Clases. Herencia

- C.F.G.S. DAW
- 6 horas semanales
- Mes aprox. de impartición: Dic
- iPasen - cjaedia071@g.educaand.es

Carmelo José Jaén Díaz

Índice

OBJETIVO

- Profundizar en el concepto de objeto.
- Conocer y manejar funciones relativas al lenguaje sobre arrays, strings, números.

GLOSARIO

Backtracking. Estrategia utilizada en algoritmos que resuelven problemas que tienen ciertas restricciones. Este término fue creado por primera vez por el matemático D. H. Lehmer en la década de los cincuenta.

BOM (Browser Object Model). Convención específica implementada por los navegadores para que JavaScript pudiese hacer uso de sus métodos y propiedades de forma uniforme.

Expresión regular. Secuencia de caracteres que forman un patrón determinado, generalmente un patrón de búsqueda.

NaN. Propiedad que indica Not a Number (valor no numérico).

Objeto window. Aquel que soportan todos los navegadores y que representa la ventana del navegador. Se estudiará en profundidad en capítulos posteriores.

URI (Uniform Resource Identifier). Cadena de caracteres que identifica un recurso en una red de forma unívoca. Una URI puede ser una URL, una URN o ambas.

GLOSARIO

URN. Localizador de recursos en la web que funciona de forma parecida a una URL, pero su principal diferencia es que no indica exactamente dónde se encuentra dicho objeto.

INTRODUCCIÓN

En esta lección vamos a ver cómo trabajar con la **herencia**. Al igual que en otros lenguajes de programación, definimos la relación entre una clase *padre* y una clase *hijo* utilizando la palabra reservada `extends`, que definiría la relación de esta segunda clase con la primera. Además, para hacer referencia a los atributos de la clase padre utilizamos la palabra `super`.

Brevemente, la sintaxis para definir una cabecera de una clase hijo sería la siguiente:

```
class ClaseHijo extends ClasePadre
```

A continuación, se definirán dos clases que heredan una de otra, y se estudiará cómo hacer referencia a las propiedades y a los métodos de una clase padre desde una clase hija.

HERENCIA

Para crear herencia de clases hay que utilizar la palabra **`extends`**.

Una clase creada con herencia, hereda todos los métodos de la clase *padre* (clase superior).

Retomando la clase definida en la lección anterior:

```
class Docente {
  constructor (nom, ape){
    this.nombre = nom;
    this.apellido = ape;
  }
}
```

SINTAXIS DE HERENCIA DE UNA CLASE

Para definir una clase que herede las propiedades (y métodos si los hubiera) de la clase **Docente**.

```
class Docencia extends Docente {
  constructor (nom, ape, mod){
      super(nom, ape);
      this.modulo = mod;
  }
}
```

- Para hacer referencia a las propiedades y métodos de la clase *padre*, utilizamos la palabra reservada **super**.
- Para hacer referencia a las propiedades y métodos de la propia clase, utilizamos la palabra reservada **this**.

De esta manera, la clase **Docencia** tendrá toda la información de la clase **Docente**, más la información adicional que implementemos en la propia clase **Docencia**.

SINTAXIS DE HERENCIA DE UNA CLASE

A continuación, vamos a implementar un método en la clase **Docente**.

```
class Docente {
  constructor (nom, ape){
    this.nombre = nom;
    this.apellido = ape;
  }

   presentacion () {
         return "Hola, soy "+this.nombre;
     }

}
```

SINTAXIS DE HERENCIA DE UNA CLASE

Y, otro método en la clase **Docencia**.

```
class Docencia extends Docente {
  constructor (nom, ape, mod){
      super(nom, ape);
      this.modulo = mod;
  }

   presentacionCompleta(){
            return this.presentacion() + " el docente de " + this.modulo;
        }

}
```

Recuerda que es obligatorio llamar a **super()** en el constructor de la clase hija antes de acceder a una propiedad mediante **this**. De lo contrario, te aparecerá el siguiente mensaje: *Uncaught ReferenceError: Must call super constructor in derived class before accessing 'this' or returning from derived constructor.*

SINTAXIS DE HERENCIA DE UNA CLASE

Al invocar, mediante la palabra reservada **super**, al constructor de la clase *padre*, **Docente**, la clase *hija*, **Docencia**, obtiene las propiedades de la clase *padre* y las inicializa con los valores pasados como argumentos.

Si instanciamos un objeto de la clase **Docencia**.

```
let docenteSAI = new Docencia ("Carmelo", "Jaén", "SAI");

console.log (docenteSAI.presentacionCompleta());
//Devuelve "Hola, soy Carmelo el docente de SAI"
```

EJEMPLO DE HERENCIA DE UNA CLASE

Si creamos la clase *padre*, **Padre** y la clase *hija*, **Hijo**:

```
class Padre {
  soloPadre() { console.log("Salida del método soloPadre en la clase
Padre"); }
  padreHijo() { console.log("Salida del método padreHijo en la clase
Padre"); }
  sobreHijo() { console.log("Salida del método sobreHijo en la clase
Padre"); }
}
```

EJEMPLO DE HERENCIA DE UNA CLASE

Si creamos la clase *padre*, **Padre** y la clase *hija*, **Hijo**:

```
class Hijo extends Padre {
  padreHijo() {
    super.padreHijo();
    console.log("Salida del método padreHijo en la clase Hijo");
  }
  soloHijo() { console.log("Salida del método soloHijo en la clase Hijo");
}
  sobreHijo() { console.log("Salida del método sobreHijo en la clase
Hijo"); }
}
```

EJEMPLO DE HERENCIA DE UNA CLASE

Si creamos una instancia de la clase hija, **Hijo**, por medio de un **new Hijo()** y ejecutamos cada uno de ellos:

```
let hijo1 = new Hijo();

hijo1.soloPadre();
//Devuelve "Salida del método soloPadre en la clase Padre"
hijo1.soloHijo();
//Devuelve "Salida del método soloHijo en la clase Hijo"
hijo1.padreHijo();
//Devuelve "Salida del método padreHijo en la clase Padre" y "Salida del
método padreHijo en la clase Hijo"
hijo1.sobreHijo();
//Devuelve "Salida del método sobreHijo en la clase Hijo"
```

EJEMPLO DE HERENCIA DE UNA CLASE

Método	Clase Padre	Clase Hija	¿Se ejecuta el método en una instancia de la clase hija?
`soloPadre()`	✓	✖	Se ejecuta porque se hereda el método del padre hacia el hijo.
`soloHijo()`	✖	✓	Se ejecuta porque simplemente existe en el hijo.
`padreHijo()`	✓	✓	Se ejecutan ambos porque `super` llama al padre primero.
`sobreHijo()`	✓	✓	Se ejecuta sólo el hijo, porque sobreescribe el heredado del padre.