JAVASCRIPT

TIPOS DE DATOS PRIMITIVOS COMPUESTOS

TIPOS DE DATOS PRIMITIVOS COMPUESTOS

- JavaScript trata a ambos como objetos:
  - Arrays
  - Objetos
- Objetos, propiedades y métodos de la vida real.
- Por ejemplo: en la vida real, un coche es un objeto. Un automóvil tiene **propiedades** como peso y color, y **métodos** como arrancar y detener:
  - Todos los coches tienes las mismas **propiedades**, pero el valor de la propiedad cambia para cada coche.
  - Todos los coches tienen los mismo **métodos**, pero los métodos son ejecutados en diferentes momentos.

OBJETOS

- Ya conocemos las variables y sabemos que son contenedores para valores de datos.

```
let car = "Fiat";
```

- Los **objetos** también son variables, pero pueden contener muchos valores.
- El siguiente código asigna **muchos valores** (Fiat, 500, blanco) a **una variable llamada coche**:

```
const car = {tipo:"Fiat", modelo:"500", color:“blanco"};
```

- Los valores son escritos como parejas de **nombre:valor**, separadas por comas ,.
- Es una práctica común declara objetos con la palabra reservada **const**.

OBJETOS

- ¿Cuándo usar const?
  - Siempre que sepamos que el valor no va a cambiar.
  - Se usa const cuando se declara:
    - Un nuevo array.
    - Un objeto nuevo.
    - Una nueva función.
    - Una nueva RegExp (expresión regular).
- Objetos constantes y arrays constantes:
  - La palabra reservada *const* puede ser un poco engañosa:
    - No define a un valor constante sino una referencia constante a un valor.
    - Por esto, NO se puede:
      - Reasignar un valor constante.
      - Reasignar un array constante.
      - Reasignar un objeto constante.
    - Pero SÍ se puede:
      - Cambiar los elementos de un array constante.
      - Cambiar las propiedades de un objeto constante.

OBJETOS
USO DE *CONST*

- Array constante

```
//Se pueden cambiar los elementos de un array constante:
const cars = ["Saab", "Volvo", "BMW"];
cars[0] = "Toyota";
cars.push("Audi"); //se pueden añadir elementos

//No se puede reasignar el array:
const cars = ["Saab", "Volvo", "BMW"];
cars = ["Toyota", "Volvo", "Audi"];    // ERROR
```

OBJETOS
USO DE *CONST*

- Objeto constante

```
//Se pueden cambiar las propiedades de un objeto constante:
const car = {type:"Fiat", model:"500", color:"white"};
car.color = "red";
car.owner = "Johnson"; // Se pueden añadir propiedades

//No se puede reasignar un objeto:
const car = {type:"Fiat", model:"500", color:"white"};
car = {type:"Volvo", model:"EX60", color:"red"};    // ERROR
```

OBJETOS

- Definición:

```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue“
};
```

- Cada pareja nombre:valor se llama propiedad:

Propiedad	Valor
firstName	John
lastName	Doe
age	50
eyeColor	blue

OBJETOS

- Acceder a las propiedades de un objeto:
  1. nombreObjeto.nombrePropiedad
  2. nombreObjeto[“nombrePropiedad"]

```
person.lastName;
person["lastName"];
```

- Los objetos en JavaScript son contenedores de valores con nombre, denominados propiedades.
- Métodos de los Objetos
  - Los objetos pueden tener métodos.
  - Los métodos son acciones que se pueden realizar sobre objetos.
  - Los métodos se almacenan en propiedades, como definiciones de funciones. Un método es una función almacenada como una propiedad.

OBJETOS

```
const person = {
   firstName: "John",
   lastName : "Doe",
   id       : 5566,
   fullName : function() {
          return this.firstName + " " + this.lastName;
   }
};
```

- Acceder a los métodos de un objeto:
  1. objectName.methodName()
  2. name = person.fullName();
- Si intentas accede a un método SIN (), te devolverá la definición de la función:

```
name = person.fullName;
```

OBJETOS

- En el ejemplo, ***this*** se refiere al objeto ***person***:
  - this.firstName significa la propiedad ***firstName*** de ***person***.
  - this.lastName significa la propiedad ***lastName*** de ***person***.
- ¿Qué es ***this***?
  - La palabra reservada ***this*** se refiere a un objeto.
  - Qué objeto depende de cómo sea invocado, usado o llamado ***this***.
  - Puede referirse a objetos diferentes:

THIS
En el método de un objeto, ***this*** es el objeto.
Solo, ***this*** es el ***objeto global***.
En una función, ***this*** es el objeto ***global***.
En un evento, ***this*** es el element que es recibido por el evento.