Javascript

JS

Objeto literal

- Un objeto es cualquier cosa de la vida representado en código.
- Los objetos tienen propiedades que se utilizan para describirlo. Podemos utilizar *strings, numbers, booleans, objetos y arrays*
- En Javascript se utilizan las llaves **{ }** para establecer un objeto literal.

```
var miObjeto = {
    propiedad1: valor,
    propiedad2: valor
};
```

Objeto literal - Propiedades

Podemos acceder a cualquier propiedad utilizando el nombre del objeto, **un punto y el nombre del atributo que quere acceder**.

- nombreObjeto**.propiedad**;

Objeto literal - Propiedades

Los objetos en Javascript tienen ínidice como string y podemos acceder a sus propiedades utilizando **[ ] y el nombre de la propiedad como string.**

- nombreObjeto**[propiedadComoString]**;

Objeto literal - Propiedades Dinámicas

Podemos utilizar una variable con el nombre del atributo.

var **propiedad** = ‘propiedadComoString’;

nombreObjeto**[propiedad]**;

Objeto literal - Establecer el valor de una propiedad

nombreObjeto.**propiedad = valor;**

nombreObjeto**[propiedadTipoString] = valor;**;

nombreObjeto**[propiedad] = valor;**;

Objeto literal - Propiedades

A un objecto se le pueden agregar propiedades aún si estas no fueron declaradas en la definición del objeto.

```
var objeto = { };

objeto.propiedad = valor;

objeto['propiedad'] = valor;
```

Objeto literal - Métodos

Los objetos también tienen métodos que nos permiten interactuar con ellos.

Los métodos son una propiedad del objeto que tienen asignado una función.

```
var miObjeto = {

    metodo: function() { }

};

miObjeto.metodo();
```

Objeto literal - Métodos y parámetros

Los métodos también pueden recibir parametros ya que son una función.

```
var miObjeto = {
        metodo: function(parametro) { }
};
miObjeto.metodo(10);
```

Objeto literal - Métodos

Al igual que las propiedades los métodos se pueden acceder por corchetes

```
var miObjeto = {
        metodo: function() { }
};
miObjeto[metodoComoString]();
```

Objeto literal - Métodos

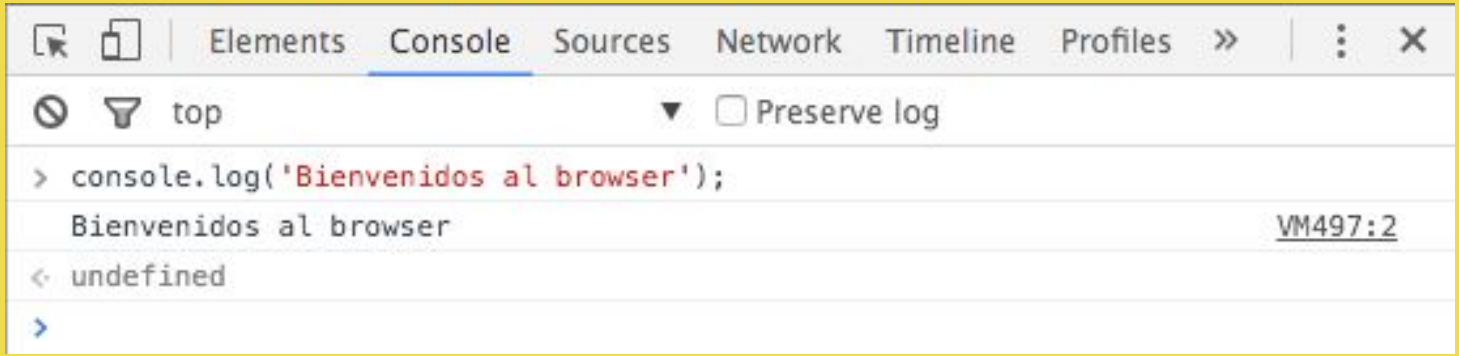
Al igual que las propiedades podemos crear un método sin estar antes definído en el objeto.

```
var miObjeto = { };

miObjeto.metodo = function() { };

miObjeto.metodo();
```

Bienvenidos al Browser

Correr Javascript en el browser

Por medio de la consola podemos correr código Javascript en el browser



JS

Agregar Javascript a un archivo HTML

Utilizando la etiqueta **script** podemos agregar código Javascript a un documento HTML

```
<head>
    <script>
        // código Javascript
    </script>
</head>
```

Agregar Javascript en un archivo externo

La etiqueta script tiene un atributo **src** que nos permite agregar código Javascript en un archivo externo.

```
<head>

    <script src="archivo.js"></script>

</head>
```

Agregar Javascript a un archivo HTML

La etiqueta script tiene un atributo **src** que nos permite agregar código Javascript en un archivo externo.

```
<head>

    <script src="archivo.js"></script>

</head>
```

Window

Los browsers tienen un objeto windows que es un objeto global a toda la aplicación.

Al definir una variable global se crea en el objeto window.

Este objeto tiene métodos propios que nos permiten interactuar con el browser.

Window - Alert

El método alert nos permite mostrar un mensaje en la pantalla con un botón para cerrarlo.

Este método se debe usar sólo para informar al usuario sin esperar confirmación del mismo.

alert(mensaje);

window.alert(mensaje)

Window - Prompt

Otro método llamado **prompt** nos permite interactuar con el usuario ya que le podemos pedir que ingrese un valor.

prompt(mensaje**)**;

var respuesta = **prompt(**mensaje**)**;

var respuesta = window.**prompt(**mensaje**)**;

Window - Confirm

Otro de los métodos del objeto windows es **confirm** que nos permite **mostrar un mensaje al usuario** y **obtener un valor booleano (true/false)** como respuesta.

confirm(mensaje**);**

var respuesta = **confirm(**mensaje**);**

var respuesta = **window.confirm(**mensaje**);**

Window - Location

El objeto location representa la URL. Por medio de sus propiedades podemos acceder a las distintas partes de la URL como también navegar a otro documento.

window.**location**;

Window - Location

Location: representa al objeto que está mapeando una URL

window.**location**;

Href: muestra la URL como string. Podemos utilizar href para asignar otra URL.

window.location.**href**;

Protocol: muestra el protocolo. ej: http, https

window.location.**protocol**;

Window - Location

Host: retorna un string con el nombre del host y el número de puerto en caso de que tenga uno.

window.location.**host;**

Hostname: muestra sólo el nombre del host

window.location.**hostname**;

Port:

window.location.**port**;

Window - Location

Pathname: muestra el path del recurso

window.location.**pathname**;

Search: muestra los parámetros del query string

window.location.**search;**

Hash: muestra el contenido de hash

window.location.**hash**;

Window - Location redirect

Por medio del atributo href podemos asignar otra url al navegador.

window.location.**href = 'http://www.google.com';**

También podemos utilizar el atributo location

Window**.location = 'http://www.google.com';**

Window - History

Window - History

El objeto history representa el historial de documentos visitados en la actual sesión del browser.

window.**history**

Length: retorna la cantidad de items del historial

window.**length**;

Window - History

Go: vuelve o avanza en el historial. Utilizamos un ínidice negativo para ir hacia atrás y uno positivo para ir hacia adelante.

window.history.**go(indice)**;

Back: vuelve al documento anterior

window.history.**back()**;

Forward: navega a la próxima URL

window.history.**forward()**;

Screen

El objeto **screen** nos proporciona valores sobre la pantalla del dispositivo que está viendo nuestro documento por medio de las propiedades **width** y **height**

window.**screen**;

window.screen.**height**;

window.screen.**width**;

Window

El objeto window tiene propiedades **innerHeight** y **innerWidth** que nos retornan el valor y ancho del documento que estamos viendo.

window.**innerHeight**;

window.**innerWidth**;

Window - Scroll

El objeto window también tiene 2 propiedades, **innerHeight** y **innerWidth** que nos permiten saber el scroll del documento.

window.**innerHeight**;

window.**innerWidth**;

Window - Open

El objeto window tiene un metodo llamado **open()** que nos permite abrir e interactuar con nuevas ventanas.

window.**open(url, nombreDeVentana, stringConPropiedades)**;

Por medio de la propiedad **opener** podemos obtener una referencia a la ventana padre.

nuevaVentana.**opener**;

Timers

Timers - setTimeout

Javascript tiene funciones nativas que nos permiten retrasar la ejecución de un código que nosotros querramos.

La función **setTimeout** se utiliza cuando queremos que nuestro código se ejecute una vez en un tiempo establecido.

window**.setTimeout(funcion, retraso)**;

setTimeout(funcion, retraso);

Timers - setTimeout

Retorna un valor numérico que se utiliza como ID de este timeout, por ejemplo para cortar la ejecución del mismo con **clearTimeout()**.

var **idTimeOut** = setTimeout(funcion, retraso);

window.**clearTimeout(idTimeout)**

Timers - setTimeout

Podemos pasarle parámetros a la función que se ejecutará en el setTimeout

window.setTimeout(funcion, retraso, **[parametros]**);

setTimeout(funcion, retraso, **[parametros]**);

Timers - setInterval

Por medio de esta función podemos ejecutar varias veces el mismo código con un retraso establecido.

window.**setInterval(funcion, retraso)**;

setInterval(funcion, retraso);

Timers - setInterval

Esta función retorna un valor numérico que se utiliza como ID y nos permite cancelar la ejecución de esta repetición con **clearInterval()**.

var **idSetInterval** = setInterval(funcion, retraso);

window.**clearInterval(idSetInterval);**

Timers - setInterval

Podemos pasarle parámetros a la función que se ejecutará en el setInterval

window.setInterval(funcion, retraso, **[parametros]**);

setInterval(funcion, retraso, **[parametros]**);