E-BOOK
DO ESTUDANTE

Introdução ao Javascript

Um simples passo para um humano.

RESILIA

SUMÁRIO

Introdução ao Javascript

Um simples passo para um humano.

Neste capítulo iremos ver uma introdução a linguagem de programação utilizada na web: o **Javascript**. Vamos entender algumas funcionalidades e dar nosso primeiro passo com essa linguagem. O Javascript nos permite **interagir com uma página**, possibilitando que tenhamos elementos que podem ser interativos.

Por exemplo: até o momento nós estávamos criando e estilizando os elementos, porém nossos botões não estavam fazendo nada, quando clicamos, não tem uma interação. Com o Javascript, nós conseguiríamos fazer com que o botão por exemplo troque de cor, exiba mensagens ou faça muitas outras coisas, isso é **interação**, ou seja, o usuário realiza uma ação em um click e nosso site ouve a ação e executa um código.

Isso acontece todo minuto por exemplo com redes sociais onde apertamos o botão de enviar uma mensagem e automaticamente uma mensagem é enviada para a outra pessoa, quem ouve o clique no botão e logo após executa um código que entrega a mensagem para o outro usuário é o Javascript.

Lembrando que a linguagem de programação da web é o Javascript. O **HTML** é uma linguagem de marcação, no qual criamos nossos elementos, o **CSS** é uma folha de estilos na qual conseguimos estimular e aplicar efeitos em nossos elementos anteriormente criados no HTML e quando falamos em programar de fato em uma **linguagem**, ela é o **Javascript**.

Está pronto(a) para iniciar nessa jornada com o Javascript? Está pronto(a) para ver como as coisas dentro da web funcionam? Está pronto(a) para aprender coisas que irão lhe auxiliar, até mudar a web, a revolucionar ou somente fazer parte desse mercado, quem sabe?! Então, para que possamos compreender Javascript, primeiramente temos que entender mais a fundo alguns conceitos dessa linguagem. Vamos lá?!

CONTEXTUALIZANDO

JavaScript é uma linguagem de programação que permite a você implementar itens complexos em páginas web.

O JavaScript foi originalmente desenvolvido como LiveScript pela Netscape, em meados da década de 1990. Mais tarde, foi renomeado JavaScript, em 1995 e tornou-se um padrão ECMA em 1997. De lá para cá o Javascript somente vem crescendo em popularidade e isso também aumenta a presença dele no mercado. Pense na seguinte estrutura: elementos (HTML), estilos (CSS) e interação (Javascript). Esta estrutura é o que compõe a web.

Para iniciarmos, de fato, no Javascript, **temos que aprender um pouco de lógica de programação**, e aí vem a pergunta, o que é lógica de programação? Antes de introduzir sobre a lógica de programação, vamos entender o conceito de lógica?

A Lógica é o pensamento computacional necessário para nos desenvolvermos e que nos ajudará a ser melhores programadores, muitas pessoas dizem que programadores bons são aqueles que tem uma boa lógica de programação.

Voltando a lógica de programação...

Quando falamos em **programar**, falamos sobre mandar o computador executar uma ação, porém ele é como um bebê que acabou de nascer e ainda não sabe absolutamente nada! Gostamos de pensar que, a cada ação na qual montamos em nosso site, ela vai desenvolvendo o computador, assim como acontece em nossas vidas, é visualizar a evolução de um bebê. Todas as linguagens necessitam da lógica, ou seja, de como orientar o computador a realizar determinada ação. E é aí que entra a lógica de programação, ligando tudo! Em seguida, vamos desenvolver nossa lógica de programação e nosso pensamento computacional.

LÓGICA DE PROGRAMAÇÃO

A lógica de programação nos ajuda a pensar como um computador pode executar determinada ação, ela nos permite sermos bons desenvolvedores, a partir dela conseguimos visualizar o que será necessário escrever e criar o código para que um computador utilizando uma linguagem em específico execute tal operação. Porém, a lógica é composta de duas partes, a primeira consiste em algoritmos e a segunda em relação computacional. Esses dois elementos juntos podem tornar sua lógica de programação maravilhosa!

ALGORITMOS

Um **algoritmo** é uma tarefa ou objetivo no qual dividimos em diversas partes menores, por exemplo: quando um bebê tem que andar, ele precisa, primeiramente, levantar, equilibrar seu corpo, e levemente movimentar uma de suas pernas para frente, uma após a outra, de forma sempre sincronizada. Quanto mais repetimos a mesma ação, isso se torna um hábito e, dessa forma, nosso cérebro passa a realizar tal ação sem precisar de muito "processamento". No entanto, para um computador, sempre será necessário descrever esse passo a passo para que ele consiga realizar alguma tarefa.

Algoritmos estão presentes no nosso dia-a-dia, e, muitas vezes, utilizados sem nem percebermos. Sabe quando temos aquela meta muito grande que precisamos ir por partes? O ato de desestruturar uma tarefa grande em subtarefas nos ajuda a concluir tarefas de forma mais eficaz.

Para **treinar** e **construir bons algoritmos**, é recomendado que você comece a usar essa "lógica" em sua vida. Se nunca usou antes, comece por pequenas coisas, aquela tarefa de casa que pode ser dividida em três partes, aquele trabalho da escola, aquela meta mensal, semanal ou anual que é composta de várias metas menores, utilizar no seu cotidiano é o que a tornará frequente e útil em sua mente.

RELAÇÃO COMPUTACIONAL

A relação computacional é como nós conseguimos relacionar aquilo que desenvolvemos com as nossas experiências e vivências do dia-a-dia. É o modo como compreendemos o código e como o lemos. Conseguir relacionar o código com a utilização de sua língua nativa é de longe a melhor das hipóteses, já que uma linguagem de programação segue um conceito/escopo no qual sempre conseguimos relacionarmos a ação com nossa língua nativa.

Vamos pensar em um código, por exemplo:

Salve o texto "`Olá mundo`" em um pote com o rótulo "`saudar`".

Exiba o conteúdo do pote "`saudar`".

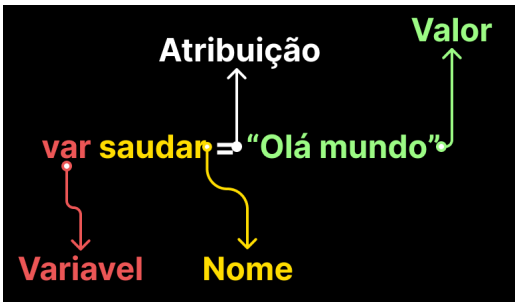
Note que neste código estamos salvando um dado "Olá mundo" em um pote que será utilizado posteriormente para exibir este valor. Quando salvamos um dado estamos permitindo que esse dado possa ser utilizado depois.

Vamos ver como essa tarefa ficaria na linguagem Javascript:

```
var saudar = "Olá mundo"
```

Nesta primeira parte do código já notamos uma diferença, a palavra “**var**" simboliza uma **variável**, ou seja, um local que guarda um valor para que ele possa ser usado posteriormente (anteriormente relacionado a um pote). A palavra “**saudar**” é o **nome da variável**, ou seja, depois, para que nós possamos pegar o conteúdo desta variável iremos precisar chamar o nome da variável que em nosso caso é o nome “saudar” (o nome da variável anteriormente foi relacionado a um rótulo), o sinal de igual simboliza a atribuição, ou seja, estaremos atribuindo um valor a nossa variável de nome “saudar” (o sinal de igual anteriormente relacionado a guardar ou por um conteúdo no pote) por fim temos a palavra “Olá mundo” que será o valor atribuído à variável. Note que esse valor está escrito com aspas, elas simbolizam um valor textual, uma string dentro do Javascript, ou seja, nossa relação ficou assim:

A variável de nome saudar está recebendo o valor “Olá mundo” e guardando este valor para utilizarmos posteriormente.



A segunda parte do código ficaria:

```
console.log(saudar)
```

Temos um **console.log**, a palavra console se refere a um terminal Javascript que pode ser acessado nos navegadores ao inspecionarmos a página e irmos na aba “console” ou “consola”. A palavra **log** define o **tipo de coisa que iremos inserir no console**, no caso um log é as informações da execução de uma aplicação, dados de execução, mensagens úteis para programadores, por fim temos a palavra “saudar”, tal palavra que é a mesma do que o nome da nossa variável analisada anteriormente, ou seja, quando chamamos a variável pelo nome dela, ela irá apresentar o seu valor contido, no nosso caso, o valor que a variável de nome “saudar” contém é a palavra "Olá mundo”, sendo assim nossa relação conjunta das duas partes do código ficou:

```
A variável de nome saudar está recebendo o valor “Olá mundo” e guardando este valor.
Na linha de baixo estamos chamando a variável de nome saudar para podermos pegar o valor nela contido e exibir este valor (no console) como um dado ou informação da aplicação para os programadores (um log).
```

Note como quando olhamos pela primeira vez o código ele não fez sentido algum, porém quando o relacionamos com a nossa linguagem nativa e com as nossas experiências pessoais, o código começou a fazer sentido e ao final estávamos **compreendendo** melhor o que cada coisa estava fazendo. Legal, né?

Utilizar a relação juntamente com os **algoritmos** nos permite entender melhor a linguagem que estamos aprendendo, além de nos permitir desenvolver habilidades que irão nos auxiliar em todas as demais linguagens e na lógica de programação. Contudo, precisamos entender que, em muitos casos, o que torna uma pessoa boa em uma **linguagem ou em lógica de programação é o treino, a prática constante.**

Quanto mais praticamos, mais passamos por situações diferentes. Situações que vão sendo armazenadas em nosso cérebro e, posteriormente, irão nos auxiliar na adaptação e rapidez da resolução quando nos depararmos com a mesma situação.

SCRIPT

Quando falamos em treinar, precisamos entender como utilizamos e interligamos o arquivo html com o Javascript. Dentro de um arquivo html temos diversas possibilidades de tags, uma delas é a tag <script></script>. Esta tag nos permite desenvolver um código Javascript dentro dela, ou seja, dentro do HTML. Dessa forma, para utilizarmos um código Javascript, sempre utilizaremos a tag script.

PRÁTICA

Vamos tentar executar o seguinte exemplo: João quer enviar uma mensagem para o usuário, de forma a saudar o usuário por estar aprendendo Javascript.

Pensando neste exemplo, para que possamos desestruturar este objetivo, precisamos utilizar o conceito de algoritmos visto anteriormente,

Reflitam sobre o exemplo e os passos que teremos que realizar. Primeiramente, sabemos que João quer enviar uma mensagem, logo precisamos de uma mensagem de saudação, em seguida exibi-la, e pôr fim a mensagem deve ser destinada a estudantes de Javascript refletindo na melhor ordem para organizar e executar. Um exemplo:

1. Criar uma variável com o nome saudar
2. Elaborar a mensagem de saudação para estudantes de Javascript
3. Atribuir a mensagem para a variável
4. Exibir o valor da variável saudar (a mensagem) no console como uma informação para programadores

Começando pela criação da variável:

```
var saudar
```

Montagem do texto:

```
Que bom que está aprendendo Javascript, essa linguagem tem futuro!!!
```

Atribuir a mensagem à variável:

```
var saudar = “Que bom que está aprendendo Javascript, essa linguagem tem futuro!!!”
```

Exibir o valor da variável no console:

```
var saudar = “Que bom que está aprendendo Javascript, essa linguagem tem futuro!!!”
console.log(saudar)
```

E assim finalizamos nossa prática! Percebam como realizar o processo por etapas nos permitiu chegar com tudo finalizado. Sempre precisamos saber por onde começar e quais serão os nossos passos. Mapear esses pontos mentalmente e construir esses passos usando o **conceito de algoritmos** nos ajudam na construção dos códigos. Foquem sempre em “por onde eu começo e qual será meu próximo passo?” Isso sempre ajudará dentro do mundo da programação.

Para refletir

- Porque devemos desestruturar uma tarefa? Qual a vantagem desse ato?
- Porque Javascript? O que esta linguagem tem de tão especial?
- O que é uma variável? Porque utilizamos variáveis? Qual a vantagem que temos ao usá-las?
- Quando pensamos em forma computacional isso pode nos ajudar na programação e principalmente na formulação da nossa lógica, o que torna o pensamento computacional diferenciado?

TIPOS DE DADOS

Dentro do Javascript temos sete tipos de dados:

Boolean: um dado onde temos duas opções: ou é verdadeiro (true) ou é falso (false). Este dado é utilizado para fazer validações, como por exemplo a validação de inscrição. Quando clicamos em se inscrever no canal, antes o que estava falso (false), se torna verdadeiro (true) e passamos a estar inscritos no canal. Exemplo:

```
var inscrito = false
var inscrito = true
```

Null: um dado nulo (null) é frequentemente usado para manter uma variável sem valor até que seja inserido. Como exemplo temos os campos de texto, quando a senha está vazia e clicamos em entrar, aparece um erro que ela está vazia, isso pode ser feito com uma variável nula, na qual somente será preenchida se o usuário digitar algum texto no campo (input). Exemplo:

```
var senha = null;
```

Undefined: um dado indefinido (undefined) é o mais comum de ser visto em erros, principalmente onde temos que utilizar dados que não estão sob nosso controle. Como quando tentamos chamar uma variável com o nome errado ou erros onde tentamos chamar variáveis antes de as criarmos. Exemplo:

```
console.log(btn) (Será indefinido pois esta variável não existe.)
```

Number: é um tipo de dado numérico. Dados numéricos são aqueles que nos permitem a utilização dos mesmos em operações matemáticas como subtração, soma, multiplicação, dentre outras. Eles são vistos normalmente em contagens, como por exemplo: a contagem de inscritos ou de produtos no carrinho. Dentro de um dado numérico temos duas abordagens:

1. Os dados inteiros (`int`) que são usados para contagens e exibição de quantidade e;

2. Os dados reais ou de ponto flutuante (`float`) que são usados normalmente para valores reais ou contagens com mais de dois números após a casa decimal.

Exemplo:

```
var quantidade = 100
```

Bigint: é um dado que “fornece um modo de representar números inteiros maiores que 2^53, que é o maior número que o JavaScript consegue, com exatidão, representar”. Dessa forma, ele permite que as criptografias possam ser mais eficazes, tais dados são frequentemente usados para criptografias ou a utilização de conjuntos binários. Exemplo:

```
var criptografia = 9007199254740991n
```

String: é um tipo de valor textual (String), podendo ser um simples caractere ou um conjunto de caracteres formando palavras e blocos de texto. String é frequentemente utilizada para composição de textos, informações, alertas dentre outras possibilidades. Exemplo:

```
var nome = “Alexandre”
```

OPERAÇÕES

Assim como temos nossos **dados numéricos dentro do Javascript**, precisamos ter nossas **operações matemáticas** para que seja possível a utilização de tal dado. Para isso temos as seguintes operações:

+ (Soma): A operação de soma nos permite somar um valor juntamente a outro. este operador é utilizado para somar valores numéricos e textuais.
Exemplo:

```
números:
var total = 15 + 15
// O resultado é 30
Textos:
var texto1 = “ola”
var texto2 = “mundo”
var mensagem = texto1 + texto2
// O resultado é “ola mundo”
```

- (Subtração): A operação de subtração nos permite subtrair valores. Este operador é usado somente entre dados numéricos.
Exemplo:

```
var nascimento = 2002
var anoAtual = 2023
var idade =  anoAtual  - nascimento
// O resultado é 21
```

/ (Divisão): A operação de divisão divide dois conjuntos numéricos. Esta operação é utilizada somente em dados numéricos.
Exemplo:

```
var total = 90 / 9
```

*** (Multiplicação):** O operador de multiplicação multiplica dois conjuntos numéricos. Este operador é usado somente em dados numéricos.
Exemplo:

```
var total = 9 * 10
// O resultado é 90
```

% (Resto): A operação de resto nos permite saber o restante de uma operação de divisão permitindo assim a validação de múltiplos. Este operador é usado somente em dados numéricos.
Exemplo:

```
var múltiplo = 10 % 2
// O resultado é 0, já que 10 dividido por 2 é igual a 5 e não sobra resto, o que o determina múltiplo. Se fosse 11 dividido por 2 seria igual a 5 e teríamos 1 de resto, o que torna o número 11 não múltiplo de 2.
```

CONDICIONAIS

As condicionais nos permitem criar condições, imagine um sistema verificando se você está inscrito ou não em um canal. Para isso, ele irá precisar de uma condição, se estiver inscrito, eu não apresento propaganda, se não for inscrito, eu apresento a propaganda. Para essa ação iremos utilizar o if & else.

O **if**, na tradução literal do inglês seria "**se**". O if inicia o ciclo de condições, ele é composto por uma condição lógica e um código a ser executado caso a condição lógica seja **verdadeira.**

Exemplo:

```
var inscrito = true
if(inscrito === true){
    console.log("Sem propagandas")
}
```

Note que dentro dos parênteses do **if,** temos nossa **condição lógica**, ou seja, estamos verificando se o valor da nossa variável “inscrito” é estritamente igual a true (verdadeiro), se esta condição for verdadeira, o código dentro das chaves do if será executado. Mas e quando a nossa variável conter o valor “false”? Para isso utilizamos o else.

O **else** na tradução literal do inglês é “**senão**”. Então, se acaso a condição lógica do if não for verdadeira, nós iremos fazer tal coisa ou ele sempre será executado acaso a condição lógica do if for falsa.

Exemplo:

```
var inscrito = false
if(inscrito === true){
    console.log("Sem propagandas")
} else {
    console.log(“Com propagandas”)
}
```

Note que neste **código a condição lógica do if não será verdadeira**, o que faz com que o **else** seja executado. Lembrando que o **else** não tem nenhuma condição lógica, ele somente espera que a condição anterior a dele seja falsa, ele é como a última opção que sempre será executada acaso as anteriores a ele forem falsas.

Neste exemplo nós olhamos somente dois valores, **verdadeiro e falso**, mas e se quiséssemos fazer condição com três ou mais possibilidades de valores? Para isso, nós conseguimos estender a condição **if & els**e usando o elseif.

O else if estende a condicional if & else, permitindo assim que possamos ter mais do que somente duas condições.

Exemplo:

```
var idade = 18
if (idade > 19) {
      console.log(“Você entra na categoria profissional”)
} elseif (idade === 18){
      console.log("você entra na categoria semiprofissional”)
}
else {
      console.log(“você entra na categoria amadora”)
}
```

Neste código temos nossa **variável** idade que está recebendo o valor 16, logo após temos a condição que está verificando se a nossa variável idade tem o valor maior do que 19. Nossa variável não tem um valor maior que 19, então ela irá verificar o elseif, o nosso else if está verificando se a nossa variável tem um valor que é estritamente igual a 18. A nossa variável tem o valor estritamente igual a 18? SIM! Logo a mensagem que será exibida é: ”você entra na categoria semiprofissional”.

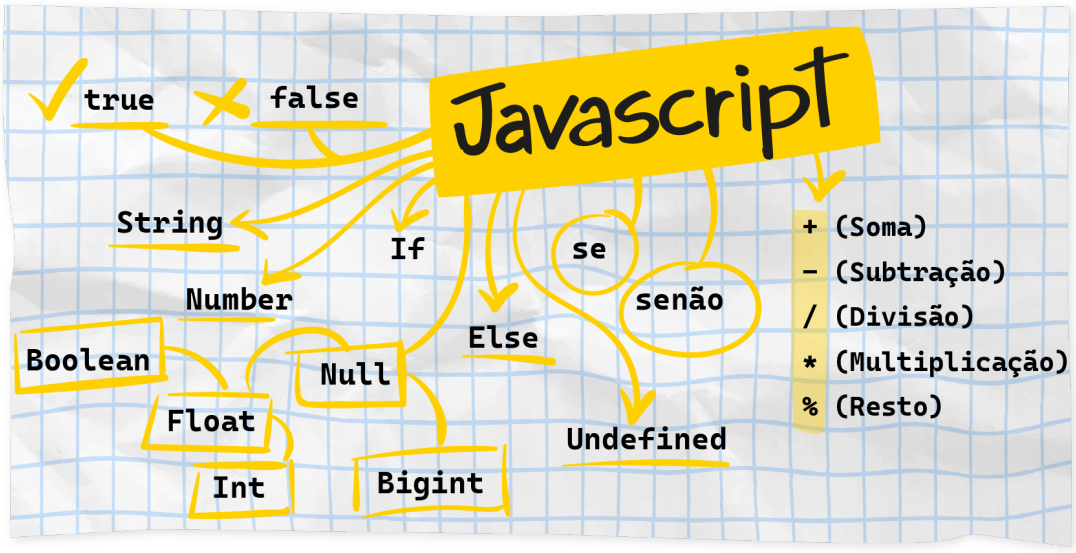
Para refletir

- Por que utilizar condicionais? Qual a vantagem da sua utilização?
- Qual a diferença entre os tipos de dados null e undefined?
- Por que temos um tipo de dado indefinido? Qual a utilidade dele? Onde frequentemente iremos encontrar o dado undefined?

Para ir além

- Os sites abaixo trazem uma introdução e complementação dos tópicos que citamos e vimos neste ebook para serem aprofundados:
  <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Introduction>
  <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Grammar_and_types>
- Para entender melhor sobre tipos de dados existentes dentro do Javascript assim como vimos a importância de os conhecer:
  <https://www.javascriptprogressivo.net/2018/07/Tipos-Dados-Valores-JavaScript.html>
- Para se aprofundar em condicionais e ver com mais profundidade recomendo o site a seguir:
  <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Building_blocks/conditionals>

NUVEM DE PALAVRAS



Até a próxima e
#confianoprocesso

RESILIA