Constantes é algo muito similar a variáveis, na verdade isso é chamado na linguagem Javascript de constante que é uma variável que não muda o seu valor. Essas regras abaixo também se aplicam para as constantes:

```
/* Sobre variáveis

// Palavras reservadas
// Nomes Claros
// Nome inicializado com Número
// Não podem ter espaços ou traços
// Utilizamos camelCase -> let nomeCompletoDoCliente
// Case-sensitive (Letra Minúscula e Letra Maiúscula) fazem diferença
// Não podemos redeclarar variáveis com let
// NÃO UTILIZE VAR, UTILIZE LET. Sempre que você pensar em alguma coisa que vai variar, alterar, no seu código utilize let
*/
```

Nós não podemos declarar primeiro uma constante e depois tentar atribuir o seu valor, que no caso seria inicializar a constante. A constante já precisa ser criada e inicializada ao mesmo tempo.

Para a gente criar uma constante utilizamos a palavra **const** e depois o nome dela:

const country = ‘Canada’;
console.log(country);

O que nós não podemos fazer é declarar uma constante sem inicializar ela porque nós não podemos modificar o valor da constante, se a gente fizer:

const nome;

Nós teremos o erro “Missing initializer in const declaration”

Está faltando inicializador na declaração da constante, nós temos que inicializar, nós precisamos sempre colocar um valor já na criação da constante.

Uma outra coisa que nós não podemos fazer com constante é modificar o valor dela, por exemplo:

```
const country = ‘Canada’;
country = ‘United States’;
```

TypeError: Assigment to constant variable.

Isso é um comportamento extremamente importante, porque na maioria dos casos o seu sistema vai depender de variáveis, de valores que estão salvos na memória com let ou com const. No caso de let é muito provável que em algum lugar do seu código, se o seu código for muito grande é muito provável que em algum lugar você possa sem querer reatribuir o valor daquela variável e mudar o comportamento do seu Software inteiro, então para resolver esse problema nós optamos por usar const e quando for necessário alterar o valor da variável a gente muda ela pra let.

```
let country = ‘Canada’;
country = ‘United States’;
```

Você pode pegar um valor de uma variável ou constante e atribuir a outra variável ou outra constante:

```
const primeiroNumero = 5;
const segundoNumero = 10;
const conta = primeiroNumero * segundoNumero;
```

Você pode usar esses dois valores (5 e 10) e fazer uma operação, no nosso caso multiplicação. E nós vamos ter a saída do resultado que é 50.

Então a gente sempre pode ir evoluindo o valor, a gente não precisa ficar alterando o valor de uma variável só sendo que a gente pode sempre recriando outras variáveis que vão pegar o valor que nós queremos das variáveis.

Então nós podemos fazer:

```
const primeiroNumero = 5;
const segundoNumero = 10;
const conta = primeiroNumero * segundoNumero;
const resultaDuplicado = resultado * 2;
console.log(resultaDuplicado);
```

Eu continuo evoluíndo meu valor, em nenhum eu estou tentando alterar algum dos valores, o resultado final será 100. E se eu tivesse alterando a minha variável, por exemplo tudo com let e viesse alterando, por exemplo eu poderia pegar o **resultado** e alterar por ele mesmo.

For example

let resultadoTriplicado = resultado * 3;

Se eu quisesse aumentar ainda mais esse *resultadoTriplicado* eu poderia utilizar o valor dessa variável mesmo e fazer uma outra operação aritmética.

resultadoTriplicado = resultadoTriplicado + 5;

Mas o resultado dela, vai ser ela mesmo + 5 (150 + 5) que vai dar 155. Só que quando eu faço isso precisamos ter muito cuidado porque nós perdemos o primeiro valor inicial da variável *resultadoTriplicado,* então nós não conseguimos resgatar esse valor. Ao contrário do que nós fizemos com const você sempre pode usar esse primeiro valor -> *primeiroNumero* quando você quiser, então se você quiser pegar o valor da variável *primeiroNumero* ele não modificado em nenhum momento no seu código você pode reutilizar ele sempre que você precisar, é muito interessante, ao invés de nós ficarmos modificando o valor, é claro que em alguns momentos nós precisamos alterar o valor da variável mesmo, mas é muito interessante sempre que possível, nós mantermos o valor inicial da sua variável e sempre ir evoluíndo o valor dela, ao invés de da gente ficar modificando o valor dela ao longo do código.

// String = Texto
// Number = Número

*Porque falamos disso?*

Isso é extremamente importante, porque o Javascript é uma linguagem de tipagem dinâmica, então a partir do momento que a gente atribuiu o valor a uma constante ou a uma variável esse valor ele já tem um tipo definido, o motor do Javascript salva esse tipo em um local. Você pode saber o tipo dele usando o *typeof* seguido do nome da variável ou constante.

For example

console.log(typeof primeiroNumero);

Se a gente fizer:

const primeiroNumero = ‘5’; // String

Não será mais um number, e sim, uma String;

Isso vai fazer uma diferença extrema quando você for utilizar o sinal de operação “+” porque ele é utilizado para concatenar strings e soma. Voltando com o valor 5 sem

aspas, vamos fazer uma soma entre a constante *primeiroNumero* + *segundoNumero*:

```
const primeiroNumero = 5; // Number
const segundoNumero = 10;
console.log(primeiroNumero + segundoNumero);
```

O valor final será 15, mas se o tipo da constante *primeiroNumero* fosse string, o comportamento é completamente diferente, agora ele vai pegar a constante *primeiroNumero* e vai concatenar com a constante *segundoNumero,* então isso vai se tornar 510, é a junção de '5' com 10, ele fez a concatenação e não a soma.

```
const primeiroNumero = '5'; // String
const segundoNumero = 10;
console.log(primeiroNumero + segundoNumero);
```

Eu poderia também utilizar o typeof:

```
console.log(type of primeiroNumero + segundoNumero);
```

Isso é uma expressão, então quando eu faço desta maneira a saída dele será *string10;* Porque ficou string + number, então não é isso que nós queríamos. Então vem os parênteses, sempre que você quer executar alguma coisa que está dentro e depois pegar o valor e verificar o tipo você utiliza os () desta maneira:

```
console.log(type of (primeiroNumero + segundoNumero));
```

Eu envolvo a minha expressão que é a soma entre () e aí eu checo a saída com typeof, o resultado final será string porque a constante *primeiroNumero* é uma string e aí o Javascript passa a tratar a constante *segundoNumero* como string também, se você quer que isso se torne um número as duas constantes precisam ser numéricas.

```
const primeiroNumero = 5; // Number
const segundoNumero = 10;
console.log(primeiroNumero + segundoNumero);
```

Então const não podem ser alteradas ao longo do código, você pode o valor de uma constante, jogar em uma nova constante ou valor de uma variável, jogar em uma nova constante, porém você não pode alterar o valor dessa constante ao longo do seu código.