**Convenções – Tipos null e undefined**

Existem várias convenções para nomes e cada linguagem de programação tem o seu. Seguem alguns deles:

* camelCase: Inicia com letra minúscula e a primeira letra de cada palavra em seguida é escrita com letra maiúscula. Por exemplo: minhaVar ou senhaDoUsuario. **Esta é a convenção utilizada pelo JavaScript para variáveis e funções.**
* snake\_case: Os espaços são substituídos pelo caractere \_ (underline), com todas as palavras em letra minúscula. Por exemplo: minha\_variavel ou senha\_do\_usuario.
* kebab-case: Similar ao anterior, porém com os espaços substituídos por hífens. Por exemplo: minha-var ou senha-do-usuario. **Esta convenção não pode ser utilizada no JavaScript para variáveis e funções.**
* PascalCase: Similar ao CamelCase, porém neste caso todas as palavras começam com letra maiúscula. Por exemplo: MinhaVar ou SenhaDoCliente.

Importante: Nunca utilize espaço nem caracteres especiais, nem inicie os nomes das variáveis com números.

Nesta aula falamos sobre três tipos primitivos: number, string e boolean. Mas existem ainda mais dois tipos que não abordamos com profundidade: null e undefined.

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);

}COPIAR CÓDIGO

Nesse caso, qual seria a diferença entre os dois casos abaixo?

let input = null;

let input2;

console.log(input); // null

console.log(input2); // undefinedCOPIAR CÓDIGO

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); // falseCOPIAR CÓDIGO

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.