Clases. Métodos get y set

- 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 crear **métodos *get* y *set*** (o también llamados *getters* y *setters*) que incluir en nuestras clases.

Estos métodos nos permiten **extraer (*get*)** y **modificar (*set*)** las propiedades de un objeto. De este modo, nosotros podemos elegir exactamente, mediante estos métodos, **qué propiedades pueden ser accedidas y modificadas y cuáles no.**

De hecho, los ***getters*** y ***setters*** determinan el fundamento del **principio de encapsulación de la programación orientada a objetos.**

Lo habitual en otros lenguajes de programación es definir los getters y setters con la palabra get o set seguida del nombre de la propiedad. Pero Javascript es un caso especial, y los getters y setters se escriben

INTRODUCCIÓN

Lo habitual en otros lenguajes de programación es definir los *getters* y *setters* con la palabra ***get*** o ***set*** seguida del nombre de la propiedad. Pero Javascript es un caso especial, y los *getters* y *setters* se escriben con la palabra **get** o **set**, separadas por **un espacio** del nombre de la propiedad, con una particularidad: **¡no podemos poner el mismo nombre al método que a la propiedad porque entraríamos en un bucle!** Por eso muchos desarrolladores utilizan el guión bajo para nombrar la propiedad.

Importante: En Javascript, NO DEBES poner el mismo nombre a un método *get/set* que a la propiedad a la que accedes porque se produciría un error!

MÉTODOS GET Y SET

Los métodos `get` y `set` se utilizan para asignar y extraer atributos de un objeto.

Es muy importante tener en cuenta que el nombre de los *getters/setters* no puede ser el mismo que la propiedad porque se produciría un bucle: al acceder a la propiedad invocaríamos al método que a su vez accede a la propiedad que invoca al método...

Por ello, muchos desarrolladores utilizan el guión bajo para nombrar la propiedad.

SINTAXIS DE LOS SETTERS Y GETTERS

```
class Docente {
  constructor (nom){
    this._nombre = nom;
  }
    get nombre() {
      return this._nombre;
    }
    set nombre(nom) {
      this._nombre = nom;
    }
  }
```

Observa el uso del guión bajo a la hora de nombrar a la propiedad y el uso del espacio para definir los métodos `get` y `set`.

SINTAXIS DE LOS SETTERS Y GETTERS

A continuación, si quisiéramos definir un objeto.

```
let docenteSAI = new Docente ("Carmelo");
```

Para invocar los métodos **get** y **set** no se hace uso de los paréntesis como ocurre con el resto de métodos (de ahí la importancia del uso del guión bajo en la nomenclatura de las propiedades).

```
//Esta invocando al metodo set, ya que la propiedad se denomina _nombre
docenteSAI.nombre = "José"; //El nuevo argumento del set se pasa así

//Esta invocando al metodo get
alert("Hola, soy "+docenteSAI.nombre);
```

USO DE SETTERS Y GETTERS CON MIEMBROS PRIVADOS

Como estudiamos en la lección anterior, si queremos evitar que las propiedades de la clase se modifiquen fuera de la clase, aportando así seguridad a nuestro código, empleamos el símbolo # antes del nombre de la propiedad al declararla.

Por tanto, para solventar el error planteado anteriormente:

```
class Docente {
    #_nombre; // Declaración de la propiedad privada explícitamente antes de
    usarla.
    constructor(nom) {
        this.#_nombre = nom;
    }
}
const docenteSAI = new Docente("Carmelo"); // { nombre: "Carmelo" }
```

USO DE SETTERS Y GETTERS CON MIEMBROS PRIVADOS

```
// error (no se puede acceder a la propiedad privada)
// Uncaught SyntaxError: Private field '#nombre' must be declared in an enclosing class
console.log(docenteSAI.#_nombre);
```

Debemos implementar los métodos **get** y **set**, ya que al ser estos métodos públicos (también podríamos definirlos privados y acceder a ellos a través de un método público definido dentro de la clase) para acceder al contenido de la propiedad privada **#nombre**.

USO DE SETTERS Y GETTERS CON MIEMBROS PRIVADOS

```
class Docente {
  // Declaración de la propiedad
privada explícitamente antes de
usarla.
  #_nombre;

//Actualmente contiene Undefined, y
es mediante el constructor cuando
almacena el dato que se le pase como
argumento al instanciarla
  constructor (nom){
    this.#_nombre = nom;
  }
  get nombre() {
    return this.#_nombre;
  }
  set nombre(nom) {
    this.#_nombre = nom;
  }
}

let docenteSAI = new Docente
("Carmelo");

console.log(docenteSAI.nombre);
// Mediante el método get
```