JavaScript: Fundamentos para Desenvolvimento Web Interativo - Turma 2025B

2.4 Tipos de dados

O mais recente padrão ECMAScript, que define o JavaScript, especifica sete tipos de dados básicos que formam a fundação da linguagem. Destes, seis são considerados tipos de dados primitivos, enquanto o sétimo, Object, abrange estruturas mais complexas. Vejamos mais informações sobre os tipos de dados no decorrer deste texto.

Tipos primitivos

Os tipos primitivos representam os dados mais simples no JavaScript. Eles são imutáveis, o que significa que seu valor não pode ser alterado após a inicialização. Vejamos abaixo cada um destes tipos primitivos:

- **Boolean:** Representa valores booleanos, que podem ser **true** ou **false**. Esses valores são usados para controle de fluxo e lógica condicional. Veja um exemplo no quadro abaixo:

```
let isActive = true;
```

- **Number:** Representa números, incluindo inteiros e decimais (ponto flutuante). O JavaScript suporta valores numéricos comuns, como 42 e 3.14159, além de valores especiais como **Infinity** e **NaN** (Not a Number). Veja um exemplo no quadro abaixo:

```
let pi = 3.14159;
```

- **String:** Cadeias de caracteres, que podem ser delimitadas por aspas simples ('), aspas duplas ("), ou crase para template literals (`). As strings podem conter texto, caracteres especiais e sequências de escape. Veja um exemplo no quadro abaixo:

```
let greeting = "Olá, mundo!";
```

- **null:** Uma palavra-chave que indica um valor nulo ou ausente. No JavaScript, **null** é usado para indicar explicitamente que uma variável não tem valor. É importante lembrar que **null** é case-sensitive, ou seja, não é o mesmo que **Null** ou **NULL**. Veja um exemplo no quadro abaixo:

```
let value = null;
```

- **undefined:** Indica que uma variável não foi inicializada ou que uma propriedade de objeto não existe. O **undefined** é um valor padrão quando uma variável é declarada, mas não atribuída. Veja um exemplo no quadro abaixo:

```
let name;
console.log(name); // Saída: undefined
```

- **Symbol:** Um tipo de dado novo no ECMAScript 6, cuja instância é única e imutável. Os símbolos são usados para criar identificadores únicos para propriedades de objetos, evitando conflitos com outras propriedades. Veja um exemplo no quadro abaixo:

```
const symbol = Symbol("unique identifier");
```

Tipos complexos

O **Object** é um tipo de dado mais complexo, que pode conter propriedades e métodos. Objetos são recipientes para armazenar valores, podendo ser representados como pares chave-valor ou como estruturas mais complexas. Veja um exemplo no quadro abaixo:

```
let person = {
     name: "Alice",
     age: 35,
};
```

Embora tecnicamente parte do tipo **Object**, as funções (**function**) desempenham um papel fundamental no JavaScript. Elas representam blocos de código que podem ser chamados para executar uma ação ou retornar um valor. Veja um exemplo no quadro abaixo:

```
function add(a, b) {
    return a + b;
}
```

Por sua vez, um **array** em JavaScript é uma estrutura de dados que permite armazenar uma coleção ordenada de valores. Para criar um **array** em JavaScript, você utiliza colchetes **[ ]** e separa os elementos por vírgulas. Esses elementos podem ser de qualquer tipo de dado, como números, strings, booleanos, objetos e até mesmo outros arrays. Veja um exemplo no quadro abaixo:

```
let arr = [1, 2, 3, 4, 5]; // array de inteiros

let names = ["John", "Mary", "Jane"]; // array de strings

let mixed = [1, "two", true, null]; // array misto
```

◄ 2.3 Elevação (hoisting) de variáveis

Seguir para...

2.5 Conversão de tipos de dados ►