JavaScript: Fundamentos para Desenvolvimento Web Interativo - Turma 2025B

2.1 Declarações

No JavaScript, cada instrução é chamada de "declaração" e geralmente é separada por um ponto e vírgula (**;**). Espaços, tabulações e novas linhas são considerados " **espaços em branco**". No entanto, ao contrário de algumas linguagens, o JavaScript pode tolerar a ausência de ponto e vírgula em certas situações, graças ao recurso de Inserção Automática de Ponto e Vírgula (Automatic Semicolon Insertion, ASI).

OBS: O ASI é um recurso do JavaScript que insere automaticamente pontos e vírgulas em alguns casos específicos, como:

- Fim de linha: Se uma linha termina sem ponto e vírgula e a próxima linha não causa conflito, o JavaScript insere um ponto e vírgula automaticamente.
- Fim de bloco: Se uma instrução termina com um bloco de código (por exemplo, em uma função), o JavaScript pode inserir um ponto e vírgula.

Vejamos no quadro abaixo dois exemplos onde ambos os blocos de código são válidos:

```
// Exemplo 1
let x = 5
x += 1
console.log(x) // Saída: 6


// Exemplo 2
let x = 5;
x += 1;
console.log(x); // Saída: 6
```

Embora o ASI pareça facilitar o código, ele pode criar problemas quando a ausência do ponto e vírgula leva a uma interpretação errada do código. Por exemplo, se uma linha termina com uma expressão incompleta e a linha seguinte pode ser interpretada como uma continuação, o ASI pode não funcionar conforme esperado.

Declarações de variáveis

As **variáveis** são usadas para armazenar valores que podem ser alterados no decorrer do tempo. Elas são referenciadas por nomes simbólicos, chamados de "**identificadores**". A nomenclatura das variáveis segue regras específicas para garantir a consistência e a compatibilidade em toda a aplicação, a saber:

- **Início do nome:** Um identificador deve começar com uma letra (maiúscula ou minúscula), um underline (_), ou um cifrão (**$**). Isso exclui números como primeiro caractere, prevenindo conflitos com literais numéricos.
- **Caracteres subsequentes:** Após o primeiro caractere, os identificadores podem conter letras, números (**0-9**), underscores, ou cifrões. Isso permite grande flexibilidade na nomeação de variáveis.
- **Case-sensitive:** O JavaScript diferencia maiúsculas e minúsculas, então **variavel** e **Variavel** são considerados identificadores distintos. Isso exige consistência no uso de maiúsculas e minúsculas ao referenciar variáveis.
- **Uso de Unicode:** Identificadores podem incluir caracteres Unicode, permitindo a inclusão de caracteres acentuados ou de outras línguas, como **ü** ou **å**.

Veja no quadro abaixo alguns exemplos de identificadores válidos em JavaScript:

- **Numeros_visitas:** Começa com uma letra e usa underscore para separar palavras.
- **temp99:** Começa com uma letra e inclui números.

- **_nome:** Começa com um underline.

Também é recomendável respeitar e seguir um conjunto de restrições e boas práticas para a nomenclatura destes identificadores de variáveis:

- **Palavras-chave reservadas:** Identificadores não podem ser palavras-chave reservadas em JavaScript, como if, function, ou return.
- **Nomes descritivos:** É recomendável usar nomes descritivos que indiquem claramente o propósito da variável. Evite nomes como x ou temp a menos que seja para uso temporário ou em contexto claro.
- **Consistência na nomeação:** Use um estilo consistente para nomear variáveis. A convenção **camelCase** é comum em JavaScript, onde a primeira letra é minúscula e as seguintes palavras começam com maiúscula (como **numeroDeVisitas**).

Cabe ainda saber que você pode declarar uma variável de três formas:

1. **Com a palavra-chave 'var':** Esta sintaxe pode ser usada para declarar tanto variáveis locais como variáveis globais. Por exemplo, **var x = 42;**.
2. **Por simples adição de valor:** Isso declara uma variável global. Porém, seu uso não é recomendado. Por exemplo, **x = 42;**.
3. **Com a palavra-chave 'let':** Essa sintaxe pode ser usada para declarar uma variável local de escopo de bloco. Por exemplo, **let y = 13;**.

Veja no quadro abaixo um exemplo de declaração de variáveis:

```
simpleValue = "Esta também é uma variável global";

var globalVar = "Esta também é uma variável global";

let blockScopedVar = "variável local de escopo de bloco";
```

Palavra-chave var

A palavra-chave **var** é a forma mais antiga de declarar uma variável em JavaScript. Ela tem um escopo de função ou escopo global, dependendo do contexto em que é usada. O escopo global significa que a variável é acessível em qualquer parte do código após a declaração, enquanto o escopo de função limita a variável ao seu respectivo contexto. Uma variável declarada

com **var** pode ser redeclarada e seu valor pode ser modificado ao longo do tempo.

Veja no quadro abaixo um exemplo de uso da palavra-chave **var**:

```
var x = 10; // Declaração global
function exemplo() {
     var y = 20; // Declaração com escopo de função
}
```

Palavra-chave let

A palavra-chave **let** foi introduzida no ECMAScript 6 (ES6) para fornecer um escopo mais restrito. Uma variável declarada com **let** tem escopo de bloco, o que significa que ela é válida apenas dentro do bloco em que foi declarada, como dentro de um loop, função ou bloco condicional. Variáveis let não podem ser redeclaradas no mesmo escopo, mas podem ter seu valor modificado.

Veja no quadro abaixo um exemplo de uso da palavra-chave **let**:

```
let x = 10; // Escopo global ou do bloco em que é declarada
if (x > 5) {
    let y = 20; // Escopo restrito ao bloco 'if'
}
```

OBS: Uma variável declarada usando a declaração **var** ou **let** sem especificar o valor inicial tem o valor **undefined**. Já, uma tentativa de acessar uma variável não declarada resultará no lançamento de uma exceção **ReferenceError**:

Veja no quadro abaixo um exemplo desta situação:

```
var a;
console.log("O valor de a é " + a); // saída "O valor de a é undefined"
console.log("O valor de b é " + b); // executa uma exception de erro de referência (ReferenceError)
```

Declaração de constantes

A palavra-chave **const** também foi introduzida no ES6 para declarar constantes. Uma constante, como o próprio nome sugere, é um valor que não pode ser alterado após a sua inicialização. Isso significa que uma vez definida, a constante permanece a mesma durante a execução do código.

A sintaxe para declarar uma constante é semelhante à de uma variável, com algumas diferenças importantes. Um identificador para uma constante deve começar com uma letra, um sublinhado (_), ou um cifrão (**$**). Os caracteres subsequentes podem incluir letras (maiúsculas e minúsculas), números e sublinhados, tal como no exemplo do quadro abaixo:

```
const minhaConstante = 42;

const _constante = "Texto";

const $constante = true;
```

Semelhante ao **let**, o **const** tem escopo de bloco, mas não permite a redefinição ou redeclaração do valor da variável. Tentar fazer isso causará um erro, porque o **const** garante que a referência não será alterada. No entanto, se o **const** se refere a um objeto ou array, os valores internos podem ser alterados, desde que a referência ao objeto ou array permaneça a mesma.

Veja um exemplo no quadro abaixo:

```
const x = 10; // Constante de escopo de bloco
x = 20; // Erro: não é possível alterar uma constante

const arr = [1, 2, 3];
arr.push(4); // Permitido: referência não muda, mas o conteúdo do array sim
```

◀ 1.4 Teste seus conhecimentos

Seguir para...

2.2 Escopo de variáveis ►