🡪 Json (JavaScript Object Notation) = formato de texto que pode ser compartilhado facilmente e ser interpretado por várias linguagens de programação

-Parece com um objeto literal dentro de um array:

‘{“nome”: ‘Brasil’, “capital”: ‘Brasília}’

-Com essa estrutura é possível converter facilmente um objeto literal e um array

-Json.parse() recebe como um parâmetro um dado e retorna um array ou um objeto literal

-Json.stringify() faz o oposto, recebe um ob. Literal ou um array e retorna uma string do tipo Json

->Json é muito similar a um ob. Literal, as diferenças são:

-Objeto literal:

\*Admite aspas simples e duplas (json só aspas duplas);

\*As chaves do objeto ficam sem aspas (no json as chaves vão entre aspas)

\* Podemos escrever em métodos (json não suporta métodos, apenas propriedades e valores)

\*Recomenda-se colocar uma vírgula na última propriedade (em json isso não pode acontecer)

\* Número também se tornam string (entre aspas)

-Escrevendo um objeto literal:

{

texto: “meu texto”,

numero: 16,

array: [‘um’,’dois’],

booleno: true,

método(): {return ‘Olá’},

}

-Escrevendo em json:

{

“texto”: “Meu texto”,

“numero”:”16”,

“array”: [“um”,”dois”],

“booleano”: true

}

* Lembrar que uma STRING nada mais é do que um ARRAY de caracteres;
* .length: retorna o tamanho de uma cadeia de texto (conta os caracteres e os espaços em branco);
* .indexOf(): pode ser executado a partir de uma string e ele recebe um parâmetro de texto que será pesquisado dentro dessa outra string
  + Ex: “Olá mundo”.indexOf(“mundo”) vai retornar 4, pois a letra “o” começa no zero. Se o que eu procuro não está na minha string ele vai retornar -1. Ele sempre vai retornar o primeiro resultado encontrado.
* .slice(): “fatia”. umaString(onde inicia, onde termina(não é obrigatório))
  + Ex: “Eu gosto de JS”.slice(3,8) // vai retornar “gosto”
* .trim(): remove espaços em branco no começo ou no final de uma string, não tira os espaços entre as palavras
  + Ex: “Sou um ser humano” vai remover os espaços em branco. Ele não recebe nenhum parâmetro.
* .split(): converte uma string em um array
  + Ex: “Sou do planeta terra”.split(“”) -> [“sou”, “do”, “planeta”, “terra”]
* .replace(): substitui uma parte do texto dentro de uma string
  + Ex: “Sou programador php”.replace(“php”,”Javascript”)
* typeOf retorna o tipo de dado
* .parseInt(): recebe uma parâmetro que será o dado que queremos converter e nos retornará uma conversão desse dado. Ele só vai converter os números, até o primeiro caracter não numérico. NÃO retorna decimais, para isso utilizar parseFloat()
  + Ex: 12E3ML -> 12 MLV123 -> NaN
* Number(): mesmo função do parseInt(), porém o Number só pode converter aquilo que é realmente um número, se tiver letra no meio ele vai retornar um NaN
  + let resultado = Number(false) -> 0
  + let resultado = Number(true) -> 1
* String(): converte um numero em uma string
  + let texto = numero.toString()
  + Trabalha com arrays ([“joão”, “maria”]), mas não trabalha com objetos({nome: “Maria”,blablalb})
* Arrow functions nos permitem escrever funções com uma sintaxe mais compacta
  + Não precisa mais utilizar a palavra reservada function
  + Elas são sempre anônimas
  + Utiliza o operador “flecha” (=>) seguida das operações que a função irá realizar
* Callback: é uma função que se passa como parâmetro de outra função
* Closures: são funções que residem em outras funções (procedimentos que existem dentro de outros procedimentos
  + Serão usados somente quando sua função contêiner for executada e após isso deixará de existir tornando a memória disponível para outros procedimentos
* Assincronismo é a falta de concordância ou simultaneidade de fatos ou fenômenos no tempo
  + É uma parte essencial da linguagem, pois permitirá que você faça várias chamadas de funções, sem precisar esperar que cada uma se concretize para continuar executando o script
  + É aquele código em que cada instrução NÃO espera que a anterior seja executada
* Linguagem single threaded: uma coisa de cada vez
* setTimeout(function(){ console.log(vouDepois())},2000) : 2 parâmetros, o 1º é uma função, o 2º é o número de milissegundos que se deve passar até a execução da função que passamos no parâmetro (se colocarmos o valor zero, ele praticamente realiza as duas funções ao mesmo tempo)
* Event Loop: é responsável por monitorar quais funções são executadas e em que momento. Tem a capacidade de saber quando a nossa Pilha de Tarefas está vazia, e só então começara a executar as funções que estão presentes na Fila de tarefas.