3 CURSO DE JAVASCRIPT

14 CAPITULO:

1 Prototipo: //estilo de programacion que se crea mediante prototipos de un objeto y no por instanciacion

prototype

1.1 prototype_chain: //cadena de prototipos es como un prototipo tiene otro prototipo

eje: let objeto = { //hereda el prototipo

"propiedad":"datos"

}

console.log(objeto);

1.2 prototype_object_t: //lo tienen todos los prototipos los heredan

1.3 caracteristicas:

1.3 1 es_definido_en_su_codigo_fuente_es_mutable: //cuando lo creamos nosotros lo podemos modificar

1.3 2 es_en_si_mismo_un_objeto_asi_como_otros: //son objetos

1.3 3 tiene_una_existencia_fisica_en_la_memoria: //insidencia real en el hardware

1.3 4 puede_ser_modificado_y_llamado: //se puede modificar y llamarlo

1.3 5 se_puede_ver_como_un_objeto_ejemplar_en_una_familia_objeto: //se puede ver asi

1.3 6 un_objeto_hereda_propiedades:(valores y metodos)_de_su_prototipo: //si yo creo un objeto

//este objeto hereda las propiedades del que yo cree

1.4 propiedad_proto:(dunder proto): //

1.5 diferencia_entre_sobreescribir_proto_y_sobreescribir_metodo: //

1.6 modo_estricto_:("use strict"): //es eliminar los errores que tiene javascript

//no perdona tira errores que java script deja pasar

//convierte errores de javascript en exepciones

//mejora la optimizaccion de errores y consigue mejores tiempose de ejecucion

//evita sintaxis usadas en posteriores a ES6

//no permite que el programador realice un "mala sintaxis"

1.6 1 usar_modo_estricto:

Eje: "use strinc";

let nombre = "lucas";

console.log(nombre);

//variables declaradas

//modificar propiedades (defineProperty() y writeable)

//agregar propiedades

//no se puede agregar propiedades a un string

//no existen las multiples variables en una funcion

```
                    //delete en objetos y variables = //se usa para eliminar propiedades
                    //las palabras reservadas no pueden  ser usadas como variables
```

2 CAPITULO:

1 Funciones_Flecha:

```
        //cuerpo sintaxis
        const saludar = ()=> {
                console.log("hola");
        }
        //si hay  solo una expresion la retorna
        const saludar = ()=> "string"; //solo una linea

        //-parentesis opcionales ante un solo paramentro(sin parametros se requiere
parentesis)
        const saludar = res => nombre = res;

        //no son adecuadas para usarse como metodos y no pueden usarse como
contructores
        //el this no puede ser usado osea hace referencia a window osea el this no existe
```

2 Funciones_Recursivas:

```
        //cuando una funcion se llama a si misma
        const validaEdad = ()=> {
                validaEdad();
        }
```

3 Operador_Ternario:

```
        //no es un condicional es un operador que funciona parecido al condicional if o else

        //? es como el if si se cumple la funcion
        //: despues de esto es como el else que es si no se cumple lo anterior
        let edad = 28;
        (edad > 18)? console.log("mayor de edad");
                        ? console.log("es igual a la edad establecidad");
                        : console.log("es menor de edad");
```

4 Operador_Spread:(/*spread operator*/)

```
        //añadir arrays a otros arrays
        //concatenar arrays
        //como argumento rest
        let arr = ["vel1", "vel2"]
        console.log(...arr)

        let arr = ["banano", "manzana"]
        let arr2 = ["pera", "kiwi"]
        arr.push(...arr2)
        console.log(arr)
```

```
let arr3 = [...arr, ...arr2]
console.log(arr3)
//nos concatena los valores pero no pierde sus propiedades
```

5 APIs:

```
//interfaz de programacion de aplicaciones
//le pasamos un dato y ella nos da un dato pero hace unas cosas internamente

1.1 objeto_Date:
    /*
    getDate() ==> dia del mes
    getDays() ==> dia de la semana
    getMonth() ==> devuelve un mes menos
    getYear() ==> devuelve el año actual menos 1900
    getHours() ==> devuelve las horas
    getMinutes() ==> devuelve los minutos
    getSeconds() ==> devuelve los segundos

    Date es un constructor y un objeto*/
    const fecha = new Date();
    console.log(fecha)

    //tarea investigar los otros metodos de Date
1.2 local_storage_y_session_storage:
    //diferencia cuando cargamos la pagina una pierde los datos y la otra los
guarda
    /*
    setItem() ==> definimos propiedades
    getItem() ==> obtenemos propiedades
    removeItem() ==> borramos una propiedad
    clear() ==> borra todas las propiedades
    */
1.3 drag_y_drop:
    //arrastrar y soltar
    /* funciona con eventos del objeto
    dragstart cuando oprimimos
    drag cuando arrastramos
    dragend cuando soltamos
    dragtode entre
    dragenter entre a la zona
    dragover si estamos en el
    drop verifica cuando soltamos
    dragleave cuando se va

    propiedad dataTransfer
```

getData()
setData()
*/

1.4 history:

//historial de la navegacion
/*
back() atras
forward() hacia adelante
tamaño del historial
go() va al sitio web indicado
pushState() modifica url y conserva info
replaceState() modifica la url y no la conserva
propiedad state y evento popstate
*/
console.log(history)

1.5 file_reader:

/*
readAsText()
readAsDataURL()

faltan otras investigar
*/

1.6 file_reader_con_drag_y_drop:
1.7 indexed_DB:
1.8 indexed_DB_con_drag_y_drop:
//investigar lo anterior

3 CAPITULO:
1 Match_Media: //se usa con resposive desing pero desde javascript

/*
matchMedia()
evento OnChange ==> parametro evento listener
*/

2 Intersection_Observer:

//definicion:
/*
Intersectionobserver()
callback y option
IsIntersecting
configurar opciones
*/

3 Notification:

/*
notification.requestPermission()
notification.permission

notification.(msg, options)
*/

4 Navigator:
/*
NavigatorID
NavigatorLanguage
NavigatorOnline
NavigatorContentUtils
NavigatorStorageUtils
NavigatorCookies
NavigatorConcurrentHardware
NavigatorPlugings
NavigatorUserMedia

hay mas pero en la pagina oficial
*/
5 Memoizacion:
//investigar esto
6 Cache:
//metodos en la estandarizacion oficial
9 Service_Workets:
//investigar esto

//investigar estos 3 siguientes
10 Cookies:
11 Objeto_Screen:
12 Objeto_Canvas: