Mestre dos Códigos perguntas

1º - VAR: possui escopo mais abrangente, e não abrange o tipo de escopo de bloco.

LET: também pode ser declarada de forma global como VAR, mas diferente dela, Let possui o tipo de escopo bloco, que é uma variável visível e utilizada apenas dentro de um único bloco.

CONST: constante variável com apenas um valor e que não pode ser alterado.

2º -  Funções normais não guardam escopo

 Funções normais guardam escopo

 Arrow function são mais rápidas

 Arrow function podem ser instanciadas

 Arrow function não guardam escopo

3º -   John

 Luke

 James

 Walter

4º -  The event 04/02/2019 will take place on Event Test

 The event undefined will take place on undefined

 Uncaught TypeError

 The event null will take place on null

5º - "document.querySelector" = seleciona qualquer elemento do DOM por id ou class

"document.querySelectorAll" = seleciona todos elementos que batem com a descrição no selector.

"document.getElementById, className, Name, TagName" = seleciona elemento por valor do Id, nome da classe, nome ou tag.

6º - Utilizar EventListener ou attachEvent

7º - Utilizar removeEventListener, detachEvent ou podemos previnir eventos default com event.preventDefault();

8º - Bubbling é a "dissipação" de um evento enviado de um componente filho que se propaga aos elementos em que ele está envolvido, ou elementos "Pais".

9º - "const object = {}" = Atribui o valor de objeto vazio;

"const object = new Object()" = Nova instancia de Object

"const object = Object.create()" = Dessa forma resultara em erro, pois Object.create() não pode ser iniciado vazio.

10º - Fetch tem como response uma promise que pode ser passada ao formato JSON, facilita a busca pois o unico parametro necessario é um string que define a url da api de onde será feita a requisição.

11º - Promisse refere-se ao Objeto que será de alguma forma a eventual conclusão ou falha em alguma operação.

new Promise((resolve, reject) => {

console.log('Initial');

resolve();

})

.then(() => {

throw new Error('Something failed');

console.log('Do this');

})

.catch(() => {

console.log('Do that');

})

.then(() => {

console.log('Do this whatever happened before');

});

12º - forEach() => percorre um array executando as operações que estiverem no bloco

filter() => retorna um array com os elementos que retornarem verdadeiro na condição

find() => retorna o primeiro elemento encontrado que passar na condição.

13º - map() => retorna um novo array dado a operação dentro do bloco, mais utilizado quando necessário criar novos conjuntos de dados com base no array existente.

reduce() => necessário quando há a necessidade de condensar todos os valores de um array.

filter() => necessário quando há a necessidade de filtrar um array e trazer apenas os elementos que passarem no condição.

14º - splice() => utilizado em conjunto com o indexOf, consegue remover um elemento independente de sua posição na lista.

15º - replace() => substitui valores de uma string.

substring() => retorna o conjunto de caracteres entre uma posição e outra da string.

trim() => retorna o texto sem os espaços em branco no inicio e fim da string.

16º -

localStorage.setItem('nome', 'valor');

localStorage.getItem('nome');

sessionStorage.setItem('nome', 'valor');

sessionStorage.getItem('nome');

17º - Criar Cookie

function setCookie(name, value, duration) {

        var cookie = name + "=" + escape(value) +

        ((duration) ? "; duration=" + duration.toGMTString() : "");

        document.cookie = cookie;

}

Ler Cookie

function getCookie(name) {

    var cookies = document.cookie;

    var prefix = name + "=";

    var begin = cookies.indexOf("; " + prefix);

    if (begin == -1) {

        begin = cookies.indexOf(prefix);

        if (begin != 0) {

            return null;

        }

    } else {

        begin += 2;

    }

    var end = cookies.indexOf(";", begin);

    if (end == -1) {

        end = cookies.length;

    }

    return unescape(cookies.substring(begin + prefix.length, end));

}

Deletar um Cookie

function deleteCookie(name) {

       if (getCookie(name)) {

              document.cookie = name + "=" + "; expires=Thu, 01-Jan-70 00:00:01 GMT";

       }

}

18º - For -> loop que envolve um bloco

For/in -> loop que corresponde a um atributo de um object

For/of -> loop referente a um array

while -> enquanto condição for verdadeira

do/while -> ate que a condição seja verdadeira.]

19º - Hoisting -> Carregamento de todas as funções e variáveis antes que o conteúdo seja carregado, dessa forma independente de onde esteja sua função ou variável ele conseguira encontrar, evitando que variáveis e funções precisem estar no inicio do código, isso é diferente para variáveis dentro do escopo de bloco, ou seja, dentro de uma função, pois o que será "içado" é apenas a função de não suas respectivas variáveis, mesmo que estejam com o mesmo nome de variáveis globais.

20º -  window

 global

 undefined

 null

21 º -

 No ínicio do meu código

 No inicio do block if

 No inicio de um loop

 no inicio de uma função