

//DIFERENÇAS ENTRE UNDEFINED E NULL:

//UNDEFINED: É quando temos um espaço de memória que não possuímos um valor definido, o espaço NÃO ESTÁ VAZIO, ele existe, porém não tem valor nenhum. No Javascript existe o elemento undefined, um elemento que não existe em várias linguagens de programação, porém não é muito bem visto o desenvolvedor usar o undefined como uma atribuição literal de uma variável. Por padrão, quando iniciamos uma variável ou let e não atribuímos valor nenhum a ela, o Javascript já a interpreta como undefined, é sempre melhor deixar que o próprio Javascript atribua undefined, invés de atribuirmos undefined por vontade própria.

//NULL: É quando temos uma variável que não aponta para lugar nenhum, ele simplesmente NÃO TEM ESPAÇO DE MEMÓRIA. Geralmente é bom usarmos null quando desejamos retirar o valor de uma variável.

// O null é um tipo especial, pois pode ser traduzido como “ausência de valor” e pode ser atribuído como valor de uma variável:

```
// let input = null;
```

```
// if (input === null) {  
//   console.log('não há informação');  
// } else {  
//   console.log(input);  
// }  
// Nesse caso, qual seria a diferença entre os dois casos abaixo?
```

```
// let input = null;  
// let input2;
```

```
// console.log(input); // null  
// console.log(input2); // undefined
```

// É aqui que entra o tipo undefined. Este tipo também representa “ausência de valor”, porém de uma outra forma: usualmente, enquanto null é um valor atribuído a uma variável que existe e foi iniciada, undefined se refere ao valor de uma variável que não foi inicializada (ou seja, não foi atribuído nenhum valor a ela).

// undefined também é o valor retornado por uma função que não tem cláusula return. Veremos mais sobre funções e return mais adiante no curso.

// É importante notar que, embora os dois tipos sejam utilizados para sinalizar ausência de valor, os operadores de comparação do JavaScript podem ou não diferenciá-los:

```
// console.log(null == undefined); // true
// console.log(null === undefined); // false
```

// No cotidiano é comum considerar undefined como uma ausência de valor “inesperada” (causada por um bug ou erro no código) e null como um tipo de dado que também significa ausência de valor, mas não de maneira inesperada. Por exemplo, um campo em uma tabela de um banco de dados que esteja sem dados ou uma informação solicitada que não seja obrigatória e não tenha sido preenchida pelo usuário pode ter valor null.

//JAVASCRIPT POR PADRÃO ATRIBUÍ UNDEFINED A UMA VARIÁVEL OU LET QUE NÃO TENHA VALOR ATRIBUÍDO:

```
let valor;
console.log(`1) ${valor}\n`); //Perceba que o valor será indefinido, ou seja, existe um espaço de memória, mas ele está vazio;
```

//PODEMOS ATRIBUIR NULL OU UNDEFINED LITERALMENTE:

```
valor = null; //esse é o jeito melhor de zerar uma variável, usando null...
console.log(`2) ${valor}`);
valor = undefined; //essa é uma forma inadequada de zerar uma variável...
console.log(`2) ${valor}\n`);
```

//QUANDO A CHAVE DE UM OBJECT É CHAMADA MAS NÃO EXISTE, O JAVASCRIPT APRESENTA UNDEFINED:

```
const produto = {};
console.log(produto)
console.log(produto.preco); //Perceba que chamamos a chave, mas ela não existe, o resultado será um undefined para essa chave...
console.log("\n");
```

produto.preco = undefined; //Declaramos a chave, e atribuímos undefined sobre o valor dela, ESSA NÃO É A MELHOR UTILIZAÇÃO DE UNDEFINED...

```
console.log(produto); //Veja o resultado da chave indefinida...
console.log(produto.preco); //Veja o valor indefinido...
console.log("\n");
```

```
produto.preco = null; //A melhor forma de zerar um valor é usando nulll...
console.log(produto) //Veja como fica o valor da chave...
console.log(produto.preco); //Veja o valor nulo...

//TESTANDO IGUALDADE EM NULL E UNDEFINED:
//Podemos ver também como a diferença conceitual entre null e undefined pode ser mostrada quando usamos operadores de igualdade comum e igualdade estrita...
console.log(`\n3) ${undefined} == ${null}? Resultado: ${undefined == null}`); //Perceba que, quando fazemos uma comparação de valor entre "null" e "undefined" elas aparentam ser iguais...
console.log(`3) ${undefined} === ${null}? Resultado: ${undefined === null}`); //Mas quando a comparação é feita de forma estrita, podemos ver que null e undefined não são a mesma coisa...
```

```
[Running] node "c:\Users\Dell\Documents\javascript\arquivos_das_aulas\035-diferencas_entre_Null_e_Undefined.js"
```

```
1) undefined
```

```
2) null
```

```
2) undefined
```

```
{}
```

```
undefined
```

```
{ preco: undefined }
```

```
undefined
```

```
{ preco: null }
```

```
null
```

```
3) undefined == null? Resultado: true
```

```
3) undefined === null? Resultado: false
```

```
[Done] exited with code=0 in 1.338 seconds
```