Matéria: Informática

Assunto: HTML, TypeScript

Resumo Teórico do Assunto

Para dominar as questões apresentadas, é fundamental compreender os conceitos de **HTML** para estruturação de conteúdo e **TypeScript** (que compila para JavaScript) para interatividade e manipulação do DOM.

---

1. HTML: Estrutura e Atributos Essenciais

HTML (HyperText Markup Language) é a linguagem padrão para criar páginas web. Ele define a estrutura e o conteúdo de uma página.

• Estrutura Básica:
* ``: Declaração do tipo de documento, informando ao navegador que é um documento HTML5.
* ``: O elemento raiz de uma página HTML. O atributo `lang` define o idioma principal do conteúdo.
* ``: Contém metadados sobre a página, como título, links para folhas de estilo (CSS) e scripts (JavaScript/TypeScript). O conteúdo dentro de `` não é exibido diretamente no navegador.
* ``: Contém todo o conteúdo visível da página web, como texto, imagens, links, formulários, etc.

• Elementos de Formulário (`` e ``):
* ``: Define um formulário HTML para entrada de dados do usuário.
* ``: Um elemento de formulário versátil usado para criar controles interativos para formulários baseados na web. Seu comportamento é determinado pelo atributo `type`.
* `type='text'`: Cria um campo de texto de linha única.
* `type='submit'`: Cria um botão para enviar o formulário.

• Atributos Chave de Elementos HTML:
* `id`: Um **identificador único** para um elemento HTML dentro de todo o documento. É crucial para a manipulação via JavaScript/TypeScript, pois permite selecionar um elemento específico sem ambiguidade.
* `class`: Um nome de classe para um elemento HTML. Vários elementos podem ter a mesma classe, permitindo que sejam estilizados ou manipulados em grupo.
* `name`: Usado principalmente para identificar dados de formulário quando são enviados ao servidor. Também pode ser usado para agrupar elementos de rádio ou checkbox.

* `value`: Define o valor inicial de um campo de entrada (``) ou o valor que será enviado com o formulário.

• Inclusão de Scripts:
* ``: Usado para vincular um arquivo JavaScript externo à página HTML. O navegador baixará e executará o script. No contexto das questões, um arquivo **TypeScript** (`.ts`) é compilado para um arquivo **JavaScript** (`.js`), que é então referenciado no HTML.

---

2. DOM (Document Object Model) e Manipulação com JavaScript/TypeScript

O **DOM** é uma interface de programação para documentos HTML e XML. Ele representa a estrutura de um documento como uma árvore de objetos, onde cada nó é um objeto que representa uma parte do documento (um elemento, um atributo, um texto, etc.). O JavaScript (e, por extensão, o TypeScript) pode usar o DOM para acessar e manipular o conteúdo, a estrutura e o estilo de documentos HTML.

• `document.querySelector()`:
* Este é um método do objeto `document` que permite selecionar o **primeiro elemento** no documento que corresponde a um **seletor CSS** especificado.
* **Sintaxe:** `document.querySelector('seletorCSS')`
* **Seletores CSS Comuns para `querySelector`:**
* `#idDoElemento`: Seleciona um elemento pelo seu atributo `id`. Ex: `#idTexto` selecionaria o elemento com `id="idTexto"`.
* `.nomeDaClasse`: Seleciona o primeiro elemento pela sua classe. Ex: `.classe-input` selecionaria o primeiro elemento com `class="classe-input"`.
* `nomeDaTag`: Seleciona o primeiro elemento pela sua tag HTML. Ex: `input` selecionaria o primeiro elemento ``.
* `[atributo=valor]`: Seleciona o primeiro elemento com um atributo específico e valor. Ex: `[name='texto']`.

• Acessando Valores de Input:
* Uma vez que um elemento `` é selecionado (por exemplo, usando `document.querySelector()`), seu valor pode ser acessado através da propriedade `.value`.
* Ex: `const meuInput = document.querySelector('#idDoInput'); console.log(meuInput.value);`

• Eventos do Navegador (`onload`):
* `onload`: É um evento que ocorre quando a página inteira (incluindo todos os recursos como imagens, scripts, etc.) foi completamente carregada. É um bom lugar para iniciar a manipulação do DOM, garantindo que todos os elementos estejam disponíveis.

• TypeScript Type Assertion (`as Tipo`):
* Em TypeScript, `as Tipo` é uma **asserção de tipo**. Ela informa ao compilador que você

sabe mais sobre o tipo de um valor do que o próprio compilador. Por exemplo, `document.querySelector(...) as HTMLInputElement` diz ao TypeScript que o elemento retornado é, de fato, um elemento de input HTML, permitindo que você acesse propriedades específicas de inputs como `.value` sem erros de compilação.

---

3. TypeScript: Funções, Parâmetros e Métodos de Array

TypeScript é um superconjunto tipado de JavaScript que compila para JavaScript puro. Ele adiciona tipagem estática opcional, classes, interfaces e outros recursos que ajudam a construir aplicações mais robustas e escaláveis.

• **Funções em TypeScript:**
* **Sintaxe de Arrow Function:** `(param1: Tipo1, param2: Tipo2) => retorno`
* Ex: `const soma = (a: number, b: number): number => a + b;`
* **Tipagem de Parâmetros e Retorno:** TypeScript permite especificar os tipos de dados esperados para os parâmetros de uma função e o tipo de dado que a função irá retornar. Isso melhora a legibilidade e ajuda a prevenir erros.

• **Parâmetros Rest (`...args`):**
* Permitem que uma função aceite um número indefinido de argumentos como um array.
* **Sintaxe:** `(...nomeDoArray: Tipo[])`
* Ex: `const logNumeros = (...numeros: number[]) => { console.log(numeros); };`
* `logNumeros(1, 2, 3)` faria `numeros` ser `[1, 2, 3]`.

• **Método `Array.prototype.reduce()`:**
* O método `reduce()` executa uma **função redutora** (fornecida por você) em cada elemento do array, resultando em um único valor de saída.
* **Sintaxe:** `array.reduce(reducerFunction, initialValue)`
* `reducerFunction`: Uma função que é executada em cada elemento do array. Ela recebe geralmente dois argumentos principais:
* `accumulator`: O valor retornado da última invocação do `reducerFunction`, ou o `initialValue` na primeira invocação.
* `currentValue`: O elemento atual do array que está sendo processado.
* (Opcional: `currentIndex`, `array`)
* `initialValue`: (Opcional) Um valor para ser usado como o primeiro argumento (`accumulator`) na primeira chamada da `reducerFunction`. Se não for fornecido, o primeiro elemento do array será usado como `accumulator` e a iteração começará a partir do segundo elemento.
* **Exemplo Comum (Soma):**
```typescript
const numeros = [1, 2, 3, 4];
const somaTotal = numeros.reduce((acumulador, valorAtual) => acumulador + valorAtual, 0);
// Na primeira iteração: acumulador = 0, valorAtual = 1 -> retorna 1
```

// Na segunda iteração: acumulador = 1, valorAtual = 2 -> retorna 3
// Na terceira iteração: acumulador = 3, valorAtual = 3 -> retorna 6
// Na quarta iteração: acumulador = 6, valorAtual = 4 -> retorna 10
// Resultado final: 10
```

* Para que `reduce` funcione corretamente, a `reducerFunction` deve aceitar pelo menos dois argumentos (acumulador e valor atual) e retornar o novo valor do acumulador.

---

Compreender esses conceitos permitirá que você analise o código HTML para identificar elementos e seus atributos, e o código TypeScript para entender como ele interage com o DOM e como as funções e métodos de array são utilizados para processar dados.

---

## Questões de Provas Anteriores

**Fonte: escriturario_agente_de_tecnologia (1).pdf, Página: 16**

**pcimarkpci MjgwNDowMTRkOjE0YTU6OTI1ODozOGQ2OjNhMGM6NTM0MzplZmI1:U3VuLCAyNyBKdWwgMjAyNSAyMzo0Nzo0MCAtMDMwMA==**

**16**
**BANCO DO BRASIL**
**AGENTE DE TECNOLOGIA - Microrregião 16 DF-TI GABARITO 1**
**47**
**Considere o código HTML a seguir.**

```
<!doctype html>
<html lang=”pt-br”>
<head>
<script src=”script.js”></script>
</head>
<body>
<form>
Texto: <input type=’text’ name=’texto’ id=’idTexto’ class=’classe-input’
value=’Texto inicial’><br>
<input type=’submit’ value=’Envia’>
</form>
</body>
</html>
```

Considere, também, o arquivo TypeScript script.ts, listado a seguir, que irá gerar o arquivo script.js no mesmo diretório
do arquivo HTML, apresentado acima.
```

onload = (event) => {
const texto = document.querySelector('???') as HTMLInputElement;
console.log('Texto inicial: ', texto.value);
};
Que texto o programador deverá utilizar no lugar de ???, no código do arquivo TypeScript script.ts, para exibir o valor do
campo HTML input na console?

(A) #classe-input
(B) #idTexto
(C) #texto
(D) .idTexto
(E) .texto
48
Em TypeScript 4, é possível usar o seguinte fragmento de código:
// definir x
const y = (...args: number[]) => args.reduce(x, 0);
Que fragmento de código apresenta uma versão compilável e executável da definição de x que poderia aparecer no lugar
do comentário "// definir x" ?
(A) const x = 1;
(B) const x = [1,2,3];
(C) const x = (a:number) => [a*2];
(D) const x = (a:number[]) => a[0];
(E) const x = (a:number,b:number) => a+b;
RASCUNHO