<table>
<tr><th colspan="3">Manera de manejar Objetos.</th></tr>
<tr><td>let coche = new Object();
coche.color = “rojo”;</td><td>let coche = {
color : “rojo”,
puertas : 3
};</td><td>
console.log(coche.color)
console.log(coche[“color”])
delete coche.color;</td></tr>
<tr><td colspan="3">for (propiedad in coche){
console.log(“Propiedad = “+ propiedad + “con valor “ + coche[propiedad])
}</td></tr>
<tr><td colspan="3">let usuario = {
nombre : “José”,
devolverDatos : function(){return “Nombre: “ + this.nombre};
}
usuario.devolverDatos();</td></tr>
<tr><td colspan="3">let usuario = new Object();
objeto.saludar = function(){ console.log(“Hola amigo!”);};</td></tr>
<tr><th colspan="3">Maneras de crear clases.</th></tr>
<tr><td colspan="2">NO USAR, ES MUY FEO
var Persona = class {
constructor(nombre) {
this.nombre = nombre;
}
}
var Coordena = class
{
constructor(x,y) { this.x =x ; this.y = y; }</td><td>class Coordenada {
constructor(x, y) {
this.x = x;
this.y = y;
}

//Forma resumida
esIgual(coordenada) {
if (this.x == coordenada.x && this.y == coordenada.y) {
return true;
}
return false;
}

//Forma tradicional
imprimirCoordenadas = function () {
console.log(`La x es ${this.x} y la y es ${this.y}`);
}
}</td></tr>
<tr><td colspan="2"></td><td></td></tr>
</table>

```
class Caracola{
static saludar () {
    console.log("Hola");
}
saludarUsu(mensaje){
  return "Hola Caracola" + mensaje;
}
}

let c = new Caracola();
console.log(c.saludarUsu("Jose"));

Caracola.saludar();
```

```
constructor(name, password, email) {
        let name     = name;
        let password = password;
        let email    = email;

        let privado  = function() {
        console.log("Accediste a un método
privado");
        }
         //Privado
        this.getPrivado = function(){ privado(); }
        this.getNombre = function(){return
name;}
  }
}
```

```
console.log("Es de clase ", perroZombi instanceof Object); TRUE O FALSE
```

ARROW.

```
nombre = (arg) => { console.log(arg) };
nombre = () => { console.log("adios")};
```

```
arg => console.log(arg);
```

Map.

```
[array].map(funcion)
Devuelve un array con la misma longitud que el que le hemos pasado:
let coches = [
  {color: red, puertas: 3},
  {color: blue, puertas: 6},
  {color:yellow, puertas: 9}
];
let mapeo = coches.map( coches => coches.color);  //ARRAY DE COLORES DE COCHES (string)
```

Filter.

```
[array].filter(funcion)
Devuelve un array con distinta longitud:
let coches = [
  {color: red, puertas: 3},
  {color: blue, puertas: 6},
  {color:yellow, puertas: 9}
];
let mapeo = coches.filter( coches => coches.color == "red");  //ARRAY DE COLORES ROJOS DE COCHES
(string)
```

<table>
<tr><th colspan="3">Reduce.</th></tr>
<tr><td colspan="3">[array].reduce(funcion)
Devuelve un array con la misma longitud que el que le hemos pasado:
let coches = [
{color: red, puertas: 3},
{color: blue, puertas: 6},
{color:yellow, puertas: 9}
];
let mapeo = coches.reduce((a,b) => a+b, valorActualAcumulador); //Devuelve la suma de todos los numeros empezando por 4</td></tr>
<tr><th colspan="3">Maneras de añadir una función a un elemento.</th></tr>
<tr><td><p onclick=”funcion(this);”>
p.onclick = cambiar;</td><td>elemento.addEventListener (“click”, funcion(){
});
elemento.removeEventListener(“click”, function());</td><td>elemento.addEventListener(“click”, funcion);</td></tr>
<tr><td colspan="2">window.event -> e.type == “click” etc</td><td>e.target == “p” || e.target.id == “idP”</td></tr>
<tr><td colspan="3">onclick=”alerta(); return false;” Primero alerta y no te lleva a ningun lado mas, si pones true, si.</td></tr>
<tr><th colspan="3">Maneras de obtener elementos y Manejarlos.</th></tr>
<tr><td colspan="2">document.getElementByid(“p”);</td><td></td></tr>
<tr><td colspan="2">array = document.getElementByTagName(“p”);</td><td>array[0].value | array[0].style.display = “” etc</td></tr>
<tr><td colspan="2">event.key -> es la tecla pulsada</td><td>a.classList.contains(“btn”) |
a.classList.remove(“”) | a.classList.add(“”)</td></tr>
<tr><td colspan="3">elemento.setAttribute(“align”, “center”) || elemento.removeAttribute(“align”)
elemento.required = true ||</td></tr>
<tr><td colspan="2">var formulario2 = document.forms["miFormulario"];</td><td></td></tr>
<tr><th colspan="3">Validaciones.</th></tr>
<tr><td>!nombre.checkValidity() ...</td><td colspan="2">elemento.setCustomValidity(“Tienes que introducir un numero!!”)</td></tr>
<tr><td colspan="3">innerHTML = elemento.validationMessage; → insertamos donde queramos el mensaje de validacion</td></tr>
<tr><td colspan="2">elemento.validity.patternMismatch</td><td>elemento.textContent -> devuelve texto
elemento.innerHTML -> devuelve html</td></tr>
<tr><td>e.validity.rangeOverflow</td><td>e.validity.rangeUnderflow</td><td>e.validity.tooLong para el maxLenght</td></tr>
<tr><td colspan="2">e.validity.valueMissing para required</td><td>e.validity.valid para ver si es valido</td></tr>
<tr><th colspan="3">Nodos.</th></tr>
</table>

e.parentNode	e.firstElementChild	e.nodo.textContent para el contenido del elemento
e.lastElementChild	e.nodo.children un array	e.nodo.innerHTML para el html del elemento
elemento.previousElementSibling -> anterior hijo igual		elemento.nextElementSibling -> siguiente hijo igual
elemento.hasChildNodes() true o false		elemento.append(meter hijo);
parrafo = document.createElement(“p”);		nodo = document.createTextNode(“Hola soy p”);
padre.removeChild(hijo)		padre.replaceChild(parrafo, sustituir)
Tablas.		
tabla.insertRow(0) principio	row.insertCell(0) celda 0	row.deleteCell(0) borra celda 0
tabla.insertRow() final	tabla.deleteRow(0) fila 0	row.deleteCell(-1) borra ultima celda
this.rowIndex, indice actual	tabla.deleteRow(-1) final	tabla.rows | cells para array de las columnas
document.querySelector(“CSS SELECTOR, etc”); document.querySelectorAll(“CSS SELECTOR, etc”);		
#identificador,	.nomClase	element -> por ejemplo “p”
element 1 > element 2	[type=”button”]	: First-child, :nth-child(numero)