EMENTA DO CURSO

JAVASCRIPT IMPRESSIONADOR

CONTEÚDO EXCLUSIVO DA HASHTAG TREINAMENTOS

PROGRAMA JAVASCRIPT IMPRESSIONADOR

1 Proposta e Objetivo do Curso

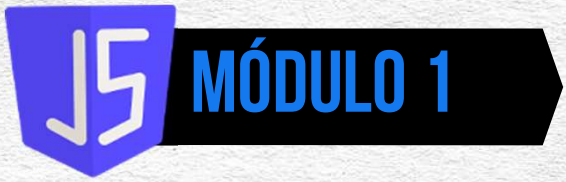
O Javascript Impressionador é um treinamento completo, onde você vai aprender tudo que precisa saber de Javascript, começando do absoluto zero mesmo, até te levar ao nível avançado em Javascript. Você vai aprender não só os fundamentos básicos, intermediários e avançados como vai também aprender a integrar o Javascript com bancos de dados, além de Node JS e React JS.

2 Apostila e Suporte

No Impressionador, temos uma equipe de experts focada no suporte, portanto conseguimos tirar dúvidas além do conteúdo, incluindo coisas que você precise fazer no seu trabalho (não é uma consultoria, então não fazemos com você, mas conseguimos tirar dúvidas de qualquer ferramenta que você esteja construindo sim), além de apostila completa, exercícios extras e atualização constante.

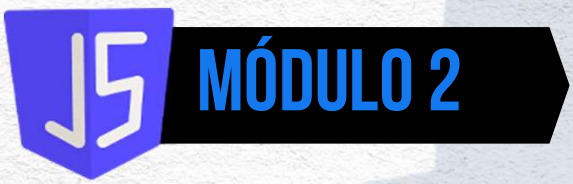
Nas próximas páginas você pode visualizar a ementa completa do curso.

PARTE 1
JAVASCRIPT



Introdução

1. Sobre o curso
2. História do JavaScript
3. O que o JS pode fazer
4. JS na prática
5. Características
6. Preparando o seu ambiente/computador

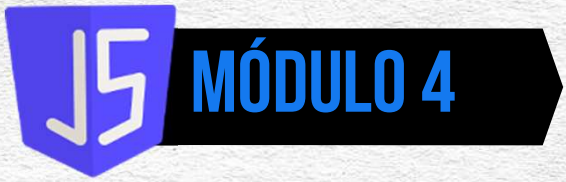


Variáveis e Tipos de Dados

1. Tipos de dados em JavaScript (string, number, boolean, null, undefined, object, symbol)
2. Variáveis e constantes em JavaScript
3. Declaração de variáveis
4. Conversão de tipos de dados em JavaScript

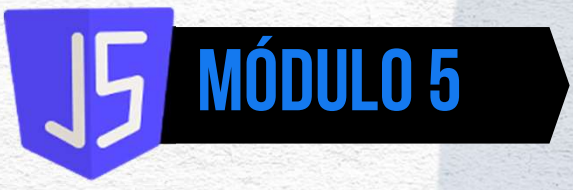
Funções e Escopo

1. O que são funções em JavaScript?
2. Como criar funções em JavaScript?
3. Escopo de variáveis em funções em JavaScript



Lidando com bugs no código

1. O que é o processo de debugging, e como ele se apresenta no dia a dia do programador?
2. Monitorando valores intermediários com "console.log"
3. Ferramentas mais rebuscadas

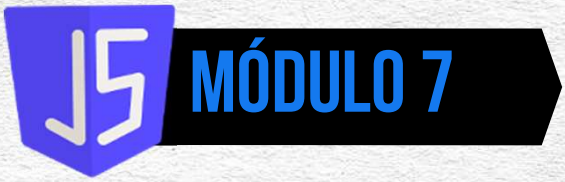


Condicionais (if/else)

1. O que são condicionais em JavaScript?
2. Como usar condicionais if/else em JavaScript?
3. Operadores de comparação em JavaScript (>, <, >=, <=, ==, ===, !=, !==)
4. Operadores lógicos em JavaScript (&&, ||, !)
5. Condicionais encadeados e aninhados em JavaScript

Loops (for/while):

1. O que são loops em JavaScript?
2. Como usar loops do while/for/while em JavaScript?
3. Recursividade



Desenvolvimento Web

1. A estrutura da Web
2. A arquitetura Cliente x Servidor
3. HTML
4. CSS
5. O que é o DOM?
6. Como acessar e manipular elementos do DOM em JavaScript?
7. Como adicionar e remover elementos do DOM em JavaScript?
8. Como manipular estilos de elementos do DOM em JavaScript?
9. Eventos do navegador em JavaScript (click, hover, submit, keydown)

MÓDULO 8

Error Handling - Tratamento de Exceção

1. Try/catch:
1.1 O que é try/catch e como ele é usado para capturar exceções em JavaScript
1.2 Sintaxe do try/catch, incluindo como lidar com exceções específicas
1.3 Como usar o bloco finally para executar código de limpeza

2. Lançamento de exceções:
2.1 O que é lançamento de exceções e quando usá-lo
2.2 Sintaxe de lançamento de exceções, incluindo como criar mensagens personalizadas
2.3 Como criar exceções personalizadas

3. Erros comuns:
3.1 Tipos de erros comuns em JavaScript, como TypeError, ReferenceError e SyntaxError
3.2 Como lidar com erros comuns em JavaScript

4. Boas práticas:
4.1 Lidando com exceções de maneira apropriada e significativa para o usuário
4.2 Não suprimindo exceções sem um motivo válido
4.3 Não capturando exceções que você não pode lidar adequadamente.

MÓDULO 9

Arrays e objetos

1. O que são arrays em JavaScript?
2. Como criar e manipular arrays em JavaScript?
3. Métodos de arrays em JavaScript (push, pop, shift, unshift, splice, slice, map, filter, reduce)
4. O que são objetos em JavaScript?
5. Como criar e manipular objetos em JavaScript?

MÓDULO 10

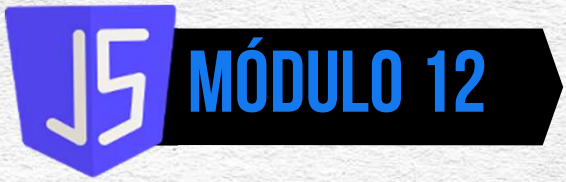
Callbacks e Promises

1. O que são callbacks?
2. Como usar callbacks em JavaScript?
3. O que são Promises?
4. Como criar e usar Promises em JavaScript?
5. Como lidar com erros em Promises?
6. Lidando com múltiplas Promises

MÓDULO 11

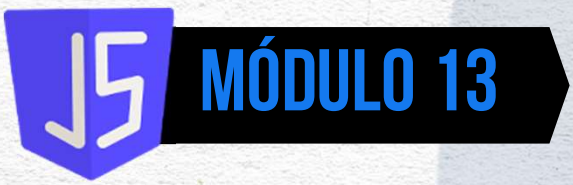
Arrow Functions:

1. O que são arrow functions?
2. Como usar arrow functions em JavaScript?
3. Como as arrow functions diferem das funções tradicionais?
4. Como lidar com o escopo das arrow functions?



Destructuring

1. O que é destructuring em JavaScript?
2. Como usar o destructuring em arrays e objetos?
3. Como o destructuring é usado em parâmetros de funções?

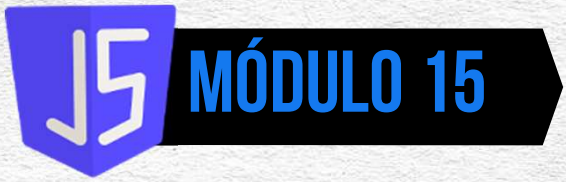


Spread syntax

1. O que é spread syntax em JavaScript?
2. Como usar o spread syntax em arrays e objetos?
3. Como o spread syntax é usado em parâmetros de funções?

Classes e Objetos

1. O que são classes em JavaScript?
2. Como criar classes em JavaScript?
3. Como herdar de classes em JavaScript?
4. Como usar o construtor em classes?
5. Como usar os métodos em classes?



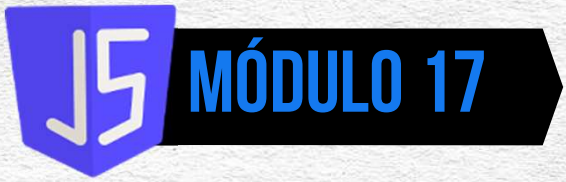
Herança e prototipagem

1. O que é herança em JavaScript?
2. Como herdar de objetos em JavaScript?
3. O que é prototipagem em JavaScript?
4. Como usar a prototipagem em JavaScript?
5. Como criar objetos a partir de funções construtoras?

Módulos e importações/exportações

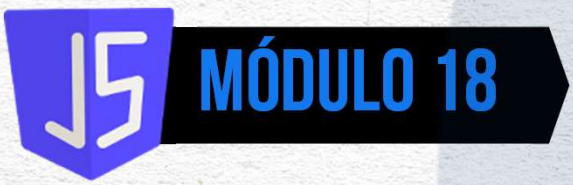
1. O que são módulos em JavaScript?
2. Como criar e exportar módulos em JavaScript?
3. Como importar e usar módulos em JavaScript?
4. Quais são as diferenças entre export default e export named em JavaScript?

PARTE 2
NODE JS



Conceitos básicos do NodeJs

1. O que é NodeJs?
2. Como instalar e configurar o NodeJs?
3. Como criar um servidor HTTP básico com NodeJs?

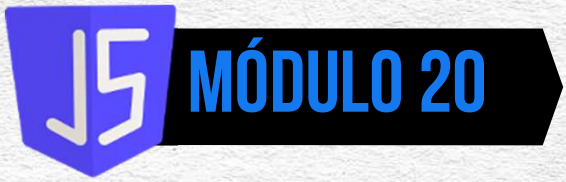


Frameworks para desenvolvimento de APIs

1. Quais são os frameworks mais populares para desenvolvimento de APIs em NodeJs?
2. Como criar uma API RESTful com o Express.js?

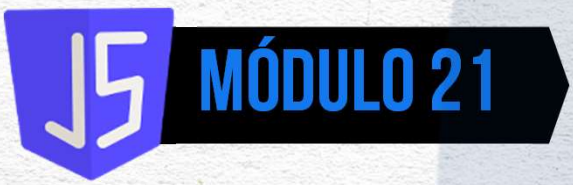
Middlewares

1. O que são middlewares em NodeJs?
2. Como usar middlewares em uma API?
3. Como criar middlewares personalizados em uma API?



Tratamento de requisições e respostas

1. *Como lidar com requisições e respostas em uma API?*
2. *Como usar os métodos HTTP (GET, POST, PUT, DELETE) em uma API?*
3. *Como validar e sanitizar os dados de entrada na API?*



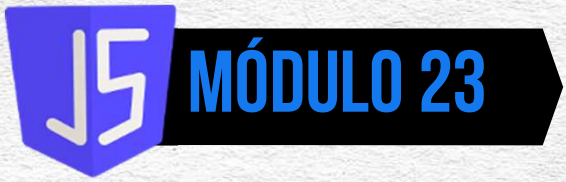
Autenticação e autorização

1. *Como implementar autenticação e autorização em uma API?*
2. *Quais são os métodos mais populares de autenticação (token, OAuth, JWT)?*
3. *Como proteger rotas específicas da API com autorização?*

Conexão com banco de dados

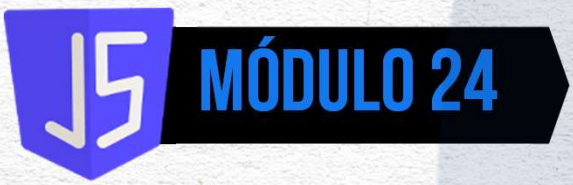
1. *Como conectar uma API com um banco de dados em NodeJs?*
2. *Quais são as bibliotecas mais populares para conexão com bancos de dados em NodeJs (MongoDB, PostgreSQL)?*
3. *Como realizar operações básicas em um banco de dados na API?*

PARTE 3
REACT JS



Componentes

1. O que são componentes e como eles são usados no ReactJS
2. Como criar componentes de função
3. Como passar dados entre componentes
4. Como usar componentes de terceiros

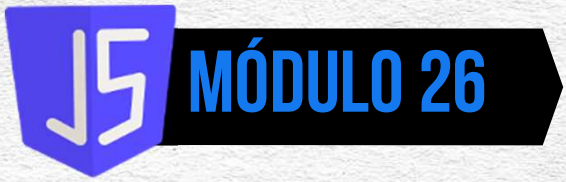


JSX

1. O que é JSX e como ele é usado no ReactJS
2. Sintaxe de JSX, incluindo como renderizar elementos, componentes e expressões
3. A diferença entre JSX e HTML

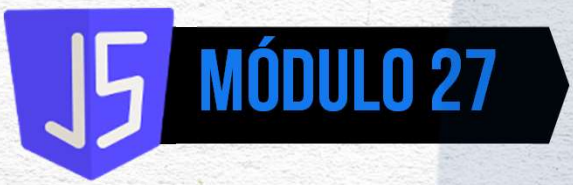
State

1. O que é o estado e como usá-lo em um componente
2. Como atualizar o estado e renderizar o componente novamente
3. Como inicializar o estado e lidar com valores iniciais



Props

1. *O que são as props e como elas são usadas em um componente*
2. *Como passar props para um componente*
3. *Como definir valores padrão para props*
4. *Como usar destructuring de props*



Eventos

1. *Como lidar com eventos em um componente ReactJS*
2. *Lista de eventos comuns, como onClick e onChange*
3. *Como passar dados de eventos de volta para o componente pai*

Hooks

1. *O que são hooks e como eles são usados em componentes funcionais*
2. *Hooks comuns, como useState e useEffect*
3. *Como criar e usar hooks personalizados*

MÓDULO 29

Gerenciamento de estados

1. O que é o gerenciamento de estado e por que é importante
2. Como definir um store e como usá-lo em um aplicativo ReactJS

MÓDULO 30

Roteamento

1. O que é roteamento e por que é importante
2. Como implementar o roteamento usando o React Router
3. Como criar rotas aninhadas e rotas protegidas

MÓDULO 31

Boas práticas

1. Separando a lógica de negócios da interface do usuário
2. Mantendo os componentes pequenos e reutilizáveis
3. Usando a propagação de propriedades (prop drilling) ou Context API
4. Evitando mutar o estado diretamente e atualizando o estado de forma imutável.