UT4_Programación con arrays y objetos definidos por el usuario.

Programación con «arrays» y objetos definidos por el usuario.

Unidad Trabajo nº 4

Autor: David García Fernández

- Saber crear arrays y recorrerlos empleando diferentes tipos de bucles.
- Conocer los métodos del objeto Array y su aplicación.
- Saber crear y usar arrays de más de una dimensión.
- Saber crear y utilizar objetos definidos por el propio usuario e implementar sus propiedades y métodos.

Introducción

Arrays en Javascript.

- Hemos estudiado los objetos nativos y predefinidos de Javascript sin haber nombrado los objetos arrays, pero lo cierto es que los arrays son objetos propios (nativos y predefinidos) de Javascript. La importancia y el tratamiento de estos han hecho que se dedique una unidad para ellos.
- Los arrays van a permitir organizar los datos que genera una aplicación de JavaScript de forma más eficientemente, evitando situaciones como la siguiente:

```
var bicicleta1 = "colnago";
var bicicleta2 = "stevens";
var bicicleta3 = "trek";
```

  En este ejemplo, si el número de datos aumenta, sería necesario crear tantas variables como datos tenedremos. = ! Esto es un problema !
- Los arrays solucionan en problema anterior sencillamente:

```
const bicicletas = ["colnago","stevens","trek"];
```

- Aunque el nombre "array" es el más utilizado para designar a este tipo de objetos, también pueden llamarse: matrices, arreglos o vectores.

Objetos definidos por el usuario.

- Objetos de usuario: Permiten crear estructuras de datos según las necesidades de programación.
- La combinación de arrays con objetos de usuario va a permitir crear aplicaciones con una amplia funcionalidad.

Arrays

El objeto Array

- Como en todos los lenguajes de programación es necesario guardar bajo un mismo identificador o variable múltiples valores.
- Podemos considerar al array como una "lista" de valores guardados en una cierta posición (índice), como se muestra en la imagen:

Nombre Array	Valor1	Valor 2	Valor 3	.....	Valor N
	[0]	[1]	[2]	.....	[N-1]

- Los índices del array son números (enteros) y a través de ellos se referencian los valores del array.
- Los arrays son un tipo especial de objeto que referencia a sus valores a través de números, mientras que los objetos en general (Javascript) son referenciados a través de un nombre.
- Tanto la longitud como el tipo de los elementos de un array son variables. Esto puede provocar que el programador deba controlar situaciones de datos no contiguos (si es lo que no desea).

```
let bicicletas = ["colnago","trek","stevens"];

let primero = bicicletas[0];
// colnago

let tercero= bicicletas[2];
// stevens
```

Propiedades del objeto Array:

Propiedad	Descripción
constructor	Devuelve la función que creó el prototipo del objeto array.
length	Ajusta o devuelve el número de elementos en un array.
prototype	Te permite añadir propiedades y métodos a un objeto.

Métodos del objeto Array:

Método	Description
concat()	Une dos o más matrices y devuelve una copia de las matrices unidas
copyWithin()	Copia elementos de la matriz dentro de la matriz, hacia y desde posiciones especificadas
entries()	Devuelve un objeto de iteración de matriz de par clave / valor
every()	Comprueba si todos los elementos de una matriz pasan una prueba
fill()	Rellena los elementos de una matriz con un valor estático
filter()	Crea una nueva matriz con cada elemento de una matriz que pasa una prueba
find()	Devuelve el valor del primer elemento de una matriz que pasa una prueba
findIndex()	Devuelve el índice del primer elemento de una matriz que pasa una prueba
forEach()	Llama a una función para cada elemento de la matriz.
from()	Crea una matriz a partir de un objeto
includes()	Verifique si una matriz contiene el elemento especificado
indexOf()	Busca un elemento en la matriz y devuelve su posición
isArray()	Comprueba si un objeto es una matriz
join()	Une todos los elementos de una matriz en una cadena
keys()	Devuelve un objeto de iteración de matriz, que contiene las claves de la matriz original
lastIndexOf()	Busca un elemento en la matriz, comenzando por el final, y devuelve su posición.
map()	Crea una nueva matriz con el resultado de llamar a una función para cada elemento de la matriz
pop()	Elimina el último elemento de una matriz y devuelve ese elemento.
push()	Agrega nuevos elementos al final de una matriz y devuelve la nueva longitud
reduce()	Reducir los valores de una matriz a un solo valor (yendo de izquierda a derecha)
reduceRight()	Reducir los valores de una matriz a un solo valor (yendo de derecha a izquierda)
reverse()	Invierte el orden de los elementos en una matriz
shift()	Elimina el primer elemento de una matriz y devuelve ese elemento
slice()	Selecciona una parte de una matriz y devuelve la nueva matriz.
some()	Comprueba si alguno de los elementos de una matriz pasa una prueba
sort()	Ordena los elementos de una matriz
splice()	Agrega / elimina elementos de una matriz
toString()	Convierte una matriz en una cadena y devuelve el resultado
unshift()	Agrega nuevos elementos al comienzo de una matriz y devuelve la nueva longitud
valueOf()	Devuelve el valor primitivo de una matriz.

Debes saber:

- Es necesario consultar la referencia de javascript para comprobar la compatibilidad de los métodos en cada navegador. (Referencia <https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array>)

Investigación:

¿ Sabes qué métodos del objeto array no son compatibles en Internet Explorer hasta la fecha?

Solución.

No son compatibles: at(), copyWithin(), entries(), fill(), find(), findIndex(), flat(), flatMap(), from(), includes(), keys(), of() y values()

Manejo y uso de arrays

¿ Cómo recorrer un array ?

- La primera consideración a la hora de trabajar con arrays es que su primer valor está almacenado en el índice [0] y el último [su longitud -1].
- Al estar delimitados su índice inicio e índice fin, podemos conocer los valores almacenados con un bucle:

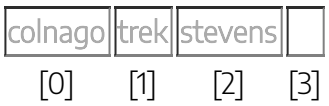
```
let bicicletas = ["colnago","trek","stevens"];

for (i=0;i<bicicletas.length;i++){
   console.log(biclieta[i]);
}
```

- Otra forma, es utilizar el método forEach(). Con este método se ejecuta la función indicada una vez por cada elemento del array.

```
const bicicletas = ["colnago","trek","stevens"];

//Forma 1ª de forEach()
bicicletas.forEach(valores => console.log(valores));

//Forma 2ª de forEach()
let txt = "";
bicicletas.forEach(miFuncion);

function miFuncion(value, index, array) {
  txt += value + "<br>";
}

// En ambos casos, aparece por pantalla:
//colnago
//trek
//stevens
```

¿ Cómo añadir valores al array ?

- La forma muy común de añadir valores al array en cualquier lenguaje de programación, es asignar, en el índice [longitud del array], el nuevo valor.
- En nuestro ejemplo de array, el tamaño del array o longitud es = 3, y es ahí donde añadiríamos el nuevo valor gracias a la propiedad length del array:




```
bicicletas[bicicletas.length] = "specialized";   // El nuevo valor es
```

- Otra forma de hacerlo, es utilizando el método push():

```
let num_bicicletas = bicicletas.push('specialized');
console.log (num_biclicletas);

// da como resultado 4. Y añade el nuevo valor.
```

- El método push() permite añadir más de un valor en la misma sentencia.

```
bicicletas.push('canion', 'orbea', 'kuota');
console.log(bicicletas);

// daría como resultado por pantalla:
//['colnago', 'trek', 'stevens','specialized','canion', 'orbea', 'kuot
```

- Si por razones de programación se necesita añadir el valor al principio del array (en el índice [0]), podemos utilizar el método unshift().

```
let nuevaLongitud = bicicletas.unshift('scott')

// Añade "scott" al inicio y devuelve el tamaño del array.<br>
//Resultando: ['scott','colnago', 'trek', 'stevens','specialized','can
```

¿ Cómo encontrar un valor en el array ?

- El método indexOf() retorna el primer índice en el que se puede encontrar un elemento dado en el array, ó retorna -1 si el elemento no esta presente.

```
var array = [2, 9, 9];
            array.indexOf(2);     // 0
            array.indexOf(7);     // -1
            array.indexOf(9, 2);  // 2
            array.indexOf(2, -1); // -1
            array.indexOf(2, -3); // 0
```

- Encontrar todas las ocurrencias de un elemento.

```
var indices = [];
            var array = ['a', 'b', 'a', 'c', 'a', 'd'];
            var element = 'a';
            var idx = array.indexOf(element);
            while (idx != -1) {
            indices.push(idx);
            idx = array.indexOf(element, idx + 1);
            }
            console.log(indices);
            // [0, 2, 4]
```

Debes saber:

El método lastIndexOf() es muy parecido al indexOf(), tan solo se diferencia en que recorre los elementos en sentido contrario.

Rellenar huecos

¿ Qué valores devolverá las siguientes sentencias que utilizan el método lastIndexOf() ?

var array = [2, 5, 9, 2];
array.lastIndexOf(2); //Resultado =
array.lastIndexOf(7); //Resultado =
array.lastIndexOf(2, 3); //Resultado =
array.lastIndexOf(2, 2); //Resultado =
array.lastIndexOf(2, -2); // Resultado=
array.lastIndexOf(2, -1); // Resultado=

Enviar

El método lastIndexOf() devuelve el último índice en el que un elemento puede encontrarse en el array, ó -1 si el elemento no se encontrara. El array es recorrido en sentido contrario, empezando por el índice fromIndex.

Sintaxis:

```
arr.lastIndexOf(buscarElemento, indiceInicio)

buscarElemento:  Elemento a encontrar en el array.
indiceInicio (opcional): El índice en el que empieza la búsqueda en se
Por defecto la longitud del array menos uno (arr.length - 1), es decir
Si el índice es mayor o igual que la longitud del array, todo el array
Si es un valor negativo, se usará como inicio el desplazamiento desde
Aún cuando el índice es negativo, el array será recorrido desde atrás
Si el índice calculado es menor de 0, se devolverá -1, es decir, el ar
```

¿ Cómo ordenar los elmentos de un array ?

- El método sort() ordena los elementos de un array localmente y devuelve el array ordenado. El modo de ordenación por defecto responde a la posición del valor del string de acuerdo a su valor Unicode (Alfanumérico).
- Para ordenar por otro criterio el método admite pasar como parámetro una función de ordenación.
- Ejemplos de uso:

Un array de elementos string, sin especificar una función de comparación:

```
var arr = [ 'a', 'b', 'Z', 'Aa', 'AA' ];
arr.sort(); //[ 'AA', 'Aa', 'Z', 'a', 'b' ]
```

Un array de elementos numéricos, sin función de comparación:

```
var arr = [ 40, 1, 5, 200 ];
arr.sort(); //[ 1, 200, 40, 5 ]
```

Un array de elementos numéricos, usando una función de comparación:

```
var arr = [ 40, 1, 5, 200 ];
function comparar ( a, b ){ return a - b; }
arr.sort( comparar ); // [ 1, 5, 40, 200 ]
```

¿ Cómo eliminar elmentos de un array ?

Eliminar el primer elemento del array.

- El método shift() elimina el elemento en el índice cero y desplaza los valores consecutivos hacia abajo, devolviendo el valor eliminado. Si la propiedad length es 0, devuelve undefined. Este método cambia la longitud del array.

```
let bicicletas = ["colnago","trek","stevens"];
primero = bicicletas.shift();
console.log(primero);  //colnago
```

Eliminar el último elemento del array.

- El método pop() elimina el último elemento de un array y devuelve su valor al método que lo llamó. Este método cambia la longitud del array.

```
let bicicletas = ["colnago","trek","stevens"];
ultimo = bicicletas.pop();
```

```
console.log(ultimo); //stevens
```

¿ Cómo obtener partes de un array ?

Método slice():

- El método slice() devuelve una **copia referenciada** de una parte del array dentro de un nuevo array empezando por inicio hasta fin (fin no incluido). El array original no se modificará.
- Sintaxis: array.slice([inicio [, fin]]). Inicio es el primer índice a recortar y fin el último (sin incluir)

```
let bicicletas = ["colnago","trek","stevens","canion","orbea",];
ultimo = bicicletas.slice(1,3);
console.log(ultimo);

//[ 'trek', 'stevens' ]
```

Método splice():

- El método splice() cambia el contenido de un array eliminando elementos existentes y/o agregando nuevos elementos.
- Sintaxis: array.splice(inicio[, borrados[, item1[, item2[, ...]]]])

```
let bicicletas = ["colnago","trek","stevens","canion","orbea",];
let eliminados = bicicletas.splice(2, 0, 'scott');
console.log(bicicletas); //[ 'colnago', 'trek', 'scott', 'stevens', 'c
console.log(eliminados); //[]

eliminados = bicicletas.splice(2, 1);
console.log(bicicletas); //[ 'colnago', 'trek', 'stevens', 'canion', '
```

Método join():

- Obtiene todos los elementos del array como una cadena separada por un delimitador dado, o por "," en su defecto.

```
let bicicletas = ["colnago","trek","stevens","canion","orbea",];
console.log(bicicletas.join());
// Devuelve colnago,trek,stevens,canion,orbea
console.log(bicicletas.join("|"));
// Devuelve colnago|trek|stevens|canion|orbea
```

Rellenar huecos

Sobre el mismo ejemplo de array de bicicletas, ¿cómo borrarías los últimos dos elementos?

let bicicletas = ["colnago","trek","stevens","canion","orbea",];
eliminados = bicicletas. [ ] ([ ] , [ ]);
console.log(bicicletas);

Enviar

Al utilizar el parámetro inicio con -2 recorremos el array desde el final hacia el principio.

El número de elementos a recorrer es 3 al no incluirse el último elemento.

¿ Cómo aplicar cambios a los elmentos de un array ?

- El método map() crea un nuevo array con los resultados de la llamada a una función dada aplicando cambios a cada uno de sus elementos.
- Sintaxis:

```
var nuevo_array = array.map(function callback(currentValue, index
// Elemento devuelto de nuevo_array
}[, thisArg])
```

Ejemplo:

```
var numeros = [1, 4, 9];
var dobles  = numeros.map(function(num) {
  return num * 2;
```

```
});

// dobles es ahora [2, 8, 18]
// numeros sigue siendo [1, 4, 9]
```

¿ Cómo validar los elementos de un array ?

Validar todos los elementos de un array:

- El método every() determina si todos los elementos en el array satisfacen una condición.
- El método ejecuta un función callback que retornará false si encuentra algún elemento que no cumple la condición, y retorna true si todos los elementos del array cumplen la condición.
- Sintaxis: array.every(callback(element[, index[, array]])[, thisArg])

```
var numeros1 = [12, 5, 8, 130, 44];
var numeros2 = [12, 54, 18, 130, 44];

function esSuficientementeGrande(elemento, indice, array) {
  return elemento >= 10;
}
numeros1.every(esSuficientementeGrande);   // false
numeros2.every(esSuficientementeGrande); // true
```

Validar si algún elemento del el array pasa una condición:

- El método some() comprueba si al menos un elemento del array cumple con la condición implementada por la función dada.
- Sintaxis: array.some(callback(element[, index[, array]])[, thisArg])

```
function esMayor10(element, index, array) {
  return element > 10;
}
[2, 5, 8, 1, 4].some(esMayor10);  // false
[12, 5, 8, 1, 4].some(esMayor10); // true
```

Crear un array nuevo con los elementos de una array que pasan una cierta condición:

- El método filter() crea un nuevo array con todos los elementos que cumplan la condición implementada por la función dada.
- Sintaxis: var newArray = arr.filter(callback(currentValue[, index[, array]])[, thisArg])
- filter() llama a la función callback sobre cada elemento del array, y construye un nuevo array con todos los valores para los cuales callback devuelve un valor verdadero. Los elementos del array que no cumplan la condición callback simplemente los salta, y no son incluidos en el nuevo array.

```
var bicicletas = ["colnago","trek","stevens","canion","orbea",];

const salida = bicicletas.filter(bici=> bici.length > 6);

console.log(salida);
// Devuelve  [ 'colnago', 'stevens' ]
```

Rellenar huecos

Sobre el siguiente código, rellena las cajas de texto para que se visualice por pantalla los ciclistas cuyo nombre empieza por "A", utilizando una expresión regular.

let ciclistas = ["Abraham Olano","Alberto Contador","Wout Van Aert","Alejandro Valverde","Samuel Sánchez"];

const result = ciclistas. [ ] (elemento => [ ] .test([ ]));

console.log([ ]);

Enviar

La expresión regular sería /^A/ a la que aplicamos el método test(elemento del array a comprobar).

El elemento del array a comprobar es pasado por parámetro mediante "=>"

UT4 Prácticas 1

UT4. Práctica 1.

Duración: 3 horas

Ejercicio 1:

Crea un array llamado **elementos** que tenga 10 datos de tipo **string** (tipo primitivo).

Sobre el array creado realiza las siguientes acciones:

1. Visualiza por pantalla todo el contenido del array, separando cada dato en líneas distintas.
2. Añade al array un dato más. (mediante el uso [longitud]
3. Añade al array dos datos más mediante utilizando un solo método.
4. Añade un dato más al principio del array.
5. Localiza un cierto dato dentro del array.
6. Elimina los últimos tres datos del array.
7. Crea un sub-array llamado **array_recortado** con los datos del array **elementos**, comprendidos entre la posición 4 y 8 (ambos inclusive).
8. Crea un nuevo array llamado **elementos_MYCLS** con los datos del array **elementos** en mayúsculas.

UT4. Proyecto Guiado

Lectura de ficheros en javascript.

- Es muy frecuente que los datos contenidos en un array hayan sido cargados desde un fichero (texto plano).
- En las primeras versiones de Javascript existía una limitación sobre el acceso a ficheros (no estaba permitido). En la actualidad, con HTM5 y la API File, es posible el uso de ficheros siempre que sea el usuario el que realice las acciones sobre ellos (por ejemplo: pulsando botones).

El siguiente vídeo describe detalladamente cómo realizar una página web con lectura de fichero:

https://youtu.be/3MG-fhHcNTM <https://youtu.be/3MG-fhHcNTM>

UT4. Práctica 1. Ejercicio 2.

1. Sigue los pasos del vídeo y genera el código necesario para cargar un fichero de texto.
2. A continuación se va a modificar el código para que en vez de aparecer en la página html el contenido del fichero, se cargue en un array llamado datos. Para ello, se debe añadir la declaración de la variable tipo array:
    - var datos = new Array( );
3. Y a continuación, se asigna el contenido de cada línea del fichero al array, teniendo en cuenta hay que dividir en partes todo el string de contenido, como la separación es línea a línea, se utiliza como delimitador '\n'-

- datos = contenido.split('\n');

4. Por último, visualizamos el array datos.

Arrays vinculados

Arrays vinculados

- Consiste en una técnica de programación por la cual se asocia información contenida en dos o más arrays a través de sus indices.
- Es labor del programador establecer la coherencia en la asociación de la información.

Ejemplo:

```
let bicicletas = ["colnago","trek","stevens","canion","orbea",];
let ciclistas = ["Abraham Olano","Alberto Contador","Wot Van Aert","Al

for (i=0;i<bicicletas.length;i++){
    console.log(`${ciclistas[i]} utiliza la marca de biclicetas: ${bic
}
```

- En el ejemplo los datos de los arrays quedan emparejados (vinculados) a través del índice, ya que se almacenan en los arrays de forma que la información esté coherente y asociada. El resultado es el siguiente:

```
Abraham Olano utiliza la marca: colnago
Alberto Contador utiliza la marca: trek
Wot Van Aert utiliza la marca: stevens
Alejandro Valverde utiliza la marca: canion
Samuel Sánchez utiliza la marca: orbea
```

Debes saber:

Cuando el lenguaje de programación permite manejar arrays de objetos, esta técnica no es tan recomendable, ya que con arrays de objetos podemos vincular información en cada objeto y tener todos los objetos almacenados en un único array

Arrays multidimensionales

Arrays multidimensionales.

- Los arrays multidimensionales son estructuras de datos que permiten almacenar información de forma ordenada en forma de "dimensiones".
- Se utilizan cuando se necesita almacenar múltiples valores del mismo tipo, donde cada dimensión representa secuencias de datos del mismo tipo.
- Ejemplo: Creación de array bidimensional.

```
var datos = new Array();
datos[0] = new Array("Cristina","Seguridad",24);
datos[1] = new Array("Catalina","Bases de Datos",17);
datos[2] = new Array("Vieites","Sistemas Informáticos",28);
datos[3] = new Array("Benjamin","Redes",26);

// O de otra forma:

var datos = [
          ["Cristina","Seguridad",24],
          ["Catalina","Bases de Datos",17],
          ["Vieites","Sistemas Informáticos",28],
          ["Benjamin","Redes",26]
      ];
```

- Para acceder a un dato en particular, de un array de arrays, se requiere que hagamos una doble referencia. La primera referencia, será a una posición del array principal, y la segunda referencia, a una posición del array almacenado dentro de esa casilla del array principal. Ésto se hará escribiendo el nombre del array, y entre corchetes cada una de las referencias: nombrearray[indice1][indice2][indice3] (nos permitiría acceder a una posición determinada en un array tridimensional).

```
document.write("<br/>Quien imparte Bases de Datos? "+datos[1][0
document.write("<br/>Asignatura de Vieites: "+datos[2][1]);
document.write("<br/>Alumnos de Benjamin: "+datos[3][2]);

// O de otra forma, mediante un bucle for:

for (i=0;i<datos.length;i++)
{
     document.write("<br/>"+datos[i][0]+" del módulo de "+datos
}
```

- Otro ejemplo, de cómo escribir los datos de un array bidimensional en una tabla html es el siguiente:

```
document.write("<table border=1>");
for (i=0;i<datos.length;i++)
{
     document.write("<tr>");
     for (j=0;j<datos[i].length;j++)
     {
          document.write("<td>"+datos[i][j]+"</td>");
     }
     document.write("</tr>");
}
document.write("</table>");
```

UT4 Prácticas 2

UT4. Práctica 2

Ejercicio 1:

Elección de delegado del aula.

Se desea programar una aplicación para elegir el delegado/subdelegado dentro de un aula. Para ello, se dispondrá de un fichero de texto llamado “alumnos_clase.txt”, con los nombres de todos los alumnos de una clase.

El programa tendrá las siguientes funcionalidades:

1. Cargar datos del fichero de alumnos de clase.
2. Decidir candidatos.
3. Registrar votos.
4. Contabilizar resultados: totales y porcentajes.

Las estructuras de datos que se deberán utilizar serán:

- Un array llamado “alumnos” con todos los alumnos del fichero “alumnos_clase.txt”
- Un array llamado “candidatos” con los alumnos que se consideren candidatos. A la vez, se creará otro array llamado “votos” asociado o vinculado al de “candidatos”, que servirá para llevar la contabilidad de votos de cada candidato.

Para el proceso de votación se presentará en la página la lista de candidatos, y se hará la acción de votar por cada alumno de la clase (incluido candidatos)

Ejercicio 2:

Selección de productos de venta:

- Crea un fichero de texto con la siguiente información:

```
<nombre_producto1>;<precio_producto_1>
<nombre_producto_2>;<precio_producto_2>
 ---
<nombre_producto_N>;<precio_producto_N>
```

- Mediante la API File carga la información en un array de objetos llamado productos.
- A continuación, visualiza y elige a través de un elemento html de selección desplegable el nombre de un producto, para que aparezca como resultado su precio.

Objetos definidos por el usuario

Objetos definidos por el usuario en Javascript.

- JavaScript no es un lenguaje orientado a objetos en sentido estricto. Se considera que, JavaScript es un lenguaje basado en objetos o prototipos. La diferencia entre orientado a objetos y basado en objetos o prototipos radica en cómo se crean los objetos, los primeros por instancias de clases y los segundos como creación directa de los mismos.
- Un **objeto** es una colección de datos relacionados y/o una funcionalidad propia (actúa sobre los datos), que generalmente consta de variables, llamadas propiedades y funciones denominadas métodos.
- Una propiedad es un par "key:value" (clave:valor), donde key es un string (también llamado "nombre clave"), y value puede ser cualquier elemento. .

- Podemos crear un objeto de distintas formas:

  1ª. Mediante la creación del "**objeto literal**" usando las llaves {...} y en interior una lista opcional de propiedades.

```
// Creación del objeto literal
let ciclista= {
  nombre: "Alejandro Valverde",
  edad: 41
};
```

2ª. Mediante el uso de un constructor de objeto.

```
function Ciclistas(nombre,edad){
  this.nombre = nombre;
  this.edad = edad;
}
var ciclista = new Ciclistas("Alejandro Valverde", 41);
console.log(ciclista.nombre , ciclista.edad);
```

```
var ciclista = new Object();
ciclista.nombre = "Alejandro Valverde";
ciclista.edad = 41;

console.log(ciclista.nombre , ciclista.edad);
// Devuelve Alejandro Valverde 41
```

3ª. Mediante el uso de clases: ES6 introdujo una mejora sintáctica sobre la herencia basada en propotipos creando "funciones especiales" llamadas clases. Esto no introduce el modelo de herencia orientada a objetos propio de la POO.
.

```
class Ciclistas{
    constructor(nombre, edad) {
      this.nombre= nombre;
      this.edad= edad;
    }
 }

 var ciclista = new Ciclistas("Alejandro Valverde", 41);
 console.log(ciclista.nombre , ciclista.edad);
```

Nota: Un nombre de función constructora o de una clase, por convenio, suelen comienzar con una letra mayúscula.

4ª Forma: Crear un objeto basándose en uno ya dado. Para ello se utiliza el método Object.create()

```
var ciclista_2 = Object.create(ciclista);
```

- La notación para referenciar las propiedades o métodos de los objetos puede ser:

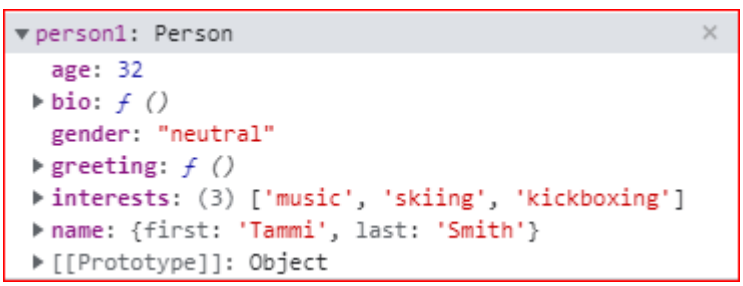
  A través de "." (notación .)
  A través de "[]" (notación []). Un aspecto útil de la notación de corchetes es que se puede usar para establecer dinámicamente no solo los valores de los miembros, sino también los nombres de los miembros.

```
var nueva_propiedad = "equipo";
var nuevo_valor = "Movistar";

ciclista[nueva_propiedad] = nuevo_valor;
console.log(ciclista['nombre'] , ciclista['edad'], ciclista['equi
// Devuelve Alejandro Valverde 41 Movistar
```

Prototipos de Objetos

- Los prototipos son un mecanismo mediante el cual los objetos en JavaScript heredan características entre sí.
- Los objetos pueden tener un objeto prototipo, el cual actúa como un objeto plantilla que hereda métodos y propiedades. La propiedad oculta [[Prototiype]], es la encargada de almacenar dicho objeto plantilla.
  - En el siguiente ejemplo se ha creado un objeto person1 a partir del constructor Person que por defecto hereda de Object.

```
let person1 = new Person('Tammi', 'Smith', 32, 'neutral', ['
```



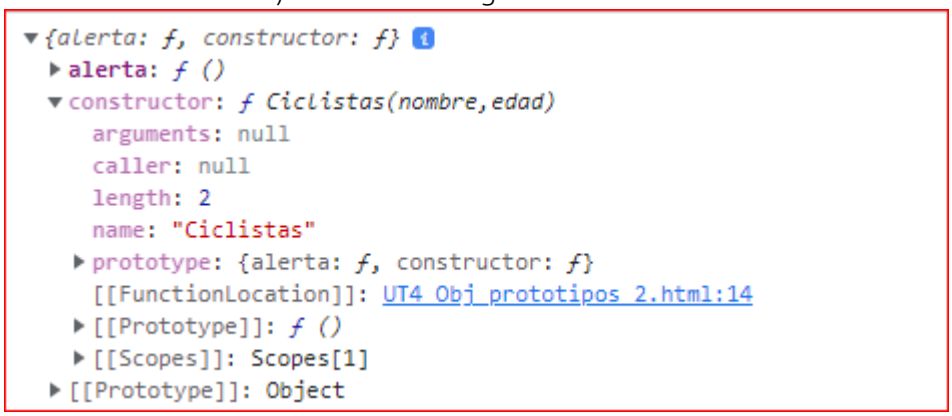
- Los prototipos van a permitir añadir métodos a los constructores existentes. Para ello se utiliza la propiedad: prototype.

```
function Ciclistas(nombre,edad){
        this.nombre = nombre;
        this.edad = edad;
}

      var ciclista = new Ciclistas("Alejandro Valverde", 41);
      Ciclistas.prototype.alerta = function() {
        alert(this.nombre + ' El objeto tiene una nueva propiedad
      };
```

- Para acceder al prototipo de un objeto se puede utilizar: Object.getPrototypeOf(obj) o la propiedad __proto__ del objeto.

```
// Añadiendo las siguiente líneas al ejemplo anterior:
      console.log(Object.getPrototypeOf(ciclista));
      console.log(ciclista.__proto__);
```

El resultado de los dos mensajes de console.log es el mismo:



UT4 Práctica 3

Actividad

Ejercicio 1:
Selección de productos de venta:

Crea un fichero de texto con la siguiente información:

```
<nombre_producto1>;<precio_producto_1>
<nombre_producto_2>;<precio_producto_2>
 ---
<nombre_producto_N>;<precio_producto_N>
```

Mediante la API File carga la información en un array de objetos llamado productos.
A continuación, visualiza y elige a través de un elemento html de selección desplegable el nombre de un producto, para que aparezca como resultado su precio.

Ejercicio 2:

Diseña una estructura de datos que almacene la información de alumnos necesaria para la votación de delegado.

Esta estructura deberá ser un array de objetos alumnos, con las propiedades y métodos necesarios para la aplicación de votación de delegado.

Ejercicio 3:

Sobre el ejercicio de elección de delegado del aula, incorpora la estructura de datos diseñada en el ejercicio anterior para sustituir los arrays vinculados. Este ejercicio trabajará directamente con el array de objetos alumnos.

Bibliografía y recursos

Tutoriales y Referencias de Javascript: <https://www.w3schools.com/js/default.asp>

- h <https://www.w3schools.com/js/default.asp> ttps://www.w3schools.com/js/default.asp <https://www.w3schools.com/js/default.asp>
- https://developer.mozilla.org/es/docs/Learn/JavaScript <https://developer.mozilla.org/es/docs/Learn/JavaScript>
- https://es.javascript.info/ <https://es.javascript.info/>

Expresiones Regulares:

- https://regex101.com/ <https://regex101.com/>

Apuntes cedidos por el Ministerio de Educación y Formación Profesional.