Expressões em JavaScript

Prof. Romerito Campos

Expressões em JavaScript

Conteúdo

- Expressões (primária, object e array, function, acesso a propriedades)
- Declarações (Condicionais, loops)

Expressões em JavaScript

- Uma expressão é uma frase em JavaScript que pode ser avaliada e produz um valor.

```
a + 10
10
a = [10, 20]
b = {x: 10, y: 20}
```

Expressões Primárias

- Definir literais: `"Hello"`, `1`, `1.5`.
- `this`, `all`, `true`:
  - `this` é avaliado de acordo com o contexto de sua aplicação.

Inicialização de Arrays e Objetos

- Para iniciar uma *array*, basta `[2, 3, 4, 5]`, onde os elementos do *array* são separados por vírgula.
- Por outro lado, utilizamos `{}` para objetos.
  - Exemplo de inicialização de objeto: `let o = {a: 10, b: 20}`

Expressão de Definição de Funções

- A expressão de definição de função corresponde ao uso da palavra reservada `function` seguida de um identificador (nome de função) e seus argumentos em parênteses. Além disso, há o corpo da função.

```
function raiz(a) {
    return Math.sqrt(a);
}
```

Invocar Expressões

- A invocação de expressão em JavaScript consiste em executar uma função `sqrt()`.
- Invocação de expressões são aplicadas a funções

Expressão de Criação de Objeto

- O operador `new` junto com o nome de uma classe é usado com expressão de criação de objeto.
- Isso é diferente de expressão de inicialização `let a={}`.

Visão Geral de Operadores

- Tipos de operações: aritméticos, comparação, lógico e atribuição.
- Operadores podem ser categorizados pelo número de operandos: **unary**, **binary**, **ternary**.
- Alguns operadores suportam mais de um tipo. Exemplo: `"3" * "5" → 15` ou `"a" + "b" → "ab"`.

Operadores relacionais

- Testa relacionamentos entre dois valores e retorna true ou false.
- Testa coisas como igual, maior que, menor que, iguadade e etc.

Operadores de Igualdade

- Os operadores `==` e `===` testam se dois valores são iguais. O segundo é estrito. Logo, tem que ser mesmo tipo e valor.
- Qual sera o resultado da operação abaixo? `"1" == true`

Operador de comparação

- `a,>, <=, e> =`.
- Pode compor qualquer coisa, mas é projetado para números e strings. Converte os outros tipos.
- A comparação de string leva em conta Case Sensitive.

Operador IN

- É usado para vigiar se uma string é na propriedade de um objeto.

```
let ponto = { s: "s", y: 2 }
"K" in ponto // true

let dado = [1, 2, 3]
"0" in dado // false
```

Operador Instance Of

- Verifica se um objeto é uma instância de uma classe.

```
class Date {
  constructor(a) {
    this.a = a;
  }
}

let d = new Date("a");
let b = { a: "a" };
b instanceof Date // false
```

Expressões lógicas

- O operador lógico AND é representado por `&&`.
- Dados dois operandos `a && b`, se `a` for falso, então `b` não é avaliado.
- O operador lógico OR é representado por `||`.
- O operador lógico NOT é representado por `!`.

Operações de Atribuições

- `let a = 50;`
- `let a, b, c = -10;`
- `let soma = a + b;`
- `let soma = 10;`

Referências