1 CURSO DE JAVASCRIPT

1 CAPÍTULO:...

1 VARIABLES: = (var, let, const)
var number = 10; //globales
let numberr = 10; //limitada
const number = 11; //siempre sera la misma variable

2 TIPOS_DE_DATOS: = (String, Number, Boolean, Symbol = mas Adelante la escribo)
String = "Cadena de texto"
Number = 19
Boolean = true or false

3 CASOS_ESPECIALES_DE_DATOS: = (undefined, null, NaN = mas Adelante)
undefined = no tiene valor definido
null = es variable vacia
NaN = no es un numero cuando se suma o se utiliza otro operador//solo numeros

multiples variables
let numero, numero1, numero2;

prompt("decime tu nombre") //se usa con variables para que lo que escribimos se almacene

4 OPERADORES:
//aritmeticos toman valores numericos(ya sean literales o variables)como sus operandos y retornan un valor numerico unico
Residuo (%)
Incremento (++)
Decremento (--)
Negación unaria (-)
Positivo unario (+)
Operador de exponenciación (**)
//de asignacion asigna un valor al operando de la izquierda basado en el valor del operando de la derecha
Operadores de asignación compuestos

Nombre	Operador abreviado	Significado
Asignación	x = y	x = y
Asignación de adición	x += y	x = x + y
Asignación de resta	x -= y	x = x - y
Asignación de multiplicación	x *= y	x = x * y
Asignación de división	x /= y	x = x / y
Asignación de residuo	x %= y	x = x % y
Asignación de exponenciación	x **= y	x = x ** y

5 CONCATENACION: //unir cadenas de texto
//ejemplo:

```
cadena1 = "Cadena";
cadena2 = "every day!";
resultado = cadena1 + cadena2;
document.write(resultado);
//forzar string
cadena1 = 5;
cadena2 = 10;
resultado = "" + cadena1 + cadena2;
document.write(resultado);
//concat al menos 1 string para que funcione
cadena1 = "every day";
cadena2 = 8;
resultado = cadena1.concat(cadena2);
document.write(resultado);
//con backtiks(``) y la variable entre ${}
cadena1 = "every day";
cadena2 = `al ${cadena1} almuerzo`;
resultado = cadena1.concat(cadena2);
document.write(resultado);
```

6 BACKTIKS:
escape comillas simples
nombre1 = "christian";
frase = "mi nombre es´cristian´ y soy pro ";

7 OPERADORES(INTERMEDIOS):
comparacion:
Operador devuelven true o false
Igual(==)
No es igual(!=)
Estrictamente igual(===)
Desigualdad estricta(!==)
Mayor que(>)
Mayor o igual que(>=)
Menor que(<)
Menor o igual(<=)
logicos:

Operador	Uso	Descripción
AND Lógico (&&)	expr1 && expr2	Devuelve expr1 si se puede convertir a false; de lo contrario, devuelve expr2. Por lo tanto, cuando se usa con valores booleanos, && devuelve true si ambos operandos son true; de lo contrario, devuelve false.
OR lógico (||)	expr1 || expr2	Devuelve expr1 si se puede convertir a true; de lo contrario, devuelve expr2. Por lo tanto, cuando se usa con valores booleanos, || devuelve true si alguno de los operandos es true; si ambos son falsos, devuelve false.
NOT lógico (!) "!expr		Devuelve false si su único operando se puede convertir a true; de lo contrario, devuelve true."

8 CAMEL_CASE:

//primera letra palabra minuscula y el comienzo de la otra mayuscula

10 CONDICIONALES: //sentencia que permite ejecutar un bloque de codigo true o false //se utiliza con comparadores
if (true) {} //si se cumple o no false o true
else if (true) {} // si lo anterior no de cumple ejecuta otra condicion
else {} //si no se cumple ninguna

11 CREAR_SOLUCIONES_HISTORIA_COFLA:
Ejemplo

```
dineroCofla = prompt("Cuanto dinero tienes Cofla?");
dineroCofla = parseInt(dineroCofla); //Funcion que permite converti a numero
if (dineroCofla > 0.6 && dineroCofla < 1) {
        alert("Comprate el helado de agua");
        alert("y te sobran" +(dineroCofla - 0.6));
}
else if (dineroCofla >= 1 && dineroCofla < 1.6) {
        alert("Comprate el helado de crema");
        alert("y te sobran" +(dineroCofla - 1));
}
else if (dineroCofla >= 1.6 && dineroCofla < 1.7) {
        alert("Comprate el helado de arequipe");
        alert("y te sobran" +(dineroCofla - 1.6));
}
else if (dineroCofla >= 1.7 && dineroCofla < 1.8) {
        alert("Comprate el helado de chocolate");
        alert("y te sobran" +(dineroCofla - 1.7));
}
else if (dineroCofla >= 1.8 && dineroCofla < 2.9) {
        alert("Comprate el helado de naranja");
        alert("y te sobran" +(dineroCofla - 1.8));
}
else if (dineroCofla >= 2.9) {
        alert("helado con chispa o con pepitas");
        alert("y te sobran" +(dineroCofla - 2.9));
}
else {
        alert("Lo siento no te alcanza para ningun helado");}
        alert("y te sobran" +(dineroCofla - 3));
```

2 CAPITULO:...

1 ARRAYS: //Arreglos //conjunto de datos //se puede guardar mas cosas que una variable //los que no estan difinidos te tira undefined
Ejemplo

```
arrays = ["Noe", "Pablo", "Juan", "Pedro"];
nombresBiblicos = ["Noe", "Pablo", "Juan", "Pedro"];
```

```
//Llamar arrays
document.write(nombresBiblicos);
//mostrar elementos de los arrays
//posiciones arrancan desde 0 en vida real es 1
document.write(nombresBiblicos[0]); //= Noe


//ArraysAsociativos //array con valor asociado //no muestra el dato que damos nos
//muestra el siguiente como 0 = 1 = 2 = 3 y asi sucesivamenete

let pc = {
        nombre:"Andres",
        procesador:"Rycen 5500g",
        ram:"16GB",
        espacio:"1TB"
};
document.write(pc["nombre"]); //y asi con las otras se puede asi
//Tambien Asi
let nombre = pc["nombre"];
let procesador = pc["procesador"];
let ram = pc["ram"];
let espacio = pc["espacio"];

frase1 = `el nombre de mi pc es: <b>${nombre}</b><br>
                el procesador es: <b>${procesador}</b><br>
                la memoria ram es: <b>${ram}</b><br>
                el espacio es: <b>${espacio}</b>`;
document.write(frase1);

2 BUCLES_E_ITERACION://definicion: son casi lo mismo que un if solo que preguntan la
misma
//condicion todo el tiempo (se repiten cada rato)
// sentencias: (while, do while, for, for in, for of)despues(break, label, continue)

//while siempre se ejecuta y cuando no es verdadera la condicion no se ejecuta
//Ejemplo
let numero5 = 0;
while (numero5 < 6){
        numero5++;
        document.write(numero5); //no ejecutar porque si no se les apaga la compu
        //si quieren solo le ponen una ej: numero+; se les traba ej2: numero++; queda bien
        //esto da igual a = 123456;
}

//do while primero pone lo que se ejecuta y luego se pregunta y si es falsa no lo ejecuta mas
//Ejemplo
let numero5 = 0;
do{
```

```
	numero5++;
	document.write(numero5);
}
while (numero5 < 6)

//break se usa con el while
//Ejemplo
let numero5 = 0;
while (numero5 < 1000){ //se va a ejecutar 1000 veces
	numero5++;
	document.write(numero5)
	if (numero == 10) {
		break; //cuando se ejecuta el while no se ejecuta mas
	}
}
document.write("fin");

//for //Bucle determinado parecido al while pero se ejecuta las veces que indiquemos se
declaran
//tiene 3 partes : declaramos luego: damos la condicion y despues: iteramos
//tambien se puede declarar por fuera pero no es tan necesario
//Ejemplo
for (let i = 0; i < 6; i++){//solamente se ejecuta en este bloque
	document.write(i + "<br>");
}
//continue //termina iteracion pero sigue //se salta los numeros seleccionados o cualquier
cosa
//
//Ejemplo
for (let i = 0; i < 12; i++){
	if (i == 12) {
		continue;
	}
	document.write(i + "<br>");
}
//for in //devuelve la posicion del elemento osea devuelve el 0 = gato
//muestra indice
let animales = ["gato", "elefante", "rex"];//primera vuelta del bucle gato y luego otra vuelta
                                                                                    //prosigue
con las otras
for (animal in animales){// nos ahorra lo del for normal
	document.write(animal);
}
//for of //nos muestra el valor de la posicion en vez de mostrar el indice
//of recorre todo
let animales = ["gato", "elefante", "rex"];
for (animal of animales){
	document.write(animal);
```

```
}
//label
//Ejemplo
array1 = ["pedro", "Christian", "rex"];
array2 = ["maria", "andres",array1];
foreveryday://si quiero terminar todos los bubles
for (let array3 in array2){
	if (array3 == 2) {
		for (let array of array1){
			document.write(array3);
			break foreveryday;//termina todo
			break;//solo termina este bloque
		}
	} else {
			document.write(array2[array3]);
		}

}

3 FUNCIONES:
// variar formas de crear funciones
// es una porcion de codigo un bloque
//agarramos todo una parte del programa y la ejecutamos de manera mas simple
//forma de uso: primero se crea y luego se llama
//Ejemplo
respuesta = prompt("hola cristian como fue tu dia");
if (respuesta == bien) {
	alert("que bien");
}
else {
	alert("que pena");
}
//esto se puede optimizar con esto y ahorrarnos codigo
//primer forma de crear una funcion
//Ejemplo
function saludar() {
respuesta = prompt("hola cristian como fue tu dia");

if (respuesta == bien) {
	alert("que bien");
}
else {
	alert("que pena");
}
}
salidar();
salidar();
//segunda formas con variables
```

```
//Ejemplo
saludar = function () {
respuesta = prompt("hola cristian como fue tu dia");

if (respuesta == bien) {
        alert("que bien");
}
else {
        alert("que pena");
}
}
//return es para devolver un valor
function saludar() {
        alert("Hola");
        return "la funcion se ejecuto correctamente";//finaliza funcion
}
let saludo = saludar();//saludo va a ser igual al return
document.write(saludo);
//parametros //las funciones no son funciones sin parametros
//Ejemplo
function suma(num1, num2)//es como si la declararamos pero dentro de la funcion
{
        let res = num1 +num2;
        document.write(res);
}
suma(12, 32);
suma(22, 55);
//scope siempre poner let para que sea regional para no tener conflictos

//funciones flecha
const saludar = function (nombre) { //en vez de usar esto se usa
        let frase = `hola ${nombre}`
        document.write(frase)
}
saludar("pedro");
//esto
const saludar = (nombre)=>{ //si solo es una linea es asi
        let frase = `hola ${nombre}`
        document.write(frase)
}
//asi
const saludar = nombre => document.write(frase); //asi se simplifica y se optimiza y se
retorna

//automaticamente
```

3 CAPITULO://programacion orientada a objetos poo:

1 POO://es un paradicma que lo que hace es actualizar la forma en la que programamos
//y haciendo que podamos programar objetos como en la vida real
//resolver problemas de mejor forma.

2 CONCEPTOS_BASICOS_DE_LA_POO:
//clase, objeto, atributo, metodo, constructor y instanciacion
//clase //crear fabrica de objetos
//objeto //es lo que crea la clase
//atributo //son las caracteristicas
//metodo //cosas que puede hacer nuestro objeto
//constructor //funcion obligatoria
//instanciacion //intanciar objeto
//Ejemplo

```
class animal { //clase, objeto, constructor
        constructor(especie, edad, color){//this se usa para saber que objeto voy a crear y el punto saber
                                                                                //que es
                //atributos
                this.especie = especie;
                this.edad = edad;
                this.color = color;
        }
        //metodo
        verInfor(){
                document.write(this.especie);
        }
}
//instanciar
const perro = new animal("perro", 6, "rojito");
document.write(perro);
```

3 CARACTERISTICAS_DEL_POO: //abstaccion, modularidad, polimorfismo y encapsulamiento
//abstaccion: // reducir los componentes basicos de un objeto
//modularidad: // es la capacidad de resolver problema grande por partes
//encapsulamiento: // consiste en hacer que todos los datos esten privados
//polimorfirmo: // es la capacidad que tiene un objeto por comportarse distinto por sus propiedades

4 OTROS_CONCEPTOS_DE_POO: //Herencia, metodos estaticos y metodos accesores(Getters, Setters)

//herencia: //crear una clase que toma todo lo que puede hacer la otra clase y agregar cosas nuevas
//Ejemplo heredando la de arriba

```
class Perro extends animal { //no se puede tener un objeto con el mismo nombre de la clase
```

```
        constructor(especie, edad, color, raza){//ademas de las que teniamos agregamos una nueva

                super(especie, edad);
                this.raza = raza;

        }
        static ladrar
}

//metodos estaticos: //no necesita que la clase se defina para ser creado

class Perro extends animal {
        constructor(especie, edad, color, raza){

                super(especie, edad);
                this.raza = raza;

        }
        static ladrar(){
                alert("wow!");
        }
}
//metodos accesores
//getters: //son para obtener un valor
//setters: //modificarlo o definirlo
class Perro extends animal {
        constructor(especie, edad, color, raza){

                super(especie, edad);
                this.raza = raza;

        }
        set setRaza(newName){//funcionan como propiedades //este es el set //forma para modificar
                this.raza = newName;
        }
        get getRaza (){ //este es get //forma para obtener un valor
                return this.raza;
        }

}
const perro = new Perro("Pedro", 5, "cafe", "pastorAleman");
perro.setRaza = "pedro";//asi se debe poner para nada de errores
document.write(perro.raza);
```

4 CAPITULO:

1 METODOS_DE_CADENA:
Propiedades de String
//Propiedades de la instancia
/*concat()
startswith()
endswith()
includes()
indexof()
lastindexof()
padstart()
padend()
repeat()
split()
substring()
tolowercase()
touppercase()
tostring()
trim()
trimend()
trimstart()
valueof() */
EJEMPLO

```
let cadena = "Cadena de prueba";
resultado = cadena.concat("Hola");//donde esta concat va las cadenas
document.write(resultado);
```

2 METODS_DE_ARRAYS:
//Transformadores
/*pop()
shift()
push()
reverse()
unShift()
sort()
splice()*/
//Accesores
/*join()
slice()
y metodos ya vistos en cadenas /*
//De Repeticion
/*filter()
forEach()*/

3 OBJETO_MATH_BASICO:
//Metodos
//sqrt() = Muestra raiz cuadrada de un numero

```
Eje:
numero = Math.sqrt(25); //seria igual a 5
//cbrt() = Muestra raiz cubica de un numero
Eje:
numero = Math.cbrt(27); //seria igual a 3
//max() = Recibe solo numeros
Eje:
numero = Math.max(4,5,7,20); //seria igual a 20 numero mayor
//min() = opuesto a max
Eje:
numero = Math.min(4,5,7,20); //seria igual a 4 numero menor
//random() = Devuelve numero aleatorio entre 0 y 1
Eje:
numero = Math.random(); //seria igual a numero entre 0 y 1 aleatorio como 0.1//tambien
puede dar un
                                                    //numero del 1 al 100
Eje:
numero = Math.random()*100;
numero = numero.toString();
num = numero[0] + numero[1];
if (numero[1] == ".") {
        num = numero[0];
}
document.write(num); // seria igual a un numero normal no decimal = 10

//round() = Devuelve un numero redondeado mas cercano
Eje:
numero = Math.random()*100;
numero = Math.round(numero);
document.write(numero);
//fround() = nos carga 15 numeros a partir del .
Eje:
numero = Math.fround(9.999999);// va  a ser igual a = 10
numero = Math.fround(9.759999);//va a ser igual a = 9.76
//floor() = Devuelve un entero que sea para abajo le saca el decimal
Eje:
numero = Math.random()*100;
numero = Math.floor(3.999); //seria igual a 3 le quita el . y los 999
//trunc() = inverso a round no redondea //elimina los decimales
Eje:
numero = Math.round(9.9); //igual 10
numero = Math.trunc(9.9); //igual 9


//Propiedades
/*
PI
SQRT1_2 raiz cuadrada de 1/2
```

SQRT2 = raiz cuadrada de 2
E = constante de euler
LN2 = logaritmo natural de 2
LN10 = logaritmo natural de 10
LOG2E = logaritmo de E con base 2
LOG10E = logaritmo de E con base 10 */

5 CAPITULO:

1 CONSOLA:

```
console.clear
console.error
console.info
console.log
console.table
console.warn
//Funciones de registro
/*
clear()//limpia la consola
error()//da error en consola
info()//mensaje informativo
log()//mensaje depuracion
table()//necesita array o objeto y a partir de ello nos hace una tabla
warn()//tira advertencia es parecido a un error pero es una advertencia
*/
//funciones de conteo
console.count
console.countReset
/*
count() //dice cuantas veces se ejecuta una funcion
countReset() // se resetea el conteo
*/
//funciones de agrupacion
console.group
console.groupEnd
console.groupCollapsed

/*
group() //crear nivel de grupo y se muestra abierto
groupEnd()// elimina los grupos creados
groupCollapsed() //el grupo se crea cerrado
*/
//funciones de temporizacion
console.time
console.timeEnd
console.timeLog
```

```
/*
time()//crea tiempo
timeLog()//cuanto tarda
timeEnd()//termina el tiempo

*/
//otra cosa que puede hacer es Modificar estilo de texto
//se añade %c asi es igual que en css
console.log("%c Christian","color:red; background: blue; border: 3px solid green");
```

6 CAPITULO:

1 DOM://Document Object Model
// una interfaz que contiene todos los elementos como html y css
1 Nodo:// un nodo es cualquier etiqueta del cuerpo, como parrafo, el mismo body o
//las etiquetas de listas
//hay varios tipos de nodos:
//Document = nodo raiz a partir del cual derivan los otros
//Element = nodos definidos por etiquetas de html
//Text = texto adentro de un nodo element(texto)
//Attribute = los atributos de las etiquetas definen nodos informacion asociada a un nodo element
//Comentarios y otros = comentarios que hacemos y otras cosas como "doctype"
2 Metodos_De_Seleccion_De_Elementos:Document-document
//getElementById() = selecciona un elemento por id
Eje:

```
document.getElementById("Player") //el id del elemento
```

//getElementByTagName() = selecciona todos los elementos que coincidan con nombre de la etiqueta
Eje:

```
document.getElementsByTagName("p") //todos elementos <p></p>
```

//querySelector() = devuelven el primer elemento que coincida con el grupo de selectores de css
Eje:

```
document.querySelector("#Player") //es como en css = #, . , *
```

//querySelectorAll() = devuelve todos los elementos que coincida con el grupo de selectores
Eje:

```
document.querySelectorAll("#Player") //todos los elementos
```

3 Metodos_Para_Definir_Obtener_y_Eliminar_Valores_De_Atributos:
//setAttribute() //modifica un valor de un atributo
Eje:

```
const jugador = document.querySelector("#Player")
jugador.setAttribute("type", "color")//lo modifica a color o como uno quiera
```

//getAttribute() //obtiene valor de un atributo
Eje:

valordeatributo = jugador.getAttribute("type") //nos da el valor del atributo = text o submit cualquiera
//removeAttribute()//Remueve valor de un atributo
Eje:
removeratributo = jugador.removeAttribute("type") //se borra el atributo

4 Atributos_Globales:

//contentEditable = true or false nos ayuda a editar un titulo o lo que sea
Eje:
const titulo = document.querySelector("#titulo");
titulo.setAttribute("contentEditable", "true")//true or false
//dir = edita igual tiene 3 valores = rtl derecha izquierda, ltr izquierda derecha
Eje:
const titulo = document.querySelector("#titulo");
titulo.setAttribute("dir", "rtl")
//hidden = indica si el elemento es o no es relevante
Eje:
const titulo = document.querySelector("#titulo");
titulo.setAttribute("hidden", "false")
//tabindex = indica si el elemento puede tener un autofocus o no si se puede seleccionar
Eje:
const titulo = document.querySelector("#titulo");
titulo.setAttribute("tabindex", "3")//numero decide como se va a desplazar
//title = a los que pasamos mouse ensima nos muestra lo que es
Eje:
const titulo = document.querySelector("#titulo");
titulo.setAttribute("title", "jajajja")//nos va a mostrar jajajja

5 Atributos_De_Inputs://accedemos directamente de java script
//className
const input = document.querySelector("#input");
document.write(input.className) //nos muestra la clase
//value
const valor = document.querySelector("#valor");
document.write(valor.value) //nos muestra el valor
//type
const tipo = document.querySelector("#type");
input.type = "color"; //cambia el tipo a color o texto o numero
//accept
const tipo = document.querySelector("#type");
input.accept = "imagen/png"; //hace una referencia a lo que solo se acepta en cualquier formato
//form actualiza los inputs que esten afuera del formulario
//minlength = minima cantidad de caracteres de un input
const tipo = document.querySelector("#type");
input.minlength = "4"; //minima de caracteres
//placeholder

```
const tipo = document.querySelector("#type");
input.placeholder = "tu nombre"; //aparece en el campo para escribir o algo
//requerid = nos permite decir si es requerido o no
const tipo = document.querySelector("#type");
input.requerid = " ";//es requerido para poder pasar al siguiente

6 Atributo_Style:
//cambiar propiedades
//propiedades camel case
const titulo1 = document.querySelector("#type");
titulo1.style.color = "red"; //pone color rojo //parecido a css
titulo1.style.backgroundColor = "red"; //se remueve el "-" background-color

7 Clases_Classlist_y_Metodos_De_Classlist:
//trabaja con clases

//Metodos:
//add() = Añadir clase
const titulo1 = document.querySelector(".titulo");
titulo1.classList.add("grande")//añade una clase = titulo grande
//remove() = Remover clase
const titulo1 = document.querySelector(".titulo");
titulo1.classList.add("grande")//elimina una clase = grande
//item() = Devuelve clase de el indice especificado
const titulo1 = document.querySelector(".titulo");
titulo1.classList.item(0)//selecciona una clase = titulo
//contains() si la clase es true o false si existe o no
const titulo1 = document.querySelector(".titulo");
titulo1.classList.contains("grande")// si es igual a ello nos devuelve true
//replace() = reemplaza una clase por otra
const titulo1 = document.querySelector(".titulo");
titulo1.classList.contains("grande","chico")//reemplaza grande por chico
//toggle() = si la tiene la elimina, si no la agrega
const titulo1 = document.querySelector(".titulo");
titulo1.classList.toggle("grande")// la agrega o la elimina si la tiene

8 Obtencion_y_Modificacion_de_Elementos:
//textContent = contenido de la etiqueta el texto desde que empieza hasta que termina
const titulo1 = document.querySelector(".titulo");
resultado = titulo1.textContent;
//innerHTML = muestra todo el codigo
const titulo1 = document.querySelector(".titulo");
resultado = titulo1.innerHTML;
document.write(resultado) //nos muestra todo el codigo
//outerHTML = nos muestra la etiqueta y todo lo que contiene
const titulo1 = document.querySelector(".titulo");
resultado = titulo1.outerHTML;
document.write(resultado)
```

9 Creacion_De_elementos:

```
//createElements()
const contenido = document.querySelector(".contenedor");
const item = document.createElement("LI")//siempre mayusculas
//createTextNode()
const contenido = document.querySelector(".contenedor");
const item = document.createElement("LI")
const elemento = document.createTextNode("esto es elemento de la lista")
//appendChild()
const contenido = document.querySelector(".contenedor");
const item = document.createElement("LI")
const elemento = document.createTextNode("esto es elemento de la lista")
item.appendChild(elemento)
console.log(item);
//createDocumentFragment()
const contenido = document.querySelector(".contenedor");
const fragmento = document.createDocumentFragment();
for (i = 0; i < 20; i++) {
        const item = document.createElement("LI")
        item.innerHTML = "esto es elemento de la lista";
        fragmento.appendChild(item);
}
contenido.appendChild(fragmento);
console.log(contenido);
```

10 Obtencion_y_Modificacion_de_Childs://(hijos)
//firstChild //obtiene primer hijo
//lastChild //ultimo hijo
//firstElementChild //obtener el primer hijo del elemento
//lastElementChild //obtiene el ultimo hijo del elemento
//childNodes //todos los nodos hijos
//children //devuelve solo etiquetas html
11 Metodos_De_Childs: //trabajar metodos hijo
//replaceChild() //se usa para reemplazar metodos hijos
//removeChild() //se usa para remover metodos hijos o padre
//hasChildNodes() //sirve para verificar si tiene o no elemento hijo
12 Propiedades_De_Parend://Padres
//parendElement //elegir elemento de un elemento //son casi los mismo los 2
//parendNode //elegir el padre de un elemento
13 Propiedades_De_Siblings://Hermanos son los que estan en el mismo nivel
//nextSibling //lo que le sigue ala etiqueta
//previusSibling //anterior a la etiqueta
//nextElementSibling //lo que le sigue al elemento
//previusElementsSibling //anterior al elemento
14 Nodos_Extras:
//closest() elemento accendente mas cercano //muestra al contenedor que tenga ese elemento mas cercano

15 Length: //propiedad de string representa la longitud de una cadena, en unidades de codigo utf-16
//length //devuelve el numero de caracteres de una cadena