

Objetos

¿Qué son los objetos?

De la realidad al código

Al examinar nuestro entorno, es evidente la presencia de una vasta cantidad de **objetos que constituyen nuestra vida diaria**, tales como automóviles, libros, computadoras y teléfonos móviles, entre otros.

En la década del 70, un grupo de visionarios en Xerox PARC, liderados por Alan Kay, imaginaron **simular estos objetos del mundo real como estructuras**, dentro de un lenguaje de programación ¡Como en la película Matrix!

Inspirados por la idea de una realidad simulada, **sentaron las bases del concepto de objetos** en programación, pilar fundamental de la Programación Orientada a Objetos (POO).



Objeto (físico o virtual): puede ser analizado en dos aspectos esenciales:		Ejemplo: Automóvil
Estado		Estado del objeto (características)
Representa las características de un objeto en un momento determinado.		<ul style="list-style-type: none">○ Es un Toyota.○ Del modelo Corolla.○ Va a 100 kilómetros por hora.○ ...
Comportamiento		Qué se puede hacer con el automóvil
Son las acciones o funciones que un objeto puede realizar o que se pueden realizar con él.		<ul style="list-style-type: none">○ Se puede acelerar.○ Puede frenar si va muy rápido.○ Girar y ajustar su dirección.○ Puede transportar pasajeros.

Las características definen **qué objeto es**. El comportamiento define **qué puede hacer el objeto**.

Estado

El **estado del objeto** es como una fotografía instantánea que describe el **conjunto de atributos** que tiene el objeto en un instante dado.

Cada atributo tiene un **nombre** y un **valor**.

Por ejemplo:

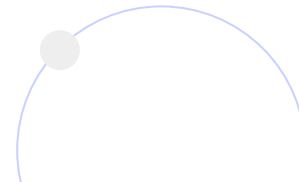
- altura: 1,95m.
- color: rojo.

Al intentar **representar un objeto** en un sistema informático se deben seleccionar los **atributos del mundo real** que resultan relevantes para el sistema específico.

Ejemplo:

No es lo mismo el estado de un auto para un sistema de carreras, que para uno de una concesionaria.

- **En el sistema de carreras**, por ejemplo, interesa saber la velocidad, la posición en la pista.
- **En el sistema de la concesionaria**, el modelo, el año, el kilometraje y el precio de venta tienen mayor relevancia.



JSON

JavaScript tiene un **formato para representar a un objeto y a su estado** llamado: *JavaScript Object Notation (JSON)*.

- Actualmente, es el **formato estándar** para representar la información y transmitirla entre aplicaciones.
- JSON es un **formato de intercambio de datos ligero y basado en texto**.

```
// Objeto JavaScript
const persona = {
  nombre: "Juan Pérez",
  edad: 30,
  profesion: "Desarrollador Web"
};
```

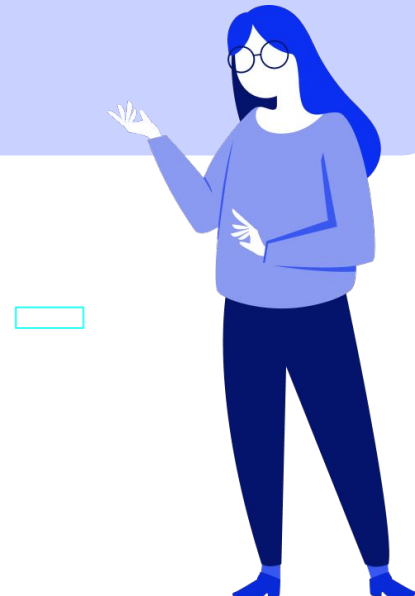


Veámoslo desde otro enfoque

En un programa extenso, **la cantidad de variables** puede crecer exponencialmente. Sin una organización adecuada, estas variables pueden volverse **difíciles de gestionar**. El código se vuelve desordenado. Se dificulta su lectura, modificación y depuración.

Es posible **agrupar variables relacionadas en objetos**. De esta manera, las variables viajan siempre **juntas en un mismo objeto** y se evita el riesgo de manejarlas por separado.

El **estado del objeto** es un conjunto de variables que, en la POO, se llaman **atributos**.



Ejemplo:

```
// Sin objetos
const nombre = "Juan Pérez";
const edad = 30;
const profesion = "Desarrollador Web";
console.log(`Hola me llamo ${nombre},
            soy ${profesion}
            y tengo ${edad} años.`);
```

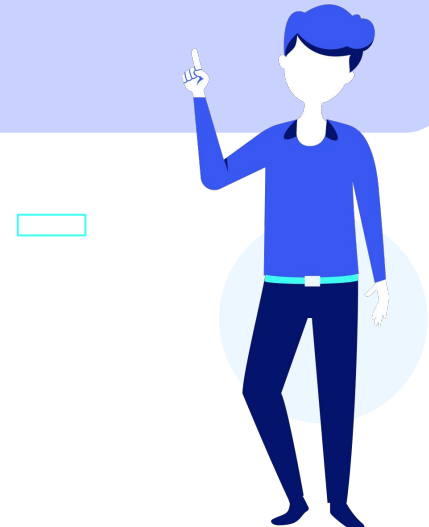
```
// Objeto persona
const persona = {
  nombre: "Juan Pérez",
  edad: 30,
  profesion: "Desarrollador Web"
};
console.log(`Hola, me llamo ${persona.nombre},
            soy ${persona.profesion}
            y tengo ${persona.edad} años.`);
```

Comportamiento

Si se focaliza únicamente en el estado, se pierde una parte fundamental de la POO: la **capacidad de los objetos para realizar acciones y colaborar**: su comportamiento.

- Sin comportamiento, solo se modelan **datos estáticos**, es decir, se pierde la esencia de la POO.
- El comportamiento es frecuentemente subestimado, pero es lo más **difícil de definir** correctamente.

En lenguajes de programación, el **comportamiento** se implementa a través de un **conjunto de métodos**.



Ejemplo del objeto automóvil

Ahora se le agrega su comportamiento.

Acciones:

- Acelerar.
- Frenar.

Otros posibles comportamientos que se podrían agregar:

- Girar y ajustar su dirección.
- Transportar pasajeros.

```
let automovil = {  
  // Estado del objeto  
  velocidad: 0,  
  //...  
  
  // Comportamiento del objeto  
  acelerar: function(aumento) {  
    this.velocidad += aumento;  
  },  
  
  frenar: function(disminucion) {  
    this.velocidad -= disminucion;  
    if (this.velocidad < 0) this.velocidad = 0;  
  }  
};
```

- Los métodos (del comportamiento) tienen **conocimiento del estado actual** del objeto.
- Actúan **directamente sobre su estado**, actualizándolo según sea necesario.
- Observar la **palabra reservada this** que se utiliza para hacer referencia al **estado del propio objeto**.
- **La interacción entre el comportamiento y el estado de un objeto es fundamental** para el diseño y la funcionalidad coherente de una aplicación orientada a objetos.



Jugar con objetos

Se utilizarán **objetos** y **ChatGPT** para representar una batalla entre **Superman** y **Omni-Man**.

Prompt:

“Crea un objeto JavaScript que represente a Superman y otro que represente a Omni-Man. Quisiera que puedan batallar entre sí. Me gustaría que me muestre como quedarían los objetos y un ejemplo de ataque de uno a otro.”



Crea un objeto JavaScript que represente a Superman y otro que represente a Omni-Man. Quisiera que puedan batallar entre sí. Me gustaría que me muestre como quedarían los objetos y un ejemplo de ataque de uno a otro.



Claro, puedo crear dos objetos en JavaScript que representen a Superman y a Omni-Man, junto con un método para que puedan batallar entre sí. Los objetos tendrán propiedades básicas como ``nombre``, ``fuerza``, ``defensa``, y ``vida``. También tendrán un método ``atacar`` que les permitirá realizar un ataque contra el otro personaje.

Copiar el código generado por ChatGPT en un archivo ***lucha.js*** y ejecutarlo con Node.js con el comando **`node lucha.js`**

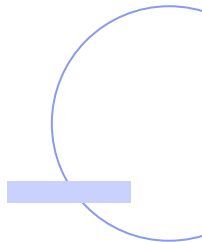


```
// Objeto para representar a Superman
const superman = {
  nombre: "Superman",
  poder: 100,
  defensa: 80,
  salud: 100,
  atacar: function(oponente) {
    // Daño aleatorio entre 10 y 20
    const damage = Math.floor(Math.random() * (20 - 10 + 1)) + 10;
    oponente.salud -= damage;
  }
};
```

```
// Objeto para representar a Omni-Man
const omniMan = {
  nombre: "Omni-Man",
  poder: 90,
  defensa: 85,
  salud: 120,
  atacar: function(oponente) {
    // Daño aleatorio entre 15 y 25
    const damage = Math.floor(Math.random() * (25 - 15 + 1)) + 15;
    oponente.salud -= damage;
  }
};
```

```
// Ejemplo de batalla entre Superman y Omni-Man
console.log("batalla entre Superman y Omni-Man:");
console.log("-----");
while (superman.salud > 0 && omniMan.salud > 0) {
  // Turno de Superman
  superman.atacar(omniMan);
  if (omniMan.salud <= 0) {
    console.log(`¡Superman gana!`);
    break;
  }

  // Turno de Omni-Man
  omniMan.atacar(superman);
  if (superman.salud <= 0) {
    console.log(`¡Omni-Man gana!`);
    break;
  }
}
```



Conclusiones

- El comportamiento del objeto alberga la **lógica de negocios** y es esencial para la **funcionalidad de las aplicaciones**.
- **Definir el comportamiento de un objeto es una tarea fundamental** que, frecuentemente, se subestima, incluso en los sistemas actuales.



Se utilizaron ejemplos en **JavaScript** debido a su capacidad para definir objetos sin necesidad de establecer previamente su estructura, para ilustrar conceptos de programación orientada a objetos de manera concisa y clara.

