** da tag <script>. O que faremos é escrever, a partir do mundo JS, um HTML. Para isto, adicionaremos document.write():

<script>

document.write()

</script>

A palavra document significa, em português, "documento". Nossa página HTML, nosso programa, é um documento. Já a palavra write corresponde à palavra "escrever". O write recebe parênteses ao final, ou seja, ele aceita receber algo para escrever no documento. Como vimos, o texto no JavaScript é incluído entre aspas.

O texto é chamado de string, tanto em JS como em outras linguagens de programação. Escreveremos a seguinte mensagem: "A idade do Flávio é"'

<meta charset="UTF-8">

<script>

document.write("A idade do Flávio é");

</script>

Salvaremos, retornaremos ao navegador e recarregaremos a página. No browser, veremos a mensagem que acabamos de escrever:

A idade do Flávio é

Inseriremos uma segunda instrução document:

<meta charset="UTF-8">

<script>

document.write("A idade do Flávio é");

document.write("18");

</script>

Salvaremos utilizando o atalho "Ctrl + S", e retornaremos ao navegador, recarregando a página. Surgirá a seguinte mensagem:

A idade do Flávio é 18

Alguém que está começando a programar pode se perguntar: há muitas informações escritas em HTML no mundo JavaScript, que poderiam ter sido escritas de forma mais simples, diretamente, apenas com a primeira linguagem?

Vamos simular a seguinte situação, em que eliminaremos o document.write, pois ele só é pertinente ao JavaScript:

<meta charset="UTF-8">

A idade do Flávio é

18

<script>

</script>

Será que obteremos o mesmo resultado?

Retornaremos ao navegador e recarregaremos a página, após o qual teremos a mesma mensagem:

A idade do Flávio é 18

Por que então utilizar o document.write() para, do mundo JavaScript, escrever em HTML? Porque o JavaScript é uma linguagem dinâmica. Podemos, eventualmente, imprimir estas informações múltiplas vezes. Apesar de escrevermos mais, no JavaScript podemos escrever um código dinâmico, mesmo em HTML.

Há duas formas de escrevermos usando document.write, as quais podemos separar em dois comandos:

<script>

document.write("A idade do Flávio é");

document.write("18");

</script>

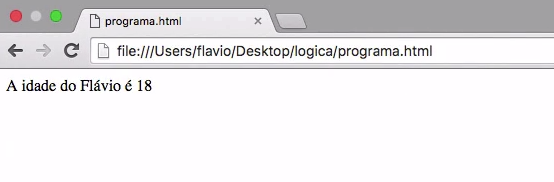
Ou inserir o valor 18 já na primeira frase:

<script>

document.write("A idade do Flávio é 18");

</script>

De ambas as formas, o resultado visual será o mesmo:



Neste momento, o método menos verboso seria inserirmos o valor 18 logo após a frase A idade do Flávio é. Entretanto, isso nem sempre é interessante. Se quisermos pular uma linha, por exemplo, podemos utilizar a tag <br>. Isso porque a frase exibida no navegador está inserida no mundo HTML, e neste contexto, utilizamos esta tag sempre que queremos pular uma linha.

Como estamos escrevendo no mundo HTML, quando o navegador for processar a informação verá o <br> e entenderá que deve pular uma linha. Sendo assim, teremos o seguinte resultado na web:

A idade do Flávio é 18

Poderíamos, alternativamente, ter escrito da seguinte forma:

<script>

document.write("A idade do Flávio é");

document.write("18");

document.write("<br>");

</script>

Salvaremos, retornaremos ao navegador, e recarregaremos a página. Obtivemos o mesmo resultado:

A idade do Flávio é 18

Portanto, na programação há diversas formas de se atingir um mesmo objetivo. Em algumas delas, escreveremos mais código, enquanto outras são mais diretas e simples. A forma a ser utilizada dependerá do tipo de problema que você está tentando resolver.

O que aconteceria se escrevêssemos o número 18 entre parênteses, sem as aspas?

<script>

document.write("A idade do Flávio é ");

document.write(18);

</script>

O JavaScript só considera algo como texto se estiver entre aspas, portanto, da forma como escrevemos 18, o número será interpretado como texto? Não. Como sua representação é numérica, ele será interpretado como um número.

Salvaremos, recarregaremos a página, e teremos o mesmo resultado na web:

A idade do Flávio é 18

Portanto, o document.write aceita trabalhar com tipo texto - "tipo" indica que é um tipo de dado -, cujo termo técnico é string, e "18", que é um número. Minha idade real não é 18, entretanto, os números nos permitem realizar operações. Por exemplo, o símbolo **+** representa soma, assim, podemos somar "18 + 20":

<script>

document.write("A idade do Flávio é ");

document.write(18 + 20);

</script>

Salvaremos, e recarregaremos a página. Vemos que a mensagem mudou, e agora temos:

A idade do Flávio é 38

Portanto, o número nos permite realizar operações matemáticas, por exemplo, podemos utilizar o símbolo **\*** para realizar multiplicações:

<script>

document.write("A idade do Flávio é ");

document.write(18 \* 20);

</script>

Com isto, obteremos o seguinte resultado:

A idade do Flávio é 360

Podemos realizar também uma operação de divisão, utilizando o símbolo **/** (barra):

<script>

document.write("A idade do Flávio é ");

document.write(18 / 2);

</script>

E nosso resultado, ao salvarmos o programa e recarregarmos a página, será:

A idade do Flávio é 9

Poderemos optar por uma subtração, usando o símbolo **-** (hífen):

<script>

document.write("A idade do Flávio é ");

document.write(18 - 2);

</script>

Cujo resultado será:

A idade do Flávio é 16.

O que acontece se colocarmos o número 18 entre aspas?

<script>

document.write("A idade do Flávio é ");

document.write("18");

</script>

O JavaScript entenderá isso como um número, ou um texto? Ele interpretará como um texto, porque está entre aspas. Faremos uma operação de adição, onde ambos os números serão colocados entre aspas:

<script>

document.write("A idade do Flávio é ");

document.write("18" + "20");

</script>

Cada número é interpretado como uma string, sendo assim, quando será a soma de "18" + "20"? Salvaremos o programa, retornaremos à página e a recarregaremos. Temos o seguinte resultado:

A idade do Flávio é 1820

Quando temos uma operação de soma envolvendo texto, o JavaScript não entende se tratar de um número, e portanto realiza uma operação à qual damos o nome de **concatenação**. Isto significa que ele juntou um texto ao outro, o que resultou em 1820. Este é então passado para o write, que o imprime desta forma.

Temos o comando document.write, e precisamos passar para ele o que desejamos imprimir. Da forma como escrevemos, ele entenderá que queremos imprimir a **concatenação** do número 18 com 20. Portanto, a primeira operação realizada pelo JavaScript é unir estes dois números para, em seguida, informar ao write que ele deve imprimir.

O que acontece se tirarmos as aspas em apenas um dos números?

document.write("18" + 20);

Estamos tentando somar um texto com um número. Uma string somada a outra resulta em uma concatenação, um número somado a outro resulta na soma numérica dos dois, mas quando temos uma string a ser somada com um número, se testarmos salvando e recarregando a página, teremos:

A idade do Flávio é 1820

Aconteceu que o JavaScript, ao receber um texto e um número, converte este último para texto, resultando em 1820. Ou seja, quando há texto e número, ele converte tudo para texto.

Episódio 2.

Faremos mais testes de operações, desta vez subtraindo 2016 - 39:

<meta charset="UTF-8">

<script>

document.write("A idade do Flávio é");

document.write(2016 - 39);

</script>

Salvaremos o código e recarregaremos a página. Vamos obter o seguinte resultado:

A idade do Flávio é 1977

Entretanto, este número não representa a idade, e sim o ano de nascimento do Flávio. O programa deve ser coerente, por isso alteraremos a mensagem:

<script>

document.write("Flávio nasceu em")

document.write(2016 - 39);

</script>

Salvaremos, e recarregaremos a página. Desta vez o browser exibe:

Flávio nasceu em 1977

É preciso ter cuidado, a cada alteração feita deve-se salvar o programa, para depois recarregá-lo no navegador. Pode acontecer de esquecermos de salvar o arquivo, ou de recarregar, e não visualizarmos a versão mais recente.

Feito isso, trabalharemos agora com outro tipo de problema para aprofundarmos nosso entendimento. O que acontecerá se incluirmos a operação na mesma frase de Flávio nasceu em?

document.write("Flávio nasceu em 2016 - 39");

Teremos um texto, e a exibição será a seguinte:

Flávio nasceu em 2016 - 39

O que acontece se colocarmos a frase em uma string separada da operação numérica?

document.write("Flávio nasceu em" + "2016 - 39")

Ao salvarmos e recarregarmos a página, teremos o mesmo resultado:

Flávio nasceu em 2016 - 39

Faremos uma nova operação para descobrirmos as médias entre três idades, para isso, deveremos somá-las e dividir por 3:

document.write("A média das idades é");

document.write(20 + 10 + 30/3);

Com isso, esperamos que o resultado seja 20. Salvaremos o programa, e recarregaremos a página no navegador. Tivemos o seguinte resultado:

A média das idades é 40

Este não é o resultado correto, que deveria ser 20.

A programação, assim como na matemática, prioriza a avaliação das multiplicações e divisões. No caso acima, o JavaScript primeiro realiza a divisão de 30/3, resultando em 10 para, em seguida, somar aos demais 20 + 10 + 10, resultando em 40.

Sendo assim, se quisermos indicar que algo deve ser avaliado primeiro, devemos inseri-lo entre parênteses - sem confundir com os parênteses que seguem o comando write. Antes da operação de divisão, queremos que o programa realize a somatória de 20 + 10 + 30, portanto, é esse trecho que isolaremos:

document.write((20 + 10 + 30)/3);

Salvaremos, e recarregaremos o navegador. Obtivemos o resultado correto:

A média das idades é 20

Vamos supor que temos a seguinte situação:

document.write("A soma das idades é" + 20 + 10 + 30);

Como não há nenhum parênteses que indique prioridade, a leitura será feita normalmente, da esquerda para a direita. Por termos string e números, será feita uma **concatenação**, assim, o resultado será:

A média das idades é 201030

Não é este nosso objetivo. Podemos salvar e recarregar a página para confirmar o resultado. Queremos que a operação de soma seja realizada primeiro, assim, recorreremos aos parênteses:

document.write("A soma das idades é" + (20 + 10 + 30));

Lembrando que devemos sempre ter o mesmo número de parênteses de um lado e de outro. Salvaremos e retornaremos ao navegador. Obtivemos o seguinte resultado:

A soma das idades é 60

Se quisermos calcular a média, manteremos a soma como está, e acrescentaremos a divisão por três (/3):

document.write("A soma das idades é" + (20 + 10 + 30)/3);

A operação que está entre parênteses será considerada primeiro. Em seguida, ele fará a divisão, e o resultado disso será concatenado com a frase A soma das idades é. Teremos como resultado final, portanto:

A soma das idades é 20

Trabalharemos a seguir com a operação:

document.write("Flávio nasceu em" + 2016 - 39);

Em seguida, isolaremos a operação matemática entre parênteses:

document.write("Flávio nasceu em" + (2016 - 39));

Dessa forma, primeiramente, será calculado o resultado de 2016 - 39, para então seguir às demais informações. Inseriremos mais um comando, indicando a idade de Joaquim:

document.write("Flávio nasceu em" + (2016 - 39));

document.write("Joaquim nasceu em" + (2016 - 20));

E outro, com a idade de Barney:

document.write("Flávio nasceu em" + (2016 - 39));

document.write("Joaquim nasceu em" + (2016 - 20));

document.write("Barney nasceu em" + (2016 - 40));

O resultado esperado é o ano de nascimento de cada um, resultado das respectivas subtrações. Salvaremos o programa, retornaremos ao navegador e recarregaremos a página. Obtivemos o seguinte resultado:

Flávio nasceu em 1977Joaquim nasceu em 1996Barney nasceu em 1976

Os valores estão corretos, entretanto, faremos com que sejam exibidos separadamente por linha. Para isso, podemos concatenar um <br>:

document.write("Flávio nasceu em" + (2016 - 39) + "<br>");

document.write("Joaquim nasceu em" + (2016 - 20) + "<br>");

document.write("Barney nasceu em" + (2016 - 40) + "<br>");

A primeira coisa que o JavaScript fará é identificar a presença de parênteses, que indiquem uma operação a ser priorizada. No caso, temos as subtrações referentes aos anos de nascimento.

Feito isso, a análise ocorrerá normalmente, da esquerda para a direita. Como temos texto e número, haverá a concatenação destes, resultando na frase Flavio nasceu em 1977, por exemplo. Por fim, será concatenado o <br>, já que o comando está inserido no código HTML, resultando na quebra de linha entre uma frase e a seguinte.

Salvaremos o programa, retornaremos ao navegador e carregaremos a página, com o seguinte resultado:

Flávio nasceu em 1977

Joaquim nasceu em 1996

Barney nasceu em 1976

Alternativamente, podemos obter o mesmo resultado escrevendo o código da seguinte forma:

document.write("Flávio nasceu em " + (2016 - 39));

document.write("<br>");

document.write("Joaquim nasceu em " + (2016 - 20));

document.write("<br>");

document.write("Barney nasceu em " + (2016 - 40));

document.write("<br>");

A forma de escrita fica a critério do programador, levando-se em consideração a facilitação do entendimento por parte do leitor.

# Aula 3: Torne seu programa dinâmico com Variaveis

## Episódio 1.

Uma variável pode guardar praticamente o que você quiser: um número, uma string ou outro pedaço de código. Sabemos que é possível calcularmos a média das idades usando a frase "A média das idades é" e concatenando com a soma das idades dividida pelo número de indivíduos.

document.write('A media das idades é ' + (39 + 20 + 41)/3);

No código, teremos o seguinte:

<meta charset="UTF-8">

<script>

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write("<br>");

document.write("Joaquim tem " + (ano - 1996) + " anos");

document.write("<br>");

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

document.write("<br>");

document.write('A média das idades é ' + (39 + 20 + 41)/3);

</script>

Salvaremos e recarregaremos a página, e obteremos o seguinte resultado:

Flávio tem 39 anos

Joaquim tem 20 anos

Barney tem 41 anos

A média das idades é 33,333333333333336

Será que há um método para que seja feita a divisão em uma variável, antes de ser inserida em write()? Por exemplo:

document.write('A média das idades é ' + media);

Sim, basta declararmos a variável media que contenha justamente a média das idades (a soma das idades dividida por 3):

var media = (39 + 20 + 41)/3;

Com isso, podemos fazer a substituição da operação pela variável, como no exemplo a seguir:

var media = (39 + 20 + 41)/3;

document.write('A média das idades é ' + media);

O resultado é o mesmo, porém mais legível, pois conseguimos diminuir a quantidade de parênteses em um único comando. Declaramos uma nova variável, media, e precisamos incluir o prefixo var, assim como que para qualquer outra variável, em seu primeiro uso deve ser antecedida do prefixo var. Se quisermos alterar seu valor posteriormente, não há necessidade de utilizarmos o prefixo novamente.

Salvaremos o programa e recarregaremos o navegador. Obtivemos o mesmo resultado. A seguir, veremos como poderemos arredondar números com muitas casas decimais. Para isto, utilizamos o document, só que, para arrendondarmos, não trabalharemos com o documento, e sim com a matemática, portanto é este o comando que chamaremos. Em seguida, chamaremos a função round (do inglês, "arredondar"):

Math.round()

Como parâmetro, passaremos a media:

var media = (39 + 20 + 41)/3;

document.write('A média das idades é ' + Math.round(media));

Com o conteúdo entre parênteses de document.write(), definimos o que será exibido na tela, e com Math.round definiremos qual valor será arredondado. Salvaremos o programa, retornaremos ao navegador e recarregaremos a página. Temos o seguinte resultado:

Flávio tem 39 anos

Joaquim tem 20 anos

Barney tem 41 anos

A média das idades é 33

Aprendemos portanto a arredondar um número; há muitos mecanismos por trás desta operação, que desvendaremos adiante. Ao observarmos o código, vemos que não é evidente qual idade corresponde a qual indivíduo. Na operação 39 + 20 + 41 não é possível inferir qual idade pertence a quem.

Para eliminarmos esta dificuldade, criaremos as seguintes variáveis:

var idadeFlavio = 39;

var idadeJoaquim = 20;

var idadeBarney = 41;

Em vez de utilizarmos os números, nossa operação ficará da seguinte forma:

var idadeFlavio = 39;

var idadeJoaquim = 20;

var idadeBarney = 41;

var media = (idadeFlavio + idadeJoaquim + idadeBarney)/3;

document.write('A média das idades é ' + Math.round(media));

Assim, fica claro que a média é referente aos valores das idades de Flávio, Joaquim e Barney. Neste caso, a variável melhora a legibilidade do código, além de facilitar sua manutenção.

Uma convenção importante: o nome de uma variável sempre se iniciará com uma letra **minúscula**. Se a variável contém duas palavras, por exemplo "idade + (nome)", a próxima palavra deve ser escrita com a primeira letra **maiúscula**. Este padrão recebe o nome de **camelCase**.

E se quisermos escrever a variável inteira em letras minúsculas, mesmo que sejam duas palavras? O seu código funcionará da mesma forma, o nome da variável apenas não vai seguir a convenção do mundo da programação. Para esclarecer que as variáveis não podem guardar somente números, criaremos o seguinte trecho:

var nome = 'Flávio';

document.write('A idade de ' + nome + ' é ' + idadeFlavio);

No JavaScript, podemos utilizar tanto as aspas simples (**'**) quanto aspas (**"**). Colocaremos as aspas duplas, mesmo porque outras linguagens de programação as utilizam.

var nome = "Flávio";

document.write("A idade de " + nome + " é " + idadeFlavio);

O comando document.write tem como parâmetro a string A idade de, concatenada com a variável nome. O JavaScript verificará qual o conteúdo da var nome, no caso, Flávio. A junção destas duas resultara em uma única string. O resultado final desta primeira parte da concatenação será:

"A idade de Flávio ".

Somaremos a isto a string "e" e, como sabemos, se somarmos duas strings teremos uma única, representando a união destas. Assim, o resultado será:

"A idade de Flávio é".

Por fim, será somado à string a variável idadeFlavio, cujo valor é um número. Como string combinado com número resulta em outra string, teremos:

"A idade de Flávio é 39.

Sendo assim, quando o comando abaixo for executado,

document.write("A idade de " + nome + " é " + idadeFlavio);

Teremos o seguinte resultado:

A idade de Flávio é 39

Salvaremos o programa, retornaremos ao navegador e recarregaremos a página. Obtivemos o seguinte resultado:

Flávio tem 39 anos

Joaquim tem 20 anos

Barney tem 41 anos

A média das idades é 33A idade de Flávio é 39

Como vemos, o texto ficou agrupado ao resultado da média das idades. Podemos incluir a tag <br> para pular uma linha:

document.write("<br>A idade de " + nome + " é " + idadeFlavio);

Ao salvarmos e recarregarmos a página, teremos o seguinte resultado:

Flávio tem 39 anos

Joaquim tem 20 anos

Barney tem 41 anos

A média das idades é 33

A idade de Flávio é 39

Deu certo! Como escrevemos a string no código HTML, o navegador processará o pulo de linha, para então exibir o texto. Alternativamente, poderíamos ter criado o mesmo resultado com o seguinte código:

<meta charset="UTF-8">

<script>

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write("<br>");

document.write("Joaquim tem " + (ano - 1996) + " anos");

document.write("<br>");

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

document.write("<br>");

var idadeFlavio = 39;

var idadeJoaquim = 20;

var idadeBarney = 41;

var media = (idadeFlavio + idadeJoaquim + idadeBarney)/3;

document.write("A média das idades é " + Math.round(media));

var nome = "Flávio";

document.write("<br>");

document.write("A idade de " + nome + " é " + idadeFlavio);

</script>

Salvaremos a página, retornaremos ao navegador e recarregaremos a página. O resultado permanece inalterado.

Episódio 2.

Aprendemos a trabalhar com variáveis, que nos facilitam a manutenção de nosso programa, e o tornam mais dinâmico.

Por exemplo, para o ano, em vez de termos o número fixo no comando, cujas alterações posteriores teriam que ser feitas manualmente em todos os locais em que ele aparece, declaramos uma variável que guarda um valor, no caso 2016. Para isso, utilizamos o prefixo var, abreviação de variable, em seguida nomeamos a variável como ano. Além disso, utilizamos o sinal de **=**, que lemos como "recebe". Portanto, a variável ano recebe 2016:

var ano = 2016

Ao lermos esta instrução, fica claro que a variável ano recebe 2016. Assim, em todo o lugar em que esta variável for acessada, o JS "pegará" o seu valor naquele momento. Quando o código do trecho abaixo for executado, a primeira linha a ser processada é a declaração da variável. Em seguida, ele imprimirá a linha do document.write() com a idade do Flávio, que é uma série de concatenações:

<meta charset="UTF-8">

<script>

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write("<br>");

document.write("Joaquim tem " + (ano - 1996) + " anos" );

document.write("<br>");

//...

No momento em que o código for executado, será analisado o valor da variável ano até este ponto, no caso, 2016. Este valor será impresso até atingir a linha em que a variável ano recebe o valor 2017. Por definição, os valores da variável mudam, e podem ser alterados em diversos pontos ao longo do programa. Portanto, a partir da nova instrução, o valor de ano será 2017:

<meta charset="UTF-8">

<script>

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write("<br>");

document.write("Joaquim tem " + (ano - 1976) + " anos");

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

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

document.write("<br>");

//...

Não há influência sobre as instruções anteriores porque elas já foram processadas, com o valor de ano de 2016. O marco de mudança do valor da variável em nosso programa é na linha ano = 2017. Vimos também que é possível realizar operações matemáticas, como adições e multiplicações, desde que elas guardem um número. Podemos guardar em uma variável, por exemplo, informações de idade:

var idadeFlavio = 39;

var idadeJoaquim = 20;

var idadeBarney = 41;

Quando compreendemos rapidamente a que o nome de uma variável se refere, sendo simples entender quais valores ela guarda, facilitamos o trabalho dos próximos programadores que alterarão o nosso código. E se tivéssemos colocado apenas uma letra como o nome da variável? Por exemplo:

var x = 39;

var y = 20;

var z = 41;

Seria difícil identificar o que cada número significa, por isso a importância de conferirmos um nome e deixar claro que tipo de valor aquela determinada variável armazena. Inclusive, podemos realizar operações matemáticas utilizando os nomes das variáveis, e guardar seu resultado em outra variável:

var idadeFlavio = 39;

var idadeJoaquim = 20;

var idadeBarney = 41;

var media = (idadeFlavio + idadeJoaquim + idadeBarney)/3;

Uma variável não guarda só um tipo texto (string), mas também um tipo número, e outros que aprenderemos no decorrer do curso. Não deixem de fazer os exercícios, e continuaremos a seguir!

# Aula 4: Crie suas próprias funcionalidades

Episódio 1.

Vamos continuar elaborando o programa. Antes, vamos remover a parte do código que não será mais necessária. O arquivo programa.html ficou com o seguinte conteúdo:

<meta charset="UTF-8">

<script>

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write("<br>");

document.write("Joaquim tem " + (ano - 1996) + " anos");

document.write("<br>");

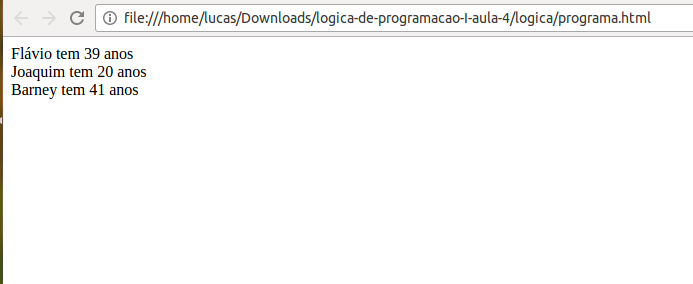
ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

document.write("<br>");

</script>

Quando utilizamos o document.write() para inserir a tag <br>, isso representa uma quebra de linha. No código acima, quantas linhas estamos pulando de uma frase até a outra? A resposta é: uma. Escrevemos uma frase e, na sequência, saltamos uma linha:



Agora, nosso chefe fez um novo pedido, ele quer que existam duas linhas de intervalo entre cada frase, pois acredita que assim as linhas ficarão mais espaçadas. Como podemos solucionar o problema? Uma opção seria adicionar mais uma linha de código com <br>:

<meta charset="UTF-8">

<script>

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write("<br>");

document.write("<br>");

document.write("Joaquim tem " + (ano - 1996) + " anos");

document.write("<br>");

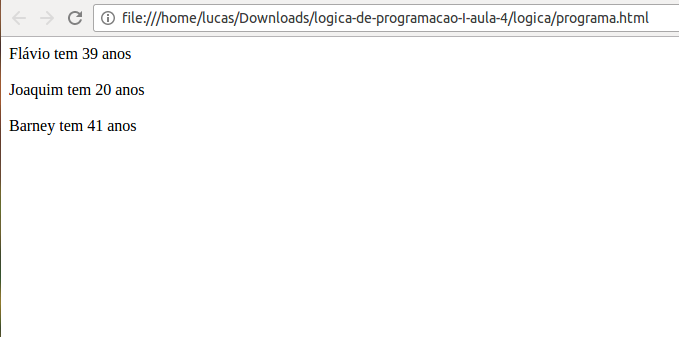
document.write("<br>");

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

Note que na última instrução document.write() não é necessário pular uma linha, porque ela está no final. A apresentação ficará da seguinte maneira:



Se o chefe pedir mais linhas, basta ir adicionando mais instruções com document.write():

<meta charset="UTF-8">

<script>

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write("<br>");

document.write("<br>");

document.write("<br>");

document.write("Joaquim tem " + (ano - 1996) + " anos");

document.write("<br>");

document.write("<br>");

document.write("<br>");

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

E se o chefe muda de ideia toda hora? Em algum momento ele pode querer voltar atrás, isto é, que o programa pule apenas uma linha:

<meta charset="UTF-8">

<script>

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write("<br>");

document.write("Joaquim tem " + (ano - 1996) + " anos");

document.write("<br>");

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

Uma solução que resolve a situação de maneira simples é adicionar a quantidade de linhas que queremos pular em uma mesma instrução. Desta forma, acrescentaremos um número de <br> equivalente à quantidade de linhas que desejamos saltar:

<meta charset="UTF-8">

<script>

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

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

document.write("Joaquim tem " + (ano - 1996) + " anos");

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

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

Porém, quando o chefe pedir novas alterações quanto ao intervalo de linhas, teremos que fazer essa alteração em todos os lugares nos quais isto ocorre! Isto vai dificultando cada vez mais a manutenção do código. Se tivermos trinta pulos de linha, algo comum em um programa, será necessário fazer alterações em n lugares diferentes.

Já vimos que as variáveis podem nos blindar desse tipo de situação. Abaixo da variável ano, criaremos a variável puloLinha, a que adicionaremos o "<br>":

var puloLinha = "<br>";

No lugar de passarmos o "<br>" para o document.write(), poderemos passar direto a seguinte variável:

document.write(puloLinha);

Na hora em que o document.write() for ser acessado, ele irá pegar o conteúdo da variável, a tag <br>, como mostra o código abaixo:

<meta charset="UTF-8">

<script>

var ano = 2016;

var puloLinha = "<br>";

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write(puloLinha);

document.write("Joaquim tem " + (ano - 1996) + " anos");

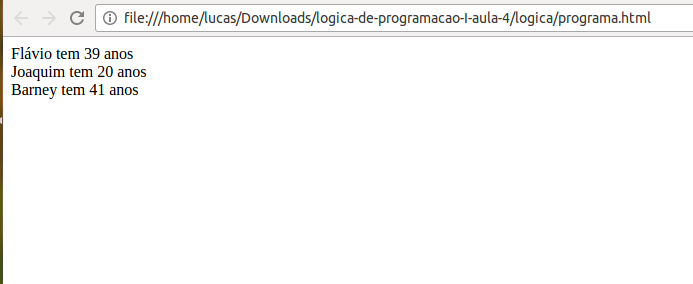
document.write(puloLinha);

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

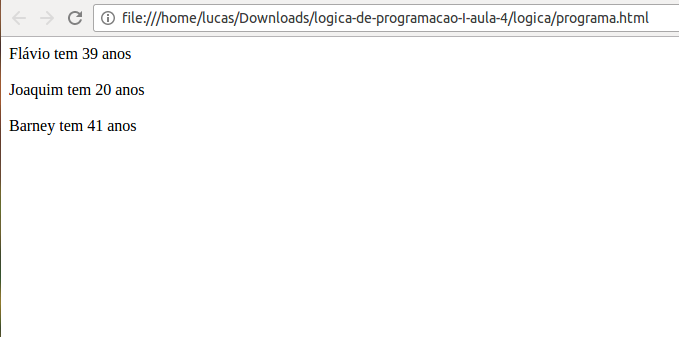
É o mesmo procedimento de quando pulamos apenas uma linha:



A partir de agora, caso seu chefe tenha interesse em pular linhas, o código pode ser ajustado em um único lugar. Imagine que ele tenha exigido que o intervalo seja de duas linhas. Para resolver esse problema faremos a seguinte modificação:

var puloLinha = "<br><br>";

E automaticamente isso muda a quantidade de linhas vazias em todo o código:



Com isso, aprendemos que o uso de uma variável nos permite facilitar a manutenção do código!

Episódio 2.

Até aqui, resolvemos um problema de manutenção. Sempre que quisermos pular uma linha, mesmo declarando a variável puloLinha, usaremos document.write(puloLinha). Isso é melhor em relação ao modelo anterior pois puloLinha pode guardar um pulo com várias linhas.

Mas se pedirmos para uma pessoa sem conhecimento em Programação nos dizer o significado da instrução document.write(), ela provavelmente não entenderá do que se trata. Isto ocorre por conta da falta clareza no código.

Além disso, é tedioso para os programadores escreverem diversas vezes a mesma instrução. A utilidade de document.write() consiste em escrever na tela aquilo que passamos como parâmetro entre os parênteses. Por exemplo, se queremos dividir 3 por 4:

document.write(3/4);

O resultado não será um número exato, e para arredondarmos, podemos utilizar o Math.round():

document.write(Math.round(3/4));

O objetivo é bem definido, e queremos arredondar o resultado da operação. Portanto, vemos que o próprio alert() tem a função delimitada, que é justamente exibi-lo na tela. Estas funcionalidades já existem no JavaScript. E se pudéssemos criar nossa própria funcionalidade, algo que tenha uma **função** bem definida? Em JavaScript ou em qualquer outra linguagem de programação, há meios para criarmos nossas próprias funcionalidades.

A seguir, eliminaremos a variável puloLinha, e excluiremos também as declarações de document.write(puloLinha):

<meta charset="UTF-8">

<script>

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

?

document.write("Joaquim tem " + (ano - 1996) + " anos");

?

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

Supondo que estamos seguindo um texto de um livro, faria sentido se o conteúdo fosse o seguinte?

document.write("Flávio tem " + (ano - 1977) + " anos");

pulaLinha

document.write("Joaquim tem " + (ano - 1996) + " anos");

pulaLinha

O texto está escrito em português, se chamarmos um colega para ler a instrução, ele entenderá que a função de pulaLinha é de pular uma linha. O JavaScript nos permite fazer essa tarefa, criar nossas funcionalidades, ou seja, as próprias funções.

No entanto, há uma peculiaridade - quando queremos executar o document.write, temos que inserir o parâmetro entre parênteses, e no caso do pulaLinha, não haverá nenhum, então a funcionalidade simplesmente será executada. De qualquer forma, incluiremos os parênteses:

document.write("Flávio tem " + (ano - 1977) + " anos");

pulaLinha();

document.write("Joaquim tem " + (ano - 1996) + " anos");

pulaLinha();

Por que isso? Retornaremos ao console do navegador, e na barra de menu superior acessaremos "Visualizar > Desenvolvedor > Console JavaScript", em que digitaremos o seguinte comando:

alert();

Pressionaremos a tecla "Enter", o que fará com que surja um alerta em forma de pop up, apesar de não termos inserido nenhum parâmetro. Isso porque se trata de uma funcionalidade já existente no JavaScript. Entretanto, para que ela pudesse ser executada, tivemos de utilizar os parênteses. Ela normalmente recebe um parâmetro, a mensagem que queremos exibir:

alert("olá");

Com isto temos o seguinte pop up:

Essa página diz:

Olá

No caso, nossa função pulaLinha() não receberá nada, sendo simplesmente declarada. Sabemos que ela na verdade terá a linha com document.write(), utilizada anteriormente:

document.write("<br>");

Como criamos uma função no JS? Convencionaremos que todas as funções que criarmos estarão nas primeiras instruções da tag <script>, ou seja, declararemos no início do código. Para declararmos uma função, utilizaremos o termo em inglês *function*, seguido de um nome, no caso, pulaLinha:

<meta charset="UTF-8">

<script>

function pulaLinha()

//...

</script>

Inserimos os parênteses por ser uma exigência do JavaScript para declararmos uma função. Queremos que esta função escreva document.write("<br>"), portanto faremos isso dentro do bloco da função, entre chaves:

<meta charset="UTF-8">

<script>

function pulaLinha() {

}

//...

</script>

Tudo que estiver entre as chaves pertence à função pulaLinha(). Um ponto importante - a função é sempre descrita por um verbo, seja ele "pula", "grita", "comemora", ou qualquer outro -, isto porque ela sempre indica uma ação a ser executada. Como document.write() pertencerá à função, adicionaremos-na dentro das chaves:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

}

//...

</script>

Este código é válido, mas sempre dentro de um bloco utilizaremos a tecla "Tab":

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

}

//...

</script>

Nós indentaremos o trecho de código. Trata-se de uma boa prática, uma convenção no mundo da programação - se estamos em um bloco, utilizamos "Tab" para criar um espaçamento na linha e, assim, o código ficará **indentado** e teremos uma melhor noção de hierarquia. Neste caso, saberemos que document.write() pertence à função pulaLinha().

Verificaremos se o código está funcionando. Salvaremos, retornaremos ao navegador e recarregaremos a página. Deu certo! Obtivemos o seguinte resultado, em que foram puladas linhas entre cada frase:

Flávio tem 39 anos

Joaquim tem 20 anos

Barney tem 41 anos

E se quisermos pular mais de uma linha? Podemos inserir mais de uma instrução na função, da seguinte forma:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

//...

</script>

Uma função pode ser um atalho para várias outras instruções, por exemplo, quando pulaLinha() for processado, é como se o JavaScript estivesse substituindo por duas quebras de linha no formato document.write("<br>"), como se passássemos disso:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

pulaLinha();

document.write("Joaquim tem " + (ano - 1996) + " anos");

pulaLinha();

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

Para isso:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write("<br>");

document.write("<br>");

document.write("Joaquim tem " + (ano - 1996) + " anos");

document.write("<br>");

document.write("<br>");

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

Assim, é como se, ao chegar em pulaLinha(), o JavaScript parasse para acessar o bloco da função e executasse as funções contidas nela. Podemos criar nossas próprias funções para deixar nossa programação mais legível, interessante, e mais fácil de se manter.

Podemos reutilizar o pulaLinha() em qualquer lugar do sistema; se quisermos alterar a maneira como esta função é utilizada, isto não afetará o restante do código. Podemos obter o mesmo resultado com apenas uma instrução:

<meta charset="UTF-8">

<script>

function pulaLinha() {

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

}

//...

</script>

Quem chama a função pulaLinha() nem saberá que ela foi alterada, desde que ela funcione da mesma forma. Salvaremos e recarregaremos o navegador, e veremos que o resultado foi o mesmo:

Flávio tem 39 anos

Joaquim tem 20 anos

Barney tem 41 anos

A vantagem de uma função é tornar legível a intenção do que queremos fazer. Se quisermos pular uma linha, por exemplo, criaremos o pulaLinha(), sempre utilizando um verbo e indicando uma ação.

Uma função representa um conjunto de instruções, e é possível ter uma, duas, cem instruções - **não há limites**. Sendo assim, será desnecessário reescrevê-las em diversos pontos no código, basta utilizarmos a função criada.

Porém, é preciso ter cuidado, pois se esquecermos de abrir e fechar os parênteses...

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

pulaLinha;

document.write("Joaquim tem " + (ano - 1996) + " anos");

pulaLinha;

Ao executarmos o código acima, teremos o seguinte resultado:

Flávio tem 39 anosJoaquim tem 20 anosBarney tem 41 anos

As linhas não foram puladas. Se abrirmos o console, veremos que nada acontece. Isso porque, para executar a função, é necessária a utilização dos parênteses. No JavaScript a função permanece guardada, isso significa que não importa quantos comandos houverem dentro dela, eles só serão executados uma vez que forem acionados.

Se, incluirmos um alerta em nossa função, e removermos os chamados pulaLinha():

<meta charset="UTF-8">

<script>

function pulaLinha() {

alert("oi");

document.write("<br>");

document.write("<br>");

}

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

document.write("Joaquim tem " + (ano - 1996) + " anos");

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

Temos três instruções. Ao salvarmos e recarregarmos a página, o alerta não será exibido, tampouco há quebras de linha. Isso ocorre pois a função não foi chamada. Se desta vez inserirmos pulaLinha() usando parênteses:

<meta charset="UTF-8">

<script>

function pulaLinha() {

alert("oi");

document.write("<br>");

document.write("<br>");

}

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

pulaLinha();

document.write("Joaquim tem " + (ano - 1996) + " anos");

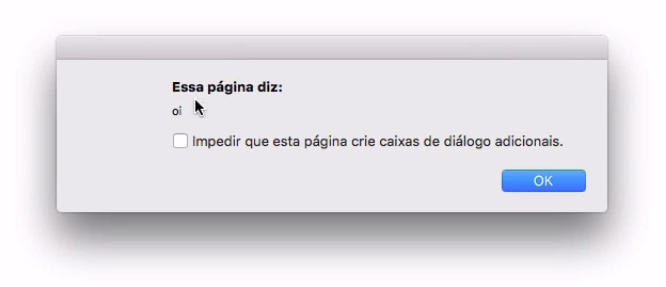
pulaLinha();

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

Todas as instruções serão executadas, o alerta e os dois pulos de linha. Ao recarregarmos o navegador, exibe-se o pop up com a mensagem oi.



Enquanto isso, no navegador veremos:

Flávio tem 39 anos

Joaquim tem 20 anos

Barney tem 41 anos

Entre as linhas, há espaços em branco. Como chamamos a função duas vezes, o alerta será exibido novamente. Removeremos o alerta, e agora entendemos que a função serve como um atalho para as instruções que queremos executar no programa.

E se chamarmos a função de x?

<meta charset="UTF-8">

<script>

function x() {

document.write("<br>");

document.write("<br>");

}

var ano = 2016;

document.write("Flávio tem " + (ano - 1977) + " anos");

x();

document.write("Joaquim tem " + (ano - 1996) + " anos");

x();

ano = 2017;

document.write("Barney tem " + (ano - 1976) + " anos");

</script>

O programa continuará funcionando, entretanto, ao visualizarmos a função x(), não temos como saber do que se trata. O nome de uma função é tão importante quanto o código que ela executa, para deixar claro o que ela faz.

Com isso, concluímos a criação de nossa primeira função, que não recebe parâmetro. O document.write() aceita receber parâmetros para, internamente, imprimir na tela aquilo que lhe é passado. Já o pulaLinha() ficou vazio e não está preparado para lidar com isso.

Nosso código pode ser melhorado ainda, como veremos adiante.

# Aula 5: Pratique Resolvendo problemas do seu dia-a-dia

Episódio 1.

insira seu código aqui

Começando deste ponto? Você pode fazer o [DOWNLOAD](https://github.com/alura-cursos/logica-de-programacao-I/archive/aula-5.zip) do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Nesta parte do curso aprenderemos a criar programas com mais utilidades, e veremos como é possível calcular nosso **IMC (Índice de Massa Corporal)**. No menu superior, selecionaremos "File > Save As...", e escolheremos outro nome, já que se trata de um novo programa. Chamaremos de imc.html.

Manteremos para este programa somente as funções pulaLinha() e mostra():

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

</script>

Todo programa que criarmos a partir de agora terá esta estrutura mínima, para podermos aproveitar o que já foi feito. O cálculo do IMC, segundo a Organização Mundial da Saúde, é composto pelo peso divido pela altura multiplicado pela altura. Portanto, criaremos uma variável chamada pesoFlavio, e outra, alturaFlavio:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var pesoFlavio = 73;

var alturaFlavio = 1.71;

</script>

Partiremos agora para o cálculo do IMC. Criaremos uma variável chamada imcFlavio, que receberá a divisão entre pesoFlavio e alturaFlavio multiplicada pela alturaFlavio:

var pesoFlavio = 73;

var alturaFlavio = 1.71;

var imcFlavio = pesoFlavio / alturaFlavio \* alturaFlavio

Se deixarmos como está, teremos um problema. Isso porque ele considerará a divisão antes de realizar a multiplicação. Para evitarmos isso, isolaremos a operação de multiplicação com parênteses:

var pesoFlavio = 73;

var alturaFlavio = 1.71;

var imcFlavio = pesoFlavio / (alturaFlavio \* alturaFlavio);

Com o resultado do IMC, utilizaremos a função mostra() para imprimi-lo na tela:

var pesoFlavio = 73;

var alturaFlavio = 1.71;

var imcFlavio = pesoFlavio / (alturaFlavio \* alturaFlavio);

mostra("O imc do Flávio é " + imcFlavio);

Salvaremos e retornaremos ao navegador. Abriremos este novo programa que acabou de ser criado, selecionando "Arquivo > Abrir arquivo..." na barra de menu superior, e escolhendo o arquivo imc.html.

Ao ser executado, o programa exibe o seguinte resultado:

O imc do Flávio é 24.9649647925858

Não vamos nos preocupar por enquanto em arredondar este resultado. Queremos calcular também o IMC de um amigo, portanto criaremos uma variável pesoAmigo, e outra, alturaAmigo:

var pesoFlavio = 73;

var alturaFlavio = 1.71;

var imcFlavio = pesoFlavio / (alturaFlavio \* alturaFlavio);

mostra("O imc do Flávio é " + imcFlavio);

var pesoAmigo = 68;

var alturaAmigo = 1.72;

Em seguida, criaremos a variável imcAmigo calculando da mesma forma como fizemos anteriormente:

var pesoFlavio = 73;

var alturaFlavio = 1.71;

var imcFlavio = pesoFlavio / (alturaFlavio \* alturaFlavio);

mostra("O imc do Flávio é " + imcFlavio);

var pesoAmigo = 68;

var alturaAmigo = 1.72;

var imcAmigo = pesoAmigo / (alturaAmigo \* alturaAmigo);

Por fim, utilizaremos a função mostra() para que esta informação seja impressa:

var pesoFlavio = 73;

var alturaFlavio = 1.71;

var imcFlavio = pesoFlavio / (alturaFlavio \* alturaFlavio);

mostra("O imc do Flávio é " + imcFlavio);

var pesoAmigo = 68;

var alturaAmigo = 1.72;

var imcAmigo = pesoAmigo / (alturaAmigo \* alturaAmigo);

mostra("O imc do meu amigo é " + imcAmigo);

Salvaremos, retornaremos ao navegador e recarregaremos a página. Obteremos o seguinte resultado:

O imc do Flávio é 24.9649647925858

O imc do meu amigo é 22.985397512168742

O que significam estes IMCs? São resultados bons ou ruins? Veremos isso adiante. Precisamos observar que, tanto no cálculo do imcFlavio quanto do imcAmigo é utilizada a mesma fórmula matemática, o mesmo cálculo. E se outro programador quiser calcular o IMC, digamos, da sua tia? Serão criadas as variáveis:

var pesoTia = 50;

var alturaTia = 1.62;

Ele terá que saber a fórmula para calcular o IMC, que é o peso dividido pela altura, multiplicada pela altura. Assim, será repetida a aplicação do cálculo de IMC já utilizada por duas vezes no programa.

Suponhamos que o cálculo do IMC é eventualmente alterado pela Organização Mundial da Saúde, assim, teremos que lembrar de todos os lugares em que o cálculo aparece. Resolveremos este problema criando uma função chamada calculaImc():

function calculaImc() {

}

var pesoFlavio = 73;

var alturaFlavio = 1.71;

var imcFlavio = pesoFlavio / (alturaFlavio \* alturaFlavio);

mostra("O imc do Flávio é " + imcFlavio);

var pesoAmigo = 68;

var alturaAmigo = 1.72;

var imcAmigo = pesoAmigo / (alturaAmigo \* alturaAmigo);

mostra("O imc do meu amigo é " + imcAmigo);

Lembrando que, para criar uma função, precisamos utilizar o function mais o nome, os parênteses e as chaves.

Não há necessidade de colocarmos o ponto e vírgula (**;**) após fecharmos a chave pois o JavaScript já entende que final e um bloco indicam o final de uma instrução. Copiaremos a linha que conta com o cálculo e inseriremos no bloco da instrução:

function calculaImc() {

var imcFlavio = pesoFlavio / (alturaFlavio \* alturaFlavio);

}

Como queremos que a mesma função sirva para várias pessoas, em vez de utilizarmos os nomes dos indivíduos nas variáveis, manteremos somente o que cada uma representa:

function calculaImc() {

var imc = peso / (altura\* altura);

}

Em seguida, removeremos as linhas de código em que temos o cálculo do IMC:

function calculaImc() {

var imc = peso / (altura\* altura);

}

var pesoFlavio = 73;

var alturaFlavio = 1.71;

var pesoAmigo = 68;

var alturaAmigo = 1.72;

Em seu lugar, chamaremos a função calculaImc():

function calculaImc() {

var imc = peso / (altura\* altura);

}

var pesoFlavio = 73;

var alturaFlavio = 1.71;

calculaImc();

var pesoAmigo = 68;

var alturaAmigo = 1.72;

calculaImc();

Falta passarmos para a função os valores que ela deverá considerar na hora de fazer os cálculos:

function calculaImc() {

var imc = peso / (altura\* altura);

}

var pesoFlavio = 73;

var alturaFlavio = 1.71;

calculaImc(pesoFlavio, alturaFlavio);

var pesoAmigo = 68;

var alturaAmigo = 1.72;

calculaImc(pesoAmigo, alturaAmigo);

Sabemos que, na programação, ao adotarmos o pesoFlavio, estamos nos referindo ao valor 73, da variável, e o mesmo acontecerá para todos os demais. Para simplificar, inseriremos os números diretamente:

function calculaImc() {

var imc = peso / (altura\* altura);

}

var pesoFlavio = 73;

var alturaFlavio = 1.71;

calculaImc(73, 1.71);

var pesoAmigo = 68;

var alturaAmigo = 1.72;

calculaImc(68, 1.72);

Deste modo, temos que o primeiro calculaImc() recebe como parâmetros, respectivamente, 73 e 1.71, e o mesmo para o segundo calculaImc():

function calculaImc() {

var imc = peso / (altura\* altura);

}

calculaImc(73, 1.71);

calculaImc(68, 1.72);

Salvaremos, retornaremos ao navegador e recarregaremos a página. Veremos que nada é exibido, e isto acontece porque não preparamos a função calculaImc() para receber dois parâmetros. Temos que indicar que os parâmetro serão x, y:

function calculaImc(x, y) {

var imc = peso / (altura\* altura);

}

calculaImc(73, 1.71);

calculaImc(68, 1.72);

Para não nos confundirmos e deixarmos mais clara a informação a qual cada valor corresponde, em vez de x e y, utilizaremos altura e peso:

function calculaImc(altura, peso) {

var imc = peso / (altura\* altura);

}

calculaImc(1.71, 73);

calculaImc(1.72, 68);

Ao chamarmos o calculaImc(), o 1.71 será passado para altura, e 73 para o peso. Assim, o IMC será calculado com base nos valores dos parâmetros. Para que os resultados sejam exibidos, incluiremos a instrução mostra() em nossa função:

function calculaImc(altura, peso) {

var imc = peso / (altura \* altura);

mostra("O imc calculado é " + imc);

}

Salvaremos e recarregaremos a página, e o resultado obtido é exibido na tela:

O imc calculado é 24.9649647925858

O imc calculado é 22.985397512168742

Em vez de espalharmos a fórmula do cálculo do IMC, criamos uma funcionalidade que está preparada - se passarmos primeiro a altura, e depois o peso - para calcular o IMC, e exibir o valor, por meio da função mostra(). Se a fórmula for alterada, o IMC é calculado pelo método antigo, + 1:

function calculaImc(altura, peso) {

var imc = peso / (altura \* altura) +1;

mostra("O imc calculado é " + imc);

}

Só precisaremos alterar a função, não todos os pontos onde ela é chamada. Nesta aula, trabalhamos com conceitos já conhecidos, entretanto estamos lidando com uma função que possui dois parâmetros.

Episódio 2.

Temos a função calculaImc(), que recebe parâmetros, realiza um cálculo, e exibe o resultado do IMC, mas e se quiséssemos somar os IMCs? Posteriormente, avaliaremos com esse resultado se estamos em boas condições de saúde, conforme os critérios da Organização Mundial da Saúde.

Uma das formas de fazermos isso seria, em vez de termos dois parâmetros altura, peso, quatro: altura1, peso1, altura2, peso2, os quais passaríamos em apenas um calculaImc():

function calculaImc(altura1, peso1, altura2, peso2) {

var imc1 = peso1 / (altura1 \* altura1);

mostra("O imc calculado é " + imc);

var imc2= peso2 / (altura2 \* altura2);

}

calculaImc(1.71, 73, 1.72, 68);

Nossa variável, então, seria escrita da seguinte forma, com o mostra() também alterado:

function calculaImc(altura1, peso1, altura2, peso2) {

var imc1 = peso1 / (altura1 \* altura1);

var imc2 = peso2 / (altura2 \* altura2);

var imcTotal = imc1 + imc2;

mostra("A soma dos imc's é " + imcTotal);

}

calculaImc(1.71, 73, 1.72, 68);

Salvaremos e recarregaremos a página. Obtivemos o seguinte resultado:

A soma dos imc's é 47.95034399142732

Deu certo! Mas e se quiséssemos somar o IMC de três indivíduos? Ou mais? Teríamos que alterar o calculoImc() diversas vezes. E se quiséssemos reduzir o número de indivíduos? Removeremos parte dos parâmetros:

function calculaImc(altura1, peso1, altura2, peso2) {

var imc1 = peso1 / (altura1 \* altura1);

var imc2 = peso2 / (altura2 \* altura2);

var imcTotal = imc1 + imc2;

mostra("A soma dos imc's é " + imcTotal);

}

calculaImc(1.71, 73);

E, ao executarmos, obteremos o seguinte resultado:

A soma dos imc's é NaN

Isso significa "not a number", ou seja, "não é um número". Algum elemento da operação ficou indefinido e fez com que a operação não fosse bem sucedida. Sempre que fazemos uma operação matemática com valores não definidos temos este resultado, NaN. Portanto, descartaremos esta solução. Retornaremos ao código no seguinte estágio:

function calculaImc(altura, peso) {

var imc = peso / (altura \* altura);

mostra("O imc calculado é " + imc);

}

calculaImc(1.71, 73);

Não podemos manter em calculaImc() a mensagem "O imc calculado é " fixa, porque assim ele sempre exibirá esta mesma mensagem. Se quisermos exibir outra, teremos inevitavelmente esta, que está fixada:

function calculaImc(altura, peso) {

var imc = peso / (altura \* altura);

}

calculaImc(1.71, 73);

Queremos que a fórmula funcione de forma genérica, para que onde inserimos uma informação de peso e altura, ela nos dê o IMC calculado. Criaremos duas variáveis, a imcFlavio e a imcAmigo, e indicaremos que cada uma delas receberá as respectivas informações de peso e altura de cada um:

function calculaImc(altura, peso) {

var imc = peso / (altura \* altura);

}

var imcFlavio = calculaImc(1.71, 73);

var imcAmigo = calculaImc(1.72, 68);

Desta forma estamos dizendo que a variável do indivíduo receberá o IMC calculado com base em suas informações individuais. Ou seja, ao chamarmos uma função, ela pode nos devolver um valor. Para exibirmos estes valores calculados, utilizaremos a função mostra() para cada uma das informações:

function calculaImc(altura, peso) {

var imc = peso / (altura \* altura);

}

var imcFlavio = calculaImc(1.71, 73);

var imcAmigo = calculaImc(1.72, 68);

mostra(imcFlavio);

mostra(imcAmigo);

Salvaremos e recarregaremos o navegador, e o resultado foi o seguinte:

undefined

undefined

Vamos entender... Nesta instrução,

var imcFlavio = calculaImc(1.71, 73);

Passamos as informações mas não indicamos que ele deve nos devolver o resultado do cálculo. Se, por exemplo, o IMC calculado fosse 50, teríamos a seguinte situação:

var imcFlavio = 50;

A variável recebe a função de calcular e, ao final, deve devolver um resultado dentro da própria variável. Só que em momento algum foi declarado na função que ela deve devolver o valor de imc. Para fazê-lo utiliza-se a palavra return, ou "retorna". Ele retornará o imc:

function calculaImc(altura, peso) {

var imc = peso / (altura \* altura);

return imc;

}

var imcFlavio = calculaImc(1.71, 73);

var imcAmigo = calculaImc(1.72, 68);

mostra(imcFlavio);

mostra(imcAmigo);

Ao calcularmos o IMC dentro da função, e a variável IMC guardar o resultado, a palavra return deixa disponível à esquerda da variável o resultado que foi calculado pela função. Salvaremos e recarregaremos a página, e então teremos o seguinte resultado:

24.96494647925858

25.985397512168742

Se inserirmos na função somente return 10, como se vê abaixo,

function calculaImc(altura, peso) {

return 10;

}

Quando ela for processada, o que será atribuído à var imcFlacio é aquilo que está em return, neste caso, 10. Assim, o valor do imcFlavio será 10. O mesmo acontecerá com o imcAmigo. Ao salvarmos e recarregarmos a página, teremos:

10

10

Entretanto, não podemos retornar 10, e sim o cálculo do IMC. Da seguinte forma:

function calculaImc(altura, peso) {

var imc = peso / (altura \* altura);

return imc;

}

var imcFlavio = calculaImc(1.71, 73);

var imcAmigo = calculaImc(1.72, 68);

mostra(imcFlavio);

mostra(imcAmigo);

Podemos pensar no calculaImc() como se fosse uma pessoa a quem estivéssemos solicitando que realizasse o cálculo do IMC. A pessoa perguntaria a altura e o peso, ao que responderíamos com os dados do nosso exemplo, ou seja, 1.71 e 73. Ficaríamos esperando o resultado, dado para quem chama a função por meio da cláusula return.

Poderíamos colocar algo como return Flavio, mas assim seriam exibidas duas vezes a palavra Flavio na tela, sem nenhum cálculo realizado. Não é isso que queremos, então definiremos como:

function calculaImc(altura, peso) {

var imc = peso / (altura \* altura);

return imc;

}

var imcFlavio = calculaImc(1.71, 73);

var imcAmigo = calculaImc(1.72, 68);

mostra(imcFlavio);

mostra(imcAmigo);

Episódio 3.

Nesta aula aprenderemos outra forma de calcularmos mais de um IMC por vez. Ao chamarmos a função calculaImc() com a altura e peso de Flávio, 1.71 e 73, foi calculado o IMC e retornada a variável imc. Para simplificar, podemos simplesmente indicar que o return será o cálculo em si:

function calculaImc(altura, peso) {

return peso / (altura \* altura);

}

var imcFlavio = calculaImc(1.71, 73);

var imcAmigo = calculaImc(1.72, 68);

mostra(imcFlavio);

mostra(imcAmigo);

Salvaremos e recarregaremos a página, e teremos o mesmo resultado:

24.96494647925858

22.985397512168742

Isso porque quando chamamos o calculaImc(), a única instrução que temos dentro dele é o return. Quando este for devolver o resultado da função, avaliará a expressão peso / (altura \* altura), e imprimirá o que dela resultar.

Entendemos o calculaImc() como um moedor de carne, em que entram as informações de peso e altura, e sai o resultado da função. Entram dois parâmetros, mas é emitido apenas um resultado.

Como o JavaScript processa isso?

Ao receber a seguinte linha:

var imcFlavio = calculaImc(1.71, 73);

Ele entenderá que a variável imcFlavio recebe calculaImc(1.71, 73), e aguardará isso ser processado, retornando o resultado do cálculo. Poderemos escolher a mensagem que queremos exibir para o IMC:

function calculaImc(altura, peso) {

return peso / (altura \* altura);

}

var imcFlavio = calculaImc(1.71, 73);

var imcAmigo = calculaImc(1.72, 68);

var totalImc = imcFlavio + imcAmigo;

document.write("A soma dos imc's é " + totalImc);

Salvaremos, e recarregaremos a página no navegador, o que resultará em:

A soma dos imc's é 47.95034399142732

Quem for utilizar a função escolherá se prefere exibi-lo na tela, receber via e-mail, ou gravar em um banco de dados. O que não podemos fazer é fixá-lo na função calculaImc(), a saída.

E se colocarmos um mostra() dentro do calculaImc(), e quisermos enviar um SMS ou um e-mail? Sua exibição na tela será sempre fixa. Quem chama a função, e pega seu valor, decidirá sobre como deseja dispor destes dados. Podemos também obter o mesmo resultado da seguinte forma:

var totalImc = calculaImc(1.71, 73) + calculaImc(1.72, 68);

Assim, estamos declarando que o valor total do IMC recebe o retorno de calculaImc(1.71, 73), somado ao retorno de calculaImc(1.72, 68). Ao se deparar com essa operação, o JavaScript tomará o resultado de cada operação separadamente, primeiro um IMC e depois o outro, para então somá-los.

Utilizar um método ou outro é uma escolha do programador. Com isso, temos um programa que abre possibilidades para criarmos outros ainda mais sofisticados.

Episódio 4.

Nesta aula aprenderemos a **interagir com usuários**. Temos nosso programa do IMC:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

function calculaImc(altura, peso) {

return peso / (altura \* altura);

}

</script>

Temos as funções pulaLinha(), mostra() e calculaImc(), e criaremos em seguida duas variáveis, alturaInformada e pesoInformado:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

function calculaImc(altura, peso) {

return peso / (altura \* altura);

}

var alturaInformada = 1.71;

var pesoInformado = 73;

</script>

Para calcular o IMC, abaixo das variáveis que acabamos de criar, teremos a var imc, que receberá o calculaImc() com os parâmetros alturaInformada e pesoInformado:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

function calculaImc(altura, peso) {

return peso / (altura \* altura);

}

var alturaInformada = 1.71;

var pesoInformado = 73;

var imc = calculaImc(alturaInformada, pesoInformado)

</script>

E, para exibir, incluiremos uma instrução document.write() abaixo da variável imc:

var alturaInformada = 1.71;

var pesoInformado = 73;

var imc = calculaImc(alturaInformada, pesoInformado)

document.write("O IMC calculado é " + imc);

</script>

Salvaremos o programa, retornaremos ao navegador e recarregaremos a página. Obteremos o resultado desejado:

O IMC calculado é 24.96494647925858

Até então não foi apresentado nenhum conceito novo.

Veremos agora como podemos calcular o IMC de um novo indivíduo, como uma visita, por exemplo. Para isso, esta pessoa teria de inserir suas informações diretamente no programa. Quanto mais pessoas quiserem saber seu IMC, maior o número de alterações pelas quais o programa terá de passar.

Faremos com que o programa seja genérico, ou seja, quando o usuário recarregar a página, será solicitado que informe sua altura e peso. Determinamos que as variáveis alturaInformada e pesoInformado receberão as informações do usuário, mas ainda não definimos como isso será perguntado.

Abriremos o console do Google Chrome e selecionaremos na barra de menu superior as opções "Visualizar > Desenvolvedor > Console JavaScript". Nele, digitaremos:

prompt()

Trata-se de uma função do JavaScript, que receberá como parâmetro a pergunta que desejamos fazer ao usuário, no caso, "Informe sua altura":

prompt("Informe sua altura");

Pressionaremos a tecla "Enter", e surgirá um pop up com a frase "Informe sua altura" e um espaço em branco abaixo para que o usuário possa incluir suas informações. Digitaremos "1.71" e pressionaremos "Ok".

Veremos que foi impresso no console a informação que acabamos de inserir:

prompt("Informe sua altura");

"1.71"

Isso porque o prompt() é uma função que retorna um valor. Portanto, a variável altura receberá o retorno do prompt():

prompt("Informe sua altura");

"1.71"

var altura = prompt("Informe sua altura");

Para confirmar, basta digitarmos altura e pressionarmos "Enter". Como foi impresso o número 1.71, sabemos que deu certo:

prompt("Informe sua altura");

"1.71"

var altura = prompt("Informe sua altura)";

altura

"1.71"

Do mesmo modo, se digitarmos alert(altura) e pressionarmos "Enter", surgirá um pop up com o número 1.71. Fecharemos o console e retornaremos ao editor de texto. No local em que há o número 1.71, colocaremos a função prompt():

var alturaInformada = prompt("Informe sua altura");

Para o peso, inseriremos prompt("Informe seu peso"):

var alturaInformada = prompt("Informe sua altura");

var pesoInformado = prompt("Informe seu peso");

A informação só será considerada quando o usuário pressionar a tecla "Ok". Salvaremos o programa, retornaremos ao navegador e recarregaremos a página. Se simplesmente pressionarmos "Ok" sem inserir nenhum dado, teremos como resultado "O IMC calculado é NaN".

Recarregaremos a página. Informaremos a altura, 1.71, e o peso, 73. Assim, o resultado exibido na tela é:

O IMC calculado é 24.96494647925858

Para que as informações de um novo usuário sejam inseridas, basta recarregarmos a página. Como o programa será executado novamente, todas as instruções serão processadas. Inseriremos novos dados, 1.62 de altura, e 55 de peso. Isto resultará em:

O IMC calculado é 20.957171162932475

Nosso programa é mais útil, porque consegue interagir com o usuário. É interessante personalizarmos ele ainda mais, criando a variável nome:

var nome = prompt("Informe o seu nome");

var alturaInformada = prompt("Informe sua altura");

var pesoInformado = prompt("Informe seu peso");

A variável nome guardará o nome que foi digitado no prompt(). Ele será concatenado nas próximas mensagens, para que fiquem mais pessoais:

var nome = prompt("Informe o seu nome");

var alturaInformada = prompt(nome + ", informe sua altura");

var pesoInformado = prompt(nome + ", informe seu peso");

var imc = calculaImc(alturaInformada, pesoInformado);

document.write(nome + ", o seu IMC calculado é " + imc);

Lemos em uma variável o nome de quem está utilizando o programa e, ao fazermos a próxima pergunta, concatenamos seu nome com a pergunta. Salvaremos e recarregaremos a página, e isto fará com que o primeiro pop up apareça, solicitando o nome do usuário. Digitaremos "Flávio" e pressionaremos "Enter".

Em seguida surge um novo pop up, com a mensagem:

Flávio, informe sua altura

Digitaremos 1.71 e pressionaremos "Ok". Surgirá então um novo pop up:

Flávio, informe seu peso

Digitaremos 73 e pressionaremos "Ok". Como resultado final, teremos:

Flávio, o seu IMC calculado é 24.96494647925858

Alteraremos a mensagem para que diga apenas "O seu IMC é":

var nome = prompt("Informe o seu nome");

var alturaInformada = prompt(nome + ", informe sua altura");

var pesoInformado = prompt(nome + ", informe seu peso");

var imc = calculaImc(alturaInformada, pesoInformado);

document.write(nome + ", o seu IMC é " + imc);

Salvaremos e recarregaremos a página. Inseriremos novas informações, como o nome "Calopsita", altura "1,79" e peso "90", e teremos o seguinte resultado:

Calopsita, o seu IMC é 28.089010954714272

Temos um programa genérico que cada pessoa que o utilizar pode recarregar para informar seus dados pessoais! var imc = calculaImc(alturaInformada, pesoInformado);

document.write(nome + ", o seu IMC calculado é " + imc);

# Aula 6: Execute Códigos Diferentes Dependendo da Condição

Episódio 1

Nesta aula, aprenderemos a resolver um novo tipo de problema. Com o programa imc.html aberto, selecionaremos as opções "File > Save as..." e salvaremos como futebol.html. Como trataremos de futebol, removeremos as partes do código relacionadas ao cálculo do IMC, e manteremos o seguinte:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

</script>

Nosso objetivo será perguntar ao usuário o número de vitórias e empates do seu time. Se você não gostar de futebol, pode inserir números aleatórios. Criaremos uma variável vitorias que receberá um prompt():

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var vitorias = prompt();

</script>

Temos que fazer uma pergunta ao usuário, no caso, utilizaremos "Entre com o número de vitórias":

var vitorias = prompt("Entre com o número de vitórias");

A próxima variável se chamará empates, e a pergunta será "Entre com o número de empates":

var vitorias = prompt("Entre com o número de vitórias.");

var empates = prompt("Entre com o número de empates.");

Salvaremos, e retornaremos ao navegador. Temos que abrir este novo arquivo e, para isso, selecionaremos "Arquivo > Abrir arquivo...> futebol.html". Quando o carregarmos, surgirá um pop up com a mensagem:

Esta página diz: Entre com o número de vitórias.

Digitaremos o número 15 e pressionaremos "Ok". Em seguida, surge um novo pop up com a seguinte mensagem:

Essa página diz: Enter com o número de empates.

Digitaremos 2 e pressionaremos "Ok". Nada acontece, pois ainda não inserimos nenhuma instrução sobre o que fazer com estes números. Com o número de vitórias e empates, calcularemos a quantidade de pontos de cada time. Uma vitória vale 3 (três) pontos, e um empate, 1 (um) ponto. Portanto, multiplicaremos a quantidade de vitórias por 3 e somaremos este resultado ao número de empates.

Criaremos uma variável pontos, que receberá a operação matemática descrita acima:

var vitorias = prompt("Entre com o número de vitórias.");

var empates = prompt("Entre com o número de empates.");

var pontos = vitorias \* 3 + empates;

Será que precisamos de parênteses? Não, pois a multiplicação será considerada em primeiro lugar, para posteriormente ser somada ao número de empates. Entretanto, é possível utilizarmos os parênteses se desejarmos deixar a equação mais clara. Em seguida, imprimiremos "Os pontos do seu time são ", concatenado com pontos:

var vitorias = prompt("Entre com o número de vitórias.");

var empates = prompt("Entre com o número de empates.");

var pontos = vitorias \* 3 + empates;

mostra("Os pontos do seu time são " + pontos);

Salvaremos, retornaremos ao navegador, recarregaremos a página, e digitaremos 3, para o número de vitórias, e 1 para o número de empates. Dessa forma, esperamos que o total de pontos seja 10. Obtivemos o seguinte resultado:

Os pontos do seu time são 91

Observamos que mesmo incluindo os parênteses e isolando a operação de multiplicação, o resultado seria o mesmo. O return da função prompt() - que não foi criada por nós, e é do JavaScript -, sempre retorna como texto, não como número.

Por que isso funcionou para nosso cálculo de IMC, e não está funcionando agora? Vamos criar um alert() para somar o número de vitórias com 10:

var vitorias = prompt("Entre com o número de vitórias.");

alert(vitorias + 10);

var empates = prompt("Entre com o número de empates.");

var pontos = vitorias \* 3 + empates;

mostra("Os pontos do seu time são " + pontos);

O número de vitórias será interpretado como um texto, então, ao recarregarmos a página e digitarmos 5, ao ser somado ao número 10, ocorrerá uma concatenação cujo resultado deve ser 510. Salvaremos o programa e recarregaremos a página.

Digitaremos 5 para o número de vitórias e pressionaremos "Ok". Conforme esperado, obteremos o seguinte resultado:

Essa página diz: 510

Sempre que usamos o prompt(), o conteúdo é lido como um texto. E se multiplicarmos um texto por um número?

var vitorias = prompt("Entre com o número de vitórias.");

alert(vitorias \* 3);

Salvaremos e recarregaremos a página. Digitaremos 5 como o número de vitórias e pressionaremos "Ok". Teremos:

Essa página diz: 15

Funcionou!

Isso acontece por conta do JavaScript. Lemos prompt() sempre como um **texto**. Se no texto houver um número e fizermos uma operação de multiplicação ou divisão, o JavaScript converte o texto para **número**, e é por isso que a multiplicação funciona.

Por definição, não pode haver multiplicação de um texto com um número, portanto, se digitarmos, por exemplo, "Flávio" no pop up, não será possível realizar a operação, o resultado será NaN, algo inválido.

Retornando ao nosso programa:

var vitorias = prompt("Entre com o número de vitórias.");

var empates = prompt("Entre com o número de empates.");

var pontos = vitorias \* 3 + empates;

mostra("Os pontos do seu time são " + pontos);

Ao chegar na variável pontos, a primeira parte da equação está correta e o JavaScript multiplica o número de vitórias por 3. A seguir, ele tentará concatená-lo com empates, mas nesse caso não há conversão, já que ao digitarmos o número de empates ele é interpretado como uma string.

Assim, teríamos 9 - resultado da multiplicação do número de vitórias por 3 - somado ao 1, que é interpretado como uma string. Como vimos, quando temos um número e uma string, eles são concatenados e, assim, o resultado será 91.

Como queremos que o prompt() seja lido como número - e não como string -, garantiremos isso nos assegurando de que nosso cálculo funcionará.

Para esta conversão, primeiramente, veremos como é possível fazê-lo pelo console do navegador. Acessaremos "Visualizar > Desenvolvedor > Exibir o código fonte", criaremos uma variável texto que recebe o número 10, e em seguida pressionaremos a tecla "Enter":

var texto = "10"

Em seguida, ao escrevermos texto e pressionarmos "Enter", teremos o seguinte resultado:

var texto = "10"

undefined

texto

"10"

texto

"10"

Este aparece entre aspas por ser um texto. Em seguida, criaremos uma variável numero, que recebe parseInt(), uma função existente no JavaScript, e preparada para receber um texto a ser convertido em número.

A função parseInt() terá como parâmetro a variável texto, que por sua vez tem no seu conteúdo uma string. Com isso, queremos que isto seja convertido para número; isto é, no caso a string 10 se tornará o número 10:

var texto = "10"

undefined

texto

"10"

texto

"10"

texto

"10"

var numero = parseInt(texto);

Pressionaremos "Enter" e, em seguida, digitaremos "texto" e apertaremos em "Enter" novamente:

var texto = "10"

undefined

texto

"10"

texto

"10"

texto

"10"

var numero = parseInt(texto);

undefined

texto

"10"

Ao digitarmos numero e pressionarmos "Enter", acontecerá o seguinte:

var texto = "10"

undefined

texto

"10"

texto

"10"

texto

"10"

var numero = parseInt(texto);

undefined

texto

"10"

numero

10

Ou seja, por não estar entre aspas, o valor foi convertido. O que **não** podemos fazer é o seguinte:

var numero = parseInt("Flavio");

A conversão não será feita, já que Flavio não é um número. Neste caso, o resultado será:

var numero = parseInt("Flavio");

undefined

numero

NaN

Foi exibida a mensagem NaN, que significa "não é um número". Retornaremos ao nosso código no editor de textos. Para convertermos o retorno do prompt(), o passaremos diretamente para parseInt():

var vitorias = parseInt(prompt("Entre com o número de vitórias."));

var empates = prompt("Entre com o número de empates.");

var pontos = vitorias \* 3 + empates;

mostra("Os pontos do seu time são " + pontos);

Esta função precisa receber algo para que possa transformá-lo em um número. Poderíamos passar uma string, no entanto usaremos o retorno do prompt(), e faremos o mesmo na variável empates:

var vitorias = parseInt(prompt("Entre com o número de vitórias."));

var empates = parseInt(prompt("Entre com o número de empates."));

Salvaremos e recarregaremos a página no navegador. Surge o primeiro pop up solicitando o número de vitórias, e digitaremos 3. Em seguida, digitaremos 1 para o número de empates. Obteremos o seguinte resultado:

Os pontos do seu time são 10

Funcionou!

Importante lembrar que se quiséssemos ler um nome, por exemplo, não seria necessário utilizar a função parseInt(), porque neste caso queremos que isto seja considerado como texto. Sempre que formos ler uma entrada, e a intenção for trabalhar com números, temos que fazer o parseInt() para que o valor digitado, que por padrão é lido como texto, seja interpretado como um número.

Episódio 2

Já conseguimos ler a quantidade de vitórias do time, de empates, e calcular o total de pontos. Entretanto, nossa intenção é dotar o programa de mais inteligência. Supondo que no ano passado o total de pontos tenha sido 28, queremos que ele seja capaz de informar se o time está pior ou melhor em relação ao ano anterior.

Faremos uma comparação do total de pontos calculado com o total de pontos do ano anterior (no caso, 28), e teremos uma mensagem personalizada para cada tipo de situação. Utilizaremos a função mostra() para imprimir a frase "Seu time está melhor do que no ano passado":

var vitorias = parseInt(prompt("Entre com o número de vitórias."));

var empates = parseInt(prompt("Entre com o número de empates."));

var pontos = (vitorias \* 3) + empates;

mostra("Os pontos do seu time são " + pontos);

mostra("Seu time está melhor do que no ano passado");

Em seguida, inseriremos a frase "Seu time está pior do que no ano passado"):

var vitorias = parseInt(prompt("Entre com o número de vitórias."));

var empates = parseInt(prompt("Entre com o número de empates."));

var pontos = (vitorias \* 3) + empates;

mostra("Os pontos do seu time são " + pontos);

mostra("Seu time está melhor do que no ano passado");

mostra("Seu time está pior do que no ano passado");

Por fim, imprimiremos a frase "Seu time está igual ao ano passado!":

var vitorias = parseInt(prompt("Entre com o número de vitórias."));

var empates = parseInt(prompt("Entre com o número de empates."));

var pontos = (vitorias \* 3) + empates;

mostra("Os pontos do seu time são " + pontos);

mostra("Seu time está melhor do que no ano passado.");

mostra("Seu time está pior do que no ano passado.");

mostra("Seu time está igual ao ano passado.");

Salvaremos e recarregaremos a página. Digitaremos os valores referentes a 3 vitórias e 1 empate. O total de pontos é 10, sendo que este número é inferior aos do ano anterior, que totalizaram 28. Sendo assim, o texto que deve ser exibido é "seu time está pior do que no ano passado". Entretanto, temos a seguinte exibição:

Os pontos do seu time é 10

Seu time está melhor do que no ano passado.

Seu time está pior do que no ano passado.

Seu time está igual ao ano passado.

O programa ainda não tem a inteligência para saber, a partir do total de pontos, qual das mensagens deve exibir. Pensando na construção das frases, só poderemos mostrar a frase que indica que o time está melhor que no ano passado se o total de pontos neste ano for maior que 28. Portanto, teremos a seguinte construção:

se pontos maior 28

mostra("Seu time está melhor do que no ano passado.");

mostra("Seu time está pior do que no ano passado.");

mostra("Seu time está igual ao ano passado.");

E assim por diante para as demais frases:

se pontos maior 28

mostra("Seu time está melhor do que no ano passado.");

se pontos menor 28

mostra("Seu time está pior do que no ano passado.");

se pontos igual 28

mostra("Seu time está igual ao ano passado.");

Se calcularmos 12 pontos, por exemplo, teremos que verificar em primeiro lugar se este resultado é maior que 28. Não sendo, passamos para a próxima frase, que avalia se os pontos são um número menor que 28 (e no caso, são), assim, a mensagem exibida será "seu time está pior do que no ano passado".

A terceira e última condição indica que os pontos permaneceram o mesmo, ou seja, 28. Como não é esse o nosso caso, não será exibida a mensagem correspondente. Salvaremos e recarregaremos a página, e nada acontece. Abriremos o console, iremos a "Visualizar > Desenvolvedor > Console JavaScript", em que teremos a seguinte mensagem:

Uncaught SyntaxError: Unexpected Identifier

Há um identificador não esperado. Ele não entendeu as frases que inserimos, pois estão em português. A primeira coisa a fazermos é trocar a palavra "se" pela sua tradução em inglês, "if":

if pontos maior 28

mostra("Seu time está melhor do que no ano passado.");

if pontos menor 28

mostra("Seu time está pior do que no ano passado.");

if pontos igual 28

mostra("Seu time está igual igual ao ano passado.");

Em seguida, trocaremos as palavras "maior" e "menor" por seus símbolos matemáticos:

if pontos > 28

mostra("Seu time está melhor do que no ano passado.");

if pontos < 28

mostra("Seu time está pior do que no ano passado.");

No caso do "igual", o símbolo (=) é utilizado para indicar uma atribuição. Em muitas linguagens de programação, para testarmos uma igualdade, e para que o JavaScript saiba que queremos atribuir ou comparar algo, temos de utilizar o símbolo duas vezes (==). O código ficará portanto da seguinte forma:

if pontos > 28

mostra("Seu time está melhor do que no ano passado.");

if pontos < 28

mostra("Seu time está pior do que no ano passado.");

if pontos == 28

mostra("Seu time está igual igual ao ano passado.");

Precisamos ter muito cuidado com isso; se quisermos realizar uma comparação, precisamos utilizar o símbolo de "igual" duas vezes (==).

Será que já funciona desse jeito?

Ainda não. O if() recebe o resultado da comparação entre a quantidade de pontos e o número 28, sendo que para cada frase haverá uma resposta, verdadeira ou falsa. Isto será representado pela inclusão das operações entre parênteses:

if(pontos > 28)

mostra("Seu time está melhor do que no ano passado.");

if(pontos < 28)

mostra("Seu time está pior do que no ano passado.");

if(pontos == 28)

mostra("Seu time está igual igual ao ano passado.");

Salvaremos tudo e recarregaremos a página. Nada é exibido, então abriremos o console do navegador, da mesma forma como fizemos anteriormente, em que digitaremos:

var pontos = 12;

Pressionaremos a tecla "Enter", e surgirá a mensagem undefined. Em seguida, digitaremos:

var pontos = 12;

undefined

pontos > 10

Apertando "Enter" de novo, surgirá a mensagem true:

var pontos = 12;

undefined

pontos > 10

true

O resultado das operações de comparação é sempre true (verdadeiro), ou false (falso). Em seguida, digitaremos pontos < 10:

var pontos = 12;

undefined

pontos > 10

true

pontos < 10

Pressionaremos a tecla "Enter" e teremos a resposta false:

var pontos = 12;

undefined

pontos > 10

true

pontos < 10

false

Assim, o if() só exibirá o mostra() caso o resultado da operação entre parênteses seja true. Precisamos definir que o if() pode ter mais de uma instrução, e pode fazer mais de uma coisa. Para isso, ele deverá ter um bloco, representado pelas chaves ({}):

if(pontos > 28) {

mostra("Seu time está melhor do que no ano passado.");

}

Se quiséssemos incluir mais de uma mensagem:

if(pontos > 28) {

mostra("Seu time está melhor do que no ano passado.");

mostra("Seu time está melhor do que no ano passado.");

}

Teríamos duas instruções dentro do bloco if(), e as mensagens só serão exibidas se o número de pontos for maior que 28. Manteremos somente uma mensagem e criaremos blocos para as demais frases:

if(pontos > 28) {

mostra("Seu time está melhor do que no ano passado.");

}

if(pontos < 28) {

mostra("Seu time está pior do que no ano passado.");

}

if(pontos == 28) {

mostra("Seu time está igual igual ao ano passado.");

}

Salvaremos e recarregaremos a página. No primeiro pop up, digitaremos 3 para o número de vitórias, e 1 para o número de empates. Teremos a seguinte exibição:

Os pontos do seu time são 10

Seu time está pior do que no ano passado.

Nosso programa é inteligente o suficiente para exibir **somente** a mensagem de que está pior, ignorando as demais, porque só entra no bloco do if() se a sua condição for verdadeira.

Faremos um novo teste. Recarregaremos a página e inseriremos um total de 10 vitórias e 2 empates, totalizando 32 pontos, o que nos trará a seguinte exibição:

Os pontos do seu time são 32

Seu time está melhor do que no ano passado.

Funcionou, ele entrou na condição em que a pontuação é maior que o ano passado e ignorou as demais. Recarregaremos a página para mais um teste. Inseriremos o total de 1 vitória e 25 empates, totalizando 28 pontos. Assim, teremos a seguinte exibição:

Os pontos do seu time são 28

Seu time está igual ao ano passado.

Agora temos um programa mais inteligente. E se ao final do código inserirmos um comando mostra() com a mensagem "FIM"?

if(pontos > 28) {

mostra("Seu time está melhor do que no ano passado.");

}

if(pontos < 28) {

mostra("Seu time está pior do que no ano passado.");

}

if(pontos == 28) {

mostra("Seu time está igual ao ano passado.");

}

mostra("FIM");

Ele exibe o "FIM"? Sim, porque não há condição nenhuma para que ele seja exibido. Ao recarregarmos a página, e inserirmos 3 vitórias e 1 empate, veremos a seguinte exibição na tela:

Os pontos do seu time são 10

Seu time está pior do que no ano passado.

FIM

Com isso, conseguimos controlar o comportamento do programa de acordo com alguma condição. Se inserirmos true para todos os ifs:

if(true) {

mostra("Seu time está melhor do que no ano passado.");

}

if(true) {

mostra("Seu time está pior do que no ano passado.");

}

if(true) {

mostra("Seu time está igual ao ano passado.");

}

mostra("FIM");

Serão exibidas todas as mensagens. Alternativamente, se inserirmos false:

if(false) {

mostra("Seu time está melhor do que no ano passado.");

}

if(false) {

mostra("Seu time está pior do que no ano passado.");

}

if(false) {

mostra("Seu time está igual ao ano passado.");

}

mostra("FIM");

Nenhuma mensagem é exibida. Por isso, temos de manter uma condição para que o programa possa trabalhar em cima dela.

Episódio 3

Na aula anterior utilizamos como exemplo dois times de futebol, com o objetivo de aprender a dotar o programa de certa inteligência para fornecer uma resposta aos usuários.

Nesta aula vamos retornar ao programa do IMC. Lembrando que a OMS determina que indivíduos com IMC menor que 18.5 estão abaixo da média, com 35 estão acima da média, e entre 18.5 e 35 estão na média aceitável.

Abriremos o nosso editor de texto, e na barra de menu superior acessaremos "File > Open... > imc.html". Faremos o mesmo processo no navegador, selecionando "Arquivo > Abrir arquivo... > imc.html". Nossos alertas pedirão certas informações, neste momento digitaremos qualquer dado, já que queremos somente abrir o programa.

Em nosso editor de texto, temos nosso programa:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

function calculaImc(altura, peso) {

return peso / (altura \* altura);

}

var nome = prompt("Informe o seu nome");

var alturaInformada = prompt(nome + ", informe sua altura");

var pesoInformado = prompt(nome + ", informe seu peso");

var imc = calculaImc(alturaInformada, pesoInformado);

document.write(nome + ", o seu IMC é " + imc);

</script>

Até o momento, a Organização Mundial da Saúde definiu os seguintes parâmetros para o IMC:

| **IMC** | **Avaliação** |
| --- | --- |
| < 18.5 | Baixo |
| 35 > | Alto |
| Entre 18.5 e 35 | Média |

Ou seja, se o indivíduo possui o IMC igual a 20, está bem, se for 34, também, entretanto, com o IMC 17 ou 36, a pessoa já não é saudável. Como temos um programa personalizado, perguntamos uma série de informações. Em nome do usuário, escreveremos Flávio, altura será 1.71, e peso, 73. Isto resultará em:

Flávio, o seu IMC é 24.96494647925858

O próximo passo será inserirmos frases que indiquem ao usuário como o resultado do seu IMC é classificado pela Organização Mundial da Saúde. Utilizaremos a condição if(), que recebe algo, e terá um bloco a ser executado conforme o resultado passado seja verdadeiro ou falso.

Nossa primeira pergunta será se o IMC for menor que 18.5, e então teremos no bloco um mostra() que exibirá a mensagem Você está abaixo do recomendado. No código, isso ficará localizado logo abaixo da mensagem nome + ", o sei IMC é " + imc.

if(imc < 18.5) {

mostra("Você está abaixo do recomendado")

}

Em seguida, criaremos uma situação para quando o IMC for maior que 35:

if(imc < 18.5) {

mostra("Você está abaixo do recomendado");

}

if(imc > 35) {

mostra("Você está acima do recomendado");

}

Salvaremos e programa, retornaremos ao navegador e recarregaremos a página. Inseriremos novamente os dados de altura e peso, 1.71 e 73, respectivamente, com que obteremos o mesmo resultado de antes:

Flávio, o seu IMC é 24.96494647925858

Não foi exibida nenhuma mensagem além desta, porque o resultado foi de aproximadamente 24, e este número não é menor que 18.5 nem maior que 35. Precisaremos criar uma condição que teste se o IMC é maior ou igual a 18.5, e menor ou igual a 35. Para isso, faremos o seguinte:

if(imc < 18.5) {

mostra("Você está abaixo do recomendado");

}

if(imc > 35) {

mostra("Você está acima do recomendado");

}

if(imc >= 18.5) {

if(imc <= 35) {

mostra("Seu IMC está excelente!");

}

}

Como o número 24 é ao mesmo tempo maior que 18.5 e menor que 35, a mensagem exibida será "Seu IMC está excelente!". Salvaremos e recarregaremos a página. Vamos testar inserindo o nome Flávio, a altura 1.71 e o peso 73. Obteremos:

Flávio, o seu IMC é 24.96494647925858Seu IMC está excelente!

Retornando ao programa, observamos que o document.write() foi utilizado para escrever nome + ", o seu IMC é " + imc:

document.write(nome + ", o seu IMC é " + imc);

if(imc < 18.5) {

mostra("Você está abaixo do recomendado");

}

if(imc > 35) {

mostra("Você está acima do recomendado");

}

if(imc >= 18.5) {

if(imc <= 35) {

mostra("Seu IMC está excelente!");

}

}

</script>

Precisaremos utilizar a função mostra() para podermos ter a quebra de linha:

mostra(nome + ", o seu IMC é " + imc);

if(imc < 18.5) {

mostra("Você está abaixo do recomendado");

}

if(imc > 35) {

mostra("Você está acima do recomendado");

}

if(imc >= 18.5) {

if(imc <= 35) {

mostra("Seu IMC está excelente!");

}

}

</script>

Salvaremos e recarregaremos a página. Inseriremos as mesmas informações que anteriormente. Teremos a seguinte exibição:

Flávio, o seu IMC é 24.96494647925858

Seu IMC está excelente!

Faremos mais um exemplo usando um IMC claramente fora do padrão. Recarregaremos a página e inseriremos as seguintes informações: para o nome, Calopsita, altura 1.61, peso 100. Obteremos o seguinte resultado:

Calopsita, o seu IMC é 38.5787583555032

Você está acima do recomendado

Temos um programa útil, o qual podemos utilizar para calcular o IMC de diversos indivíduos. A forma como estamos fazendo isso está verbosa, ou seja, há bastante informação escrita. Para ajudar, o JavaScript possui um atalho. Queremos testar se o IMC é maior ou igual a 18, **e** menor ou igual a 35 e, para isso, poderemos escrever da seguinte forma:

if(imc >= 18.5 && imc <= 35) {

mostra("Seu IMC está excelente!");

}

Assim, o JavaScript primeiro se questionará se o IMC é maior ou igual a 18.5, sendo verdade, ele passará a analisar o && e se questionar se, além disso, **também** é menor ou igual a 35. Sendo as duas afirmações verdadeiras, o navegador imprimirá a mensagem Seu IMC está excelente!. Atenção, se uma das condições for verdadeira e a outra não, nada será exibido, uma vez que a função mostra() só será executada se **as duas condições forem verdadeiras**.

Anteriormente, havia um if() dentro de outro:

if(imc >= 18.5) {

if(imc <= 35) {

mostra("Seu IMC está excelente!");

}

}

Agora não mais. Tiramos o segundo if() do bloco e incluímos as duas condições em uma só linha:

if(imc >= 18.5 && imc <= 35) {

mostra("Seu IMC está excelente!");

}

Salvaremos e recarregaremos a página. Inseriremos as mesmas informações que anteriormente e veremos que o programa está funcionando:

Flávio, o seu IMC é 24.96494647925858

Seu IMC está excelente!

Episódio 4

Já aprendemos a criar um programa útil para nos ajudar a calcular o IMC e descobrir se alguém está ou não saudável. Criaremos agora o nosso primeiro jogo, que será de adivinhação. Clicaremos em "File > Save As..." e salvaremos como jogo\_advinha.html. Manteremos as funções pulaLinha() e mostra() do nosso programa:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

</script>

Teremos em nosso programa uma variável chamada numeroPensado, que recebe o valor 5:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = 5;

</script>

Nosso objetivo é criar um jogo em que o usuário adivinhe qual é o número pensado. O indivíduo se depara com a página, contendo um alerta perguntando qual é o chute, o usuário inserirá um número entre 1 e 10, e se este número for igual ao numeroPensado, teremos um acerto.

Portanto, teremos uma variável chute, que lerá a informação digitada no teclado:

var numeroPensado = 5;

var chute = parseInt(prompt("digite seu chute!"));

Lembrando que sempre que queremos interpretar uma entrada como um número, devemos utilizar o parseInt(). O programa lerá o número digitado como uma string, o qual será convertido para um número e será guardado em chute. Agora, testaremos a possibilidade de chute ser igual a numeroPensado:

var numeroPensado = 5;

var chute = parseInt(prompt("digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você acertou!");

}

Em seguida, veremos o que acontece quando chute **não é igual** ao número pensado:

if(chute != numeroPensado) {

mostra("Você errou, o número pensado foi " + numeroPensado);

}

Se acertarmos, surgirá uma mensagem indicando isso. Se errarmos, ele indicará o erro e ainda exibirá qual era o número pensado. Salvaremos o programa, retornaremos ao navegador e abriremos o arquivo jogo\_advinha.html, indo a "Arquivo > Abrir arquivo... > jogo\_advinha.html". Surgirá um pop up para digitarmos um chute, e inseriremos o número 2, para testarmos, pressionando "Ok" em seguida. É exibida a seguinte mensagem:

Você errou, o número pensado foi 5

Recarregaremos a página e digitaremos o número 5. Veremos:

Você acertou!

Podemos melhorar nosso código... No caso, estamos fazendo um teste, e desejamos executar algo sempre que o resultado for verdadeiro, e também quando for falso. Isto é, temos o chute e, para ele, descritas duas situações, uma para quando erramos e outra para quando acertamos. Sendo assim, em vez de utilizarmos a função if(), podemos implementar o else, que equivale a "se não" (ou "caso contrário"). Se for o número pensado temos uma situação, e **se não**, temos outra:

if(chute == numeroPensado) {

mostra("Você acertou!");

} else {

mostra("Você errou, o número pensado foi " + numeroPensado);

Com o else não foi preciso repetir em um novo if() a condição contrária. Ele entende que, se a condição do if() não for verdadeira, só pode existir a outra condição.

Salvaremos, recarregaremos a página, digitaremos o número 5, e teremos a seguinte mensagem:

Você acertou!

Recarregaremos a página. Digitaremos o número 3, e a mensagem é a seguinte:

Você errou, o número pensado foi 5

Funcionou! O programa, entretanto, pode ser melhorado. Da forma como está, quando o usuário erra uma primeira vez, já é informada a resposta correta. Se quisermos que várias pessoas joguem, teríamos que recarregar a página todas as vezes, além de alterarmos, no código, o numeroPensado.

E se o programa gerasse o numeroPensado aleatoriamente, a cada vez que fosse feita uma tentativa? Aprenderemos a fazer justamente isso. Abriremos o console no navegador e selecionaremos "Visualizar > Desenvolvedor > Console JavaScript".

Há a função Math.random(), que gera um número aleatório. Pressionando a tecla "Enter", ele gerou um número entre 0 e 1, e todas as vezes em que pressionarmos "Enter" surgirá um novo número. Entretanto, queremos que seja gerado um número entre 0 e 10.

Queremos andar uma casa decimal para direita, e para isso multiplicaremos o número gerado por 10, da seguinte forma:

Math.random() \* 10

Ao pressionarmos "Enter", é movida uma casa decimal para a direita. Por exemplo, onde anteriormente tínhamos um número 0.21, agora temos 2.1. Podemos apertar "Enter" n vezes, e em todas elas surgirá um número diferente, com uma estrutura similar a esta:

Math.random() \* 10

2.1240615385242934

O primeiro número antes da vírgula sempre estará entre 0 e 9. Como queremos que o resultado esteja entre 0 e 10, utilizaremos a função Math.round() para arredondarmos o resultado de Math.random():

Math.round(Math.random() \* 10);

Ao pressionarmos "Enter", teremos sempre um novo número aleatório entre 0 e 10:

Math.round(Math.random() \* 10);

1

Math.round(Math.random() \* 10);

6

Math.round(Math.random() \* 10);

7

Math.round(Math.random() \* 10);

10

Retornaremos ao editor de texto para alterarmos nosso código, diretamente na variável numeroPensado, localizada logo abaixo da função mostra():

var numeroPensado = Math.random() \* 10;

Mas e se quiséssemos um número de 0 a 1000? Basta trocarmos 10 por 1000 no código acima. Agora, falta fazermos com que o número seja arredondado, utilizando-se a função Math.round():

var numeroPensado = Math.round(Math.random() \* 10);

Salvaremos e recarregaremos a página. Surgirá um pop up solicitando um chute, digitaremos o número 3 e receberemos o seguinte retorno:

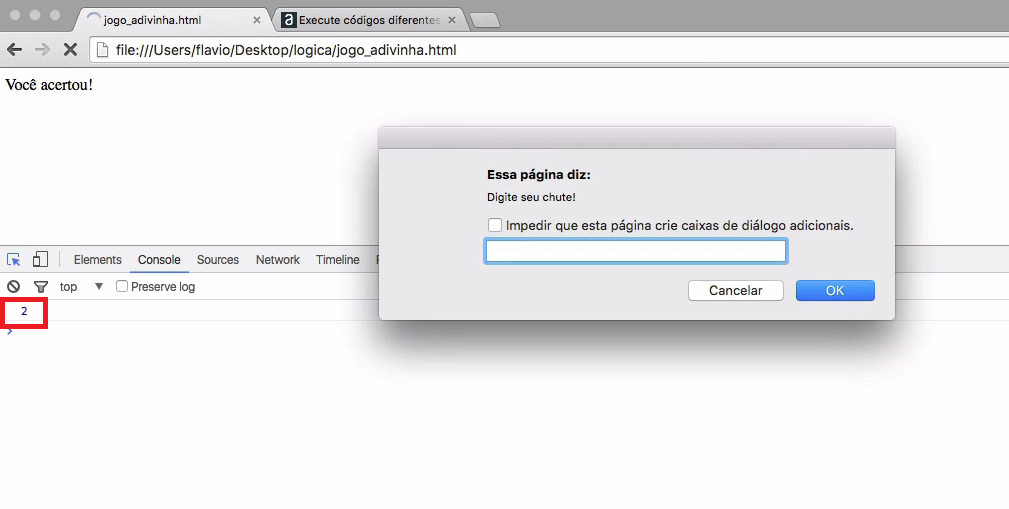
Você errou, o número pensado foi 5

Podemos testar várias vezes até acertarmos. Nesta hipótese, o navegador exibirá a mensagem: Você acertou!. Em seguida criaremos um teste para descobrirmos se o usuário errou ou acertou, utilizando o console.log():

var numeroPensado = Math.round(Math.random() \* 10);

console.log(numeroPensado);

Com isso, o numeroPensado será exibido no console do navegador, em que poderemos visualizar antes mesmo de inserirmos nosso palpite:



Isso nos ajudará a testar ambas condições, de acerto e erro. Para nos certificarmos disso, vamos testar com um número maior, que multiplicaremos por 100:

var numeroPensado = Math.round(Math.random() \* 100);

Retornaremos ao navegador e recarregaremos a página. Digitaremos um palpite de 34, após o qual receberemos a mensagem:

Você errou, o número pensado foi 73

Seria muito difícil acertarmos, a menos que tivéssemos mais chances de testarmos. Assim, finalizamos nosso primeiro jogo.

# Aula 7: repita tarefas

Episódio 1

Nesta aula, aprenderemos a criar um programa que exiba todos os anos em que houve copas do mundo de futebol, desde 1930, data do primeiro campeonato.

Criaremos um novo arquivo, a partir do nosso jogo de adivinhação, selecionando a opção "File > Save As... > ano\_copa.html". Manteremos apenas as funções mostra e pulaLinha:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

</script>

Declararemos uma variável anoCopa que receberá o ano 1930:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var anoCopa = 1930;

</script>

Criaremos um alert para exibir a frase "Teve copa em", concatenada com a variável anoCopa:

var anoCopa = 1930;

alert("Teve copa em " + anoCopa);

Salvaremos e retornaremos ao navegador. Abriremos o arquivo do programa ano\_copa.html, selecionando "Arquivo > Abrir arquivo...". Surge um pop up com a seguinte mensagem:

Teve copa em 1930

Nosso objetivo é fazer com que o programa exiba todos os anos em que houve copas do mundo.

Retornaremos ao código.

Precisamos informar ao programa que os anos de copa serão anoCopa mais 4:

var anoCopa = 1930

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4

Assim, o anoCopa passa a valer 1934. Repetiremos o processo, por diversas vezes, para que em cada uma delas tenhamos um ano de copa diferente:

var anoCopa = 1930;

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4

Salvaremos, retornaremos ao navegador e recarregaremos a página. Assim surgirá um alerta referente a cada um dos anos em que houve Copa do Mundo.

Entretanto, é impraticável termos que copiar e colar tantas vezes até termos o número total de copas.

Como podemos observar, todas as instruções são iguais. E se pudéssemos criar uma função para repeti-las?

Primeiro, manteremos apenas uma das instruções em nosso código:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var anoCopa = 1930;

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

</script>

Criaremos uma função repita e abriremos um bloco:

var anoCopa = 1930;

repita {

}

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

Inseriremos a mensagem que queremos repetir dentro do bloco da função, e tudo que estiver lá será replicado:

var anoCopa = 1930;

repita {

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

}

Se funcionar, isso nos permitirá declarar o anoCopa como 1930, e repetirá infinitas vezes a operação anoCopa = anoCopa + 4. Criaremos um alerta, para indicar o fim:

var anoCopa = 1930;

repita {

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

}

alert("FIM");

Como a operação está inserida no bloco da função repita, o programa não acionará o alert("FIM"), em vez disso, retornará ao bloco e acionará a instrução infinitas vezes.

Como podemos fazer isso em JavaScript?

A primeira ferramenta de repetição que aprenderemos é chamada while, que recebe true:

var anoCopa = 1930;

while(true) {

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

}

alert("FIM");

O while(true) indica que sempre será repetido o conteúdo do bloco dessa função. Se o while receber false, nada será repetido. Se salvarmos o programa, e recarregarmos a página, somente será exibido um alerta, com a mensagem FIM. A repetição não foi exibida.

Se trocarmos o false por true, significa que, enquanto isso for verdadeiro, haverá repetição, portanto temos:

var anoCopa = 1930;

while(true) {

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

}

alert("FIM");

Salvaremos o programa, retornaremos ao navegador e salvaremos a página.

Surge um pop up com a mensagem: Teve copa em 1930, a cada vez que pressionamos "Ok", surge um novo ano de copa, como 1934, 1938, e assim sucessivamente. Entretanto, o programa não para, enquanto pressionarmos "Ok" sempre surgirá um novo ano de copa.

O Google Chrome apresenta uma opção, que diz "Impedir que esta página crie caixas de diálogo adicionais", ao marcarmos na caixa e pressionarmos "Ok", não surgirão mais pop ups. Ao fazermos isso, temos que fechar a aba atual e abrir uma nova, e utilizá-la para visualizar o programa.

Não queremos que sejam impressos anos infinitos, mas sim que o faça até que anoCopa seja menos que 2016, representamos isso da seguinte forma:

while(anoCopa <= 2016)

Que, no código, é inserido da seguinte forma:

var anoCopa = 1930;

while(anoCopa <= 2016) {

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

}

alert("FIM");

Assim como a condição if, o while testará se anoCopa é menor ou igual a 2016, se for verdadeiro, as instruções contidas no bloco serão executadas. Isso significa que o programa acessará o bloco, enquanto anoCopa <= 2016 for verdadeiro.

Ao chegarmos em 2017, é verificado se o número é menor ou igual a 2016, não sendo, a repetição cessará, o programa sairá da instrução e imprimirá o alerta FIM.

Vamos testar. Salvaremos o programa, retornaremos ao navegador e abriremos o programa ano\_copa.html.

Surgirá o primeiro alerta, com o texto:

Teve copa em 1930

Pressionaremos "Ok", até chegarmos em 2014, quando ao clicar em "Ok" será exibida a mensagem FIM.

Conseguimos implementar uma condição de repetição para que a condição do while, em determinado ponto, seja false.

Recapitulando, o anoCopa inicia em 1930 e, como este número é menor que 2016, o programa executará a instrução contida no bloco da função.

E se colocarmos que o ano inicial é 2017?

var anoCopa = 2017;

while(anoCopa <= 2016) {

alert("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

}

alert("FIM");

O JavaScript questionará se 2017 é menor que 2016, como a resposta é negativa, ele jamais executará as instruções dentro do bloco do while, passando direto para o alert("FIM"). Vamos testar. Salvaremos o programa e recarregaremos a página.

Surgirá um pop up com a mensagem:

Essa página diz:

FIM

Retornaremos para 1930.

Exibir o alert pode não ser agradável para o usuário. Assim, utilizaremos a função mostra:

var anoCopa = 1930;

while(anoCopa <= 2016) {

mostra("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

}

mostra("FIM");

Salvaremos e recarregaremos a página. Obtivemos uma lista contendo todos os anos em que houve copa:

Teve copa em 1930

Teve copa em 1934

Teve copa em 1938

Teve copa em 1942

Teve copa em 1946

Assim sucessivamente, até a copa de 2014:

Teve copa em 2014

FIM

Com poucas linhas de código, conseguimos repetir a instrução, enquanto a condição anoCopa <= 2016 for verdadeira. Se alterarmos a condição para while(true), será repetida a instrução ao infinito, fazendo com que o navegador trave. Salvaremos e recarregaremos a página.

O navegador não conseguirá processar essa informação, por isso o fecharemos. É preciso ter cuidado, há navegadores em que isso não é possível, é feita a impressão infinita no HTML, que acaba travando a máquina.

Retornaremos para a condição anterior.

Em vez de imprimirmos até 2016, podemos configurar de modo que imprima até o limite dos anos de copa.

Declararemos uma variável limite, que receberá um prompt solicitando ao usuário que digite a data limite:

var limite = prompt("Entre com a data limite");

var anoCopa = 1930;

while(anoCopa <= 2016) {

mostra("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

}

mostra("FIM");

Já que estamos tratando de um número, é uma boa prática colocarmos oparseInt:

var limite = parseInt(prompt("Entre com a data limite"));

var anoCopa = 1930;

while(anoCopa <= 2016) {

mostra("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

}

mostra("FIM");

Substituiremos o 2016 pela variável limite:

var limite = parseInt(prompt("Entre com a data limite"));

var anoCopa = 1930;

while(anoCopa <= limite) {

mostra("Teve copa em " + anoCopa);

anoCopa = anoCopa + 4;

}

mostra("FIM");

Salvaremos e recarregaremos. Surgirá um pop up solicitando uma data limite, digitaremos 1998. Pressionaremos "Ok". Temos como resultado a lista com todos os anos de copa, até 1998 e, após isso, a mensagem FIM.

Se digitarmos outro número, além de 1998 e que não seja 2002, não há problema, ele exibirá somente até o último ano em que houve copa do mundo. Se digitarmos 2016 serão impressos os anos até 2014.

Adiante, veremos outras estruturas de repetição possíveis.

Episódio 2

Nesta aula, aprenderemos mais uma forma de utilizar repetições em nossos programas.

Desde crianças, aprendemos que a tabuada nada mais é que uma repetição... Criaremos um programa chamado tabuada. Com nosso programa ano\_copa aberto, selecionaremos a opção "Salvar Como" e daremos a ele um novo nome: tabuada.html. Apagaremos os trechos que não lidam com a tabuada e manteremos somente as funções pulaLinha() e mostra():

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

</script>

Criaremos a tabuada de **7**, que segue um mesmo padrão, o número 7 multiplicado por outro (entre 1 e 10), ou seja, o multiplicador varia. Para isso, criaremos a variável multiplicador:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var multiplicador = 1;

</script>

Em seguida, usaremos mostra() para exibir o resultado das operações:

var multiplicador = 1;

mostra(7 \* multiplicador);

Retornaremos ao navegador, e abriremos o programa tabuada.html. É exibido para nós apenas o número 7 na tela. Entretanto, nossa intenção é que as multiplicações se repitam até o número 10. Criaremos uma situação em que, enquanto o multiplicador for menor ou igual a 10, o número 7 é multiplicado por ele:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var multiplicador = 1;

while(multiplicador <= 10) {

mostra(7 \* multiplicador);

}

</script>

Dessa forma, o JavaScript sempre entenderá o multiplicador como 1, e precisamos criar um mecanismo que indique que este número deve crescer, até o limite estabelecido. Para isso, declararemos que multiplicador é igual a multiplicador + 1:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var multiplicador = 1;

while(multiplicador <= 10) {

mostra(7 \* multiplicador);

multiplicador = multiplicador + 1

}

mostra("FIM");

</script>

Assim, quando o multiplicador atingir o limite (10), a condição de while() deixará de ser verdadeira e, então, o JavaScript poderá imprimir FIM. Vamos testar salvando o programa, retornando ao navegador e recarregando a página. Teremos a seguinte exibição:

7

14

21

28

35

42

49

56

63

70

FIM

Funcionou como gostaríamos, temos a tabuada de 7. Existe ainda outra maneira de aplicarmos a repetição, que passaremos a estudar agora. Além do while(), existe o for(), veremos como este funciona. Primeiro, removeremos o while() e o substituiremos pelo for(), que possui uma estrutura com três espaços:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var multiplicador = 1;

for(espaço1; espaço2; espaço3) {

mostra(7 \* multiplicador);

multiplicador = multiplicador + 1

}

mostra("FIM");

</script>

Cada um dos espaços é separado por um ponto e vírgula (;). Neste caso, a inicialização do multiplicador fica localizada no espaço1, diferentemente do while(), em que ficava fora:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

for(var multiplicador = 1; espaço2; espaço3) {

mostra(7 \* multiplicador);

multiplicador = multiplicador + 1

}

mostra("FIM");

</script>

No while(), havia uma condição para repetição - que o multiplicador fosse menor ou igual a 10. Em espaço2 do for, esta condição será inserida:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

for(var multiplicador = 1; multiplicador <= 10; espaço3) {

mostra(7 \* multiplicador);

multiplicador = multiplicador + 1

}

mostra("FIM");

</script>

Este pedaço do for() conterá a condição para o JavaScript saber se deve continuar a repetir. O terceiro espaço servirá para acrescer o multiplicador de + 1. Se não fizermos isto, a condição multiplicador <= 10 nunca será falsa.

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

for(var multiplicador = 1; multiplicador <= 10; multiplicador = multiplicador + 1) {

mostra(7 \* multiplicador);

}

mostra("FIM");

</script>

A estrutura do for() possui a mesma funcionalidade do while(), com a diferença de que ela é divida em três partes. A primeira é uma variável para inicializá-lo, a que será incrementada, o segundo contém a condição de repetição, a última parte do for() é o incremento da variável multiplicador, declarada no espaço1. Vamos salvar e recarregar a página. Teremos o mesmo resultado:

7

14

21

28

35

42

49

56

63

70

FIM

Com o while(), o mesmo resultado é obtido da seguinte forma:

var multiplicador = 1;

while(multiplicador <= 10) {

mostra(7 \* multiplicador);

multiplicador = multiplicador + 1;

}

Na primeira linha, o for() deixa claro tudo o que é necessário para definir a repetição, ou seja, a variável que será incrementada, a condição que a compara com o valor e, por último, o critério de incremento da variável.

Precisaremos de uma variável a ser incrementada, pois caso contrário a condição de repetição jamais será falsa e o programa nunca parará de repetir. No while() ocorre o mesmo, entretanto, a inicialização da variável é externa, e o incremento da variável acontece dentro de seu bloco.

Antes de continuarmos, veremos uma prática de programação que nos permite escrever menos linhas de código. Digamos por exemplo que tenhamos uma variável chamada total, que recebe 0:

var total = 0;

Se quisermos incrementar este total adicionando 1 ao seu valor, temos que declarar que a variável receberá o que já tem, **mais** 1:

var total = 0;

total = total + 1;

Assim, o total valerá 1. Se repetirmos esta operação:

var total = 0;

total = total + 1;

total = total + 1;

Ele passará a valer 2. Há um atalho para incrementarmos uma variável, sempre somando 1. Trata-se do total++:

var total = 0;

total++;

Esta ferramenta equivale à sintaxe:

total = total + 1;

É um atalho que nos permite escrever menos para realizarmos uma operação corriqueira. Portanto, substituiremos todos os multiplicador = multiplicador + 1 por multiplicador++:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

for(var multiplicador = 1; multiplicador <= 10; multiplicador++) {

mostra(7 \* multiplicador);

}

mostra("FIM");

</script>

Isso se chama **pós incremento**. Outra vantagem é quando trabalhamos com variáveis muito grandes. Vamos criar uma para teste, chamada quantidadeDePalavrasDoLivroQueEuCompreiAnoPassado:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

for(var multiplicador = 1; multiplicador <= 10; multiplicador++) {

mostra(7 \* multiplicador);

}

mostra("FIM");

var quantidadeDePalavrasDoLivroQueEuCompreiAnoPassado = 0;

</script>

Se quisermos incrementar + 1, teremos que utilizar novamente o nome da variável, e indicar que ela recebe ela mesma mais 1:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

for(var multiplicador = 1; multiplicador <= 10; multiplicador++) {

mostra(7 \* multiplicador);

}

mostra("FIM");

var quantidadeDePalavrasDoLivroQueEuCompreiAnoPassado = 0; quantidadeDePalavrasDoLivroQueEuCompreiAnoPassado = quantidadeDePalavrasDoLivroQueEuCompreiAnoPassado + 1;

</script>

Isto dá certo, mas podemos evitar de escrever tanto fazendo o seguinte:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

for(var multiplicador = 1; multiplicador <= 10; multiplicador++) {

mostra(7 \* multiplicador);

}

mostra("FIM");

var quantidadeDePalavrasDoLivroQueEuCompreiAnoPassado = 0; quantidadeDePalavrasDoLivroQueEuCompreiAnoPassado++;

</script>

Removeremos esta variável do nosso código, já que foi criada somente para teste. Adiante, faremos melhorias nos programas que criamos até o momento.

Episódio 3

Nesta aula, criaremos outro programa! Selecionaremos a opção "Salvar Como", e daremos ao novo arquivo o nome de media\_idades\_familiares.html. Apagaremos o trecho do programa referente à tabuada de 7, mantendo somente as funções pulaLinha() e mostra():

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

</script>

Nosso objetivo é calcular a média das idades de nossos familiares, e utilizaremos nomes e idades fictícios, os quais serão inseridos na forma de variáveis:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var idadePedro = 28;

var idadeMarta = 32;

var idadeJorge = 60;

var idadeBete = 22;

</script>

Para termos a médias entre as idades, precisaremos somá-las e dividir pela quantidade. Criaremos uma variável totalIdades:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var idadePedro = 28;

var idadeMarta = 32;

var idadeJorge = 60;

var idadeBete = 22;

var totalIdades = (idadePedro + idadeMarta + idadeJorge + idadeBete);

</script>

Em seguida, para calcularmos a média entre elas, criaremos a variável mediaIdades, que receberá totalIdades dividida por 4. Para imprimir, utilizaremos a função mostra():

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var idadePedro = 28;

var idadeMarta = 32;

var idadeJorge = 60;

var idadeBete = 22;

var totalIdades = (idadePedro + idadeMarta + idadeJorge + idadeBete);

var mediaIdades = totalIdades/4;

mostra(mediaIdades);

</script>

Salvaremos o programa, retornaremos ao navegador e selecionaremos as opções "Arquivo > Abrir arquivo... > media\_idade\_familiares.html", para abrirmos o programa. Obteremos o seguinte resultado:

35.5

Se quisermos ampliar a utilização do programa, veremos que cada família possui uma quantidade diferente de familiares. Se quisermos tornar o programa genérico, para que funcione com qualquer família, teremos sempre que alterar todas as variáveis, e não queremos esta solução.

Queremos que o programa funcione de modo a perguntar ao usuário a quantidade de familiares a serem analisados. O usuário insere a resposta e, em seguida, queremos que o programa pergunte as idades dos familiares. O número de vezes que o programa perguntará é igual ao número de familiares inseridos pelo usuário.

Assim, utilizaremos a função prompt(), pedindo para que seja lida uma informação do teclado, tantas vezes quanto houver familiares. Por exemplo, se há cinco familiares, as idades serão solicitadas cinco vezes.

Além disso, teremos que somar as idades inseridas para podermos, ao final, calcular a média. Apagaremos o código criado até o momento, mantendo somente as funções pulaLinha() e mostra():

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

</script>

A primeira informação que precisamos perguntar ao usuário é a quantidade de familiares, portanto, teremos uma variável chamada totalFamiliares:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares =

</script>

Para perguntarmos ao usuário, utilizaremos o prompt() com a seguinte pergunta "Quantidade de familiares?":

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = prompt("Quantidade de familiares?");

</script>

Já que queremos ler esta resposta ao prompt() como um número, utilizaremos o parseInt():

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = parseInt(prompt("Quantidade de familiares?"));

</script>

Com isto, temos o total de familiares. Para descobrirmos as idades de cada um, perguntaremos tantas vezes quanto o número de familiares. Primeiro, inseriremos a pergunta, e como leremos a resposta como um número, utilizaremos o parseInt():

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = parseInt(prompt("Quantidade de familiares?"));

var idade = parseInt(prompt("Informe idade do familiar"));

</script>

Salvaremos o programa e retornaremos ao navegador, onde recarregaremos a página. Surgirá um primeiro pop up solicitando o número de familiares, e digitaremos 3. A seguir, surgirá um pop up solicitando a idade do familiar. Já que temos 3 familiares, para que o programa funcione, precisamos que isso seja perguntado 3 vezes, entretanto, ao pressionarmos "Ok" veremos que só foi perguntado uma vez.

Isso acontece porque não indicamos ao programa que ele precisa fazer esta pergunta de acordo com a quantidade de familiares inserida. Para isso, criaremos um while():

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = parseInt(prompt("Quantidade de familiares?"));

while(? <= totalFamiliares) {

var idade = parseInt(prompt("Informe idade do familiar"));

}

</script>

Mas como o while() saberá que já perguntou uma, duas ou três vezes? Precisaremos de uma variável de incremento para que seja alterada a cada leitura de idade de um familiar, fazendo com que ela saiba quantas idades já leu. Então, criaremos a var numero que receberá 0:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = parseInt(prompt("Quantidade de familiares?"));

var numero = 0;

while(? <= totalFamiliares) {

var idade = parseInt(prompt("Informe idade do familiar"));

}

</script>

Enquanto numero for menor ou igual ao totalFamiliares, o prompt() "Informe idade do familiar" será executado:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = parseInt(prompt("Quantidade de familiares?"));

var numero = 0;

while(numero <= totalFamiliares) {

var idade = parseInt(prompt("Informe idade do familiar"));

}

</script>

Supondo que o total de familiares seja 3, e que já temos uma leitura, teremos var numero = 1:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = parseInt(prompt("Quantidade de familiares?"));

var numero = 1;

while(numero <= totalFamiliares) {

var idade = parseInt(prompt("Informe idade do familiar"));

}

</script>

Enquanto 1 for igual ou menor a 3, será feita a pergunta. Temos que incrementar numero para que, ao ultrapassar 3, parem de ser feitas perguntas. Ao final, imprimiremos a palavra FIM:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = parseInt(prompt("Quantidade de familiares?"));

var numero = 1;

while(numero <= totalFamiliares) {

var idade = parseInt(prompt("Informe idade do familiar"));

numero++;

}

mostra("FIM");

</script>

Assim, temos que para um determinado número de familiares será feita uma determinada quantidade de perguntas. Se tivermos 10 familiares, faremos 10 perguntas. Salvaremos o programa e recarregaremos a página. Isto fará com que surja o primeiro pop up, solicitando o número de familiares. Digitaremos 3, de modo que deverão ser feitas 3 perguntas.

O próximo pop up surgirá, solicitando a idade do familiar, e digitaremos 10. Surgirá um novo pop up, em que digitaremos 20, e por último, digitaremos 30. Ao pressionarmos "Ok" neste último pop up, não deverão surgir mais pop ups.

O programa funcionou, e após perguntar a quantidade de familiares e a idade destes, o programa exibiu FIM na tela. Conseguimos configurar as perguntas, precisamos em seguida programar o cálculo da média das idades. Este cálculo envolverá a soma de todas as idades inseridas. Criaremos uma variável chamada totalIdades, que iniciará em 0:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = parseInt(prompt("Quantidade de familiares?"));

var numero = 1;

while(numero <= totalFamiliares) {

var totalIdades = 0;

var idade = parseInt(prompt("Informe idade do familiar"));

totalIdades = totalIdades + idade;

numero++;

}

mostra("FIM");

</script>

Leremos a idade do usuário, e declararemos que totalIdades receberá totalIdades + idade. A ideia é guardarmos todas as idades em totalIdades, a qual deverá acumular todos os números inseridos. Em seguida, criaremos a variável mediaDasIdades, que recebe o totalIdades dividido pela quantidade de familiares, ou seja, totalFamiliares:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = parseInt(prompt("Quantidade de familiares?"));

var numero = 1;

while(numero <= totalFamiliares) {

var totalIdades = 0;

var idade = parseInt(prompt("Informe idade do familiar"));

totalIdades = totalIdades + idade;

numero++;

}

var mediaDasIdades = totalIdades/totalFamiliares

mostra("FIM");

</script>

Ao final, imprimiremos uma mensagem "A média das idades dos familiares é", concatenada à mediaDasIdades:

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var totalFamiliares = parseInt(prompt("Quantidade de familiares?"));

var numero = 1;

while(numero <= totalFamiliares) {

var totalIdades = 0;

var idade = parseInt(prompt("Informe idade do familiar"));

totalIdades = totalIdades + idade;

numero++;

}

var mediaDasIdades = totalIdades/totalFamiliares

mostra("A média das idades dos familiares é " + mediaDasIdades);

mostra("FIM");

</script>

Salvaremos e recarregaremos a página. Digitaremos uma quantidade de familiares igual a 3, e, em seguida, as respectivas idades: 10, 20 e 30. Obteremos o seguinte resultado:

A média das idades dos familiares é 10

FIM

A média deveria ter resultado em 20, portanto, não deu certo.

Como a inicialização do totalIdades está inserida no while(), todas as vezes que o while() for inicializado, a variável totalIdades será zerada, pois recebe 0. Assim, não serão somadas as idades, mas sim consideradas individualmente, até que o totalIdades receba somente o último valor a ser inserido, que no caso foi 30. Como havia 3 indivíduos, a divisão é 30/3, cujo resultado é 10.

Para solucionarmos isso, a inicialização da variável totalIdades não pode estar dentro do while(), pois este rodará várias vezes e, se em cada uma delas for zerado o valor, nada será acumulado. Salvaremos o programa e recarregaremos a página. Digitaremos 3 familiares, e as idades 10, 20 e 30. Obteremos o seguinte resultado:

A média das idades dos familiares é 20

Este processo, no qual precisamos acumular um valor durante uma repetição, é bastante comum. Podemos, por exemplo, utilizá-lo para calcular os salários de todos os funcionários de uma empresa.

Episódio 4

No editor de texto, abriremos novamente o programa de adivinhação, selecionando na barra de menu superior "File > Open... > jogo\_advinha.html". Em seguida, o abriremos no navegador selecionando "Arquivo > Abrir arquivo... > jogo\_advinha.html" Surgirá um pop up com a mensagem "Digite seu chute!". Digitaremos o número 5, e obteremos o seguinte resultado:

Você errou, o número pensado foi 6

Nosso objetivo será fornecer ao usuário três chances para descobrir o número. Retornaremos ao programa, no editor de texto. Precisamos analisar e identificar qual parte do código teremos que repetir. No caso, será a pergunta do chute, e todo o teste que verifica se é o número correto:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = Math.round(Math.random() \* 10);

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você acertou!");

} else {

mostra("Você errou, o número pensado foi " + numeroPensado);

}

</script>

Queremos repetir o bloco a cada chute do usuário. Para isso, criaremos um while():

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = Math.round(Math.random() \* 10);

while() {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você acertou!");

} else {

mostra("Você errou, o número pensado foi " + numeroPensado);

}

}

</script>

Como queremos dar 3 chances, o while() será executado sempre que algo for menor ou igual a este número, no caso, a quantidade de tentativas. Vamos criar a variável tentativas:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = Math.round(Math.random() \* 10);

var tentativas = 1;

while() {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você acertou!");

} else {

mostra("Você errou, o número pensado foi " + numeroPensado);

}

}

</script>

Assim, o while() terá como parâmetro tentativas, desde que esta corresponda a um número menor ou igual a 3:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = Math.round(Math.random() \* 10);

var tentativas = 1;

while(tentativas <= 3) {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você acertou!");

} else {

mostra("Você errou, o número pensado foi " + numeroPensado);

}

}

</script>

Enquanto o número de tentativas for menor ou igual a 3, o bloco while() será inteiramente executado. Para que o número de tentativas seja acumulado, precisaremos incrementar a variável tentativas:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = Math.round(Math.random() \* 10);

var tentativas = 1;

while(tentativas <= 3) {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você acertou!");

} else {

mostra("Você errou, o número pensado foi " + numeroPensado);

}

tentativas++;

}

</script>

Salvaremos e recarregaremos a página. Aparecerá um pop up solicitando um chute. Digitaremos qualquer valor, e se o resultado for negativo, surgirá novamente um pop up. Teremos que inserir um número, em seguida surgirá mais um pop up e, ao digitarmos um número incorreto, não haverá mais tentativas, e teremos o seguinte resultado impresso:

Você errou, o número pensado foi 6

Você errou, o número pensado foi 6

Você errou, o número pensado foi 6

Não faz sentido informarmos ao usuário que ele errou e, ao mesmo tempo, imprimir qual é o número pensado. Assim, alteraremos a mensagem para simplesmente "Você errou!":

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = Math.round(Math.random() \* 10);

var tentativas = 1;

while(tentativas <= 3) {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você acertou!");

} else {

mostra("Você errou!");

}

tentativas++;

}

</script>

Salvaremos e recarregaremos a página. Ao realizarmos novos testes, com valores errados em todas as tentativas, obteremos o seguinte resultado:

Você errou!

Você errou!

Você errou!

O programa está funcionando como gostaríamos. Adicionaremos a função mostra() para que, ao final, seja impressa a mensagem FIM:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = Math.round(Math.random() \* 10);

var tentativas = 1;

while(tentativas <= 3) {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você acertou!");

} else {

mostra("Você errou!");

}

tentativas++;

}

mostra("FIM");

</script>

Salvaremos e retornaremos ao navegador. Recarregaremos a página e digitaremos números para realizar mais um teste. Erraremos em todas as tentativas e obteremos o seguinte resultado:

Você errou!

Você errou!

Você errou!

FIM

Mas e se acertarmos, por exemplo, na segunda tentativa? Veremos que ele ainda assim insistirá em uma terceira tentativa. Por isso, faremos com que o programa pare de perguntar assim que o usuário acertar o número pensado. Vamos retornar ao código.

A primeira coisa a fazer é alterar a grafia das frases, deixando "errou" e "acertou" em letras maiúsculas, e alterando a frase para quando o usuário acerta:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = Math.round(Math.random() \* 10));

var tentativas = 1;

while(tentativas <= 3) {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você ACERTOU, o número pensado era " + numeroPensado);

} else {

mostra("Você ERROU!");

}

tentativas++;

}

mostra("FIM");

</script>

Nosso objetivo é que no caso de um acerto com uma ou duas tentativas ainda disponíveis, fazer com que o JavaScript saia do loop e, assim, pare de solicitar chutes. Para isso, podemos utilizar a função break():

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = Math.round(Math.random() \* 10);

var tentativas = 1;

while(tentativas <= 3) {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você ACERTOU, o número pensado era " + numeroPensado);

break;

} else {

mostra("Você ERROU!");

}

tentativas++;

}

mostra("FIM");

</script>

Ao se deparar com o break(), o JavaScript sai do loop e parte para a próxima instrução, que no caso é mostra("FIM"). Para testarmos, definiremos o numeroPensado como 4:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = 4;

var tentativas = 1;

while(tentativas <= 3) {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você ACERTOU, o número pensado era " + numeroPensado);

break;

} else {

mostra("Você ERROU!");

}

tentativas++;

}

mostra("FIM");

</script>

Salvaremos e recarregaremos a página. Erraremos a primeira tentativa e, na segunda, digitaremos 4 - sabendo que é este o resultado correto. Ao fazermos isso, temos a seguinte mensagem:

Você ERROU!

Você ACERTOU, o número pensado era 4

FIM

Poderíamos, alternativamente, inserir no loop a informação tentativas = 4:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = 4;

var tentativas = 1;

while(tentativas <= 3) {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você ACERTOU, o número pensado era " + numeroPensado);

tentativas=4;

} else {

mostra("Você ERROU!");

}

tentativas++;

}

mostra("FIM");

</script>

Desta forma, inevitavelmente o loop seria quebrado e obteríamos o mesmo resultado. Ou seja, podemos tanto utilizar o break() quanto alterar a variável da condição, conferindo a ela um valor para o qual ela cessará de repetir. Em nosso código, manteremos o break():

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = 4;

var tentativas = 1;

while(tentativas <= 3) {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você ACERTOU, o número pensado era " + numeroPensado);

break;

} else {

mostra("Você ERROU!");

}

tentativas++;

}

mostra("FIM");

</script>

Assim, temos um código inteligente, que dá mais chances ao usuário para que ele possa adivinhar o número. Retornaremos a fórmula do numeroPensado como estava, antes de alterarmos para 4:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase) {

document.write(frase);

pulaLinha();

}

var numeroPensado = Math.round(Math.random() \* 10);

var tentativas = 1;

while(tentativas <= 3) {

var chute = parseInt(prompt("Digite seu chute!"));

if(chute == numeroPensado) {

mostra("Você ACERTOU, o número pensado era " + numeroPensado);

break;

} else {

mostra("Você ERROU!");

}

tentativas++;

}

mostra("FIM");

</script>

Salvaremos e recarregaremos a página. Podemos tentar acertar mais uma vez para conferir se o código está adequado. As estruturas de repetição são capazes de resolver muitos problemas no mundo da programação.

Episódio 5

Nesta aula, veremos mais um conceito para nos ajudar a fixar os loops. Salvaremos nosso arquivo do jogo de adivinhação como um novo, chamado estrelas.html. Apagaremos os trechos referentes ao jogo, mantendo somente as funções pulaLinha() e mostra():

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase){

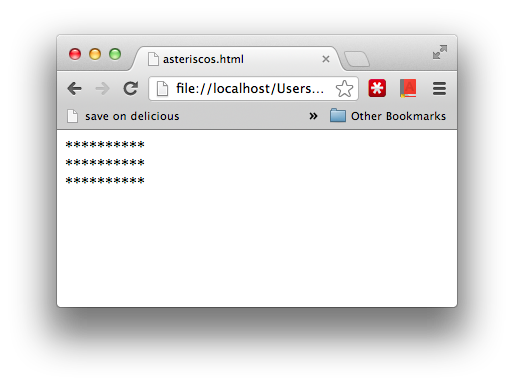
document.write(frase);

pulaLinha();

}

</script>

Supondo que queremos imprimir na tela três linhas, com dez asteriscos cada:



Sabemos que queremos repetir uma mesma linha, portanto teríamos:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase){

document.write(frase);

pulaLinha();

}

mostra("\*\*\*\*\*\*\*\*\*\*");

</script>

Como faremos para repetir três vezes? Podemos utilizar, por exemplo, o for(), em que a inicialização será var Linha = 1, a condição linha <= 3 e, por fim, o incremento linha++:

<meta charset="UTF-8">

<script>

function pulaLinha() {

document.write("<br>");

document.write("<br>");

}

function mostra(frase){

document.write(frase);

pulaLinha();

}

for(var linha = 1; linha <= 3; linha++) {

mostra("\*\*\*\*\*\*\*\*\*\*");

}

</script>

Lembrando que o for() conta com três partes - a primeira é a variável de inicialização, a que será incrementada, a segunda é a condição para sabermos quando ele deve parar de repetir o que está em seu bloco, por fim, temos o incremento.

Salvaremos e retornaremos ao navegador. Na barra de menu superior, selecionaremos "Arquivo > Abrir arquivo... > estrelas.html", e ao abrirmos o programa veremos que foram exibidas três linhas com asteriscos, conforme havíamos programado:

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

Como a função pulaLinha() contêm dois pulos, a alteraremos para que haja apenas uma quebra de linha:

<script>

function pulaLinha() {

document.write("<br>");

}

function mostra(frase){

document.write(frase);

pulaLinha();

}

for(var linha = 1; linha <= 3; linha++) {

mostra("\*\*\*\*\*\*\*\*\*\*");

}

</script>

Salvaremos e recarregaremos a página, assim, teremos o seguinte resultado:

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

Nosso objetivo não é digitar os dez asteriscos, mas sim, queremos digitar apenas um e fazer com que ele seja repetido em sua linha para, em seguida, ser replicado em mais linhas:

<script>

function pulaLinha() {

document.write("<br>");

}

function mostra(frase){

document.write(frase);

pulaLinha();

}

for(var linha = 1; linha <= 3; linha++) {

mostra("\*");

}

</script>

Só que, desta forma, se salvarmos e recarregarmos a página, veremos que há três linhas com apenas um asterisco em cada, ou seja, não foi feita a replicação para 10 asteriscos:

\*

\*

\*

Nosso objetivo é que, com um mostra() contendo apenas um asterisco, sejam impressas 3 linhas com 10 asteriscos em cada. Como o mostra pula uma linha automaticamente, utilizaremos em seu lugar o document.write():

<script>

function pulaLinha() {

document.write("<br>");

}

function mostra(frase){

document.write(frase);

pulaLinha();

}

for(var linha = 1; linha <= 3; linha++) {

document.write("\*");

}

</script>

Para criarmos os 10 asteriscos utilizaremos um for(), em que há uma variável coluna que, enquanto for menor que 10, será incrementada como coluna++. Dentro do bloco for() inseriremos nosso document.write():

<script>

function pulaLinha() {

document.write("<br>");

}

function mostra(frase){

document.write(frase);

pulaLinha();

}

for(var linha = 1; linha <= 3; linha++) {

for(var coluna = 1; coluna < 10; coluna++) {

document.write("\*");

}

}

</script>

Isto é, teremos um for() dentro de outro, sendo que o primeiro é responsável por criar a primeira, segunda e terceira linhas. Já o for() que está inserido nele é responsável por repetir as colunas. Para cada linha criada pelo primeiro, o segundo imprimirá 10 colunas, criando assim 3 linhas com 10 asteriscos cada, como gostaríamos. Salvaremos o programa e recarregaremos a página, após o qual obteremos o seguinte resultado:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Não funcionou. Temos uma única linha com 30 asteriscos, porque não pedimos ao programa que pulasse uma linha após a conclusão do segundo for(). Sendo assim, nosso código ficará da seguinte forma:

<script>

function pulaLinha() {

document.write("<br>");

}

function mostra(frase){

document.write(frase);

pulaLinha();

}

for(var linha = 1; linha <= 3; linha++) {

for(var coluna = 1; coluna < 10; coluna++) {

document.write("\*");

}

pulaLinha();

}

</script>

Salvaremos e recarregaremos a página. Como vemos, temos 10 asteriscos em cada linha, sendo que há 3 linhas. Mas será que temos **10 asteriscos** mesmo? Ao observarmos atentamente, veremos que cada linha conta apenas com 9 asteriscos, não 10. Isso ocorre pois quando construímos as colunas, colocamos apenas que a coluna < 10, quando na verdade deveria ser coluna <= 10:

<script>

function pulaLinha() {

document.write("<br>");

}

function mostra(frase){

document.write(frase);

pulaLinha();

}

for(var linha = 1; linha <= 3; linha++) {

for(var coluna = 1; coluna <= 10; coluna++) {

document.write("\*");

}

pulaLinha();

}

</script>

Isso é um erro comum, que pode ocorrer quando trabalhamos com loops. Salvaremos o programa e recarregaremos a página, e o resultado será:

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

Funcionou! Adiante, veremos outras ferramentas que melhoram nossas aplicações.

# Aula 8: Intereja de maneira diferente com o Usuário

Episódio 1.

Nesta aula aprenderemos a capturar e executar os códigos de nossos programas de uma maneira diferente e elegante. No editor de texto, criaremos um arquivo utilizando o atalho "Ctrl + N", e o salvaremos com o nome adivinha\_mais.html. Ele será um jogo de adivinhação diferente daquele que criamos anteriormente.

Em primeiro lugar, colocaremos no código o meta charset="UTF-8":

<meta charset="UTF-8">

<script>

</script>

Retornaremos ao navegador e abriremos nosso programa. Nosso objetivo é exibir, no canto superior esquerdo da tela, um campo de entrada para o usuário, acompanhado de um botão. Criaremos isto em HTML, quando utilizamos o <input/>, fazendo com que seja aberto um campo para entrada de texto no navegador.

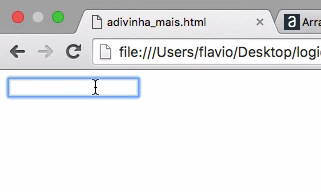
<meta charset="UTF-8">

<input/>

<script>

</script>

Ao salvarmos o programa e recarregarmos a página, veremos o seguinte:



A palavra "*input*" vem do inglês, e sua tradução é "entrada". Em seguida, criaremos um botão, e a palavra em inglês para isso é "*button*", assim, vamos usá-la em nosso código:

<meta charset="UTF-8">

<input/>

<button></button>

<script>

</script>

Esta *tag*, como podemos observar, abre e fecha, diferentemente da <input>. Dentro da *tag* <button> inseriremos a frase "Compare com o meu segredo":

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

</script>

Salvaremos e recarregaremos a página. Teremos na tela, no canto superior esquerdo, o campo para inserir texto e, em seu lado direito, o botão com o texto conforme configuramos. Nosso objetivo é que o usuário digite um número e, ao apertar o botão, que ele seja comparado ao numeroPensado para que se verifique se ele acertou ou errou. Para este novo programa, chamaremos o numeroPensado de segredo.

Neste capítulo, aprenderemos o mínimo possível para que possamos interagir com o usuário por meio da caixa de texto, e do botão. O <input> e o <button> estão inseridos em qual mundo? HTML ou JavaScript? Eles estão no mundo **HTML**, pois estão fora da tag <script>. Se as inserirmos na tag <script> o JavaScript nem seria capaz de compreender isso.

Precisaremos descobrir uma maneira de acessar estas tags do mundo HTML a partir do mundo JavaScript. Para isso, criaremos uma variável input:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var input

</script>

Ela receberá a função document.querySelector() - atenção para o "S" maiúsculo. Utilizaremos a querySelector() para inserir aquilo que está no mundo HTML na variável input. Ela receberá como parâmetro o nome da *tag* que desejamos utilizar:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var input = document.querySelector("input");

</script>

Isso permite trabalharmos com o <input> no universo JavaScript. O nome da variável não importa, podemos escolher qualquer um, apenas precisamos nos atentar ao nome da tag, que deve ser passado exatamente como é para o querySelector() poder acessá-lo. No caso, chamaremos a variável de input apenas por motivos didáticos.

Para sabermos qual o valor que está inserido na variável, utilizaremos o input.value:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var input = document.querySelector("input");

input.value

</script>

Podemos criar um alert() para imprimi-lo, mas queremos comparar se o valor do input é igual ao segredo. Por isso, teremos uma variável segredo que receberá o valor 5:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

input.value

</script>

Para testar se o que o usuário digitou é igual ao segredo, utilizaremos o if(), se este for o caso, será exibido um alert() com a mensagem "Você ACERTOU!", caso contrário, utilizaremos o else para que seja exibido Você ERROU!!!!!!!!:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

</script>

Recapitulando:

* No HTML temos um campo de *input*, utilizado para receber textos;
* Há também um botão;
* Nosso objetivo é, ao carregarmos a página, digitarmos um número em nosso campo, clicarmos no botão, e somente então, que o programa compare o número inserido ao segredo.

Salvaremos o programa e recarregaremos a página. Surgirá um pop up com a mensagem de que erramos antes mesmo de termos clicado no botão para verificar o número. Isso acontece porque o programa está fazendo o teste com um valor vazio. Podemos confirmar que não há valor criando um alert() para o input.value:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

alert(input.value);

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

</script>

Se salvarmos e recarregarmos a página, veremos que o valor de input não existe, pois surgirá apenas um alerta, sem nenhuma mensagem. Feito isso, poderemos remover este alert() de nosso código.

O problema é que precisamos sinalizar ao programa que o teste entre os números só pode ser feito a partir do momento em que o usuário clicar no botão, e não ao carregar a página. Como sabemos, é possível "guardar" um código para chamá-lo posteriormente, por meio de **funções**.

A primeira coisa a fazer é garantir que o teste **não seja executado** ao carregar a página. Criaremos uma *function* para a verificação:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

function verifica() {

}

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

</script>

Ela não recebe parâmetros. Em seu bloco, inseriremos todo o mecanismo de teste do número, da seguinte forma:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

}

</script>

Salvaremos o programa e recarregaremos a página. Não foi exibido nenhum alerta, pois o verifica() está guardado para ser executado posteriormente. No navegador, clicaremos na opção "Visualizar" na barra de menu superior, e selecionaremos "Desenvolvedor > Console JavaScript". No campo de *input* digitaremos o número 4. No console, chamaremos a função verifica() que acabamos de criar, digitando:

verifica()

Ao pressionarmos "Enter", aparecerá um pop up com a mensagem "Você errou!". Não é interessante para o usuário ter que abrir o console para fazer esta verificação, nosso objetivo é que o verifica() seja chamado sempre que o botão for clicado. Para isso, precisaremos trabalhar com o <button> do HTML no JavaScript. Isso porque queremos associar a função verifica() ao clique do botão.

Declararemos uma variável chamada button e daremos ao document.querySelector() o parâmetro button - o nome da tag:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

}

var button = document.querySelector("button");

</script>

Para associar ao botão a execução do verifica() utilizaremos o button.onclick(), em português, "*on click*" significa "no clicar", ou seja, queremos que a verificação seja feita ao clique:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

}

var button = document.querySelector("button");

button.onclick = verifica();

</script>

Salvaremos o programa e recarregaremos a página. Surgirá um pop up imediatamente, o que não é o resultado desejado. Isso acontece pois, da maneira como declaramos, o onclick recebe verifica(), mas quando utilizamos os parênteses estamos chamando a função, portanto ela é que será executada ao recarregar a página.

É necessário fazer com que o verifica() esteja inserido no onclick. Retornaremos ao navegador e abriremos o console. Digitaremos:

verifica()

E, ao pressionarmos "Enter" veremos que a função é executada pois surge um pop up. Mas o que acontece se escrevermos somente verifica?

verifica()

undefined

verifica

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!"});

} else {

alert("Você ERROU!!!!!!!!");

}

}

Ao fazermos isso, é exibido o código da função. Assim, há uma grande diferença entre utilizar os parênteses, ou seja, executar a função, e não utilizá-los, situação em que teremos acesso ao código da função. Para solucionarmos isto, retornaremos ao código e, quando formos remover os parênteses do verifica() em button.onclick, teremos:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Dessa forma, toda a informação contida em verifica, isto é, o código, está vinculado ao onclick do botão. Assim, ele estará preparado para executar a verificação sempre que o botão for clicado.

Salvaremos o programa e retornaremos ao navegador. Fecharemos o console e recarregaremos a página. Inseriremos o número 3 no *input* e clicaremos no botão. Surgirá um pop up com a mensagem "Você errou!". Ao colocarmos o número correto, no caso 5, e clicarmos no botão, aparecerá um pop up com a mensagem "Você acertou!".

No mundo JavaScript, conseguimos capturar os elementos do mundo HTML, verificar o valor destes elementos, e associar a execução de uma função a um clique de botão. Vimos que, ao abrirmos o console do navegador e fazermos o verifica(), estamos executando a função de forma manual. No entanto, se digitarmos verifica e pressionarmos "Enter", é impresso o código da função.

Como nosso objetivo é executar a função ao clique do botão, removemos os parênteses no código, fazendo com que ela fique inserida no onclick e, quando clicamos no botão do navegador, é ele quem insere os parênteses para chamar a função. Esta estratégia é mais interessante para o usuário.

Episódio 2

A próxima melhoria que faremos em nosso programa implica em fazermos com que o campo de texto seja esvaziado quando o usuário errar uma tentativa. Para isso, após a função verifica(), declararemos que o input.value receberá uma string em branco:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

input.value = "";

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Da mesma forma como conseguimos pegar o <input> do mundo HTML, também conseguimos lhe conferir um valor a partir do JavaScript. Salvaremos o programa e recarregaremos a página. Inseriremos 4 e clicaremos no botão, e surgirá um pop up informando que o número está incorreto. Ao clicarmos no botão Ok", veremos que o campo para digitação de texto foi limpo.

Em seguida, daremos um foco no campo de texto sempre que o usuário errar o número utilizando a função input.focus():

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Salvaremos e recarregaremos a página. Assim, ao digitarmos um palpite incorreto, e após o surgimento do pop up, a caixa de texto ficará em foco, ganhando uma moldura colorida, indicando que pode ser feita mais uma tentativa. Isso melhora a experiência do usuário.

Nosso objetivo neste curso é aprender a lógica de programação, com noções básicas de HTML e JavaScript. Conseguimos aprender como este interage de forma dinâmica com o HTML, deixando nosso programa com aspecto profissional.

Até agora trabalhamos com um segredo fixo, em seguida, faremos com que ele seja mutável, utilizando o Math.random() e o Math.round():

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = Math.round(Math.random() \* 10);

var input = document.querySelector("input");

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Salvaremos o programa e recarregaremos a página. Podemos digitar diversos números para realizar testes. Para aprimorá-lo, faremos com que a caixa de texto ganhe foco sempre que recarregarmos a página, com o input.focus() logo após chamarmos o input para o mundo JavaScript:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = Math.round(Math.random() \* 10);

var input = document.querySelector("input");

input.focus();

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Salvaremos e recarregaremos a página, e desta vez o campo de texto está em foco e com a borda colorida. Neste capítulo criamos um ambiente mais agradável para captação dos dados do usuário.

# Aula 9: Trabalhe com Muitos Dados

Episódio 1

Continuaremos a trabalhar com o programa advinha\_mais.html, como na aula anterior. Nosso programa faz uma comparação entre aquilo que o usuário digita no campo de input e o **segredo** estabelecido em nosso código. Ao rodar, ele processa uma série de instruções:

* Em primeiro lugar, ele gera um número aleatório dentro da variável segredo;
* Em seguida, ele busca o input do mundo HTML para o JavaScript, fazendo com que o campo de texto ganhe foco, em razão do input.focus();
* Posteriormente, declaramos a função verifica(), a qual recebe o valor digitado pelo usuário no input e o compara com o segredo. Se correto, é exibido o alerta "Você acertou", caso contrário, surge um alerta com a mensagem "Você errou!";
* Acertando ou errando, o campo de texto é limpo sempre que um número é digitado, e é exibido um pop up. Além disso, este campo limpo também ganha foco; e
* Como queremos que a função seja executada somente ao clique do botão, criamos uma variável button, que busca esta funcionalidade do mundo HTML, e a associamos ao clique, por meio de onclick.

Nossa intenção agora é aumentar as chances de acerto para o usuário. Criaremos uma série de variáveis, cada uma contendo um valor de segredo diferente:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo1 = 5;

var segredo2 = 6;

var segredo3 = 7;

var segredo = Math.round(Math.random() \* 10);

var input = document.querySelector("input");

input.focus();

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Temos portanto três segredos, e queremos testar se o palpite do usuário, inserido no campo de texto, é igual ao segredo1, segredo2 ou segredo3. Poderíamos fazer isto copiando a fórmula que já criamos e a utilizando para cada uma das variáveis, da seguinte forma:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo1 = 5;

var segredo2 = 6;

var segredo3 = 7;

var segredo = Math.round(Math.random() \* 10);

var input = document.querySelector("input");

input.focus();

function verifica() {

if(input.value == segredo1) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

if(input.value == segredo2) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

if(input.value == segredo3) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Salvaremos, retornaremos ao navegador e recarregaremos a página. Digitaremos 3 e pressionaremos o botão ao lado. Surgirá um pop up com "Você errou!", pressionaremos "Ok", mas o pop up aparecerá novamente, e assim sucessivamente, todas as vezes em que pressionamos "Ok", até esgotarmos todos os três segredos.

Se digitarmos, por exemplo, 5, acertamos o primeiro segredo, mas o programa ainda assim conferirá o input com os demais, ou seja, surgirá um primeiro pop up com a mensagem "Você acertou!", seguido de mais três pop ups com "Você errou!".

Poderíamos acertar um **ou** outro segredo e, se tivéssemos mais segredos, seria necessário criar mais variáveis, além de fazer mais cópias do nosso if(). Esta não é a solução ideal, pois quanto mais segredos tivermos mais linhas de código serão necessárias. Então vamos editar o código para deixá-lo como estava antes de criarmos os segredos adicionais, e estabeleceremos o valor do segredo em 5:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = 5;

var input = document.querySelector("input");

input.focus();

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Faremos com que a variável segredo não guarde somente um valor, mas uma lista com diversos números. **Não podemos** fazer isso da seguinte forma:

var segredo = 5 = 6 = 10 = 20;

Isto é **incorreto**. Precisaremos utilizar um novo tipo de dado, chamado ***array***, inserindo os valores entre **colchetes** (**[]**):

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredo = [];

var input = document.querySelector("input");

input.focus();

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Se a variável conterá vários segredos, temos como boa prática de programação colocar seu nome no plural, portanto, de var segredo passaremos para var segredos:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [];

var input = document.querySelector("input");

input.focus();

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Em seguida, inseriremos alguns valores para segredos:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2];

var input = document.querySelector("input");

input.focus();

function verifica() {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Dentro de um *array*, separamos os conteúdos por vírgula (**,**). Ao clicarmos em verifica(), o número inserido será comparado individualmente com cada um dos valores contidos no *array* de segredos. Retornaremos ao navegador e acessaremos o console, em que declararemos a variável segredos.

Para selecionar o console do navegador, basta clicar em "Visualizar" na barra de menu superior e, em seguida, "Desenvolvedor > Console JavaScript".

var segredos = [5,7,10,2]

Ao pressionaremos "Enter", surgirá a palavra undefined abaixo:

var segredos = [5,7,10,2]

undefined

Se escrevermos em seguida a palavra segredos e apertarmos "Enter" de novo, nosso *array* será impresso:

var segredos = [5,7,10,2]

undefined

segredos

[5,7,10,2]

E se quisermos acessar somente um elemento do *array*? O primeiro, por exemplo? Para fazermos isso, teremos que passar a sua posição para o console. Como queremos o primeiro, utilizaremos o número 1 e o inseriremos entre os colchetes:

var segredos = [5,7,10,2]

undefined

segredos

[5,7,10,2]

segredos[1]

Clicando em "Enter", veremos que é exibido o número 7, isto é, a segunda posição:

var segredos = [5,7,10,2]

undefined

segredos

[5,7,10,2]

segredos[1]

7

Isto aconteceu porque no JavaScript - e em algumas outras linguagens de programação -, a primeira posição é representada pelo número 0. Sendo assim, se quisermos isolar o primeiro elemento do *array*, precisaremos utilizá-lo:

var segredos = [5,7,10,2]

undefined

segredos

[5,7,10,2]

segredos[1]

7

segredos[0]

5

... E assim por diante. Neste *array*, temos um total de quatro posições, portanto, elas são representadas pelos números 0, 1, 2 e 3. Se digitarmos segredos[4] e pressionarmos "Enter" diversas vezes, ele retornará undefined:

var segredos = [5,7,10,2]

undefined

segredos

[5,7,10,2]

segredos[1]

7

segredos[0]

5

segredos[1]

7

segredos[3]

2

segredos[4]

undefined

Aprendemos a criar um *array* no JavaScript e a acessar cada elemento individualmente. Para fazermos a verificação, a função deverá percorrer cada elemento do *array*. No console do navegador, apagaremos tudo que escrevemos até este ponto e digitaremos var segredos = [5,7,10,2], em seguida pressionaremos "Enter" e teremos o seguinte resultado:

var segredos = [5,7,10,2];

undefined

Se o palpite for, por exemplo, 4, é necessário examinar os segredos da seguinte forma:

var segredos = [5,7,10,2];

undefined

segredos[0] == 4

false

O que fizemos foi digitar segredos e acompanhar da posição que queremos analisar, no caso, 0. Como vimos, o duplo sinal de igual (**==**) representa igualdade, diferente de quando é utilizado sozinho, que lemos como "recebe". Assim, estamos perguntando se o número da posição 0 é igual a 4. Clicando em "Enter", é feita a verificação, e teremos como resposta, false. Tentaremos outra posição, a 1:

var segredos = [5,7,10,2];

undefined

segredos[0] == 4

false

segredos[1] == 4

false

E assim por diante, até a posição 3:

var segredos = [5,7,10,2];

undefined

segredos[0] == 4

false

segredos[1] == 4

false

segredos[2] == 4

false

segredos[3] == 4

false

Se formos testar o número 7, precisamos digitar toda esta operação novamente:

var segredos = [5,7,10,2];

undefined

segredos[0] == 4

false

segredos[1] == 4

false

segredos[2] == 4

false

segredos[3] == 4

false

segredos[0] == 7

false

segredos[1] == 7

true

Como o número da segunda posição é 7, obtivemos um resultado verdadeiro. A verificação deve começar na posição 0 e seguir até o limite da lista, que em nosso caso é a posição 3.

Em regra, o número de posições equivale ao de elementos menos um, por exemplo, quando temos 4 elementos, o número de posições será 4 - 1, portanto, 3.

Alternativamente, poderíamos ter uma variável posicao que recebe o valor 0 e, em seguida, faríamos a verificação declarando:

segredos[posicao] == 9

Estamos perguntando se o segredo da posição 0 é o número 9 e, no caso, não é:

var segredos = [5,7,10,2];

undefined

segredos[0] == 4

false

segredos[1] == 4

false

segredos[2] == 4

false

segredos[3] == 4

false

segredos[0] == 7

false

segredos[1] == 7

true

posicao = 0

0

segredos[posicao] == 9

false

Assim, faremos a incrementação da posição, que passará a ser posicao++, e de 0, ela foi para 1. Testaremos novamente:

var segredos = [5,7,10,2];

undefined

segredos[0] == 4

false

segredos[1] == 4

false

segredos[2] == 4

false

segredos[3] == 4

false

segredos[0] == 7

false

segredos[1] == 7

true

posicao = 0

0

segredos[posicao] == 9

false

posicao++

0

segredos[posicao] == 9

false

Acabamos de testar a primeira posição, agora incrementaremos a variável mais uma vez para testarmos a próxima. Em seguida, realizaremos um novo teste, com a variável posicao, considerando que agora ela passou a equivaler à segunda posição:

var segredos = [5,7,10,2];

undefined

segredos[0] == 4

false

segredos[1] == 4

false

segredos[2] == 4

false

segredos[3] == 4

false

segredos[0] == 7

false

segredos[1] == 7

true

posicao = 0

0

segredos[posicao] == 9

false

posicao++

0

segredos[posicao] == 9

false

posicao++

1

segredos[posicao] == 9

false

Limparemos o console novamente e realizaremos novos testes, com o número 10:

var segredos = [5,7,10,2];

undefined

var posicao = 0;

undefined

segredos[posicao] == 10

false

posicao++

0

segredos[posicao] == 10

false

posicao++

1

segredos[posicao] == 10

true

Temos que fazer uma repetição para testarmos todas as posições. Retornaremos ao código e, na função verifica(), criaremos um for(), lembrando que ele conta com três espaços:

for(espaco1; espaco2; espaco3)

Deste modo, nossa verificação contida no if() será inserida neste for():

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2];

var input = document.querySelector("input");

input.focus();

function verifica() {

for(espaco1; espaco2; espaco3) {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Por que colocar dentro do for()? Porque precisamos repetir a verificação do que foi digitado, com o segredo, para cada item do nosso *array*.

No for(), declararemos uma variável chamada posicao, que inicialmente recebe o valor 0. A condição para repetição é que a posicao seja menor do que 4 e que, enquanto não atingir este limite, continue a se repetir. Por fim, a variável será incrementada, portanto utilizaremos posicao++:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2];

var input = document.querySelector("input");

input.focus();

function verifica() {

for(var posicao = 0; posicao < 4; posicao++) {

if(input.value == segredo) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Assim, quando posicao alcançar 4, o JavaScript perguntará se este número é menor que 4, e como não é, ele deixará de repetir o teste. Isso porque não existe a posição 4 em nosso *array*. Ao fazermos o teste, agora não o compararemos mais com segredo, mesmo porque esta variável não existe mais. Em seu lugar, inseriremos o *array* de segredos:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2];

var input = document.querySelector("input");

input.focus();

function verifica() {

for(var posicao = 0; posicao < 4; posicao++) {

if(input.value == segredos) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Além disso, faremos com que a comparação seja feita na posição escolhida, aquela determinada pelo nosso for():

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2];

var input = document.querySelector("input");

input.focus();

function verifica() {

for(var posicao = 0; posicao < 4; posicao++) {

if(input.value == segredos[posicao]) {

alert("Você ACERTOU!");

} else {

alert("Você ERROU!!!!!!!!");

}

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Salvaremos o programa e recarregaremos a página. Digitaremos 3 e pressionaremos o botão, o que fará com que surjam quatro pop ups, e em todos eles seremos informados de que erramos o número.

Em seguida, digitaremos 10, e como este número está na posição 2, ele deve testar duas vezes e acertar na terceira. Surge um primeiro pop up com a mensagem de erro, um segundo com a mesma mensagem, e um terceiro informando que acertamos, mas ele continua realizando o teste na última posição - e não queremos que isto aconteça.

Se acertarmos, não faz sentido continuarmos a verificação. Por isso, utilizaremos a função break:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2];

var input = document.querySelector("input");

input.focus();

function verifica() {

for(var posicao = 0; posicao < 4; posicao++) {

if(input.value == segredos[posicao]) {

alert("Você ACERTOU!");

break;

} else {

alert("Você ERROU!!!!!!!!");

}

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Salvaremos o programa e recarregaremos a página. Digitaremos o número 7, que corresponde a posição 1, portanto o primeiro teste retorna um erro e, no segundo, acertamos. Após isso, não são feitos mais testes - nosso código funcionou!

Outro ponto em nosso programa é que ele exibe um pop up com mensagem para cada erro cometido, e não queremos que isto aconteça. Nosso objetivo é que seja exibida uma única mensagem. Por isto, removeremos o else que contém a mensagem de erro:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2];

var input = document.querySelector("input");

input.focus();

function verifica() {

for(var posicao = 0; posicao < 4; posicao++) {

if(input.value == segredos[posicao]) {

alert("Você ACERTOU!");

break;

}

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Salvaremos o programa e recarregaremos a página. Digitaremos 2 e clicaremos no botão, e receberemos somente um aviso de acerto, conforme pretendido. Em seguida, usaremos 6 que, como sabemos, não está em nosso *array*. Pressionaremos o botão, e nada acontece. Quando o usuário comete um erro não é exibida nenhuma mensagem.

Um detalhe ao qual precisamos estar atentos é: se alterarmos o número de elementos em um *array*, precisaremos alterar também nosso for(), para que os testes sejam feitos até o limite da nova posição. Por exemplo, se em vez de 4 tivéssemos 5 elementos em nosso *array*, nosso código passaria a ser o seguinte:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2,3];

var input = document.querySelector("input");

input.focus();

function verifica() {

for(var posicao = 0; posicao < 5; posicao++) {

if(input.value == segredos[posicao]) {

alert("Você ACERTOU!");

break;

}

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Para não termos que alterar nosso for() todas as vezes que quisermos adicionar um novo elemento, poderemos utilizar o .length para termos sempre o *array* atual no for():

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2,3];

var input = document.querySelector("input");

input.focus();

function verifica() {

for(var posicao = 0; posicao < segredos.length; posicao++) {

if(input.value == segredos[posicao]) {

alert("Você ACERTOU!");

break;

}

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Isso gera um retorno da quantidade de elementos contidos naquele *array*, para a função. Em seguida, resolveremos um problema que vimos anteriormente, que é a falta de um alerta indicando ao usuário que ele errou o número. Inseriremos esta referência ao final do for(), ou seja, após a varredura em todos os elementos do *array*, por meio da inserção do alert():

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2,3];

var input = document.querySelector("input");

input.focus();

function verifica() {

for(var posicao = 0; posicao < segredos.length; posicao++) {

if(input.value == segredos[posicao]) {

alert("Você ACERTOU!");

break;

}

}

alert("Você errou!");

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Recapitulando: o *array* será varrido por completo, todas as posições serão testadas e, ao final, se não houve compatibilidade em nenhum elemento, será exibida a mensagem "Você errou!".

Salvaremos o programa e recarregaremos a página. Digitaremos 3 e pressionaremos um botão, aparecerá um pop up com a mensagem "Você acertou!", em seguida surge um novo pop up com "Você errou!". Nossa ideia é exibir o alerta "Você ERROU!" somente quando o usuário não conseguir acertar nenhum dos números em nosso *array*.

Para resolvermos isso, antes do for(), declararemos uma variável chamada achou, que recebe false:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2,3];

var input = document.querySelector("input");

input.focus();

function verifica() {

var achou = false;

for(var posicao = 0; posicao < segredos.length; posicao++) {

if(input.value == segredos[posicao]) {

alert("Você ACERTOU!");

break;

}

}

alert("Você errou!");

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Quando o número digitado corresponder a um dos elementos do *array*, declararemos achou = true:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2,3];

var input = document.querySelector("input");

input.focus();

function verifica() {

var achou = false;

for(var posicao = 0; posicao < segredos.length; posicao++) {

if(input.value == segredos[posicao]) {

alert("Você ACERTOU!");

achou = true;

break;

}

}

alert("Você errou!");

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Com isso, informamos ao JavaScript que um dos números é compatível. Caso o número digitado não esteja presente em segredos, informaremos que if(achou == false), ou seja, se achou for false, deve ser exibido um aviso informando ao usuário que ele errou:

<meta charset="UTF-8">

<input/>

<button>Compare com o meu segredo</button>

<script>

var segredos = [5,7,10,2,3];

var input = document.querySelector("input");

input.focus();

function verifica() {

var achou = false;

for(var posicao = 0; posicao < segredos.length; posicao++) {

if(input.value == segredos[posicao]) {

alert("Você ACERTOU!");

achou = true;

break;

}

}

if(achou == false) {

alert("Você ERROU!");

}

input.value = "";

input.focus();

}

var button = document.querySelector("button");

button.onclick = verifica;

</script>

Salvaremos o programa e recarregaremos a página. Vamos testar com 1, e surgirá um pop up com "Você ERROU!". Em seguida, digitaremos 3, e o pop up aparecerá com a mensagem "Você acertou!" e em seguida, nada mais, ou seja, funcionou!