Esse curso visa aprender sobre as principais estruturas de javascript, veremos primeiro atribuição de valores, tipos e uma atividade prática no final.

Na nomeação de variaveis, existem alguns bons costumes que devem ser mantidos para melhorar a compreensão do texto, um exemplo desses bons costumes, é **nomear variaveis com mais de um nome com cada nome em letra maiúscula**, na imagem vemos alguns outros casos desse tipo.

Essas regras devem ser seguidas para garantir uma sintaxe e leitura do código.

**Var** declara o nome da variavel, ela é **global**, a **let** ela é uma variavel que **não é global**, ou seja, não é possivel acessar uma variavel let de fora do bloco em que ela esta localizada. **O uso de let evita erros ao usar variaveis de mesmo nome** dentro de diferentes blocos.

**Hoisting** é atribuir um valor a uma variavel antes de declarar ela. Isso se refere ao fato de que em JavaScript não é necessário declarar uma variavel **var** antes de utiliza-la (no início do código), na verdade nem é necessário sempre declarar a variavel, pois ao utilizar um nome não especificado, o próprio JS vai assumir que aquilo é uma variável.

**Redeclaração** é atribuir um valor a uma variavel já existente com o var na frente (não é possível redeclarar uma let), sendo a **reatribuição** atribuir um valor a uma variavel já existente sem o var na frente. Um jeito facil de notar quando ocorre uma redeclaração de uma variavel ou um comando é observar que uma variavel redeclarada (em outros palavras que se torna “inútil”) é que ela se torna levemente acizentada.

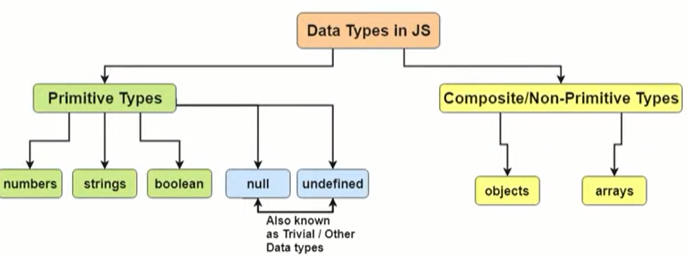
**camelCase** é o nome dado ao bom costume de nomear os arquivos que vimos antes.

**É considerado seguro utilizar let em vez de var, pois o var pode ficar exposto e tornar o código vulnerável.**

--

A **const** é preferencialmente declarada em SNAKE\_UPPER\_CASE e **não faz hoisting**. Const, referente a constante, são variaveis que **não variam** não tem jeito facil de dizer isso.... **é necessário declarar a const com o seu valor antes de utiliza-lo**, tentar declarar a const sem dar um valor e depois dar um valor conta como uma tentativa de reatribuição, o que não é permitido.

==

O javascript é uma linguagem de **tipagem dinâmica**, onde não é necessário **especificar o tipo dele antes de declarar o tipo dele**.

Os tipos de dados em JS são os do quadro, tipos **primitivos** não possuem métodos dentro deles, tipos **compostos/não primitivos** possuem métodos, e usamos geralmente o objeto.

O que separa dinamicamente objetos de primitivos, é que **objetos são escritos com a primeira letra maiúscula**, uma string é diferente de uma String, pois string é primitivo e String um objeto.

**Strings** são textos que podem ser escritos de três maneiras:

“isso é um exemplo usando aspas”

‘isso é um exemplo usando o tracinho que não sei o nome’

Dito isso é possivel escrever “dentro” dessas `e esse é o ${exemplo} que usa a crase, esse

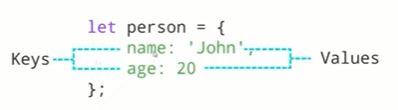
strings usando o ‘...’ + exemplo + “...”. exemplo permite colocar comandos e

isso vai mostrar “...exemplo...” Cortes de linhas dentro da mesma sentença;`

é de bom senso usar aspas ou ‘’ ao fazer strings que não usem valores de variaveis, mas é sempre ideal usar as crases quando necessário usar os valores, pois torna a frase mais limpa.

Da para usar o javascript dentro do console dos browsers, isso facilita um pouco o estudo e permite fazer algumas coisas interessantes, mas nada que nos importa tanto, no momento, é apenas uma forma interessante de estudar.

**Arrays** são listas interaveis de elementos, com seu index começando em 0. Possue diversos comandos, array.indexof(x) mostra o index, array.add(x,y,z) adiciona elementos à array na ordem especifica, array.pop() retira o ultimo elemento, array.shift() tira o primeiro elemento, array.unshift(x) adiciona um numero **ao início** da lista, etc..

**objetos** são estruturas do tipo chave e valor (como visto ao lado), objetos permitem o trabalho utilizando a POO/OOP, ao servir como uma biblioteca de informações especificas sobre um objeto.

**Empty/undefined/null** são valores que retornam algo **falso** ou “**nada**”.   
o **empty** significa que **não há nada** (zero), sem valores dentro. let eggs = “”;

O **null** é retornado quando o valor **não é definido**, não foi dado nada que o defina. let eggs;

**Undefined** é quando não há nada, inexistente. como pedir uma variavel não declarada.

**Falsy** **values** são valores que são tratados como falso mesmo sem ser falso, como o null, undefined e empty, eles vão ser sempre falsos, porem ao perguntar se eles são falsos, eles também seram falsos.

Agora teremos duas pequenas aulas práticas, usaremos o .js.