Resumo Completo com Explicações: Programação Front-End (Unicesumar, 2025)

UNIDADE 1: Introdução à Programação Front-End

Esta unidade apresenta o que é desenvolvimento front-end, sua importância e tecnologias essenciais para iniciar nesta área.

Explicações:

- **Front-end vs. Back-end**: Front-end é a parte visual, com que o usuário interage; back-end é a parte lógica que roda no servidor.
- **Principais linguagens**:
- **HTML**: estrutura do conteúdo.
- **CSS**: aparência e estilo visual.
- **JavaScript**: interação e comportamento dinâmico.
- **Cliente-servidor**: O navegador (cliente) faz requisições a um servidor, que retorna dados (ex: páginas web, informações de uma API).
- **Ferramentas essenciais**: VS Code (editor), navegadores com DevTools (Chrome/Firefox), servidores locais como Live Server.
- **Frameworks**: São bibliotecas prontas para acelerar o desenvolvimento. Ex: React.js. Porém, o ideal é aprender antes sem eles.

Perguntas e Respostas:

1. **Qual a função do front-end?** R: Criar interfaces e interações com o usuário.
2. **Cite as três linguagens base do front-end.** R: HTML, CSS e JavaScript.
3. **O que é o modelo cliente-servidor?** R: É a comunicação entre navegador e servidor usando requisições e respostas.

UNIDADE 2: Criando Páginas com HTML e CSS

Explicações:

- **HTML** é uma linguagem de marcação que utiliza tags para definir títulos, parágrafos, imagens, links, etc.
- **Tags semânticas** como `<header>`, `<main>`, `<footer>` ajudam na acessibilidade e SEO.
- **Exemplo de estrutura básica**:

```
<!DOCTYPE html>
<html>
<head><title>Exemplo</title></head>
<body>
  <h1>Olá Mundo</h1>
  <p>Primeiro parágrafo</p>
</body>
</html>
```

- **CSS** é usado para estilizar. Pode ser:
- Inline: dentro da tag.
- Interno: no `<head>` com `<style>`.
- Externo: arquivo `.css` separado (recomendado).
- **Modelo de caixa (box model)**: Cada elemento tem `content`, `padding`, `border` e `margin`.
- **Seletores**: `p` (tag), `.classe`, `#id`.
- **Flexbox**: usado para alinhar elementos com facilidade. Ex:

```
.container {
  display: flex;
  justify-content: space-between;
}
```

Perguntas e Respostas:

1. **Para que serve o HTML?** R: Estruturar o conteúdo da página.
2. **O que é o CSS?** R: Uma linguagem para estilizar o HTML.
3. **O que significa "box model"?** R: Modelo que representa a estrutura visual dos elementos (conteúdo, padding, borda e margem).
4. **Qual a diferença entre classe e id no CSS?** R: Classe pode ser reutilizada, id deve ser único.

UNIDADE 3: Dando Vida aos Sistemas com JavaScript

Explicações:

- **JavaScript** é uma linguagem de programação interpretada pelo navegador, usada para tornar páginas interativas.
- **DOM (Document Object Model)**: Representa a estrutura da página. Permite acessar e modificar elementos HTML via JS.
- **Eventos**: Ações do usuário, como clique ou digitação.
- **Exemplo**:

```
document.querySelector("#botao").addEventListener("click", () =>
alert("Clicado!"));
```

- **Variáveis**: `let`, `const`, `var`. `const` não muda, `let` muda, `var` é mais antigo.
- **Funções, arrays, objetos** são estruturas básicas da linguagem.

Perguntas e Respostas:

1. **O que é o DOM?** R: Representação dos elementos da página que pode ser manipulada via JS.
2. **Para que serve o JavaScript?** R: Tornar as páginas interativas e dinâmicas.
3. **Diferença entre let e const?** R: `let` pode mudar, `const` não.

UNIDADE 4: Sistemas que Conversam entre Si

Explicações:

- **APIs** permitem que sistemas se comuniquem.
- **HTTP** é o protocolo usado para isso. Métodos:
- `GET`: buscar dados
- `POST`: enviar dados
- **JSON** é o formato comum de dados entre sistemas.
- ``: método JS para consumir APIs.

```
fetch("https://api.exemplo.com")
  .then(res => res.json())
  .then(data => console.log(data));
```

- **Acessibilidade (WCAG)**: práticas para tornar sites utilizáveis por todos.

Perguntas e Respostas:

1. **O que é uma API?** R: Interface de comunicação entre sistemas.
2. **Qual método é usado para buscar dados?** R: GET.
3. **Por que acessibilidade é importante?** R: Garante que todos possam usar o site, inclusive pessoas com deficiências.

UNIDADE 5: Otimização de Desempenho e Testes

Explicações:

- **Desempenho**:
- Reduzir tamanho de arquivos (minificar).
- Usar imagens otimizadas e lazy loading.
- Aproveitar cache do navegador.
- **Testes**:
- **Manuais**: feitos pelo desenvolvedor/testador.
- **Automatizados**: feitos com ferramentas como Jest.
- **Testes de unidade**: verificam partes isoladas do código.

```
test("Soma de 2 e 2", () => {
  expect(2 + 2).toBe(4);
});
```

Perguntas e Respostas:

1. **O que é minificação de arquivos?** R: Remoção de espaços e comentários para reduzir o tamanho.
2. **Qual vantagem dos testes automatizados?** R: Evita erros e economiza tempo em atualizações.
3. **O que é lazy loading?** R: Carregamento sob demanda de conteúdo (como imagens).

Dicas de Estudo:

- Visite o MDN Web Docs.
- Refaça os exemplos do livro.
- Teste códigos com o DevTools dos navegadores.
- Use sites como W3Schools para prática interativa.

Ideias Extras:

- Monte um portfólio aplicando os conhecimentos.
- Crie projetos com APIs públicas.
- Aplique boas práticas de acessibilidade desde o início.