

HINTS

GETTERS Y SETTERS

Un **Getter** (*captador*) es un método que obtiene el valor de una propiedad específica. Mientras que un **Setter** (*definidores*) es un método que establece el valor de una propiedad específica. Se pueden definir captadores y definidores en cualquier objeto principal predefinido, o en un objeto definido por el usuario que admita la adición de nuevas propiedades.

Éstos pueden ser definidos usando inicializadores de objeto, o agregándolos posteriormente a cualquier objeto, y en cualquier momento, usando un método de adición **Getter** o **Setter**. Al definir captadores y definidores, utilizando inicializadores de objetos, todo lo que necesita hacer es prefijar un método captador con **get**, y un método Setter con **set**. Por supuesto, el primero no debe esperar un parámetro, mientras que el segundo espera exactamente un parámetro (el nuevo valor a establecer).

Por ejemplo:

```
1 //objeto
2 var puerta = {
3     //Propiedades
4     cerrada: false,
5
6     //Getter
7     get getCerrada() {
8         return this.cerrada;
9     },
10
11     // Setter
12     set setCerrada(valor) {
13         return this.cerrada = valor;
14     },
15
16     //Métodos
17     abrir: function () {
18         console.log('puerta abierta');
19     },
20     cerrar: function () {
21         console.log('puerta cerrada');
22     }
23
24 };
```

Como se puede apreciar, se emplean las palabras clave `get` y `set`, para establecer los Getters y Setters. En este caso, el Getter retorna el valor de la propiedad “cerrada”, y el Setter recibe un parámetro para establecerlo como el valor de la propiedad “cerrada”. Mediante las siguientes líneas de código, se ejemplifica el uso de ambos.

```
1 console.log(puerta) // Llamamos al objeto
2 console.log(puerta.cerrada) //Llamamos a la propiedad "cerrada" del objeto
3 "puerta"
4 // = false
5 console.log(puerta.setCerrada) // Utilizamos el Setter
6 console.log(puerta.cerrada) //Verificamos Cambio efectuado por el setter
7 // = true
```

El resultado por consola es el siguiente:

```
► {cerrada: false, abrir: f, cerrar: f}
false
true
true
```

De esta forma, queda demostrado cómo usar los Getters y Setters en un objeto de JavaScript.

OBJETOS NATIVOS EN JAVASCRIPT

JS ofrece la habilidad de utilizar objetos nativos del idioma en nuestras implementaciones. Esto se puede asemejar mucho a lo que se hace en otros idiomas, como Java. Unos objetos que podemos utilizar en JavaScript son los siguientes:

Date()

Math()

El primero nos permite obtener información con respecto a la fecha, o a la hora en tiempo real. Por ejemplo, podemos utilizar las siguientes líneas de código JavaScript para visualizar toda la información que podemos obtener de este **objeto**:

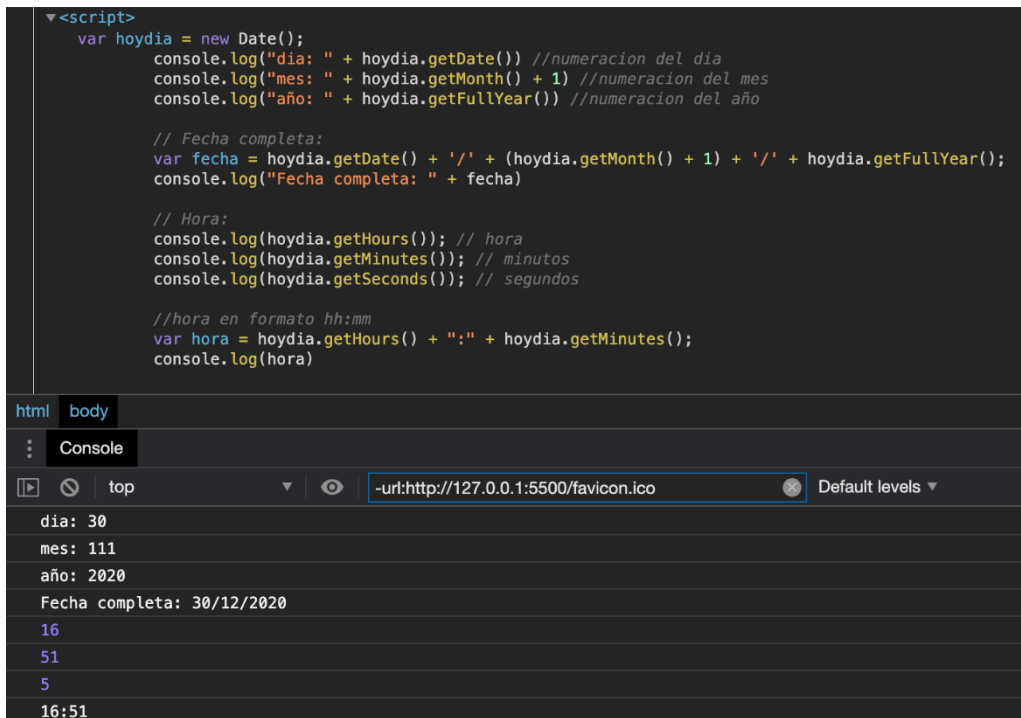
```
1 var hoydia = new Date(); //objeto "Fecha"
2 console.log("dia: " + hoydia.getDate()) //numeracion del dia
```

```

3 console.log("mes: " + hoydia.getMonth() + 1) //numeracion del mes
4 console.log("año: " + hoydia.getFullYear()) //numeracion del año
5
6 // Fecha completa:
7 var fecha = hoydia.getDate() + '/' + (hoydia.getMonth() + 1) + '/' +
8 hoydia.getFullYear();
9 console.log("Fecha completa: " + fecha)
10
11 // Hora:
12 console.log(hoydia.getHours()); // hora
13 console.log(hoydia.getMinutes()); // minutos
14 console.log(hoydia.getSeconds()); // segundos
15
16 //hora en formato hh:mm
17 var hora = hoydia.getHours() + ":" + hoydia.getMinutes();
18 console.log(hora)

```

La siguiente imagen muestra los resultados por consola de las líneas de código anteriores:



```

<script>
  var hoydia = new Date();
  console.log("dia: " + hoydia.getDate()) //numeracion del dia
  console.log("mes: " + hoydia.getMonth() + 1) //numeracion del mes
  console.log("año: " + hoydia.getFullYear()) //numeracion del año

  // Fecha completa:
  var fecha = hoydia.getDate() + '/' + (hoydia.getMonth() + 1) + '/' + hoydia.getFullYear();
  console.log("Fecha completa: " + fecha)

  // Hora:
  console.log(hoydia.getHours()); // hora
  console.log(hoydia.getMinutes()); // minutos
  console.log(hoydia.getSeconds()); // segundos

  //hora en formato hh:mm
  var hora = hoydia.getHours() + ":" + hoydia.getMinutes();
  console.log(hora)

```

html body

Console

top

-url:http://127.0.0.1:5500/favicon.ico

Default levels

dia: 30

mes: 11

año: 2020

Fecha completa: 30/12/2020

16

51

5

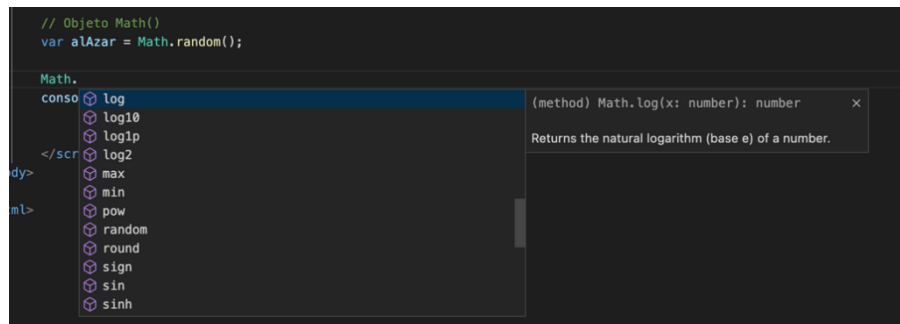
16:51

Como se puede apreciar, el objeto `Date()` o "Fecha()", nos permite acceder a atributos y métodos que nos ayudan a delimitar la información que queremos utilizar. En sí, su funcionalidad es bastante extensa, y por

esa razón se recomienda explorar la [documentación](#) de este objeto, para así familiarizarse con el alcance de funciones que tiene.

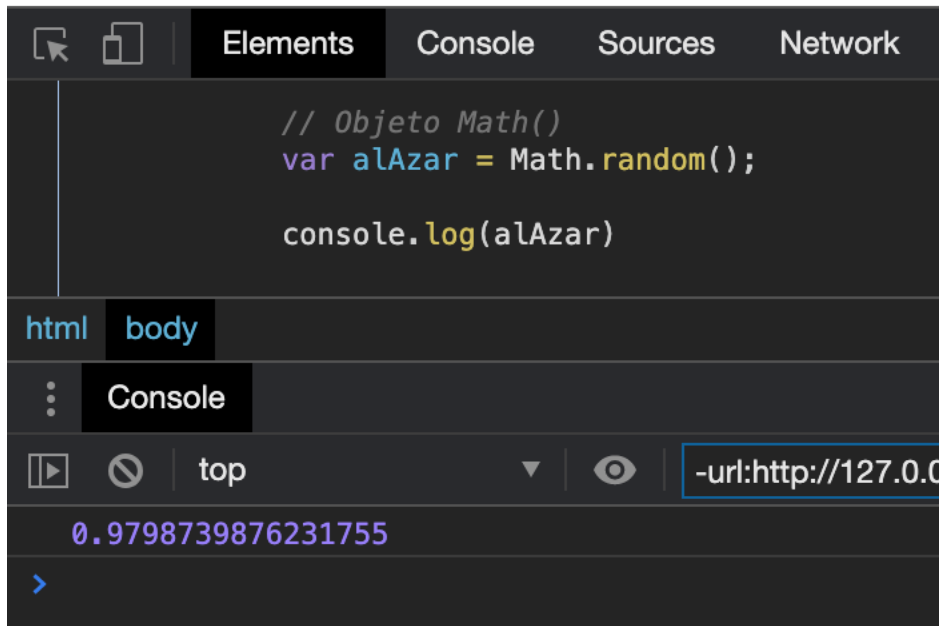
En cuanto al objeto **Math()** o “Matematicas()”, éste nos permite utilizar una amplia gama de métodos y atributos. Sus funcionalidades varían desde la generación de números al azar, a métodos de trigonometría.

A continuación, se puede observar como generar un valor al azar empleando el objeto **Math()**, y posteriormente, visualizar todas las funcionalidades que contempla dicho objeto al usar la combinación de teclas: **CONTROL + ESPACIO**.



Al utilizar las siguientes líneas de código, obtendremos este resultado por consola:

```
1 // Objeto Math()
2 var alAzar = Math.random();
3 console.log(alAzar)
```



```
// Objeto Math()
var alAzar = Math.random();

console.log(alAzar)
```

html body

Console

top -url:http://127.0.0.1

0.9798739876231755

>

Se recomienda investigar e indagar más con respecto a este objeto, en la siguiente [documentación](#).

NOTACIÓN DE CORCHETES

Como se ha aprendido en este CUE, la notación de puntos implica acceder a propiedades y métodos de un objeto, simplemente escribiendo su nombre seguido de un punto, y luego el nombre de la propiedad o método que se requiere usar.

¿Sabías que hay otra forma de acceder a las propiedades de los objetos?: será usando la "notación de corchetes".

Se puede acceder a las propiedades de un objeto, especificando su nombre, seguido del nombre de una propiedad entre paréntesis. La sintaxis es así:

```
1 nombreObjeto["nombrePropiedad"] .
```

El siguiente es un ejemplo de la notación de corchetes, donde se evidencia el sonido que hace un gato:

```
1 let obj = {  
2   gato: 'miau',  
3   perro: 'guau'  
4 };  
5 let sonido = obj['gato'];  
6 console.log(sonido); // miau
```

Prueba esta notación, para que puedas corroborar que ayuda a trabajar con objetos de una forma más fácil.