🔢 Converter variáveis para número em JavaScript

1️⃣ `Number()`

👉 **Converte o valor inteiro** — a string precisa ser totalmente válida

```
Number("10");         // 10
Number("10.33");      // 10.33
Number(" 10 ");       // 10
Number(true);         // 1
Number(false);        // 0
Number("10,33");      // NaN
Number("10 33");      // NaN
Number("John");       // NaN
```

📌 Importante

- Se **qualquer caractere inválido existir**, retorna `NaN`
- É o método **mais rigoroso**

☑️ Use quando:

- Você **espera um número válido**
- Quer evitar conversões "parciais"

2️⃣ `parseInt()`

👉 Extrai um número inteiro do início da string

```
parseInt("10");         // 10
parseInt("10.33");      // 10
parseInt("10 anos");    // 10
parseInt("10 20");      // 10
parseInt("anos 10");    // NaN
```

📌 Regras

- Ignora espaços iniciais
- Para quando encontra algo que não é número
- **Não arredonda**, apenas corta

⚠️ Boa prática:

```
parseInt("10", 10); // define base decimal
```

☑️ Use quando:

- Você quer **apenas números inteiros**
- Está lendo dados "sujos" (inputs, textos)

3️⃣ `parseFloat()`

👉 Extrai número com casas decimais

```
parseFloat("10");         // 10
parseFloat("10.33");      // 10.33
parseFloat("10 anos");    // 10
parseFloat("10.5kg");     // 10.5
parseFloat("anos 10");    // NaN
```

📌 Funciona como o `parseInt`, mas **mantém decimais**

☑️ Use quando:

- Você precisa de **números quebrados**
- Está lidando com entradas do usuário

🗓️ `Number()` com datas

```
Number(new Date("1970-01-01")); // 0
Number(new Date("1970-01-02")); // 86400000
```

📌 Retorna **milissegundos desde 01/01/1970** (Unix Epoch)

🧠 Métodos do objeto `Number`

Esses **NÃO são usados em variáveis**, apenas assim:

❌ Errado:

```
x.isInteger(); // erro
```

☑️ Certo:

```
Number.isInteger(x);
```

🔍 `Number.isInteger()`

```
Number.isInteger(10);   // true
Number.isInteger(10.5); // false
```

♾️ `Number.isFinite()`

```
Number.isFinite(123);      // true
Number.isFinite(Infinity); // false
Number.isFinite(NaN);      // false
```

❓ `Number.isNaN()`

```
Number.isNaN(NaN);    // true
Number.isNaN(123);    // false
```

📌 **Forma correta** de testar `NaN`

```
NaN === NaN // false ❌
```

🔒 **`Number.isSafeInteger()`**

```
Number.isSafeInteger(10);                    // true
Number.isSafeInteger(9007199254740991);    // true
Number.isSafeInteger(9007199254740992);    // false
```

📌 Limite seguro:

```
-(2^53 - 1) até +(2^53 - 1)
```

📜 **`Number.parseInt()` e `Number.parseFloat()`**

São **idênticos** aos globais:

```
parseInt("10") === Number.parseInt("10"); // true
```

☑️ Criados para:

- Organização
- Uso fora do browser (Node.js, módulos)

🆚 Resumo rápido

Situação	Melhor escolha
Conversão rigorosa	`Number()`
Inteiro de texto	`parseInt()`
Decimal de texto	`parseFloat()`
Verificar inteiro	`Number.isInteger()`
Verificar NaN	`Number.isNaN()`
Verificar número válido	`Number.isFinite()`