



*Instantiva*

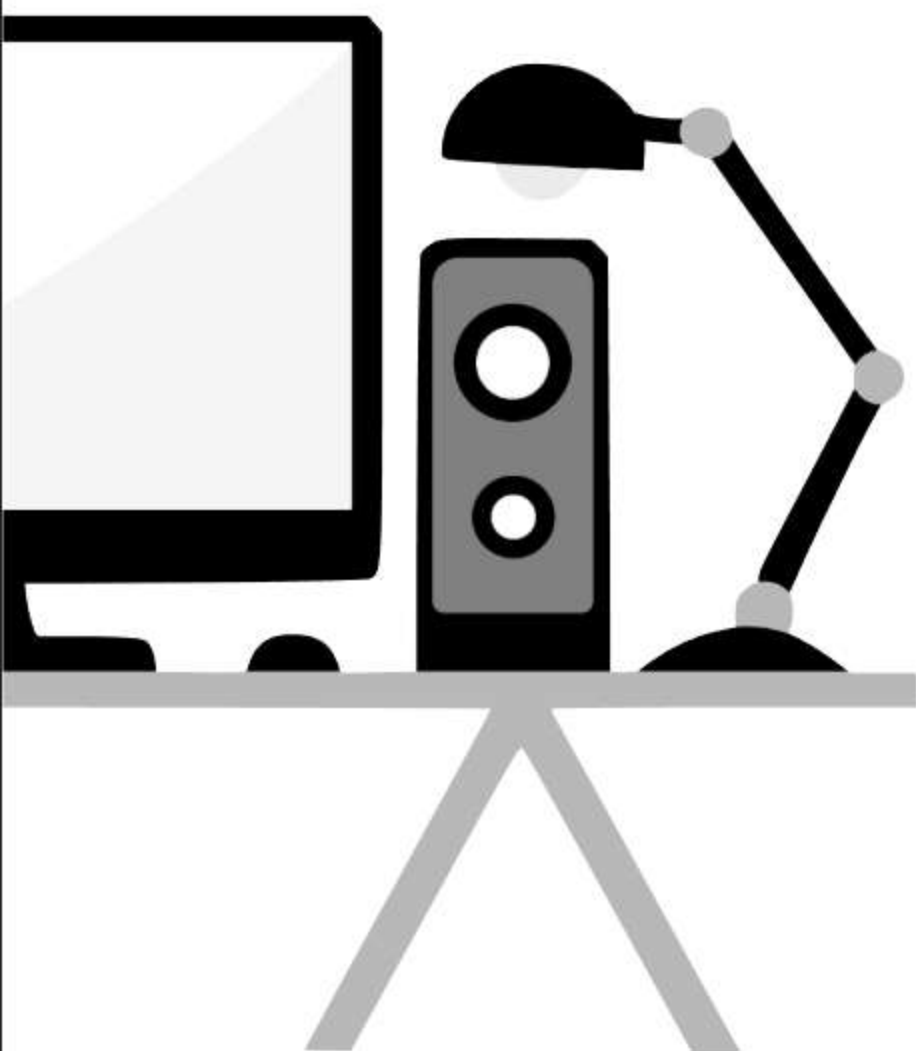
CON • SENTIDO

# OBJETOS

Lección 05

Utilizar objetos preconstruidos para la codificación de un algoritmo que resuelve un problema acorde al lenguaje JavaScript.





## **Objetos**

1. ¿Qué es un objeto?
2. Elementos de un objeto
3. Notación punto
4. Notación corchete
5. Métodos de un objeto
6. Objetos preconstruidos en JavaScript

## **Objeto Math**

1. ¿Qué es un Math? Y para qué sirve
2. Propiedades del objeto Math
3. Métodos del objeto Math

## **Objeto String**

1. ¿Qué es un String? Y para qué sirve
2. Creación de un objeto String versus uno primitivo
3. Métodos del objeto String



# ¿Qué es un objeto?

El concepto de objeto en programación se puede asimilar al concepto de objeto en la vida real. Para una mejor comprensión un objeto se puede entender como algo tangible en la vida real.

En JavaScript un objeto es una entidad independiente con propiedades y tipos

Por ejemplo una taza es un objeto que tiene como propiedades: su color, su diseño, su peso, material, etc.



Color: “Negro”  
Diseño: “Logo NodeJS”  
Peso: 100g  
Material: “Loza”

# Elementos de un objeto

Un objeto se crea de manera literal identificando cada propiedad de este entre llaves { }, las propiedades y métodos del objeto se asignan separadas por coma.

En JavaScript los objetos son dinámicos, esto quiere decir que sus propiedades no tienen por que ser definidas en el momento de instanciar el objeto, por ende sus propiedades se pueden añadir en otra instancia de ejecución del programa.

```
// Ejemplo de un objeto
let objeto = {
  "propiedad1" : "prueba",
  "propiedad2" : function() { console.log("¡Hola Mundo!"); }
};
```



Una vez instanciado el objeto podemos acceder a cada propiedad llamando a la variable del objeto seguido de un punto y el nombre de la propiedad.

De esta misma forma podemos agregar nuevas propiedades al objeto o editar las propiedades actuales del objeto.

```
// Ejemplo de un objeto
let objeto = {
  "propiedad1" : "prueba",
  "propiedad2" : function() { console.log("¡Hola Mundo!"); }
};

console.log(objeto.propiedad1); // Muestra "prueba"

// Agregando una nueva propiedad
objeto.propiedad3 = "este es el elemento 3";

console.log(objeto.elemento3); // Muestra "este es el elemento 3"
```

De manera similar a la notación punto podemos realizar el mismo procedimiento de agregar o editar una propiedad con la notación corchete.

```
// Ejemplo de un objeto
let objeto = {
  "propiedad1" : "prueba",
  "propiedad2" : function() { console.log("¡Hola Mundo!"); }
};

console.log(objeto["propiedad1"]);
// Muestra "prueba"

objeto["propiedad3"] = "este es el elemento 3";

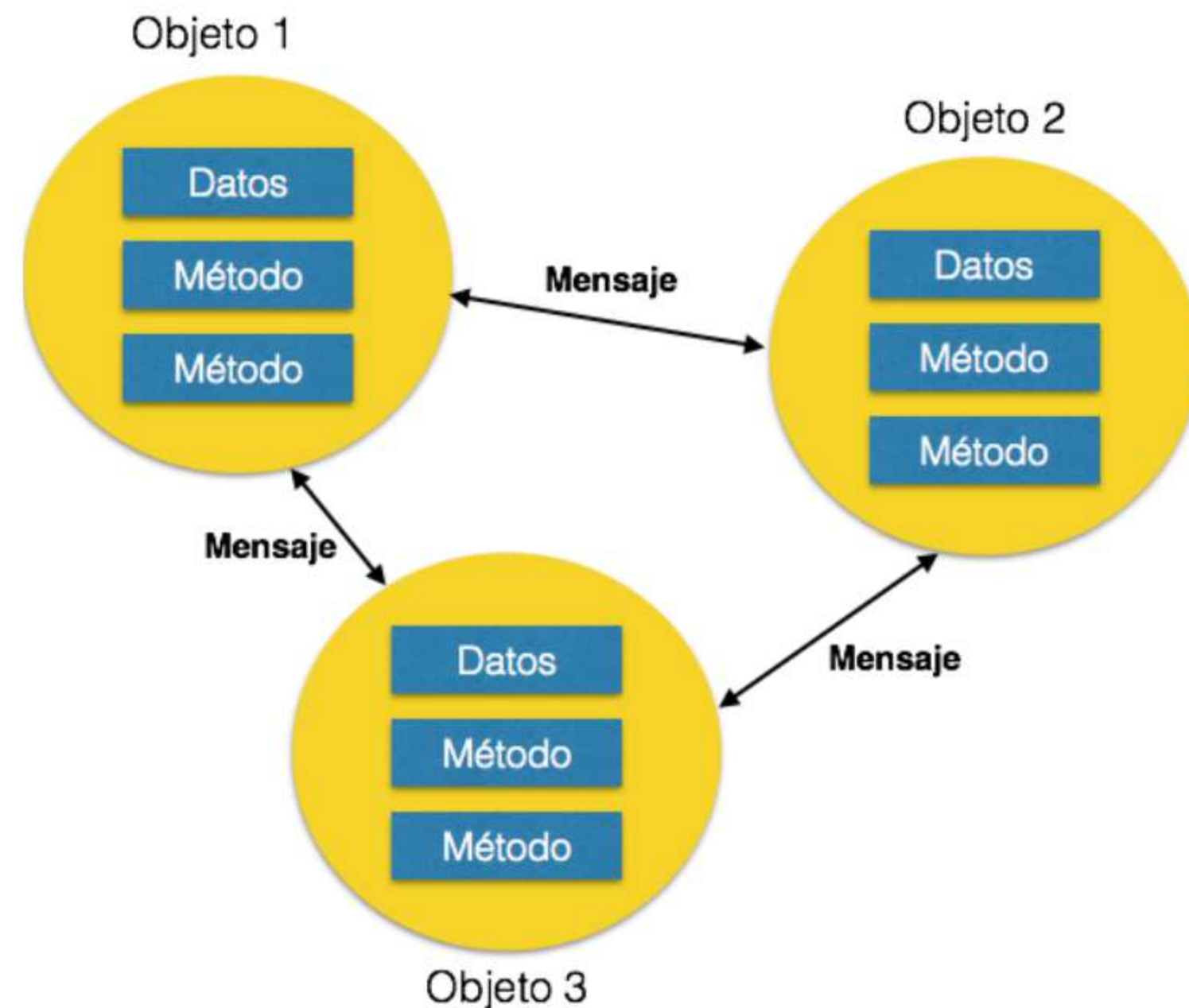
console.log(objeto["elemento3"]); // Muestra "este es el elemento 3"
```



# Métodos de un objeto

Todos los objetos en JavaScript derivan de la primitiva **Object**, Esta primitiva cuenta con una serie de métodos ya incorporados para simplificar el trabajo con objetos, estos métodos son

***Object.create()***  
***Object.keys()***  
***Object.values()***  
***Object.entries()***  
***Object.assign()***  
***Object.freeze()***  
***Object.seal()***





## ***Object.create()***

Se utiliza para crear un nuevo objeto y vincular este objeto a un objeto ya creado.

```
// Ejemplo de un objeto
const trabajo = {
  cargo: "Programador",
  tipo: "Por hora",
  jornada: "de Lunes a viernes",
  detalles() {
    console.log(`el cargo de ${this.cargo} se trabaja ${this.tipo}
    y su jornada es ${this.jornada}.`);
  }
};

const arquitecto = Object.create(trabajo);
arquitecto.cargo = "Arquitecto";
arquitecto.detalles(); // Muestra el cargo de Arquitecto se trabaja
Por hora y su jornada es de Lunes a viernes.
```

# Métodos de un objeto

## ***Object.keys()***

Crea una arreglo que contiene todos los nombres de las propiedades y atributos de un objeto.

```
// Ejemplo de un objeto
const trabajo = {
  cargo: "Programador",
  tipo: "Por hora",
  jornada: "de Lunes a viernes",
  detalles() {
    console.log(`el cargo de ${this.cargo} se trabaja ${this.tipo}
    y su jornada es ${this.jornada}.`);
  }
};

console.log(Object.keys(trabajo));
// Muestra [ "cargo", "tipo", "jornada", "detalles" ]
```



## ***Object.values()***

Crea una arreglo que contiene todos los valores de cada atributo en el objeto

```
// Ejemplo de un objeto
const trabajo = {
  cargo: "Programador",
  tipo: "Por hora",
  jornada: "de Lunes a viernes",
  detalles() {
    console.log(`el cargo de ${this.cargo} se trabaja ${this.tipo}
y su jornada es ${this.jornada}.`);
  }
};

console.log(Object.values(trabajo));
// [ "Programador", "Por hora", "de Lunes a viernes", f detalles{...}]
```

## ***Object.entries()***

Crea una matriz que asocia al nombre de la propiedad con su valor

```
// Ejemplo de un objeto
const trabajo = {
  cargo: "Programador",
  tipo: "Por hora",
  jornada: "de Lunes a viernes",
  detalles() {
    console.log(`el cargo de ${this.cargo} se trabaja ${this.tipo}
y su jornada es ${this.jornada}.`);
  }
};

console.log(Object.entries(trabajo));
// muestra [ "cargo", "Programador"], [ "tipo", "Por hora"], [
"jornada", "de Lunes a viernes"], [ "detalles", f detalles {...}] ]
```



## ***Object.assign()***

Permite fusionar dos objetos en un tercer objeto con las propiedades y atributos de ambos objetos.

```
// Ejemplo de un objeto
const nombre = {
  Nombre: 'Philip',
  Apellido: 'Fry'
};

const trabajo = {
  Empresa: 'Delivery Boy',
  Cargo: 'Planet Express'
};

let persona = Object.assign(nombre, trabajo);

console.log(persona);
// muestra { Nombre:"Philip", Apellido:"Fry",Empresa:"Delivery Boy",
Cargo:"Planet Express" }
```

## ***Object.freeze()***

Impide que se modifique propiedades o valores de un objeto.

```
// Ejemplo de un objeto
const nombre = {
  Nombre: 'Philip',
  Apellido: 'Fry'
};
let persona = Object.freeze(nombre);
persona.Nombre = "Raul";
console.log(persona);
// muestra error: Uncaught TypeError: Cannot assign to read only
property 'Nombre' of object '#<Object>'
```



## ***Object.seal()***

Impide que se añadan nuevas propiedades al objeto, pero permite su modificación.


```
// Ejemplo de un objeto
const nombre = {
  Nombre: 'Philip',
  Apellido: 'Fry'
};
let persona = Object.seal(nombre);
persona.Nombre = "Raul";
console.log(persona);
// muestra {Nombre:"Raul", Apellido:"Fry" }

persona.Auto = "BMW";
// muestra error: Uncaught TypeError: Cannot add property Auto, object
is not extensible
```

# Objetos pre-construidos

JavaScript cuenta con una serie de objeto ya contruidos que nos ayudan a tratar algunos casos especiales, estos objetos son:

1. Math
2. String
3. Data
4. Number
5. RegExp



```
Math.floor( 45.95); // 45
Math.floor( 45.05); // 45
Math.floor(  4   ); //  4
Math.floor(-45.05); // -46
Math.floor(-45.95); // -46
```



El objeto Math incluye una serie de propiedades y constantes matemáticas, este objeto no se puede editar y nos ayuda en cálculos complejos o funciones matemáticas y trigonometría.

```
// properties
console.log(Math.PI);
console.log(Math.E);
console.log(Math.LN10);
// methods
console.log(Math.sin(0)); // 0
console.log(Math.round(6.343)); // 6
console.log(Math.random());
console.log(Math.min(34,4));
```

# Propiedades del objeto Math

Las propiedades del objeto Math son las siguientes:

Propiedad	Descripción
Math.E	Constante de Euler, aprox: 2,718
Math.LN2	Ln de 2, aprox: 0,693
Math.LN10	Ln 10, aprox: 2,303
Math.PI	Valor de $\pi$ (pi), aprox: 3.1416



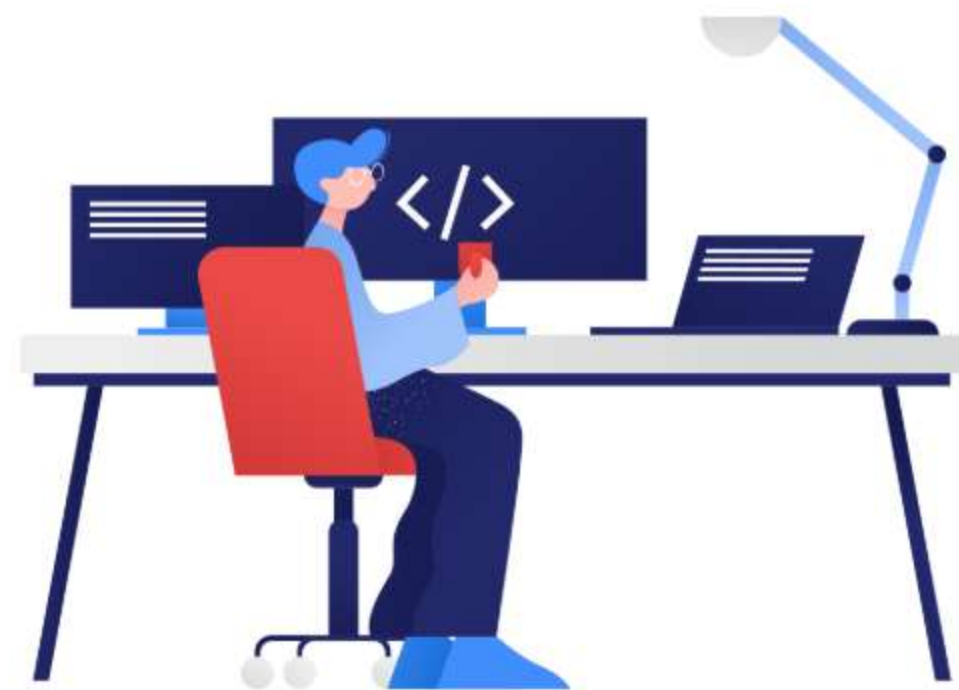
Algunos de los métodos del objeto Math son:

Métodos	Descripción
Math.abs(x)	Devuelve el valor absoluto de X
Math.cos(x)	Devuelve el coseno de X (en radianes)
Math.log(x)	Devuelve el logaritmo natural (base e) de X
Math.pow(x, y)	Devuelve X elevado a la potencia de Y
Math.random()	Retorna un numero aleatorio entre 0 y 1
Math.sign(x)	Retorna 1 si el x es positivo o -1 si el x es negativo.
Mat.round(x)	Redondea x al numero entero mas cercano

# Objeto String

El objeto **String** permite realizar operaciones y manipulaciones sobre cadenas de texto. Cuando se declara una cadena como una primitiva (por ejemplo: `let saludo = "Hola";`), JavaScript convierte automáticamente esa primitiva en un objeto String de forma temporal para permitir el uso de sus propiedades y métodos.

Esto significa que incluso las cadenas primitivas pueden acceder a métodos del objeto String, como `.length`, `.toUpperCase()`, `.slice()`, entre otros, gracias a la conversión automática que realiza el motor de JavaScript.





# Objeto String

Algunas de las propiedades y métodos que implementa el objeto String son las siguientes:

Propiedad	Descripción
String.length	Retorna la longitud del texto

Método	Descripción
char.At(i)	Retorna el carácter del string en la posición i
Replace(buscar,nuevo)	Remplaza el texto buscado por un nuevo string
Split(separador)	Convierte el string en un array indicando el separador
toLowerCase()	Pone todo el string a minúsculas
toUpperCase()	Pone todo el string a mayúsculas

# Ejercicio práctico

Un teléfono celular cuenta con un identificador de llamadas, esto permite que al ingresar una llamada el propietario sepa quién llamo si ese número se encuentra en su agenda.

Realizar un programa que imite esta funcionalidad, dado un teléfono, buscar este una estructura de objetos y muestre a quien pertenece dicho número.







*Instantiva*

CON • SENTIDO