Tera

JAVASCRIPT PARA INICIANTES

Do Zero ao Primeiro Código

SUMÁRIO

INTRODUÇÃO

O JavaScript é uma linguagem poderosíssima. Une uma série de aplicações às incríveis vantagens de usá-la nesses inúmeros contextos. Além disso, está sempre sendo atualizada, com novas bibliotecas e frameworks para superar limitações e alcançar maior versatilidade. Por isso, é obrigatória para quem quer seguir no front-end e **para quem precisa avançar no mundo full stack**.

Apesar de tudo, JavaScript também é uma linguagem simples. Envolve conhecimentos básicos acerca de programação e acerca da infraestrutura da internet, mas uma vez que você já os conhece, tudo fica mais fácil.

Por isso, que bom que você está lendo este e-book. Nele, você entenderá quais são esses conceitos fundamentais para dar os primeiros passos, como o JS atua em uma página web, a importância da linguagem e entenderá na prática como dar esses passos iniciais com alguns códigos que ensinaremos.

Mostraremos os conceitos-base como um guia, de uma forma que ajude você a entender o que deve estudar. Depois, mostraremos como fazer. Acompanhe e aprenda mais sobre essa fascinante tecnologia.

PARA COMEÇAR:
POR QUE É UMA BOA IDEIA APRENDER JAVASCRIPT?

Vamos começar com uma pergunta: por que aprender JavaScript? Se existem tantas tecnologias eficazes no mundo afora, por que focar essa linguagem?

Responderemos com três tópicos. No primeiro, abordaremos o que é possível fazer com essa linguagem hoje. Assim, você terá uma visão de sua versatilidade. No segundo, falaremos sobre a função do JS em um site comum. Por fim, falaremos sobre as oportunidades e o campo de visão para quem lida com JavaScript no dia a dia.

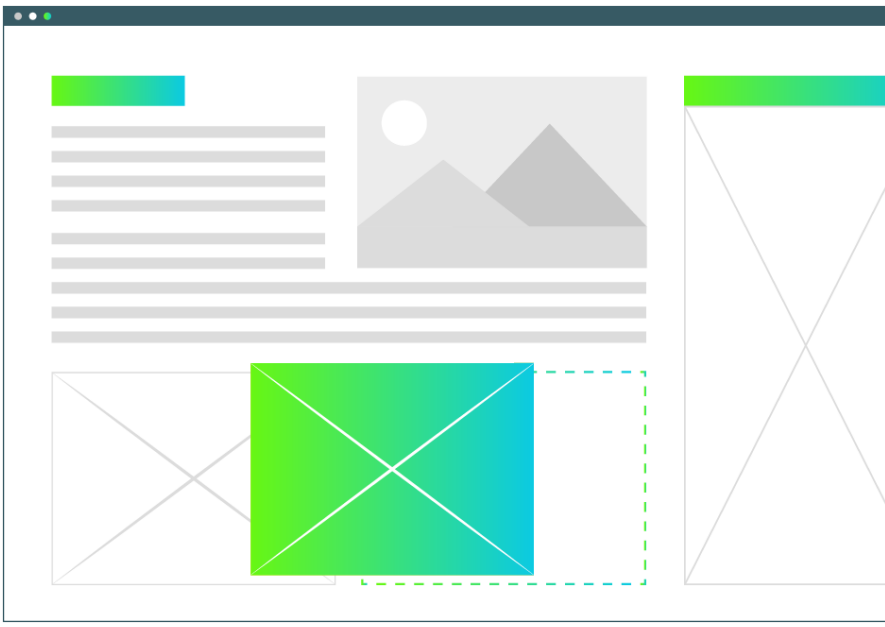
O QUE VOCÊ PODE FAZER COM JAVASCRIPT

A principal função do JS é **gerar dinamismo para uma página web**. Considerando um esqueleto já criado com HTML e CSS, o JavaScript surge para adicionar uma lógica de programação e dar vida a uma série de recursos cruciais para o cumprimento dos requisitos.

Inclusive, boa parte dos frameworks e bibliotecas focam nisso: a parte de apresentação dos sites, o chamado front-end.

Contudo, JS não para por aí. Ela já é adotada para o back-end atualmente também, com o framework Node.js. Assim, ela é usada para lidar com APIs, autenticar fluxos de informações e outras ações do lado do servidor.

Também podemos ter JS para **testes e para automação de tarefas**—como compactação de CSS, otimização do site e outros processos importantes. Para a automação, utiliza-se o Gulp, uma biblioteca específica.



Há registros de aplicações para mobile também, com desenvolvimento de interfaces responsivas e de aplicações típicas para telas menores. O JavaScript também manipula o banco de dados, com o MongoDB.

Em usos mais diferentes e interessantes, temos o JS aplicado para desenvolvimento de jogos, com as bibliotecas Impact e Phaser.

Vale citar também a aplicação para **inteligência artificial**, um tema muito importante nos nossos dias, e para sistemas operacionais. Para lidar com IA, temos a biblioteca tensor.JS, versão da

conhecida TensorFlow. Para sistemas operacionais, temos o Node.OS.

Nas iniciativas voltadas para o back-end, muitas pessoas têm conseguido bons resultados com aplicações de conversas em tempo real, como em um chat.

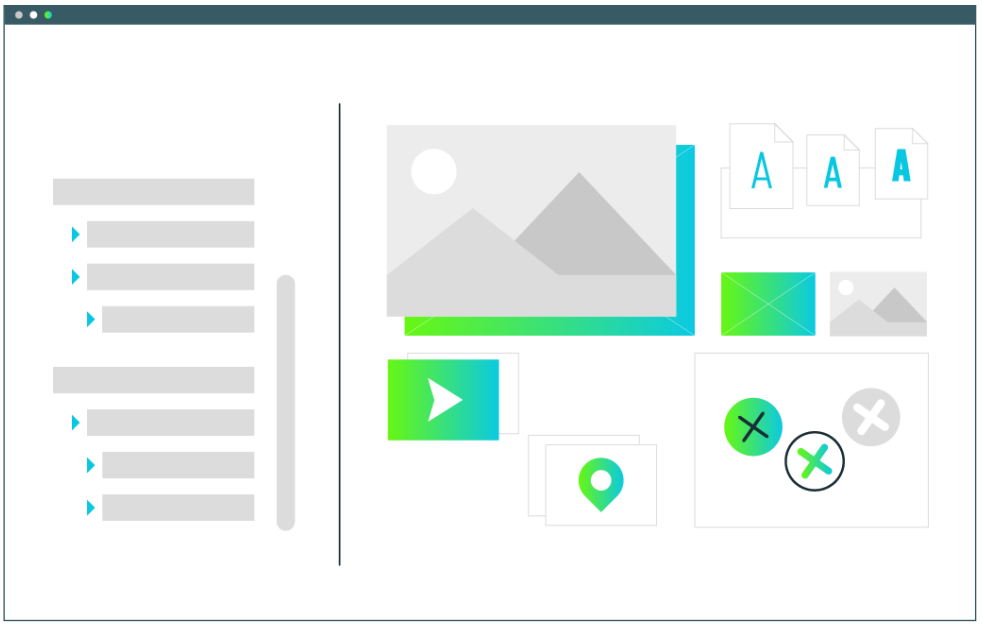
QUAL A FUNÇÃO DO JAVASCRIPT EM UM SITE?

Em sites, a principal aplicação do JS é na otimização do front-end. **Uma vez que o HTML e o CSS fornecem uma estrutura visualmente já personalizada, o JS é usado para dar os toques finais.** (E esses toques finais são muito importantes, pois permitem que o site interaja com o usuário ativamente.)

Imagine, por exemplo, um elemento HTML com uma determinada cor. A partir do HTML somente, não é possível alterar a cor dinamicamente, com o clique do usuário, digamos. Contudo, com uma lógica embutida no JS, você consegue. A partir da configuração do tratamento do clique, é possível

mudar a cor de forma automática (veremos mais sobre isso na parte de eventos deste e-book).

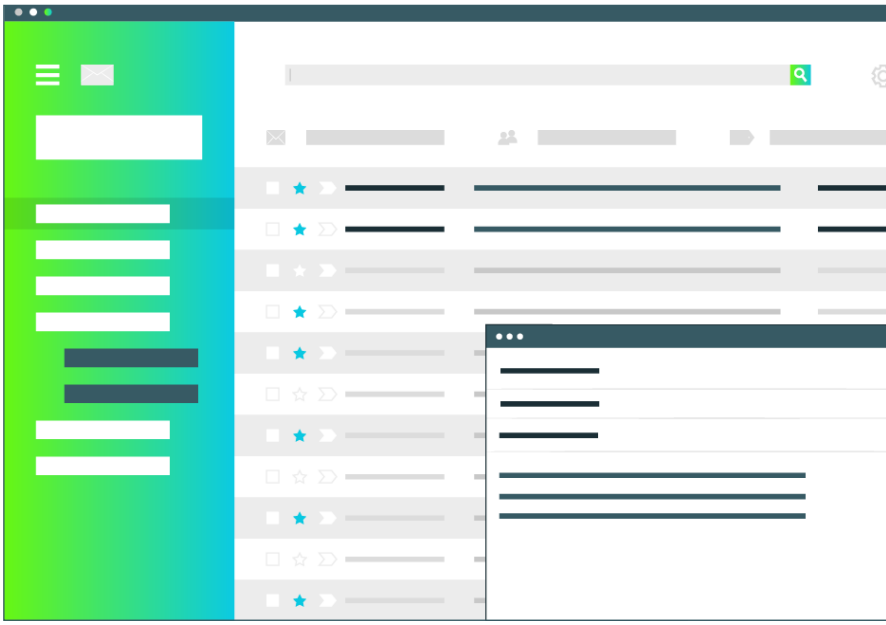
Isso se torna uma possibilidade para outras ações também: é viável fazer algo aparecer ou sumir da tela dinamicamente e outros tipos de uso.



Além disso, é possível dar zoom em imagens, mostrar um contador atualizando automaticamente na tela, sortear um número, mostrar menus animados ou interativos e mostrar carrosséis. Na interface da Netflix e de sites como YouTube, o mecanismo que permite arrastar horizontalmente

para ver mais filmes ou vídeos é possível com o JS.

Recentemente, temos a importância da **single--page application**, um exemplo de página que consegue atualizar elementos sem mudar a página inteira. Um exemplo disso? O Gmail, com a função de enviar e-mails e checar a caixa de entrada ao mesmo tempo. Ou o Twitter com a função de checar os tópicos mais falados enquanto a timeline é atualizada.



QUEM TRABALHA USANDO ESSA LINGUAGEM?

A pessoa que lida com JS no dia a dia geralmente é a pessoa do **front-end**. Ela cuida da interface gráfica, da experiência e do visual no geral. Contudo, profissionais full stack também precisam dominar a linguagem, pois poderão utilizá-la para o front, para o banco de dados ou para o back.

> Além disso, outras pessoas podem utilizar essa tecnologia: cientistas de dados, pessoas que criam jogos, bem como devs mobile.

Depois desse primeiro tópico com uma visão geral da linguagem, você provavelmente já notou que está diante de uma das mais importantes tecnologias da atualidade. Agora, vamos seguir para **conceitos que você precisa saber antes de começar a programar em JS**.

GLOSSÁRIO:
UM MERGULHO NOS TERMOS DA LINGUAGEM JAVASCRIPT

Entender mais sobre alguns termos do universo de JavaScript é essencial para dar os primeiros passos antes mesmo de codar. Separamos alguns conceitos para que você entenda um pouco da linguagem técnica que envolve o processo de programação em JavaScript.

- **VARIÁVEIS**
- **FUNÇÕES OU MÉTODOS**
- **OPERADORES**
- **TIPOS DE DADOS**
- **ESTRUTURAS DE REPETIÇÃO**
- **ESTRUTURAS CONDICIONAIS**
- **FRAMEWORKS**

I VARIÁVEIS

Comecemos com as variáveis: **elas representam um espaço na memória disponível para alocar um determinado valor**. O objetivo é registrar uma informação importante para utilizá-la posteriormente. A vantagem de usar uma variável, em vez de declarar o valor diretamente, é poder manipular esse espaço na memória e submeter essa pequena estrutura a mudanças na lógica.

Por exemplo, uma variável numérica pode começar com um valor 10. Contudo, com a lógica subsequente, processamentos de adição são feitos e essa variável passa a valer 20. Depois, muda novamente para 354.

Para que você não precise manter esse número na cabeça depois das mudanças dele, você o guarda no espaço de memória. Esse espaço vai ser acessado sempre para realizar as mudanças automaticamente (a pessoa que programa nem precisa saber dessas mudanças, só mesmo da lógica).

As variáveis também servem como uma forma de **tornar o código mais limpo** e fácil de compreender. Em vez de ter um número solto no meio do código, a pessoa programadora pode definir uma variável com o nome Renda_Total, que indica muito claramente do que se trata.

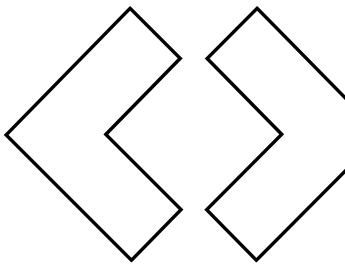
FUNÇÕES OU MÉTODOS

Funções são pedaços de código que executam alguma tarefa específica. Assim, toda vez que chamamos essa função, **obtemos a sua solução de forma simples e encapsulada**. A alternativa ao uso de funções seria um código complexo e desorganizado em que precisaríamos desenvolver um conjunto de código toda vez que precisássemos daquela tarefa.

Digamos que você precise de um código para devolver a porcentagem de um número com relação a outro. Esse é um cálculo um pouco mais complexo de ser feito e requer um conjunto de passos. Assim, caso você faça isso da maneira mais intuitiva, escrevendo todo o código toda vez que precisar, o seu código-fonte vai se tornar uma bagunça. Fora que você terá o trabalho de escrever novamente sempre.

Com as funções, você encapsula aquela tarefa e sua lógica de funcionamento em um termo específico (o nome da função). Então, só tem que chamar a função e passar o que chamamos de parâmetros — que são os valores com os quais a função trabalha. Em nosso exemplo, parâmetros seriam os números que eu quero usar para o cálculo.

Veremos ao longo deste e-book várias funções que fazem tarefas prontas, com as quais nem mesmo nos preocupamos com a lógica, somente com a tarefa executada. Assim, entenderemos melhor como uma função é usada.



I OPERADORES

Em JavaScript e em outras linguagens de programação, os operadores são símbolos que representam uma operação. Eles podem ser operadores aritméticos, que se assemelham basicamente ao que aprendemos em matemática na escola: adição, subtração, multiplicação e divisão. Sendo que para multiplicação, usamos "*" e para divisão, usamos "/".

Na divisão, temos especificamente a divisão inteira, que é representada pelo "%". Ele retorna apenas a parte inteira da resposta.

Outros tipos de operadores importantes são os de comparação de números, que compreendem o maior que (>), menor que (<), igual (== ou ===) e diferente (!=).

Por fim, temos a classe dos operadores lógicos. Esses são aprendidos quando estudamos lógica booleana. São eles: && (e), || (ou) e ! (não). O "e" só é satisfeito quando duas condições são satisfeitas, o "ou" é satisfeito quando uma das condições é satisfeita, ao passo que o "não" nega o que vem depois, então só é satisfeito quando a condição é negativa.

I TIPOS DE DADOS

Os tipos de dados são padrões que usamos em programação para determinar diferentes modelos dos dados com os quais estamos trabalhando. Desse modo, torna-se mais fácil desenvolver lógicas e fluxos lógicos.

Parte do princípio de que tipos de dados diferentes não podem ser processados como semelhantes, exemplo: um número não é do mesmo tipo que uma string (conjunto de letras), portanto, **não podemos simplesmente realizar um cálculo entre um número e uma string**.

Ou seja, com essa delimitação entre os tipos, é possível saber exatamente quando poderemos realizar um cálculo ou processamento entre variáveis sem obter erros. Além de números e strings, temos o tipo booleano (verdadeiro ou falso) e o nulo (que determina um valor vazio).

O grande destaque do JS com relação aos tipos de dados é que a linguagem não é fortemente tipada. Assim, se você define uma variável, não precisa especificar o tipo de dado estritamente e pode mudar o tipo quando desejar. Ou seja, quando atribui um valor à sua variável, você define o tipo dela temporariamente.

ESTRUTURAS DE REPETIÇÃO

Um dos elementos mais importantes da lógica de programação é a capacidade de criar loops. **São controladores de fluxo que repetem a mesma operação até que uma determinada condição seja cumprida.**

Você pode ter, por exemplo, um conjunto de dados que deseja exibir na tela. Com um loop, você exibe esse conjunto até que o contador de elementos do conjunto chega ao seu limite, o número de elementos do conjunto.

Uma das principais aplicações de uma estrutura de repetição é percorrer estruturas de dados (que são um agrupamento de variáveis), seja para exibir os elementos, seja para realizar algum processamento com eles.

ALGUNS TIPOS DE ESTRUTURA DE REPETIÇÃO SÃO: for, while, forEach E do-while.

Basicamente, eles estabelecem um elemento contador e executam um bloco de código (um bloco encapsulado, que lembra uma função) até que o contador chegue ao limite estabelecido.

> É sempre importante tomar cuidado com a lógica, pois uma estrutura dessas mal formulada leva o sistema ao que chamamos de **loop infinito** e pode travar a aplicação.

ESTRUTURAS CONDICIONAIS

Outros elementos de controle de fluxo de extrema importância são as estruturas condicionais. As duas principais são o bloco **“if-else”** e o **“switch”**.

O bloco if-else funciona de modo muito simples. Você define uma condição de comparação (if X < 2, por ex) e coloca um bloco de código dentro do if que será executado caso aquela condição seja cumprida (se satisfizer, adicione 1 ao X, por ex). O “else” é o elemento em contraste, que executa uma determinada instrução somente no caso da condição do if não passar (se o X não for < 2).

```
if (x < 2){
    //codigo do if
} else{
    document.write(“<h1>teste do if </h1>”);
}
```

Em um código comum de uma aplicação, você verá muitos “ifs”. Inclusive, é comum que tenhamos um bloco if-else dentro de um loop, para testar condições em elementos de uma determinada estrutura de dados somente.

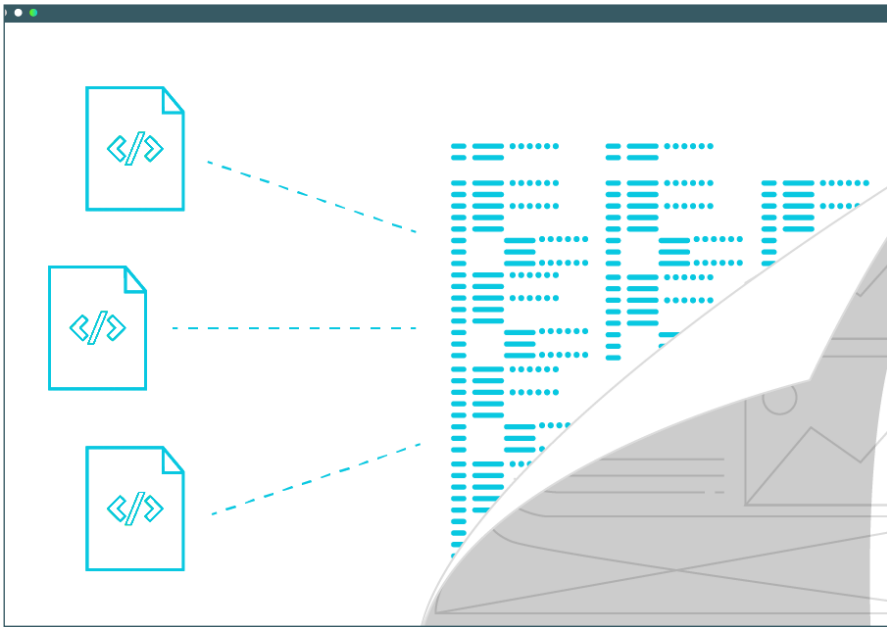
Já o switch analisa uma variável e estabelece condições para executar diferentes pedaços de código. Por exemplo, digamos que o switch analise nossa variável X, os casos podem ser: case > 2, case < 2 ou case == 2. Para cada caso, temos uma instrução que deve ser executada.

As estruturas condicionais dividem o fluxo em uma lógica similar a de uma árvore. Cada condição permite que o sistema vá para um lado ou para outro da ramificação.

I FRAMEWORKS

Frameworks são padrões de bibliotecas e classes que podem ser usados em diferentes projetos. Ou seja, são uma espécie de código pronto que pode agilizar o trabalho de programação.

JavaScript é uma linguagem conhecida pelo seu incrível número de bibliotecas e frameworks. São eles, aliás, que fazem a tecnologia ser tão versátil e poderosa para diversas aplicações.



PRINCIPAIS FRAMEWORKS

Como falamos, os frameworks de JS são essenciais para que pessoas programadoras ganhem agilidade na criação de códigos. Assim, é interessante conhecer os principais frameworks do mercado, aqueles que mais surgem em discussões e em vagas de emprego para ajudar você a se preparar.

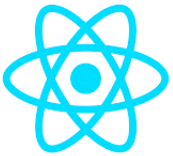
Antes de começar, vale um aviso: **os frameworks JavaScripts mudam bastante**, de uma forma rápida. Sempre surgem novas versões e até mesmo novos frameworks para melhorar a vida das pessoas programadoras. Por isso, este guia a seguir visa apenas apresentar os principais.

ANGULAR

Um dos mais famosos é sem dúvidas o Angular, cujo propósito é o desenvolvimento tradicional front-end. Requer um processo mais complexo de instalação e configuração, porém gera muitos efeitos positivos para as empresas. Uma de suas vantagens é o fato de que é **um framework open-source**.

Trata-se de uma ferramenta muito útil para quem precisa manipular o DOM (estrutura de elementos de uma página) e para quem busca facilidade na gestão do data binding e dos testes. Assim, as mudanças do código já são projetadas na parte de visualização, o que possibilita melhor desempenho para as interfaces.

Por ser provavelmente o mais famoso e adorado em vagas de emprego, a **comunidade do Angular é muito grande**. Uma vez que você se dedique a esse padrão, encontrará muitas pessoas prontas para ajudar com problemas e desafios na internet.



REACT

A tecnologia criada pelo Facebook tinha um propósito muito claro: permitir o desenvolvimento de interfaces SPA (single-page applications) com facilidade. Ou seja, possibilitar a criação de interfaces com elementos que mudassem sem precisar recarregar a página inteiramente.

Desse modo, tornou-se um padrão muito relevante para quem quer chegar a páginas dinâmicas. É muito útil também por encorajar o **encapsulamento e a reutilização de código**, o que possibilita que funções e scripts criados com uma função sejam usados em vários projetos.

De forma similar ao Angular, **o React é conhecido por envolver uma larga comunidade e ser muito importante para vagas de front-end e full stack**.

NODE.JS

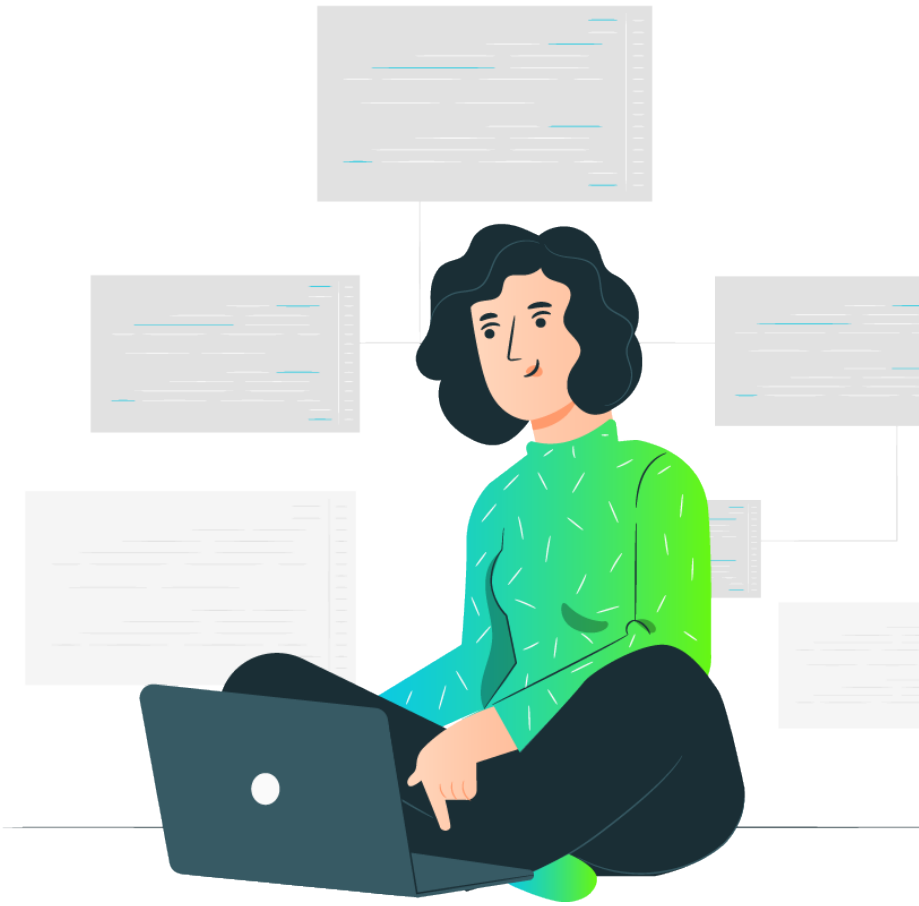
O Node.js é uma importantíssima solução que revolucionou o uso da linguagem JS ao permitir que ela fosse usada fora do browser. O próprio framework funciona como um ambiente de execução. Com isso, o uso da linguagem se expandiu para as aplicações que já mencionamos, incluindo sistemas operacionais e o back-end.

Aliás, em se tratando de back-end, o Node entrega possibilidades bem interessantes. Uma delas é a chance de trabalhar com **microsserviços e arquitetura sem servidor** (serverless).

Isso significa uma capacidade muito maior de produzir elementos independentes que podem ser alterados sem que os outros sejam afetados (uma seção, um menu, um sidebar). O uso de microsserviços é uma tendência moderna.

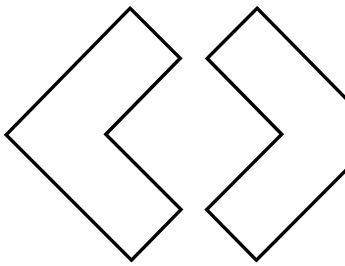
EMBER JS

É um dos concorrentes do Angular, com foco bastante desenvolvido em performance. Também apresenta uma comunidade extensa e interessante para ajudar nos projetos diários. Um dos seus usos é justamente a criação de single-page applications, bem como também para testes robustos.



Considerado um ótimo concorrente do React e do Angular, o Vue.js é uma opção mais simples que vem ganhando espaço. Mesmo que não seja tão querido por pessoas que recrutam devs, é ainda muito importante para a comunidade front-end. Permite desenvolver single-page applications, com um bom controle do data-binding também.

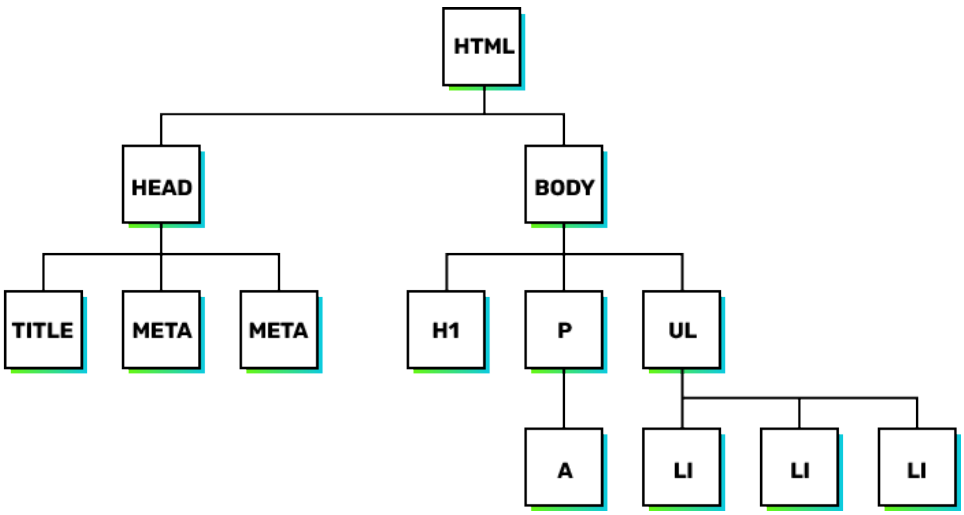
É um framework mais leve, utiliza uma lógica de DOM virtual e possibilita o **reuso** dos seus componentes com facilidade. O seu grande destaque é o fato de que o Vue é um framework progressivo, ou seja, que possibilita o uso e a adaptação a ele em pequenas partes do sistema por vez.



INTERAÇÃO DO JAVASCRIPT COM O DOM

Falamos bastante sobre o DOM no último tópico. Agora vamos definir o que vem a ser esse conceito. A sigla significa Document Object Model e representa uma árvore de elementos da estrutura de uma página web.

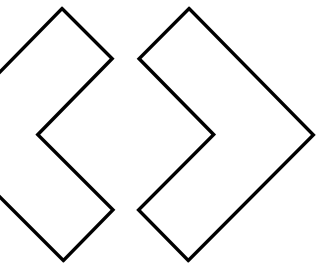
Se falamos em estrutura, você já deve pensar em HTML. Exatamente. **O DOM é o conjunto de tags do HTML, dispostas em uma hierarquia no modelo de árvore.** Por exemplo: o nó-principal de toda página é a tag HTML, que é a principal em todo documento web. Dentro dessa tag, temos as tags HEAD e BODY. E assim por diante.



Fonte: Tableless

Sendo o DOM o esqueleto da página, o JavaScript tem como uma das suas principais funções a manipulação dele. É possível adicionar elementos, remover elementos, reagir a acontecimentos que afetam esses elementos, entre outras ações. A partir de um objeto específico que manipula o documento (document), **você é capaz de trazer elementos do HTML e adicioná-los à lógica do JS para controle**.

Assim, é possível utilizar aqueles conceitos que já vimos, como estrutura de repetição e estrutura condicional para trabalhar com os componentes da nossa página.



Nesse sentido, é importante tratar de dois conceitos do HTML que você verá muito em códigos JavaScript: id e class. São duas estratégias para nomear elementos e permitir que eles sejam identificados e recuperados no código JS, de modo a fazer parte de sua lógica.

A diferença entre eles? O id é geralmente usado como uma forma de identificar um único elemento. Ao passo que a class é usada para agrupamentos de elementos similares. Assim, você consegue agrupar elementos semelhantes ao chamá-los todos de um mesmo class.

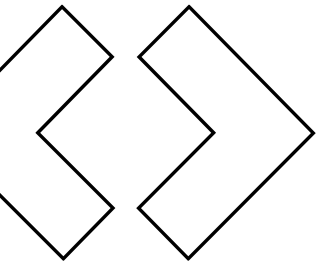
Isso porque temos funções e métodos que ajudam a tratar class e id. Um deles é o “document.getElementById”, que vai permitir recuperar quaisquer elementos com aquele id, e outro é o document.getElementsByClassName(), que vai possibilitar buscar elementos pelo nome da classe.

É importante notar aqui que nesse assunto o JavaScript se torna uma tecnologia altamente dependente do HTML e do CSS. **Então, para dominar o JS e usá-lo corretamente no dia a dia, você precisará também de uma boa base de HTML/CSS.**

EVENTOS EM JAVASCRIPT

Outro importante conceito na programação em JS é o de **evento**. Em suma, um evento é um acontecimento que muda o status de algum elemento na página e que pode ser manipulado na lógica de programação. O evento geralmente ocorre por conta de alguma ação do navegador ou da pessoa usuária.

Quando entramos na parte de eventos, o nosso JavaScript começa a ficar mais interessante e divertido. Pois é nessa parte que entendemos como adicionar real interatividade à nossa página e realmente lidar com as ações das pessoas que usam um site.



Para isso, a gente precisa de um elemento que fica pronto para notar qualquer mudança no status dos elementos do DOM. O termo para isso na programação web é “listen” ou “ouvir”. É como um vigia sempre alerta para captar essas alterações.

Um exemplo disso é o clique. Um clique é um evento, pois é uma ação da pessoa usuária a partir do mouse.

A informação é registrada pelo sistema operacional e chega, então, ao navegador que interpreta aquilo como um evento. Caso você tenha desenvolvido uma lógica para tratar aquele evento, algo vai acontecer. Por exemplo, podemos definir que nossa página mude de cor sempre que o usuário clicar em algo.

O tratamento de eventos segue uma lógica muito similar à das estruturas condicionais e à das estruturas de repetição que discutimos no glossário. Só que essa lógica ocorre de forma automatizada, sendo que nós só temos acesso a sua versão final. Funciona assim: se um evento é detectado, então faça algo. Se não, continue tentando captá-lo.

Outros exemplos de eventos são: carregamento completo da página, quando uma tecla é pressionada, carregamento de uma imagem, quando o mouse é colocado por cima de um objeto, digitação de algo no campo de formulário, etc.

Com esses tratamentos, as possibilidades de animação aumentam e permitem uma interface mais viva. Por exemplo, quando o usuário passa o mouse por cima do menu, ele pode aumentar de tamanho; ou quando o usuário passa o mouse na seta do carrossel, ele arrasta horizontalmente, mostra outros elementos e atualiza.

Com isso, a página se torna dinâmica e melhora a experiência do usuário, oferecendo respostas mais rápidas.

É interessante perceber como os temas DOM e eventos se conectam. Um evento nada mais é do que uma mudança que afeta todos os elementos do DOM em uma sequência hierárquica. Lembra da árvore? Pois, uma mudança em um elemento-filho começa a ser detectada pelos nós-pais até chegar a ele.

Posteriormente, o evento é propagado de volta ao elemento-raiz, subindo na hierarquia na ordem contrária à da captura.

PRIMEIROS PASSOS
ANTES DE PROGRAMAR

Pois bem, você já conhece os principais conceitos para programar em JS: estruturas condicionais, operadores, variáveis, funções, eventos, DOM, etc. Já entende o porquê de usar a linguagem em seus projetos e reconhece a importância de seguir estudando. Chegou a hora de entender quais são os passos preparatórios para programar de fato.

Um deles é a **configuração do seu ambiente**. Esse termo pode parecer complexo, mas é muito mais simples do que parece, já que programar para web não requer nem sistemas instalados. Você pode fazer o seu primeiro exemplo com o bom e velho bloco de notas.

Contudo, caso queira algo um pouco mais útil, com mais recursos, pode instalar o notepad++, uma espécie de bloco de notas com upgrades.

Outra opção é procurar uma IDE mesmo. Uma IDE é uma plataforma completa para desenvolvimento de aplicações em linguagens de programação. Oferece recursos de reconhecimento da sintaxe, autocompletar, compilação e execução e integração com bibliotecas. Ainda é possível testar e analisar o código em busca de erros (que são apontados com clareza).

Ou seja, é uma suíte completa para programação. Entre as conhecidas, temos o Vim, o Visual Studio Code, o WebStorm, o NetBeans e outras.


A instalação geralmente é muito simples e depende de outros pacotes/recursos específicos da linguagem utilizada. Para o JavaScript puro, você precisará de poucos minutos.

Para o caso de JavaScript com frameworks, para programar é preciso configurar o ambiente e os pacotes de código. O Angular, por exemplo, depende da instalação de um mecanismo chamado de NPM, que pode ser feito em um terminal de comando.

Uma grande vantagem dos últimos anos no universo da programação é a possibilidade de não precisar instalar nada em sua máquina e mesmo assim obter os recursos robustos de uma IDE. **É o uso de softwares de programação na nuvem.** Para desenvolvimento web, temos o Code Pen (que funciona também como um hub de projetos e uma rede social) e o Replit.

Com eles, você começa instantaneamente e já consegue ver o seu código funcionar. Inclusive, recomendamos o uso deles para o próximo tópico.

MÃO NA MASSA: CRIANDO SEU PRIMEIRO CÓDIGO

Aqui estamos. Agora que você aprendeu conceitos básicos e a lógica por trás do JS, é hora de colocar a mão na massa. Neste tópico, vamos ensinar duas aplicações simples do JavaScript para você entender como começar.

OLÁ, MUNDO

A tradição ao iniciar uma nova tecnologia é começar com um simples “olá, mundo”. Então, esse é justamente o nosso primeiro exemplo.

Recomendamos que você abra o site Code Pen para esses exemplos. Na parte superior esquerda da home page, clique em “start coding”.

Primeiro, comece com a estrutura de uma página web padrão na seção do HTML. Inclui uma tag “html”, uma tag “head”, uma tag “title”, uma tag “meta”, uma tag “body” e uma tag de “script”. Todas com seu devido fechamento.

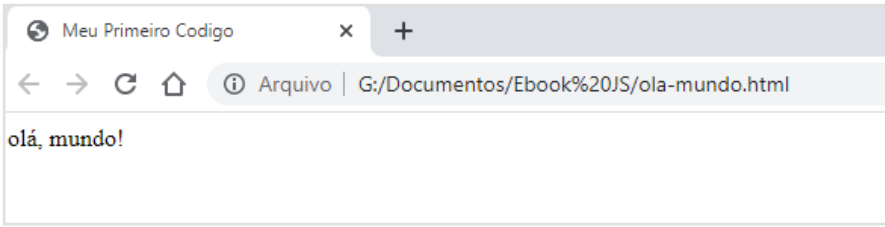
```
<html>
    <head>

    <meta charset=“utf-8”/>
    <title>Meu Primeiro Codigo</title>

    <body>

    </body>

    </head>

</html>
```

Agora, vamos ao nosso arquivo js. Lá, escreveremos “document.write (“olá, mundo!”);”. **Em suma, estamos pedindo para nossa função de escrever no DOM que ela imprima uma mensagem.**

JS

```
document.write (“olá, mundo!”);
```

O resultado desse código vai aparecer em uma página web desta forma:



I FORMULÁRIO SIMPLES

Para o segundo exemplo, mantenha a estrutura já criada do HTML.

Para nosso formulário, precisaremos de uma tag “form”. Dentro dela, os elementos do formulário são pertencentes às tags “label” (nome do campo) e “input” (espaço em branco para digitação).

```
<form name="meu_form">

    <h1>Cadastre-se</h1>

    <label for="nome">Nome</label>
    <input type="text" id="nome-id" placeholder="nome e sobrenome" required="required" name="nome" />
    <br>
    <br>

    <label for="email">Email</label>
    <input type="email" id="email-id" placeholder="fulano@mail.com" name="email" />
    <br>
    <br>

    <input type="submit" class="enviar" onclick="Enviar();" value="Enviar" />
    </form>
```

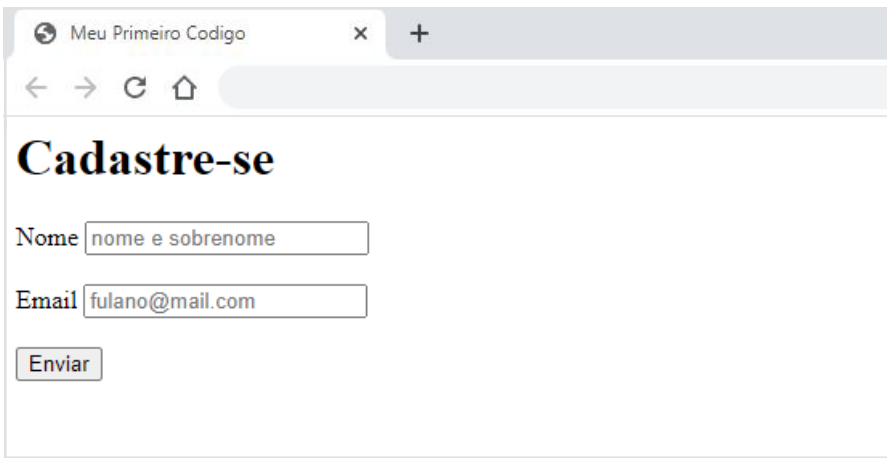
O primeiro input é do tipo texto. Definimos que esse campo será obrigatório, então adicionamos o “required”. O input do e-mail é do tipo e-mail, ao passo que o do botão de enviar é do tipo “submit”.

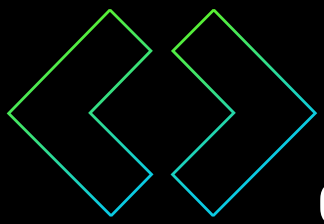
Pronto. Nosso HTML do formulário está pronto. Agora, vamos partir para o JS.

Lá, criaremos uma função com o nome “enviar”. A primeira parte dessa função vai pegar o elemento que tem o id “nome-id” do nosso DOM e passar para a variável “nome”.

```
function Enviar () {

    var nome = document.getElementById(“nome-id”);

    if (nome.value != “”) {
        alert(‘Obrigado, ‘ + nome.value + ‘, os seus dados foram encaminhados com sucesso’);
    }

}
```

Depois, fazemos uma verificação: se o valor da variável não é nula, podemos então supor que a pessoa preencheu o nome. **Assim, exibimos uma mensagem com a função “alert”.** O resultado é o formulário:





CONCLUSÃO

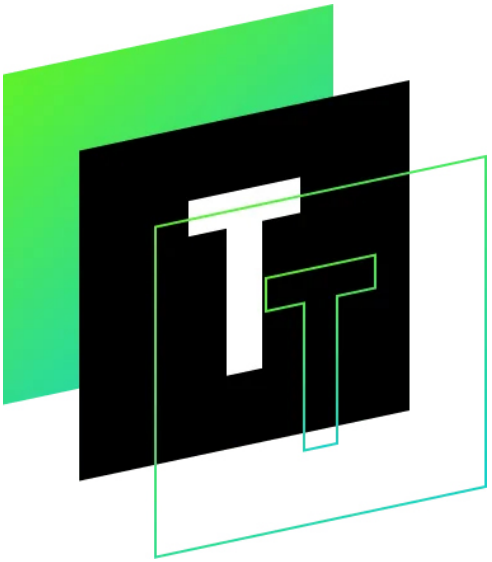
Parabéns! Se você chegou até aqui, já deu seus primeiros passos nessa linguagem e, agora, basta continuar se desenvolvendo e explorando outras ideias e frameworks.

Como você viu, o JavaScript é uma tecnologia versátil e essencial para a construção de páginas web modernas, responsivas e dinâmicas. Também é um padrão muito requerido no mercado, sendo assim fundamental para o desenvolvimento full stack e front-end.

Com os conhecimentos básicos, você já consegue entender como a linguagem funciona e tentar escrever os primeiros códigos. Esperamos que este seja apenas o início da sua jornada em desenvolvimento web.

Sente que está na hora de avançar nessa carreira? Então conheça o novo curso da Tera, Full Stack Development, que vai formar a próxima geração de profissionais Full Stack!

CONHEÇA O CURSO



SOBRE A TERA

Somos mais do que uma escola, somos uma comunidade de pessoas apaixonadas por educação e tecnologia.

Acreditamos que um mundo melhor nasce do trabalho de pessoas conscientes, responsáveis e corajosas que se apropriam da tecnologia para servir ao coletivo.

EQUIPE EDITORIAL

Redação
Gabriel Sacramento

Revisão
Rebeca Nascimento

Direção de arte
Tatiane Rocha

Diagramação
Marina Ferreira