



# Curso de Front-End

## Clase 14



# Agenda de la clase



# Agenda

- Antes del repaso...
- Repaso.
- Objetos.
- Ejercicios.



# Antes del repaso...

¿De qué deberían ser capaces a esta altura del curso?

# A esta altura deberían poder...

¿Hay algo que les gustaría repasar?



- Usar las etiquetas **HTML** más importantes → Para definir el **QUÉ**.  
`h1, h2, p, ul, li, div, span, form, input, select, etc.`
- Usar las propiedades **CSS** más importantes → Para definir el **CÓMO**.  
`font-size, color, background-color, margin, padding, position, display, etc.`
- Usar, en **JavaScript**, variables, condicionales (`if/else`), *loops* (`for/while`), **funciones** y *arrays*.
- Manipular el **DOM**.
- Usar frameworks o librerías como **Bootstrap**.
- Usar Google y **Stack Overflow** para consultar dudas.
- Leer una **documentación**.
- Indentar su código y usar espacios de forma correcta.  
Ayúdense con herramientas como **Prettier**.  
Recuerden que un código mal indentado es mucho más difícil de leer, mantener y los hace parecer “amateurs”.  
Además, existen lenguajes en los que un código mal indentado ni siquiera funciona.
- Usar la **consola** para detectar errores en su código (*debug / debuggear*).



# Repaso



# ¿Qué es un evento?

- Es “algo” que sucede mientras un usuario interactúa con una página web.
- Ejemplos de eventos:
  - El usuario presiona una tecla.
  - El usuario hace click en un botón.
  - El usuario agranda o achica una ventana.
  - Una página se terminó de cargar.
- Los eventos se detectan con la función `addEventListener` de JavaScript.
- Documentación: <https://developer.mozilla.org/en-US/docs/Web/Events>.



# Eventos más usados

En realidad depende de cada aplicación, pero en general los más usados son:

- `click`
- `submit`
- `change`
- `input`
- `load`
- `keydown`
- `scroll`





# Detectar un evento con JavaScript (1/2)

La función [addEventListener](#) permite **detectar eventos** que suceden sobre un elemento cualquiera de la página.

```
var unElemento = document.querySelector("#elem");  
  
unElemento.addEventListener("nombreDeUnEvento", nombreDeUnaFunción);
```

👉 El segundo parámetro de la función `addEventListener` (llamada *callback*) también se podría haber declarado como una **función anónima**.



Y antes de pasar al siguiente tema (“objetos”), repasemos qué es un *array*



# ¿Qué es un *Array*? (en JavaScript)

- Es una **estructura de datos** que permite almacenar un **conjunto** de valores en formato lista. Se lo puede pensar como una “lista de elementos”.
- En lugar de manipular los valores por separado, se manipula el conjunto.
- El largo de los *arrays* es variable y pueden contener elementos repetidos.
- Los valores contenidos en un *array* no tienen por qué ser del mismo tipo. Pueden ser *strings*, *numbers*, *booleans*, funciones, otros *arrays*, etc (todo mezclado). De todas maneras, lo más usual es que los elementos de un *array* sean del mismo tipo.
- En español se les llama “Arreglos” (o incluso “Vectores”).

Nota técnica: un *array* en JS es un *objeto*. Lo veremos más adelante. Ver [más información](#).



# Objetos



# Prueben en la consola...

```
var usuarios = ["María", "José", "Juan", "Martín", "Lucía"];  
typeof(usuarios); // ¿Que retorna la función typeof?
```

¿De qué tipo es la variable `usuarios`? ¿Es de tipo `Array`? ¿`Array` es un tipo de datos?



# Problema: ¿Cómo guardar datos de una persona? (1/3)

Supongamos que necesitamos guardar los datos de una persona como, por ejemplo, nombre, apellido, edad, email y lista de nacionalidades. ¿Cómo lo harían?

Opción 1:

```
var nombre = "María";  
var apellido = "Rodríguez";  
var edad = 36;  
var email = "maria.rod@gmail.com";  
var nacionalidades = ["Uruguay", "España"];
```

Mmm... ¿y si quisiésemos guardar los datos de otra persona?



## Problema: ¿Cómo guardar datos de una persona? (2/3)

Opción 2:

```
var persona1 = [  
    "María", "Rodríguez", 36, "maria.rod@gmail.com", ["Uruguay", "España"]  
];  
  
var persona2 = [  
    "José", "Pérez", 40, "jperez@hotmail.com", ["Uruguay"]  
];
```

Mmm... ¿y ahora cómo sabemos que lo que está en el índice 1 corresponde al apellido y lo que está en el índice 4 corresponde a la nacionalidad? ¿Cómo sabemos que 36 y 40 son las edades y no el talla del pie?

Precisaríamos una especie de diccionario que indique a qué corresponde cada posición del *array*.



# Problema: ¿Cómo guardar datos de una persona? (3/3)

Opción 3: ¡usando **objetos**!

```
var persona1 = {  
  nombre: "María",  
  apellido: "Rodríguez",  
  edad: 36,  
  email: "maria.rod@gmail.com",  
  nacionalidad: ["Uruguay", "España"]  
};
```





# ¿Qué es un objeto? (JavaScript)

Un objeto es una **entidad** que contiene una **colección de propiedades**.

Una propiedad es una asociación de clave-valor.

Las propiedades son similares a las variables comunes de JavaScript, la diferencia es que las propiedades son variables que están unidas a un determinado objeto.

El valor de una propiedad puede ser una función y en ese caso a la propiedad se le llama **método**.

```
var nombreDelObjeto = {  
  clave1: valor1,  
  clave2: valor2,  
  clave3: valor3  
};
```

Propiedades. (Cada propiedad se separa con una coma ",").


Las claves y valores se separan con dos puntos ":".



# El concepto de *objeto* es similar al de la vida real...

Ejemplo: un **auto**. Básicamente es un objeto que tiene un montón de propiedades como: color, peso, marca, modelo, velocidad máxima, kilometraje, etc.

```
var auto = {  
  marca: "VW",  
  color: "gris",  
  peso: 1200,  
  velmax: 220,  
  kilometraje: 45000  
};
```



Propiedades



# Acceder a una propiedad de un objeto

Dado el objeto llamado `persona1`:

```
var persona1 = {  
  nombre: "María",  
  apellido: "Rodríguez",  
  edad: 36,  
  email: "maria.rod@gmail.com",  
  nacionalidad: ["Uruguay", "España"]  
};
```

En este caso, el valor de la propiedad es un *array*. De hecho, las propiedades pueden tener cualquier tipo de valor, incluso otros objetos.

Se accede sus propiedades de esta forma:

```
persona1.apellido; // Devuelve "Rodríguez".  
persona1.edad; // Devuelve 36.
```



# Entonces... ¿los *arrays* son objetos? (1/2)

Sí. Se podría decir que los *arrays* son un caso particular de objetos en los que las claves de las propiedades son números naturales (que empiezan en cero).

Ejemplo:

```
var usuarios = {  
  0: "María",  
  1: "José",  
  2: "Juan",  
  3: "Martín",  
  4: "Lucía"  
};
```

```
usuarios[2]; // Retorna el string "Juan".
```



## Entonces... ¿los *arrays* son objetos? (2/2)

Visto de otra forma, los objetos son *arrays* donde en lugar de estar obligados a usar números naturales como índices podemos crear nuestros propios índices.

Entonces, también es posible acceder a las propiedades de un objeto de esta forma:

```
persona1["apellido"]; // Devuelve "Rodríguez". Esta notación se llama "Bracket Notation".
```

Equivalente a:

```
persona1.apellido; // Devuelve "Rodríguez". Esta notación se llama "Dot Notation".
```

Nota: A veces, a los objetos en JavaScript se les llama “*Arrays Asociativos*” (o incluso “*Hash Maps*”).



# Asignar una propiedad a un objeto

Visto de otra forma, los objetos son *arrays* donde en lugar de estar obligados a usar números naturales como índices podemos crear nuestros propios índices.

Entonces, también es posible acceder a las propiedades de un objeto de esta forma:

```
persona1["apellido"] = "Pérez";  
persona1.apellido = "Pérez";
```



# Métodos de un Objeto

Cuando una propiedad recibe como valor una función, se le llama método.

```
var persona1 = {  
  nombre: "María",  
  apellido: "Rodríguez",  
  edad: 36,  
  email: "maria.rod@gmail.com",  
  nacionalidad: ["Uruguay", "España"],  
  nombreCompleto: function () {  
    return this.nombre + " " + this.apellido;  
  },  
};
```

La palabra `this` es una palabra reservada del lenguaje. En JavaScript tiene un comportamiento muy especial, y por lo tanto se recomienda leer estos [docs de MDN](#).

```
persona1.nombreCompleto(); // Retorna "María Rodríguez"
```



# Para pensar... (casi) todo en JS es un objeto.

- Los objetos son objetos (obviamente).
- Los arrays son objetos.
- Las funciones son objetos.
- `string`, `number`, `boolean`, `undefined`, `null` no son objetos, se le llama primitivas.

Pero... prueben lo siguiente:

```
var dos = 2;           // Variable de tipo number.
var hola = "Hola";     // Variable de tipo string.
dos.toString();        // ¿Se llama a un método de un number?
hola.length;           // ¿Se llama a una propiedad de un string?
```

Lo que sucede es que JavaScript genera, en tiempo real, un objeto *wrapper* que contiene el valor. En este caso se genera un objeto `Number` y `String` (en mayúscula):

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Number](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number)

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/String](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String)





# Ejercicio 1



# Ejercicio 1 – Curriculum Vitae

Supongamos que los desarrolladores de Back-End nos enviaron un **objeto** JavaScript que contiene información sobre una persona.

Descargar este [archivo JavaScript](#) (`person.js`) con dicho objeto.

Ustedes deberán **inventar** el diseño de un Curriculum Vitae Online y mostrar los datos recibidos.

Pueden inspirarse en otros sitios. Ver ejemplos [aquí](#), [aquí](#) y [aquí](#).

Partir del proyecto base (.zip), **ya** compartido por Teams.

```
const person = {
  firstname: "María",
  lastname: "Rodríguez",
  birth_date: "1983-04-19",
  email: "maria.rod@gmail.com",
  nationality: ["Uruguay", "España"],
  image: "https://thispersondoesnotexist.com/image",
  education: [
    {
      institution: "Hack Academy",
      course: "Front-End",
      year: 2017,
    },
    {
      institution: "Hack Academy",
      course: "Back-End (PHP)",
      year: 2018,
    },
  ],
  jobs: [
    {
      company: "ACME S.A.",
      title: "Junior Developer",
      year_from: 2018,
      year_to: 2019,
    },
  ],
}
};
```