Novedades ES6

- **Clases**
  - Se ha introducido la palabra reservada class
  - Una clase **no precisa de argumentos** (parámetros) en su definición.
  - No es necesario el punto y coma ';' final.
  - Opcional, pero como práctica habitual en la POO, el nombre de la clase comienza con una letra mayúscula.
  - Al tratarse de una clase, **el sistema no permite su uso como una función**, sino que se reserva como un constructor. No hay salida de datos con return.
  - Sólo puede haber un constructor, no existe el concepto de sobrecarga
  - Se utiliza la palabra reservada super para llamar al constructor de la clase padre.

```
class Rectangulo {
    constructor(base, altura) {
        this.base = base;
        this.altura = altura;
    }
}
```

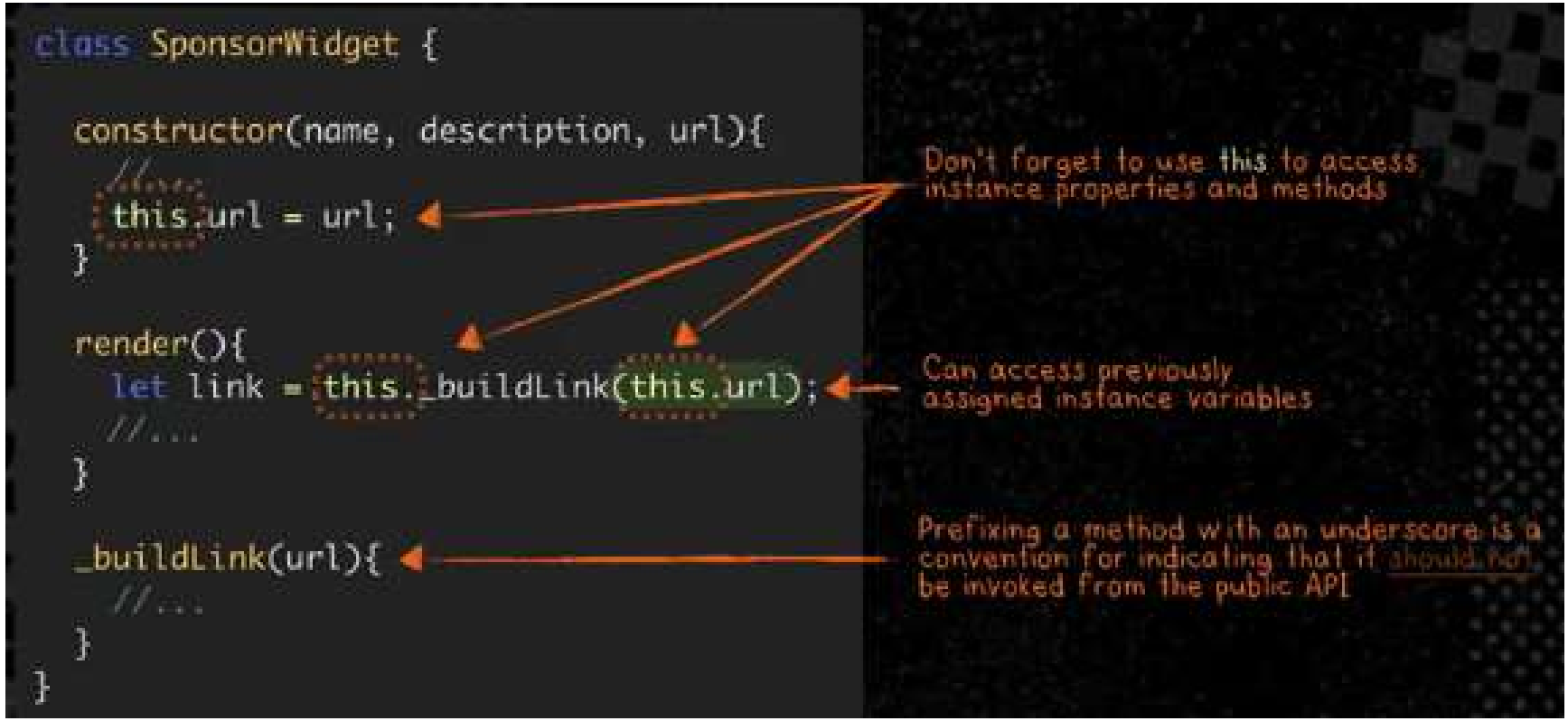
Novedades ES6

- **Clases**

  - Se utiliza _ delante de métodos y propiedades para indicar que no deberían ser públicos.
  - La definición de métodos ahora es más sencilla, ya no hay que escribir la palabra *function* ni declararlos de forma explícita con *var*, *let* o *const*, solo el nombre del método y su implementación.
  - El contenido de una clase **se ejecuta en modo estricto** de forma automática.
  - Las declaraciones de clases **no siguen las reglas de hoisting** como sí lo hacen las declaraciones de funciones. Esto quiere decir que solo existen tras ser declaradas.
  - De forma implícita un clase se comporta como una constante, no siendo posible redeclararla más adelante en un mismo ámbito o *scope*.

Novedades ES6

- **Clases**
  - **Otro ejemplo**



Novedades ES6

- **Clases**
  - El uso de clases incorpora nuevas palabra reservadas:
    - Constructor
    - Get
    - Set
    - Static - (Métodos estáticos, son llamados sin crear una instancia de la clase)

Novedades ES6

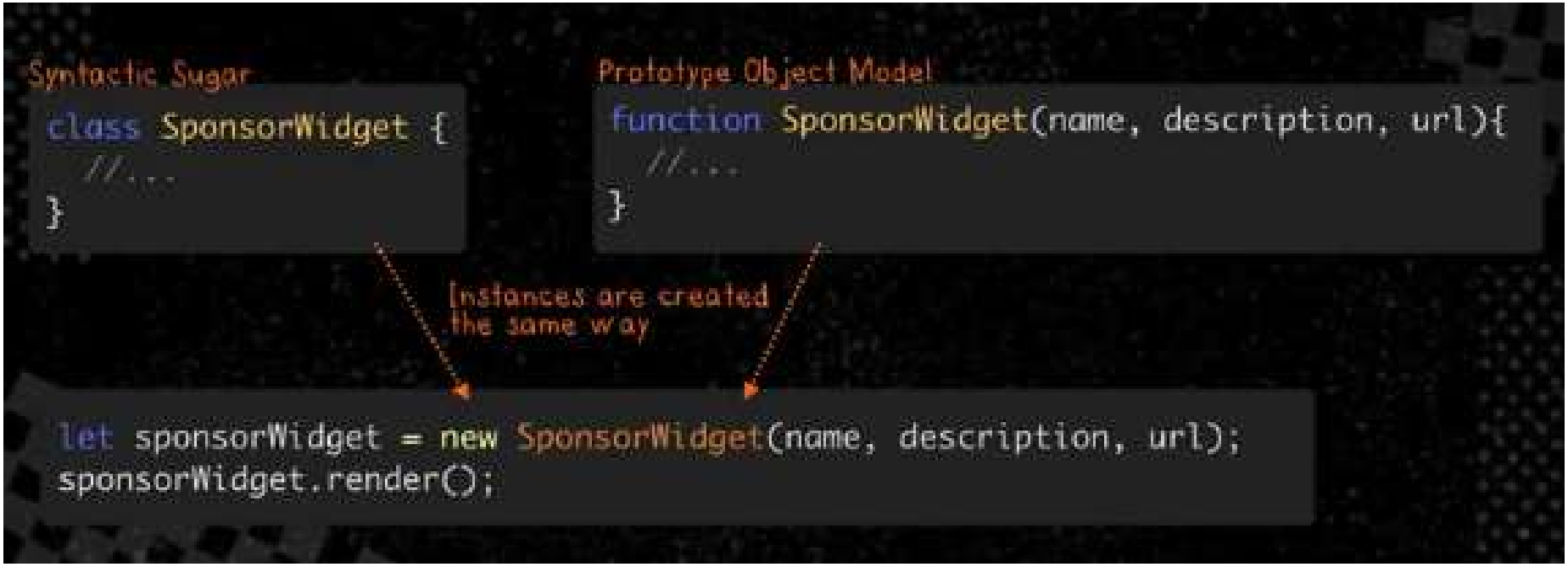
- **Clases**

```
class Persona {

    constructor(nombre,apellidos){
        this._nombre = nombre;
        this._apellidos = apellidos;
    }

    get nombre(){
        return this._nombre;
    }

    set nombre(nombre){
        this._nombre = nombre;
    }

    get apellidos(){
        return this._apellidos;
    }

    set apellidos(apellidos){
        this._apellidos = apellidos;
    }

    get nombreCompleto(){
        return `${this._nombre} ${this._apellidos}`;
    }

    static info(){
        return 'Método estático';
    }
}
```

```
var p = new Persona('Pablo', 'Pérez');
console.log(p.nombre);
console.log(p.apellidos);
console.log(p.nombreCompleto);

console.log(Persona.info());
```

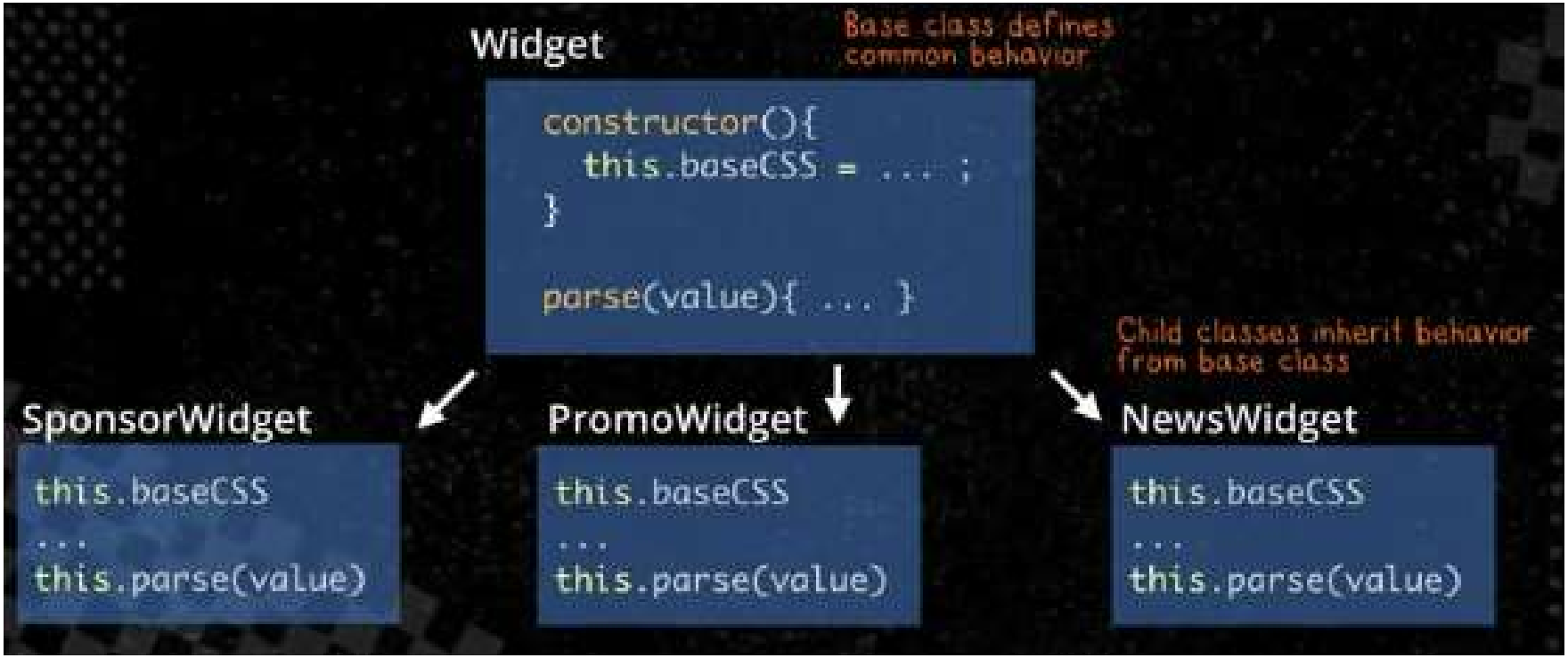
Novedades ES6

- **Clases**



Novedades ES6

- **Herencia de clases**

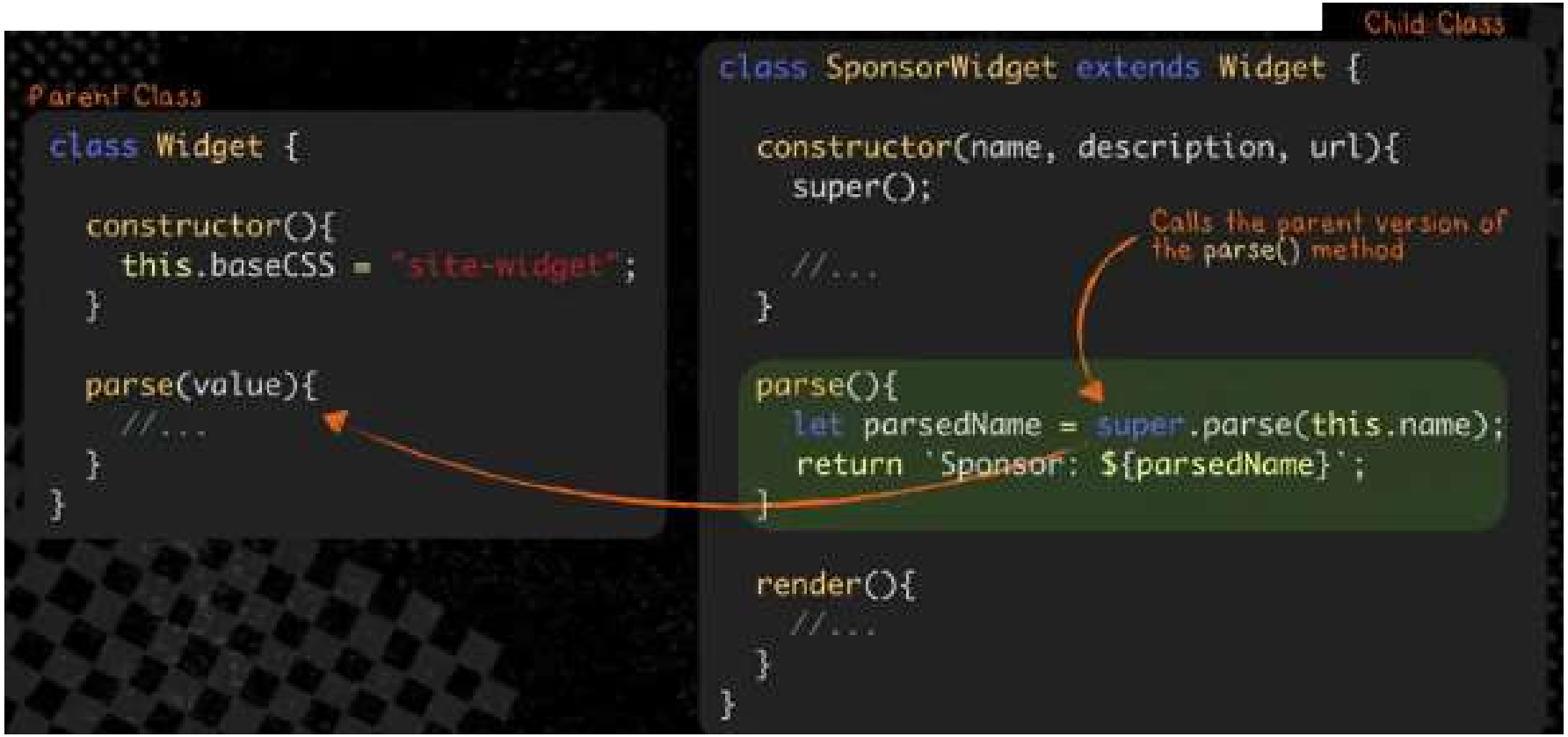


Novedades ES6

- **Herencia de clases**

Novedades ES6

- **Herencia de clases**



Novedades ES6

- **Herencia de clases**

```
class ShoppingCart {
  constructor(userId){
    this.userId = userId;
    this.products = [];
  }

  addProducts(product){
    this.products.push(product);
  }

  calculate(){
    //... complex math
  }
}
```

```
class ForumShoppingCart extends ShoppingCart {
  constructor(userId){
    super(userId);
  }

  calculate(){
    let partialCost = // call parent class `calculate` method here
    return partialCost - _calculateDiscount();
  }

  _calculateDiscount(){
    //... complex math
  }
}
```

- **Desde la clase ForumShoppingCart ¿cómo podemos invocar el método calculate de su clase padre ShoppingCart?**

- parent.calculate()
- global.calculate()
- super.calculate()