16. Métodos estáticos, getters y setters

En este capítulo, exploraremos tres conceptos avanzados en JavaScript: **métodos estáticos**, **getters** y **setters**. Estas herramientas permiten controlar el acceso a propiedades, encapsular lógica y definir funcionalidades a nivel de clase (no de instancia). Usaremos los ejemplos de animales para ilustrar su uso.

---

1. Métodos Estáticos

Los **métodos estáticos** pertenecen a la clase en sí, no a las instancias. Se definen con la palabra clave `static` y se suelen usar para funcionalidades que no dependen de datos específicos de un objeto.

Ejemplo: Método Estático en `Mamifero`

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

  // Método estático
  static especie() {
    return "Mamífero";
  }
}

// Uso
console.log(Mamifero.especie()); // "Mamífero"
```

Caso de Uso

- Crear utilidades relacionadas con la clase (ej: validaciones, fábricas de objetos).
- Acceder a datos o métodos sin necesidad de instanciar la clase.

---

2. Getters y Setters

Los **getters** y **setters** permiten controlar el acceso a las propiedades de un objeto. Son útiles para:

- Validar datos al asignar valores.
- Calcular propiedades dinámicamente.
- Encapsular lógica compleja.

Ejemplo: Getter y Setter en `AnimalDomestico`

```
class AnimalDomestico extends AnimalDeGranja {
    constructor(nombre) {
        super(false);
        this._nombre = nombre; // Convención: _propiedad para indicar privacidad
        this._enCasa = false;
    }

    // Getter para nombre
    get nombre() {
        return this._nombre;
    }

    // Setter para nombre (valida que no esté vacío)
    set nombre(nuevoNombre) {
        if (typeof nuevoNombre !== "string" || nuevoNombre.trim() === "") {
            throw new Error("El nombre debe ser un string no vacío");
        }
        this._nombre = nuevoNombre;
    }

    // Getter para estado del animal
    get estado() {
        return this._enCasa ? "en casa" : "fuera de casa";
    }
}

// Uso
const perro = new AnimalDomestico("Lassie");
console.log(perro.nombre); // "Lassie" (getter)
perro.nombre = "Max"; // (setter)
console.log(perro.nombre); // "Max"
console.log(perro.estado); // "fuera de casa"
```

Convenciones

- Usa `_propiedad` para indicar que una propiedad es "privada" (aunque no hay verdadera privacidad en JS).
- Los getters/setters se definen con `get` y `set`.

---

3. Propiedades Estáticas

Las **propiedades estáticas** son variables asociadas directamente a la clase, no a las instancias.

Ejemplo: Propiedad Estática en `Mamifero`

```
class Mamifero {
    static HOCICO = true; // Propiedad estática
```

```
}

console.log(Mamifero.HOCICO); // true
```

4. Ejemplo Integrado: `AnimalDomestico`

```
class AnimalDomestico extends AnimalDeGranja {
  constructor(nombre) {
    super(false);
    this._nombre = nombre;
    this._enCasa = false;
  }

  // Getter y Setter para nombre
  get nombre() {
    return this._nombre;
  }

  set nombre(nuevoNombre) {
    if (!nuevoNombre) throw new Error("Nombre inválido");
    this._nombre = nuevoNombre;
  }

  // Getter para enCasa
  get enCasa() {
    return this._enCasa;
  }

  // Setter para enCasa (valida el valor)
  set enCasa(valor) {
    if (typeof valor !== "boolean") {
      throw new Error("El valor debe ser booleano");
    }
    this._enCasa = valor;
  }

  // Método estático para crear un animal predeterminado
  static crearPerro() {
    return new AnimalDomestico("Perro");
  }
}

// Uso
const mascota = AnimalDomestico.crearPerro(); // Método estático
console.log(mascota.nombre); // "Perro"
```

```
mascota.enCasa = true; // Setter
console.log(mascota.enCasa); // true (Getter)
```

---

5. Ventajas de Getters y Setters

1. **Validación**:

```
set edad(nuevaEdad) {
  if (nuevaEdad < 0) throw new Error("La edad no puede ser negativa");
  this._edad = nuevaEdad;
}
```

2. **Propiedades Computadas**:

```
get resumen() {
  return `${this.nombre} (${this.estado})`;
}
```

3. **Encapsulación**:
   Ocultan la implementación interna de una propiedad.
