2 CURSO DE JAVASCRIPT

7 CAPITULO:

1 WINDOW: ====>//objeto de javascript es el objeto mas grande de la herarquia de javascript

//hereda propiedades de eventtarget

1.1 open: ===>//abre una ventana nueva con la url de ella

```
window.open();
```

1.2 close: ===>//ciera un ventana abierta

```
window.close();
```

1.3 closed: ===>//nos dice si una ventana esta cerrada o no

```
window.closed();
```

1.4 stop: ===>//hace que se detenga una ventana

```
window.stop();
```

1.5 alert: ===>//muestra un cuadro de alerta

```
window.alert();
```

1.6 print: ===>//muestra cuadro para poder imprimir el documento actual

```
window.print();
```

1.7 prompt: ===>//muestra un cuadro donde solicita un dato al usuario

```
window.prompt();
```

1.8 confirm:===> //muestra un cuadro en el que nos da dos botones (aceptar y cancelar)

```
window.confirm();
```

1.9 screen: ===> //devuelve una referencia al objeto de pantalla asociado a la ventana

```
window.screen;
```

1.10 screenLeft: ===> //devuelve distancia horizontal entre el borde izquierdo del navegador

```
//y el borde izquierdo de la pantalla de la pc
//propiedad solamente de lectura
window.screenLeft;
```

1.11 screenTop: ===> //delvuelve distancia vertical entre el borde de arriba del navegador y el

```
//borde de arriba de la pantalla de la pc
//propiedad solamente de lectura
window.screenTop;
```

1.12 scrollX: ===> //devuelve el numero de px que el documento se desplaza horizontalme

```
window.scrollX;
```

1.13 scrollY: ===> //devuelve el numero de px que el documento se desplaza verticalmente

```
window.scrollY;
```

1.14 scroll_scrollTo: ===> //desplazan la ventana a un lugar particular del documento.

```
//con opciones y posiciones
```

//funcionan exactamente igual son casi lo mismo

window.scroll(0, 100);
window.scrollTo(0, 100);

1.15 resizeBy: ===> //cambia el tamaño de la ventana actual en una cantidad especifica
window.resizeBy(50, 100);
1.16 resizeTo: ===> //redimensiona dinamicamente la ventana
window.resizeTo(50, 100);
1.17 moveBy: ===> //mueve la ventana a una ubicacion relativa
window.moveBy(50, 100);
1.18 moveTo: ===> //mueve la ventana a una ubicacion absoluta
window.moveTo(50, 100);

1.19 objetos_barprop: ===> //nos dice si esta visible o no //para ver las distintas barras
1.19.#1 locationbar: window.locationbar.visible
1.19.#2 menubar :window.menubar.visible
1.19.#3 personalbar: window.personalbar.visible
1.19.#4 scrollbars: window.scrollbars.visible
1.19.#5 statusbar: window.statusbar.visible
1.19.#6 toolbar: window.toolbar.visible

2 Location:
2.1 window.location.href ==> //devuelve el href(url) de la pagina actual
2.2 window.location.hostname ==> //devuelve el nombre del dominio del servidor web
2.3 window.location.pathname ==> //devuelve la ruta y el nombre del archivo de la pagina actual
2.4 window.location.protocol ==> //devuelve el protocolo web utilizado(http: 0 https:)
2.5 window.location.assing() ==> //carga un nuevo documento

8 CAPITULO:
1 Herramientas_De_Desarrollo_Del_Navegador_CHROME:

2 Pestaña_Elements:
2.1 filtros_y_palabras_clave:
2.2 modificar_crear_y_eliminar_etiquetas:
2.3 modificar_propiedades_css:
2.4 event_listeners:

2.5 properties:
2.6 opciones_para_copiar:
2.7 seguir_explorando:

3 Pestaña_Resources:
3.1 ver_y_guardar_archivos:
3.2 informacion_ofrecida:
3.3 cambios_en_tiempo_real:
3.4 snippets: //ejecutar codigo por fragmentos

4 Pestaña_Networks:
4.1 interfaz:
4.2 sort_by_y_filtros_de_busqueda:
4.3 limpiar_cookies_y_cache:
4.4 importar_y_exportar:

5 Pestaña_TimeLine:
5.1 uso_grabacion:
5.2 screenshots_de_la_grabacion:
5.3 controls:
5.4 overview: //fps, cpu y net
5.5 flame_chart:
5.6 details:
5.7 sugerencias_de_grabacion:
5.8 hay_otras_pero_me_dio_pereza_escribirlas:

6 Pestaña_Application:
6.1 application:
6.2 storage:
6.3 cache:
6.4 background_services:
6.5 frames:

//estas son las mas usadas las otras tambien ayudan pero poco

9 CAPITULO:
1 Los_Eventos: //los eventos son cualquier cambio que suceda despues de cargar la pagina

1 event_handlers: //de eventos
2 event_listeners: //eventos de escucha //para que funcione el script debe estar en body a lo ultimo
3 objeto_Event:
4 event_flow: //flujo de eventos
event_bubbling_vs_event_capturing:
const button = document.getElementById("contenedor");
button.addEventListener("Click", (e)=>{
document.write("me apretaste")});

6 event_stop_propagation: //eventstopPropagation() //termina los eventos

2 Los_Eventos_Del_Mouse:
1 click: //ocurre con un click
2 dblclick: //ocurre con doble click
3 mouseover: //ocurre cuando el puntero se mueve sobre un elemento o uno de sus hijos
4 mouseout: //ocurre cuando el puntero se mueve fuera de un elemento o elemento secundario
//---otros---
5 contextmenu: //ocurre con el click derecho para abrir el menu contextual
6 mouseenter: //ocurre cuando el puntero se mueve sobre el elemento
7 mouseleave: //ocurre cuando el puntero se mueve fuera del elemento
8 mouseup: //ocurre cuando el usuario suelta un boton del mouse sobre el elemento
9 mausemove: //ocurre miestras el puntero se mueve mientras esta sobre el elemento
10 mousedown: //ocurre cuando el usuario aprieta un boton del mouse sobre el elemento

3 Los_Eventos_Del_Teclado: //cuando pasa algo con las teclas del teclado
1 keydown: //ocurre cuando una tecla se presiona
2 keypress: //ocurre cuando una tecla se deja de presionar
3 onkeyup: //ocurre despues de que los dos eventos anteriores hayan concluido consecutivamente

4 Los_Eventos_De_La_Interfaz:
1 error: //ocurre cuando sucede un error en la carga del archivo multimedia
2 load: //ocurre cuando un objeto se ha cargado
3 beforeunload: //ocurre antes de que el documento se cargue o se valla a otro lugar
4 unload: //ocurre una vez se halla descargado la pagina
5 resize: //ocurre cuanndo se cambia el tamaño de vista de un documento
6 scroll: //ocurre cuando se deplaza la barra de desplazamiento de un elemento
7 select: //ocurre despues de que el usuario selecciona algun texto de <input> o <textarea>

5 Mas_En: //https://www.w3schools.com/jsref/obj_events.asp //link donde estan todos los eventos

6 Timers_Temporizadores: //nos permite hacer trabajo con tiempo
Eje:

```
setTimeout(()=>{
        document.write("hola");
}, 2000) //se escribe la cantidad de segundos en milisegundos
```

1 setTimeout: //setTimeout() //ejecuta una funcion en la cantidad de segundos
2 setInterval: //setInterval() //ejecuta siempre una funcion en la cantidad de segundos
//lo ejecuta infinitas veces
3 clearTimeout: //clearTimeout() //se usa para que nunca se ejecute la de arriba porque eliminamos
4 clearInterval: //clearInterval() //se usa para que nunca se ejecute la de arriba porque la elimina

10 CAPITULO:

1 Control_De_Flujo_Y_Manejo_De_Errores:
1 sentencias_de_bloque: //crear un nuevo ambito de variables
2 sentencias_de_control_de_flujo: //if, else if y else
3 sentencias_de_manejo_de_excepciones: //sirve para manejar errores

2 Sentencia_Switch:
1 sintaxis_y_clausula_case: //es parecido a if, else if y else solo que todo dentro

```
let expr = 3;
switch(expr){
        case 1: console.log("1");
        break
        case 2: console.log("2");
        break
        case 3: console.log("3");
        break
}
```

2 break: //termina sentencias
3 defaul: //defaul es como un else se ejecuta cuando las demas no cumplen la funcion

3 Excepciones_Y_Tipos_De_Excepciones:

1 excepciones_ECMAScript: //son las de java script
2 DOMExcepcion_y_DOMError: //son las del DOM

4 Try_Catch: //se usa cuando una minima probabilidad de error
1 sintaxis: //se usan tambien typeof que nos dice que es

```
try { //obligatorio acompañamiento con catch o finally

}
catch (){

}
```

2 objeto_error:
3 catch_incondicional: // el que no tiene condicion adentro
4 catch_condicional: //cuando hay una condicion como if dentro de de eso
5 sentencia_throw: //tira un error tambien tira objetos

6 Finally: //ejecutarse a toda costa se ejecuta pase lo que pase

11 CAPITULO:

1 Desventajas_De_Trabajar_De_Manera_Obsoleta:

//¿cuando un codigo es obsoleto?

1 "deprecated": //obsoleto

2 inutil:

3 no_recomendado:

4 con_bugs_o_fallos:

5 esta_por_ser_reemplazado:

6 hay_mejores_formas_de_hacerlo:

//los efectos negativos (aplicando a metodos, clases y propiedades) son desventajas:

1 uso_excesivo_de_recursos:

2 codigo_con_bugs_o_fallos:

3 codigo_innecesariamente_largo:

4 SEO: //la pagina detecta errores

//¿como verificar si esta obsoleto?

1 1:_de_cada_3_webs_utiliza_librerias_de_javascript_absoletas:

2 verificar_si_tienen_o_usan_funciones_metodos_objetos_o_metodologias_absoletas:

3 verificar_en_los_sitios_basados_en_estandares_oficiales:

2 Detectar_Navegadores_Obsoletos:

1 deteccion_del_navegador_y_cobertura_a_multiples_navegadores:

3 Recomendaciones:

1 verificar_si_algo_es_estandar:

2 investigar_en_las_paginas_oficiales:

3 estar_constantemente_revisando_lo_nuevo_de_javascript:

4 mirar_como_otros_desarrolladores_hacen_eso_de_una_manera_mas_optimizada:

12 CAPITULO:

1 Callbacks: //funcion dentro de otra funcion y es una funcion que llama a otra funcion

1 eje:

```
function prueba(callback) {
    callback("Cris");
}
function decirNombre(nombre) {
    console.log("nombre");
}
prueba(decirNombre);
```

2 problemas_de_los_callbacks: //que nos pone a hacer cosas infinitas y para eso son las promesas

2 Promesas: //son un objeto (resolve,reject)

1 terminacion_de_una_operacion_asincrona:

2 fracaso_de_una_operacion_asincrona:

3 eje: then() //nos ayuda a resolver las promesas

```
let nombre = "christian";
const promesa = new Promise((resolve,reject)=>{
    if (nombre !== "christian") { reject("lo siento no es pedro");}
    else resolve(nombre);
})
promesa.then((resultado)=>{ //se usa para acceder a los elementos
console.log(resultado)
}).catch((e)=>{console.log(e)})
```

4 await_y_async: //funciones asincronas funcionan con promesas

//await = obtener informacion

```
eje: const resulta = async ()=>{
        resultado = await obtenerValor();
    } resulta();
```

13 CAPITULO:

1 Peticiones_HTTP: //es una peticion que enviamos a un servidor y el servidor nos devuelve una

//informacion c ==> s y devuelve c <== s

1 cliente_y_servidor: //el cliente es el navegador y el servidor es la pagina

2 no_guardan_informacion: //no guarda la informacion

2 Datos_Estructurados_json: //(JSON) //json se usa con "" porque si no trae muchos problemas

```
eje: objeto = {"variable1" : "Chris",
                "variable2" : "Andres"}
                //para poder enviar datos a un server tienen que ser string
```

1 serializacion_y_deserializacion: //serializacion = es una cadena de texto `{"variable1" : "Chris"}`

//deserializado = es un objeto asi: {"variable1" : "Chris"}

2 metodo_parse: //parse() // convierte string con estructura json a formato json(deserializar)

```
eje: const serializar = JSON.parce(objeto); //los convierte en json
```

3 metodo_stringify: //stringify() // convierte en dato de javascript en una cadena de texto

//json(serializar)
eje: const serializar = JSON.strigify(objeto); //lo convierte en string

4 json_polyfill: //JSON Polyfill //un link que ahorita lo pongo
// funciones funcionalidad de cualquier cosas de javascript en navegadores
//como internet explorer que no soportaba polyfill

3 AJAX: //javascript asincrono //la pagina se actualiza cada vez que enviamos o damos enter
//enviar solicitud en paralelo asi c <== s y c <== AJAX <== s nos responde en paralelo
//no es soportado por todos lo navegadores
1 objeto_XMLHttpRequest: //objeto para enviar peticiones a un servidor
eje: const peticion = new XMLHttpRequest();
2 enviar_peticiones_GET: //open()= sirve para abrir, send()= sirve para enviar
eje: peticion.open("GET", "") //segundo parametro es la url
3 trabajar_el_resultado_de _las_peticiones:
4 objeto_ActiveXObject: //se usa mas que todo para internet explorer se usa por si no esta el 1 objeto
5 nueva_forma_de_trabajar_resultado: //con JSON hay que convertir a json para usarlo
6 enviar_peticiones_POST: // los datos se envian a travez del metodo post
7 diferencias_entre_GET_y_POST: //imagen url =

4 Fetch: //reemplazo de ajax
eje: peticion = fetch("URL"); //por defecto va GET
1 basado_en_promesas: //encapsulada
2 objeto_fetch:
3 text: //text() promesa desencapsulada en un texto
4 json: //json() promesa desecapsulada a un objeto json
5 blob: //blob() nos crea ruta imaginaria en donde se almacena una peticion y si actualiza hace otra
6 formdata: //formData()
7 arrayBuffer: //arrayBuffer()

5 Libreria_Axios:
1 Instalacion: //github.com/axios/axios
2 basado_en_promesas:
3 objeto_axios:
4 metodo_get_y_post: //get() y post()
5 formas_de_enviar_los_datos:
6 ventajas:

6 Fetch_Y_Axios_Con_Await_Y_Async:
    1 implementacion:
    2 importancia_el_try_catch: