

//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 usar null quando desejamos retirar o valor de uma variável.

//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 null...  
console.log(produto) //Veja como fica o valor da chave...
```

```
console.log(produto.preco); //Veja o valor nulo...
```

RESULTADO...

The screenshot shows the Visual Studio Code interface with the file explorer on the left, the editor in the center, and the terminal at the bottom. The file explorer shows a folder named 'ARQUIVOS_DAS_AULAS' containing several JavaScript files. The editor displays the file '035-diferencas_entre_Null_e_Undefined.js' with the following content:

```
1 //DIFERENÇAS ENTRE UNDEFINED E NULL:
2
3 //UNDEFINED: É quando temos um espaço de memória que não possui 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 atribuímos
```

The terminal at the bottom shows the output of running the file:

```
C:\Users\MasterGabriel\Documents\JAVASCRIPT\ARQUIVOS_DAS_AULAS>node "c:\Users\MasterGabriel\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

C:\Users\MasterGabriel\Documents\JAVASCRIPT\ARQUIVOS_DAS_AULAS>
```

The status bar at the bottom indicates the current line and column (Ln 10, Col 1), the number of spaces (Espaços: 4), the encoding (UTF-8), the line ending (CRLF), the language (JavaScript), and the time (07:10 31/08/2021).