**Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente**

**Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente**

**Uma imagem contendo Diagrama

Descrição gerada automaticamente**

[**Iniciando · Jest (jestjs.io)**](https://jestjs.io/pt-BR/docs/getting-started) **– DOCUMENTAÇÃO OFICIAL JEST**

[**import - JavaScript | MDN (mozilla.org)**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import) **– DOCUMENTAÇÃO OFICIAL JS PRA IMPORTAÇÃO DE ARQUIVOS E BIBLIOTECAS**

**Os testes automatizados são uma prática essencial no desenvolvimento de software, pois garantem que as aplicações funcionem corretamente sem a necessidade de intervenção manual constante. Aqui estão os principais pontos sobre testes automatizados:**

1. **Definição: Testes automatizados utilizam scripts e ferramentas para verificar automaticamente se o software atende aos requisitos esperados. Eles são executados por meio de bibliotecas específicas, como o Jest, que permitem simular entradas e validar saídas.**
2. **Benefícios:**
   * **Consistência: Os testes são realizados da mesma forma todas as vezes, reduzindo a chance de erros humanos.**
   * **Rapidez: Um único comando pode executar uma bateria de testes, economizando tempo em comparação com testes manuais.**
   * **Repetibilidade: Os testes podem ser executados sempre que necessário, como após alterações no código, garantindo que novas funcionalidades não quebrem o que já estava funcionando.**
3. **Tipos de Testes Automatizados:**
   * **Testes Unitários: Verificam se funções ou métodos individuais retornam os resultados esperados.**
   * **Testes de Integração: Avaliam se diferentes partes do sistema funcionam bem juntas.**
   * **Testes End-to-End: Simulam o comportamento do usuário em toda a aplicação, testando fluxos completos.**
4. **Ferramentas Comuns:**
   * **Jest: Muito utilizado no ecossistema JavaScript, ideal para testes unitários e de integração.**
   * **Cypress e Selenium: Usados para automação de testes em navegadores, permitindo a simulação de interações do usuário.**
5. **Importância: A automação de testes é crucial para garantir a qualidade do código antes de ser colocado em produção, evitando problemas que poderiam impactar a experiência do usuário, como falhas em uma loja online.**

**Para um entendimento mais aprofundado, você pode consultar os seguintes conteúdos:**

* [**Tipos de teste**](https://aulas.cubos.academy/conteudo/8c821a4a-1ffc-4d6b-9f08-ed536a6d4262)
* [**Teste Manual vs Teste Automatizado**](https://aulas.cubos.academy/conteudo/e1c9c840-0e61-42a3-b4f6-b542e70647cd)
* [**Introdução a testes automatizados**](https://aulas.cubos.academy/conteudo/75b71f4f-f5dd-420e-babf-df719eb6f552)

**Jest**

**Para baixar o jest: google > jest > site jestjs.io/pt-BR/docs/getting-started**

Esta aula é essencial para os desenvolvedores que desejam aprimorar suas habilidades de criação de testes automatizados e garantir a eficácia e precisão do código escrito. A aula aborda o conceito de testes automatizados, seu funcionamento, e por que eles são necessários. Os alunos terão a oportunidade de explorar a biblioteca Jest, uma das mais prevalentes para testes automatizados no ambiente JavaScript. A aula descreve claramente como escrever um teste usando a biblioteca Jest e como conduzir a implementação de software através de testes. Além disso, a aula também orienta sobre como instalar a biblioteca Jest usando o Node Package Manager (npm). Este conteúdo é altamente benéfico para aqueles que querem aprimorar seu desenvolvimento de software e assegurar a qualidade do código através de testes rigorosos e eficazes.

1. Mocking:
   * Conceito: trata-se de simular um comportamento para evitar a dependência de outros componentes / sistemas em um teste automatizado.
   * Utilização no Jest: exemplos práticos de utilização e como o mocking pode facilitar a execução de testes isolados.
2. Teste de integração:
   * Definição: teste que visa verificar se a combinação de diferentes partes do software funciona corretamente em conjunto.
   * Como criar um teste de integração utilizando Jest: passo a passo para implementação.
3. Manutenção de testes automatizados:
   * Como manter a base de testes atualizada conforme o software evolui.
   * Práticas para assegurar que testes continuem válidos e úteis conforme novas funcionalidades são adicionadas ao software.
   * O impacto positivo da manutenção de testes para a qualidade do software.

**Exportanto e importando módulos**

Esta aula oferece uma visão detalhada sobre o mundo dos testes automatizados, um componente essencial para qualquer desenvolvedor de software. Ela serve como um forte recurso para revisar os conceitos-chave na construção de aplicações JavaScript, além de focar nas diferentes formas de importar bibliotecas, oferecendo exemplos pragmáticos de como e quando utilizar require e import. Além disso, a aula ensina como exportar funcionalidades e demonstra como objetos podem ser usados para exportar múltiplas funcionalidades. A aula também enfatiza a desestruturação de objetos ao importar e a diferença crucial entre exportar bibliotecas e funcionalidades implementadas pelo próprio desenvolvedor. Esta aula será de grande valia para aqueles interessados em aprofundar seus conhecimentos em JavaScript e em otimizar seus processos de teste e desenvolvimento.

* Compreensão do paradigma da programação orientada a objetos
* Conceitos fundamentais de design de software
* Introdução aos princípios de teste de unidades
* Uso de bibliotecas de assertivas para implementação de testes
* Exploração da biblioteca Jest para realização de testes
* Discussão sobre boas práticas de escrita de teste de unidade
* Conceito de mock e sua aplicação em testes de software
* Análise de código para identificação de erros e bugs.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

As importações em arquivos diferentes em JavaScript podem ser feitas de duas maneiras principais: utilizando **CommonJS** e **ES Modules**. Vamos explorar como isso funciona em cada um desses métodos.

**1. Usando CommonJS**

No Node.js, a forma tradicional de importar e exportar módulos é através do CommonJS, que utiliza require e module.exports.

**Exemplo de Exportação e Importação**

* **Exportação**: Vamos criar um arquivo chamado soma.js que contém uma função para somar dois números.

// soma.js

function soma(a, b) {

return a + b;

}

module.exports = soma; // Exporta a função soma

* **Importação**: Agora, em um arquivo index.js, você pode importar a função soma e utilizá-la.

// index.js

const soma = require('./soma'); // Importa a função do arquivo soma.js

console.log(soma(2, 3)); // Saída: 5

**2. Usando ES Modules**

Com a introdução dos ES Modules, a sintaxe de importação e exportação foi modernizada, permitindo uma abordagem mais clara e organizada.

**Exemplo de Exportação e Importação**

* **Exportação**: Você pode criar um arquivo soma.mjs e usar a palavra-chave export.

// soma.mjs

export function soma(a, b) {

return a + b;

}

* **Importação**: Para importar a função em um arquivo index.mjs, você usaria a sintaxe import.

// index.mjs

import { soma } from './soma.mjs'; // Importa a função soma

console.log(soma(2, 3)); // Saída: 5

**Considerações Importantes**

* **Configuração para ES Modules**: Para usar ES Modules, você precisa configurar o arquivo package.json do seu projeto com "type": "module".
* **Desestruturação**: Ao exportar múltiplas funções ou variáveis, você pode usar a desestruturação para importar apenas o que precisa.

**Links Úteis**

Para mais detalhes sobre como realizar importações e exportações em arquivos diferentes, você pode assistir às seguintes aulas:

* [Exemplo de importação de arquivos](https://aulas.cubos.academy/conteudo/73c50fde-7dcd-47ed-92c6-8dcf60f3699c)
* [Import x Require (ES Modules ou CommonJS)](https://aulas.cubos.academy/conteudo/86aacd36-1b7a-4de1-964e-e222cc8947fc)
* [Uso dos ES Modules](https://aulas.cubos.academy/conteudo/1b71dcd1-6c2d-42ad-bff7-2f2817b931b0)

Esses conceitos são fundamentais para organizar e modularizar o seu código em projetos JavaScript.

Para poder importar um arquivo do mesmo projeto/nível, (require(./nome\_do\_arquivo) a gente precisa usar o module.export= nome da função ou o que eu quero exportar. pra exportar e poder importar em outro arquivo. (bibliotecas já fazem a exportação automática por isso n precisamos fazer nelas)

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Exportando objeto

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Exportando e importando módulos com ES Modules**

**É a forma mais nova. Mais moderna**

Esta aula faz um aprofundamento em aplicações modernas, focando no desenvolvimento de aplicativos com a utilização de padrões modernos do JavaScript. Desde a inicialização de um projeto com o npm, até a exportação e importação com S Modules, o curso fornece orientações práticas e exemplos concretos para ajudar a aprimorar seus conhecimentos.

O curso também destaca erros comuns e suas resoluções, tornando-o uma excelente fonte de aprendizado não apenas para novatos na programação, mas também para aqueles que querem se atualizar sobre as práticas mais recentes. Além disso, a explicação clara sobre o uso de Export e Import faz com que o conteúdo seja facilmente aplicável. Esta aula é indispensável para quem está investindo no desenvolvimento moderno em JavaScript. Você será prometido futuras discussões sobre variações nas práticas de exportação e importação para aprimorar ainda mais suas habilidades.

1. Trabalhando com pacotes de terceiros
   * Importação e utilização de pacotes externos.
   * A importância de verificar a compatibilidade dos pacotes com S Modules.
2. Exercícios práticos e revisões
   * Revisão dos tópicos abordados através de exemplos concretos.
   * Prática de resolução de erros de importação e exportação.
   * Desenvolvimento de pequenas aplicações para solidificar o conhecimento adquirido.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Precisa iniciar o npm init -y pra iniciar o projeto e ir em package.json e adicionar em qualquer lugar o “type”: “module”,

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

**Exportação individual com ES Modules**

Este curso aborda o conceito de exportação de módulos em JavaScript, uma habilidade essencial para os programadores. Os alunos aprendem como exportar seletivamente informações de um módulo, de forma a personalizar o que é exposto no código. O instrutor usa exemplos claros para mostrar como usar a palavra-chave 'export' para exportar informações, bem como o que acontece se você tenta usar uma variável ou função que não foi exportada.

Este curso vai além da mera exportação de módulos; também ensina como essas informações exportadas podem ser usadas de maneira eficiente em outros arquivos de código. Essa habilidade impulsiona a flexibilidade e a eficácia do código, permitindo que os programadores exponham apenas os elementos necessários de um módulo. O curso é prático e interativo, incentivando os alunos a experimentar e ver os resultados na prática.

1. Módulo padrão vs Módulo individualizado:
   * Discussão sobre a diferença entre a exportação de um módulo inteiro e o avanço da exportação individual dentro dele.
2. Prática e Exercícios:
   * Realização de exercícios para reforçar o entendimento sobre exportação de módulos e exportação individual. Os exemplos utilizados incluem a criação de diferentes cenários de exportação e problemas a serem resolvidos pelos alunos.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Exportação default com ES Modules**

Esta aula é muito útil para aqueles interessados em progredir suas habilidades em JavaScript, especialmente no que diz respeito à importação e exportação de módulos. O enfoque é dado na exportação individual e na exportação com o módulo padrão (default), juntamente com suas respectivas características e contextos de utilização. Além disso, as aulas apresentam como nomear sua importação, dando liberdade para definir como preferir. No fim deste módulo, a habilidade adquirida será extremamente valiosa, pois é aplicável em praticamente todos os projetos JavaScript, tornando o código mais modular, manutenível e fácil de entender. Portanto, é uma grande oportunidade para aprofundar seus conhecimentos e tornar-se mais proficiente em desenvolvimento JavaScript.

* Tópico principal: Exportação e Importação no JavaScript

1. Definição e uso da Exportação Individual.
2. Conceito e aplicação da Exportação com Módulo Padrão (default).
3. Aprimorando o uso do módulo padrão no JavaScript.
4. Conceito de Nomeação da Importação e como aplicá-la.
5. Uso prático da Nomeação da Importação em programas.
6. Exemplificação de como usar a exportação e importação no JavaScript.
7. Discussão da importância pragmática do uso de exportação e importação.
8. Aplicação desse conceito em projetos futuros de programação.

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

O QUE EU EXPORTAR COM DEFAULT, QUE TEM QUE SER UM CONJUNTO COM TUDO QUE EU QUERO, N POSSO CRIAR VÁRIOS, EU CRIO UM IMPORT (ALGUMACOISA – QUE VIRA UM MÓDULO) E AÍ POSSO ACESSAR AS PROPRIEDADES EXPORTADAS. NÃO CONSIGO FAZER DESESTRUTURAÇÃO PQ FOI IMPORTADO COMO MÓDULO PADRÃO

FICANDO DESSA FORMA ABAIXO:

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

**Criando meu primeiro teste automatizado**

Esta aula é um tutorial completo sobre a configuração e o uso do Jest, uma potente ferramenta de testes JavaScript. À medida que você progride na aula, aprenderá como instalar o Jest como uma dependência de desenvolvimento, criar e configurar um arquivo de teste, e usar o comando npx jest para executar seu primeiro teste. Adicionalmente, o curso também oferece uma visão prática e útil de como utilizar as funções test e expect para escrever e verificar os resultados dos testes. Por fim, você também será guiado através do processo de configuração do arquiv package.json para usar o ECMAScript Module (ESM) no Jest. Esta aula é uma oportunidade imperdível para qualquer um que queira aprofundar seus conhecimentos em JavaScript e testes automatizados.

* Introdução ao framework de teste Jest
* Compreensão de como instalar e configurar o Jest
* Criação e nomeação de arquivos de teste para reconhecimento pelo Jest
* Escrevendo testes básicos utilizando a função test
* Verificando os resultados do teste com a função expect
* Executando testes comandos Jest no Terminal
* Configurando Jest para suportar módulos ES6 com 'npx jest'
* Adicionando configurações personalizadas no arquivo package.json para Jest

Para configurar o Jest após a instalação, você pode seguir as etapas abaixo:

1. **Inicialização do Jest**:
   * Utilize o comando npx jest --init para criar automaticamente um arquivo de configuração do Jest (jest.config.js ou jest.config.ts, dependendo do seu projeto).
2. **Responda às Perguntas**:
   * Durante a inicialização, você será solicitado a responder a algumas perguntas, como:
   * Qual ambiente você deseja usar? (por exemplo, Node)
   * Você deseja coletar cobertura de código?
   * Deseja limpar automaticamente os mocks?
3. **Ajustes no Arquivo de Configuração**:
   * Após a criação do arquivo de configuração, você pode abri-lo e fazer ajustes conforme necessário. Por exemplo, definir o preset para ts-jest se estiver usando TypeScript.
4. **Scripts no package.json**:
   * Adicione scripts no seu arquivo package.json para facilitar a execução dos testes. Por exemplo:

"scripts": {

"test": "jest",

"test:watch": "jest --watch",

"test:coverage": "jest --coverage"

}

1. **Executando os Testes**:
   * Para rodar os testes, utilize o comando npm run test ou npx jest.
2. **Verificação**:
   * Após configurar, crie um teste simples para garantir que tudo está funcionando corretamente. Por exemplo, teste uma função de soma e verifique se o resultado está correto.

Pra rodar o arquivo jest: npx jest

**Configuração do GIT e primeiro commit**

Esta aula é ideal para quem deseja se familiarizar com o Git, uma ferramenta essencial de controle de versão usado por desenvolvedores de todo o mundo. Com a capacidade de rastrear alterações no código, o Git é uma habilidade necessária não apenas para esta série de aulas, mas também para qualquer projeto de desenvolvimento de software futuro. A aula ensina desde os primeiros passos, como inicializar o Git em um projeto e adicionar arquivos para acompanhamento, até a execução de seu primeiro commit. Uma parte crucial da aula é dedicada ao arquivo .gitignore, uma ferramenta útil para excluir arquivos e pastas desnecessários do rastreamento do Git, importante para manter os repositórios limpos e organizados. Com exemplos práticos e passo-a-passo, esta aula é uma excelente maneira de aprimorar suas habilidades de desenvolvimento de software e se preparar para futuros projetos.

* Utilização do Git para rastrear alterações de código.
* Inicialização do Git em um projeto utilizando o comando "Git init". \*\*\*
* Significado e utilização do arquivo .gitignore para excluir arquivos/pastas desnecessários.
* Adição de alterações na "stage" do Git através do comando "Git add .".
* Prática de realizar um commit através do comando "Git commit -m".
* Gerenciamento de repositórios e importância do Git para futuros projetos.
* Análise de um exemplo prático de configuração inicial do Git em um projeto.
* Estudo de regras para adição de arquivos e pastas ao arquivo .gitignore.