# 15. Clases y Herencia

JavaScript utiliza un modelo de **herencia prototípica**, pero desde ES6, se introdujo una sintaxis más clara y familiar para trabajar con clases y herencia, similar a otros lenguajes orientados a objetos.

---

## 1. Clase Base: `Mamifero`

```
class Mamifero {
  constructor(comestible) {
    this.patas = 4;
    this.cola = true;
    this.comestible = comestible;
  }

  correr() {
    console.log("Este animal corre");
  }

  saltar() {
    console.log("Este animal salta");
  }

  meLoComo() {
    if (this.comestible) {
      console.log("Te has comido un mamífero");
    } else {
      console.log("No puedes comerte este animal");
    }
  }
}
```

**Características**:

- El método `constructor` inicializa los atributos (`patas`, `cola`, `comestible`).
- Los métodos (`correr`, `saltar`, `meLoComo`) se definen directamente en la clase.

---

## 2. Subclase: `AnimalDeGranja`

Hereda de `Mamifero` usando `extends` y `super`:

```
class AnimalDeGranja extends Mamifero {
  constructor(comestible) {
    super(comestible); // Llama al constructor de Mamifero
    this.enChoza = false;
  }
```

```
    guardarEnChoza() {
        this.enChoza = true;
    }

    sacarDeChoza() {
        this.enChoza = false;
    }

    // Sobrescribir el método heredado
    meLoComo() {
        if (this.comestible) {
            console.log("Te has comido un animal de granja");
        } else {
            console.log("No puedes comerte este animal");
        }
    }
}
```

**Detalles**:

- `extends Mamifero`: Indica que hereda de la clase `Mamifero`.
- `super(comestible)`: Ejecuta el constructor de la clase padre.
- Se pueden agregar nuevos métodos (`guardarEnChoza`, `sacarDeChoza`).

---

## 3. Subclase: `AnimalDomestico`

Hereda de `AnimalDeGranja` y agrega más funcionalidades:

```
class AnimalDomestico extends AnimalDeGranja {
    constructor(nombre) {
        super(false); // Llama a AnimalDeGranja con comestible = false
        this.nombre = nombre;
        this.enCasa = false;
    }

    guardarEnCasa() {
        this.enCasa = true;
    }

    sacarDeCasa() {
        this.enCasa = false;
    }

    // Sobrescribir el método heredado
    meLoComo() {
        console.log(`No te puedes comer a ${this.nombre}`);
        throw new Error("Este método no está disponible en la subclase");
```

```
    }

    // Sobrescribir y extender un método usando super
    saltar() {
        super.saltar(); // Llama al método saltar de Mamifero
        console.log("¡Salto como mascota!");
    }
}
```

**Detalles**:

- `super(false)`: Llama al constructor de `AnimalDeGranja` con `comestible = false`.
- `super.saltar()`: Reutiliza el método `saltar` de la clase padre (`Mamifero`).
- `throw new Error`: Genera un error si se intenta ejecutar `meLoComo`.

---

# 4. Ejemplos de Uso

## Crear Instancias

```
// Instancia de Mamifero
const ciervo = new Mamifero(true);
ciervo.correr(); // "Este animal corre"
ciervo.meLoComo(); // "Te has comido un mamífero"

// Instancia de AnimalDeGranja
const cerdo = new AnimalDeGranja(true);
cerdo.guardarEnChoza(); // Modifica enChoza
cerdo.meLoComo(); // "Te has comido un animal de granja"

// Instancia de AnimalDomestico
const perro = new AnimalDomestico("Lassie");
perro.guardarEnCasa(); // Modifica enCasa
perro.guardarEnChoza(); // Método heredado de AnimalDeGranja
```

## Sobrescritura de Métodos

```
// El perro sobrescribe el método saltar
perro.saltar();
// Salida:
// "Este animal salta"
// "¡Salto como mascota!"

// Intentar comérselo
try {
    perro.meLoComo();
} catch (error) {
```

```
    console.log(error.message); // "Este método no está disponible en la subclase"
}
```

## 5. Verificación de Herencia

JavaScript permite verificar la cadena de herencia con `instanceof`:

```
console.log(perro instanceof AnimalDomestico); // true
console.log(perro instanceof AnimalDeGranja); // true
console.log(perro instanceof Mamifero); // true

console.log(cerdo instanceof AnimalDomestico); // false
console.log(cerdo instanceof AnimalDeGranja); // true
```

## 6. Conceptos Clave

1. `extends`:
   - Indica que una clase hereda de otra.
   - Ejemplo: `class AnimalDeGranja extends Mamifero`.
2. `super`:
   - **En el constructor**: Llama al constructor de la clase padre.

     ```
     super(comestible); // Ejecuta el constructor de Mamifero
     ```

   - **En métodos**: Llama a un método de la clase padre.

     ```
     super.saltar(); // Ejecuta el método saltar de Mamifero
     ```

3. **Sobrescritura de Métodos**:
   - Redefine un método en la subclase para personalizar su comportamiento.