Introducción a Javascript

Clase 5

Objetos I

Objetos: introducción

¿Qué es un objeto?

- Es un espacio de la memoria con un nombre de referencia para identificarlo, que puede contener:
  - **Propiedades:** son los atributos del objeto (variables de cualquier tipo de dato, menos las funciones).
  - **Métodos:** son las acciones del objeto (variables del tipo función).

- Para Javascript TODO es un objeto y ya utilizamos métodos y propiedades de ellos
  - Ejemplo:
    - document.write() // Método write del objeto document
    - unArray.length // Propiedad length del objeto unArray

- Los objetos son simples de utilizar, son una variante al array asociativo (de hecho, los arrays asociativos, son objetos y viceversa, pero ya lo veremos).
- JSON (acrónimo de JavaScript Object Notation) es utilizado por las principales librerías basadas en Javascript y es la forma de transportar datos entre el lado cliente y el lado servidor.

Objetos: Creación

Objetos: creación

¿Cómo se crea un objeto?

- Existen cuatro formas de crear un objeto, ahora se explican 2:
    - Usando la versión constructora:
        - let objetoConstructor = new Object();
    - Usando la versión literal:
        - let objetoLiteral = { };
- Generalmente utilizaremos la versión literal a la hora de crearlos

Objetos: Contenido

Objetos: contenido

¿Cómo se le guardan valores a un objeto?

- Existen dos formas de agregar contenido a un objeto, sin importar si fue creado con la versión constructora o literal:
  - Crear un objeto vacío y agregando propiedades o métodos por notación de puntos
    ```
    let objeto = { }
    objeto.propiedad = 'un valor';
    objeto.metodo = function () { };
    ```
- Lo que está después del punto será la propiedad (variable) o método (función).
- Los métodos son funciones, por lo que deberán crearse como tales y luego se los podrá ejecutar.
- Se pueden ir armando una secuencia de propiedades y métodos a través de los puntos, por ejemplo
  - objeto.propiedad.otraPropiedad.unaPropiedadMas;
  - objeto.propiedad.otraPropiedad.unMetodo();

Objetos: contenido

¿Cómo se le guardan valores a un objeto?

- Crear un objeto de manera literal (sí o sí), asignando con dos puntos y separando con coma:

```
let objeto = {
        propiedad: 'un valor',
        metodo: function () { },
};
```

Notación de objetos de Javascript (JSON):

- Es muy probable que optemos por cargar la información del objeto al crearlo de manera literal con información inicial. De ser así hay que respetar las siguiente sintaxis:
    - Dentro del objeto no se utiliza el igual, sino que los dos puntos (:)
    - Cada propiedad o método se separa del siguiente con una coma (,)

Objetos: this

Objetos: métodos que usan propiedades propias de el

¿Qué pasa si en un método del objeto quiere utilizar una propiedad de ESTE mismo objeto ya existente o que está siendo creado?

- Podemos utilizar el nombre de la variable que le pusimos al objeto:

```
let Persona = {
        nombre: 'Nombre',
        Presentarse: function () {
                // Utilizo la propiedad del objeto:
                console.log('Hola soy ' + Persona.nombre);
        },
};

// Ejecuto el método de la persona:
Persona.Presentarse(); // En consola saldrá 'Hola soy Nombre';
```

Objetos: métodos que usan propiedades propias de el

¿Qué pasa si en un método del objeto quiere utilizar una propiedad de ESTE mismo objeto ya existente o que está siendo creado?

- O podemos hacer referencia a ***ESTE*** objeto que me contiene con el objeto ***this***:

```
let Persona = {
        nombre: 'Nombre',
        Presentarse: function () {
                // Utilizo la propiedad de este mismo objeto:
                console.log('Hola soy ' + this.nombre);
        },
};

// Ejecuto el método de la persona:
Persona.Presentarse(); // En consola saldrá 'Hola soy Nombre';
```

Objetos: métodos que usan propiedades propias de el

¿Qué pasa si en un método del objeto quiere utilizar una propiedad de ESTE mismo objeto ya existente o que está siendo creado?

- **this** es un objeto de ámbito, que **s*olamente existe dentro de los métodos o funciones de un objeto***, muy utilizado para autoreferenciar al objeto en sí.

- **Al utilizar** métodos con **this**, los mismos **no pueden ser utilizando la función flecha**.

- Veamos un ejemplo para mayor comprensión ***(index-objetos.html)***

Objetos o arrays asociativos

Objetos o arrays asociativos

¿Un array asociativo es lo mismo que un objeto?

- Los arrays pertenecen al tipo de dato object
- Los arrays asociativos, en particular, son una versión adaptada de un objeto
- Esto significa que se puede utilizar los conceptos aprendidos hasta el momento, tanto para arrays asociativos, como para objetos, combinándolos
- Por ejemplo
    - Se puede crear un array asociativo, crear índices asociativos y luego agregar propiedades para cargar nuevos valores por notación de puntos

```
let materia = [ ];                                  // Creo un array
materia['Nombre'] = 'Intro a JavaScript';           // Cargo un valor en un índice
materia['Profesor Titular'] = 'Juli Pacilio';       // Cargo un valor en un índice
materia.Horario = '19 a 22';                        // Cargo un valor en una propiedad
```

Arrays: contenido

¿Un array asociativo es lo mismo que un objeto?

- Por ejemplo
  - Se puede crear un objeto, crearle sus elementos y luego recorrerlo con un for in

```
let materia = {
        Nombre: 'Intro a JavaScript',
        'Profesor titular': 'Juli Pacilio',
        Horario: '19 a 22',
};

for (let dato in materia) {
        console.log(materia[dato]);
};
```

- Si una propiedad tiene caracteres especiales, debe ir entre ' '

Objetos: Consideraciones

Objetos: últimas consideracione

A tener en cuenta

- En vista de lo que acabamos de aprender, entonces, una posición asociativa es lo mismo que una propiedad.
- Esto nos deja entonces la posibilidad de acceder por corchetes o notación de puntos

```
let profesor = {
        Nombre: 'Julian',
        Apellido: 'Pacilio',
        'Fecha de nacimiento': '02-02-1999',
};
```

- Si voy a utilizar índices, con comillas
  - console.log(**profesor['Nombre']**);
- Si voy a utilizar propiedades, sin comillas
  - console.log(**profesor.Apellido**);
- Si es una propiedad con caracteres especiales, con comillas, sí o sí
  - console.log(**profesor['Fecha de nacimiento']**)

Objetos II

Objetos: Repaso

¿Qué es un objeto y cuál es su contenido?

- Es un tipo de dato
- Puede tener propiedades (variables sin acciones) y métodos (variables del tipo función)

¿De qué formas se pueden crear objetos en Javascript?

- De forma constructora
- De forma literal
- Mediante Clases
- Método create()

¿Cómo se agrega contenido a un objeto?

- Al crear un objeto de manera literal, con JSON
- Con notación de puntos
- Ambos se pueden combinar

Objetos: Repaso

¿Qué es this?

- Es una variable de ámbito
- Hace referencia al objeto que ejecuta el método que contiene dicho this
- Sirve para acceder a propiedades o métodos del mismo objeto
- Solamente existe dentro de las funciones

¿Un array asociativo es un objeto?

- Efectivamente, podemos utilizar las características de ambos, es decir, acceder a sus propiedades / métodos como si de índices se tratara o al revés, acceder a un índice a través de la notación de puntos
- Lo único que se debe tener en cuenta es que este intercambio de características es posible solamente si los índices o propiedades / métodos, respetan las normas de asignación de nombre de variable.

Objetos II: Inconvenientes

Objetos: no permiten la duplicación

¿Qué "inconveniente" presenta Javascript en cuanto a los objetos?

- El objeto no permite crear copias de sí mismo directamente … ¿cómo es eso?

Números

let n1 = 10;
let n2 = n1;
n2 = n2 + 5;

¿Cuánto valen n1 y n2?
n1 vale 10 n2 vale 15

Números
let c1 = 'Hola'; let c2 = c1; c2 = ' y chau';
¿Cuánto valen c1 y c2?
c1 vale 'Hola' c2 vale ' y chau'

Objetos
let o1 = { }; o1.nombre: 'Juli'; let o2 = o1; o2.nombre = 'Nico';
¿Cuánto valen o1.nombre y o2.nombre?
o1.nombre vale 'Nico' o2.nombre vale 'Nico'

- Esto significa que para hacer varios objetos “iguales”, tendríamos que crear cada variable “igual”
- Al no ser muy práctico, se pueden crear objetos a partir de un molde
  - let objeto = new MoldeParaElObjeto();


Guys I have bad news
You can't learn JavaScript in 30 days
DEVS.LOL


learn Javascript
in one lesson
2:52 / 35040:04

Clases

Clases: introducción

¿Qué es una clase?

- Es una plantilla para crear objetos. Encapsula datos con código
- Se crean con la palabra reservada ***class*** seguida del nombre de la clase y luego **{ }**:
  - class NombreClase { }

Propiedades

- Al escribir variables dentro de las clases, las mismas representan las propiedades del objeto
- Existen propiedades públicas y privadas
  - Públicas
    - Se crean sin anteponer ninguna palabra al nombre
    - Pueden ser accedidas por los métodos de la clase y por el objeto creado
  - Privadas
    - No están estandarizadas en todos los navegadores
    - Se escriben anteponiendo el símbolo #: #propiedadPrivada ;
    - Solamente pueden ser accedidas por los métodos de la clase

Clases: introducción

Métodos

- En las clases los métodos se escriben colocando el nombre, seguido de ( ) y las { }. No se coloca la palabra **function** ni "la flecha" **=>**
  - nombreMetodo( ) { }
- Los métodos también puede ser públicos o privados (pero tampoco están estandarizados):
  - metodoPublico() { }
  - #metodoPrivado() { }
- Para referirse a una propiedad dentro de un método, se debe colocar this. a la propiedad:
  - this.propiedadPublica ;
  - this.**#**propiedadPrivada ;
- Así mismo, para ejecutar un método dentro de otro, se debe colocar this. a la ejecución del método:
  - this.metodoPublico() ;
  - this.**#**metodoPrivado() ;

Clases: introducción

Métodos

- Puntualmente, para que una clase funcione necesita tener el método constructor: constructor() { }

- Dicho método es el primero que se ejecuta al crear un nuevo objeto a partir de la clase (instanciación del objeto).

- Veamos un ejemplo para ver qué sucede ***(clases-ejemplo-1.html)***.

Clases: Parámetros

Clases: parámetros

Agregando parámetros a las clases

- Como la clase en sí no tiene paréntesis, los parámetros los tiene le método constructor
- Al ser un método, sigue las reglas de una función (se pueden inicializar los parámetros)
- Como los parámetros son privados, deberán guardarse en variables públicas de la clase.propiedad
- Veamos un ejemplo para ver qué sucede ***(clases-ejemplo-2.html)***.

Clases: getters y setters

Clases: setters y getters

Trabajando con las propiedades mediante get y set

- Son métodos que establecen (set) o leen (get) el valor de una propiedad
- Pero, si bien son métodos, al interactuar con el objeto que los contiene se los maneja como propiedades
- Veamos un ejemplo para ver qué sucede ***(clases-ejemplo-3.html)***.

Clases: props y métodos estáticos

Clases: Propiedades y métodos estáticos

Propiedades y métodos estáticos

- Las propiedades o métodos estáticos permiten acceder a ellos sin necesidad de instanciar un objeto. Directamente se coloca el nombre de la clase, seguido de la propiedad o método

- Además, guardan una relación con las otras instancias de los objetos creados

- La palabra static se antepone a la propiedad o método para hacerlo estático

- También se puede aplicar para getters

- Veamos un ejemplo para mayor comprensión ***(clases-ejemplo-4.html)***

Clases: últimos detalles

Clases: últimos detalles

Orden ideal de los elementos dentro de una clase

1. Propiedades y métodos estáticos.
2. Propiedades públicas.
3. Propiedades privadas.
4. Constructor.
5. Setters y Getters.
6. Métodos públicos.
7. Métodos privados.

Clases: Una forma más de crear objetos

¿Cómo crear objetos mediante el método create()?

- Los objetos en Javascript también se pueden crear a partir del método create():

```
let objeto = Object.create();
```

- El primer parámetro puede recibir un objeto vacío, así como uno ya existente.

```
let objeto1 = Object. create( { } );
objeto1.nombre =  'Julián';
objeto1.Soy = function () {
  ■   console.log(`Soy ${this.nombre}`);
}
let objeto1 = Object. create(objeto1);
objeto2.nombre =  'Nicolás';
objeto2.Soy();
```

- Esta es otra forma de "clonar" objetos también válida

Fin de la clase

Este es el espacio para dudas y/o preguntas existenciales