Facultad Regional Rosario
Universidad Tecnológica Nacional

Curso full stack

Introducción a javascript

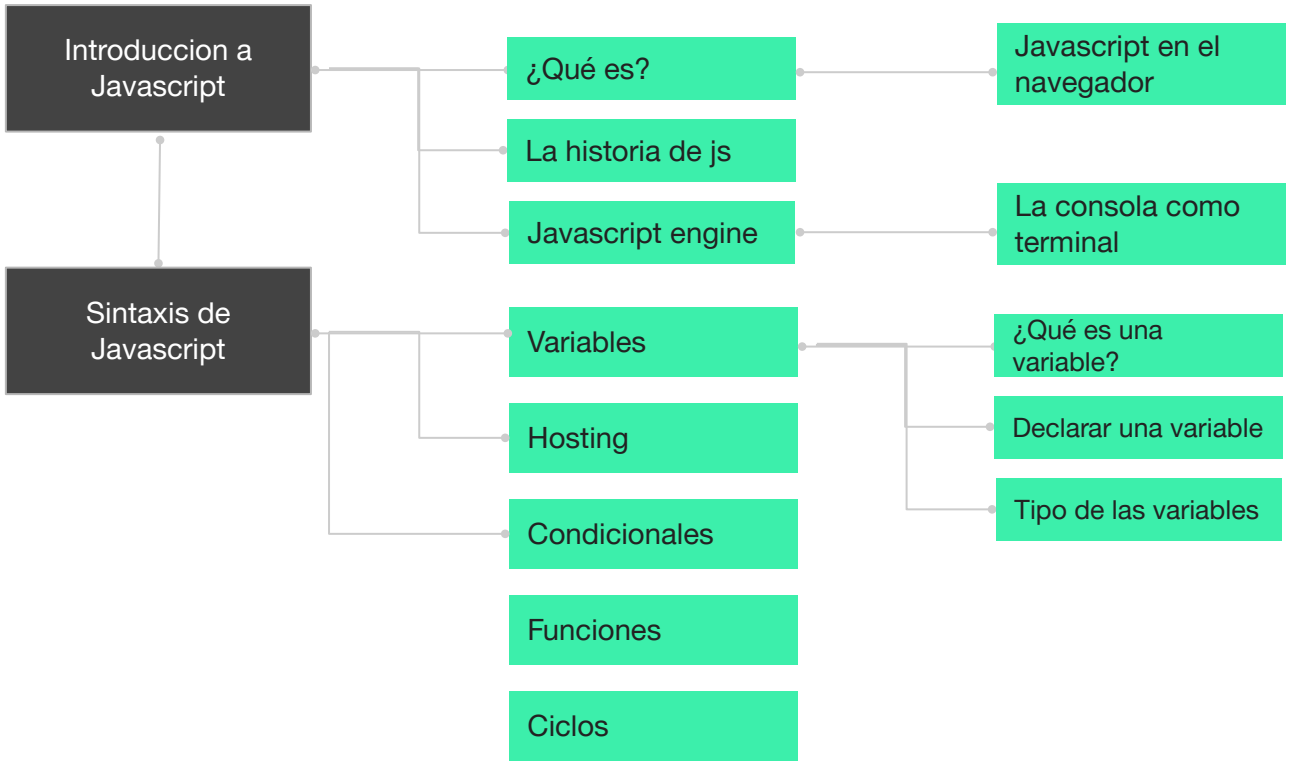
OBJETIVOS DE LA CLASE

- Introducción a Javascript.
- Sintaxis de Javascript.

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS DE LA CLASE: Javascript moderno

¡Para recordar!



Javascript

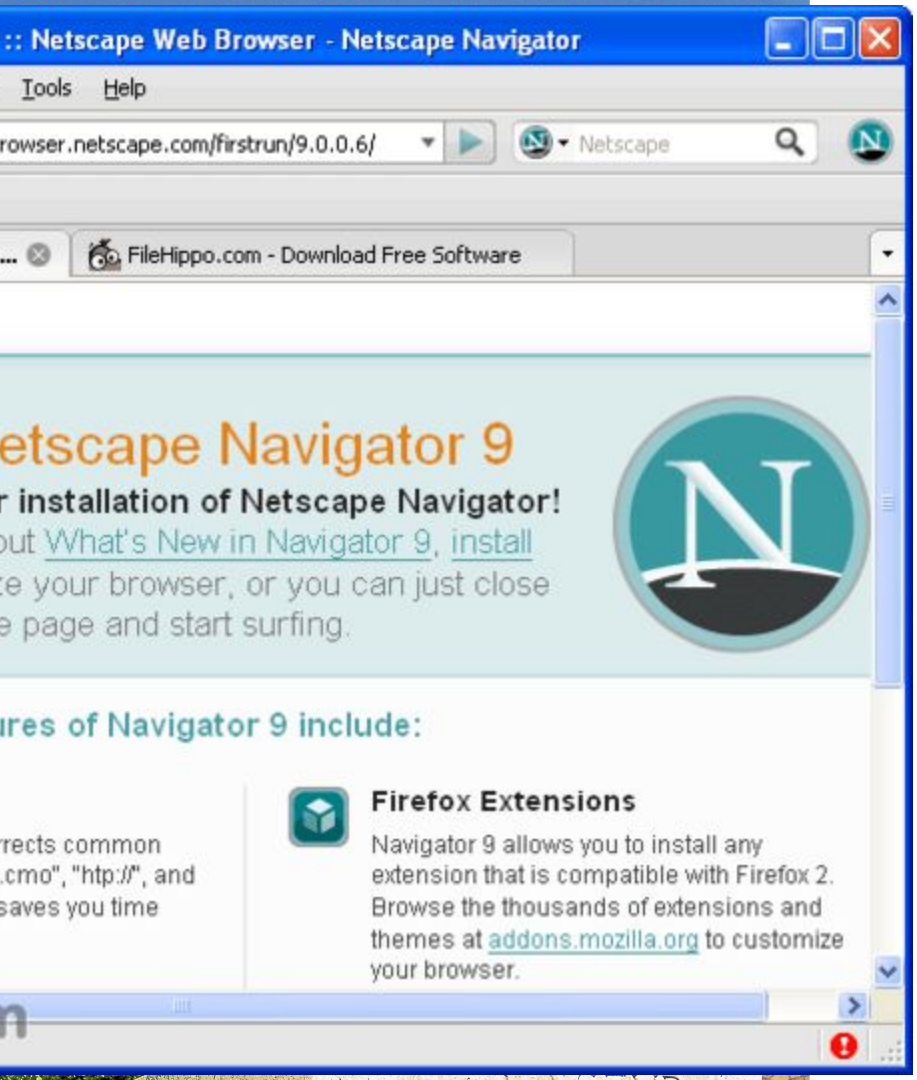
¿QUÉ ES?

- Es un lenguaje de programación que junto a html y css forman los pilares del desarrollo web desde el lado del cliente.
    - Es utilizado para hacer las webs más interactivas
    - Para crear sitios webs como Facebook, Youtube, Linkedin.
    - Se pueden crear sitios webs, aplicaciones móviles, aplicaciones de escritorio, se puede usar desde el lado del servidor y usar en base de datos.

Javascript en el navegador

¿Que puedo hacer?

- Editar contenido de una página html.
- Hacer sitios web responsive.
- Detectar navegador del usuario.
- Crear cookies.
- Validar formularios.
- Animaciones .
- Crear aplicaciones web (corriendo en distintos dispositivos).

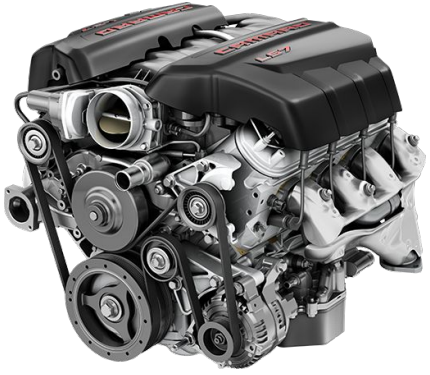


La historia de js

JavaScript fue desarrollado originalmente por Brendan Eich de Netscape. En 1997 los autores propusieron a JavaScript para que fuera adoptado como estándar de la European Computer Manufacturers 'Association ECMA.

En junio de 1997 fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también como un estándar ISO.

Javascript Engine

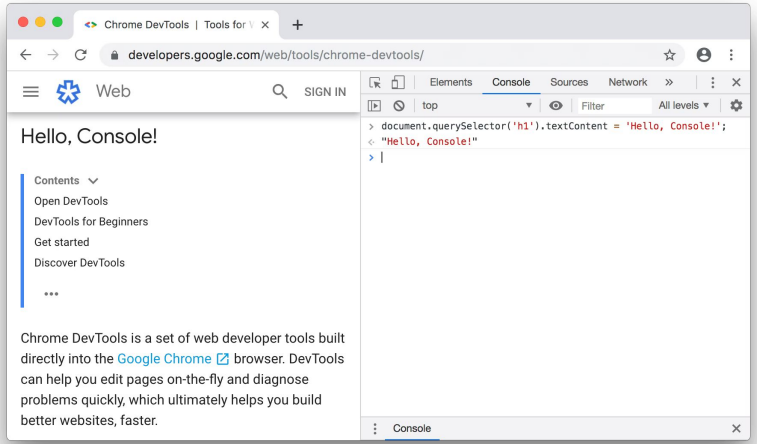


Los motores de javascript son los encargados de que la computadora pueda entender nuestro código javascript. Por lo general en otros lenguajes de programación se necesitan compiladores para que la computadora pueda interpretar el código, en javascript esa es la función de los motores.

Existen diversos motores de javascript, los cuales son definidos como **ECMAScript engines (v8, spidermonkey, chakra).**

Entre los motores de más renombre se encuentra el v8 motor creado por google para potenciar las posibilidades de proceso en su navegador. Este motor fue lanzado en 2008 y está desarrollado en c++.

La consola como terminal



Gracias a los **motores de javascript** podemos correr código javascript en la consola de nuestro navegador:

Veamos las posibilidades que nos da la consola a la hora de correr nuestro codigo!

¿Qué es una variable?

Una variable es un contenedor para un valor, como un número que podríamos usar en una suma, o una cadena que podríamos usar como parte de una oración. Pero una cosa especial acerca de las variables es que los valores que contienen pueden cambiar. Veamos un sencillo ejemplo:

HTML:

```
<button>Presióname</button>
```

javascript:

```
const button = document.querySelector('button');


button.onclick = function() {
 let name = prompt('¿Cuál es tu nombre?');
 alert('¡Hola ' + name + ', encantado de verte!');
}
```

En este ejemplo, al presionar el botón se ejecutan un par de líneas de código. La primera línea muestra un cuadro en la pantalla que pide al lector que ingrese su nombre y luego almacena el valor en una variable. La segunda línea muestra un mensaje de bienvenida que incluye su nombre, tomado del valor de la variable.

Declarar una variable

Para usar una variable, primero debes crearla precisamente, a esto lo llamamos declarar la variable. Para hacerlo, escribimos la palabra clave var o let seguida del nombre con el que deseas llamar a tu variable:

Una vez que hayas declarado una variable, la puedes iniciar con un valor. Para ello, escribe el nombre de la variable, seguido de un signo igual (=), seguido del valor que deseas darle. Por ejemplo

```
var a = 5;
var b = 10;


if (a === 5) {
  let a = 4; // El alcance es dentro del bloque if
  var b = 1; // El alcance es global
}
```

Números

Puedes almacenar números en variables, ya sea números enteros como 30 (también llamados enteros — "integer") o números decimales como 2.456 (también llamados números flotantes o de coma flotante "number").

No es necesario declarar el tipo de las variables en JavaScript, a diferencia de otros lenguajes de programación. Cuando le das a una variable un valor numérico, no incluye comillas:

```
let myAge = 17;
```

Cadenas de caracteres (string)

Las string (cadenas) son piezas de texto. Cuando le das a una variable un valor de cadena, debes encerrarlo entre comillas simples o dobles; de lo contrario, JavaScript intenta interpretarlo como otro nombre de variable.

En JavaScript, puedes escoger entre comillas simple y dobles para envolver tus cadenas. Ambas funcionarán correctamente:

```
let dolphinGoodbye = 'Hasta luego y gracias por todos los peces';


var simp = 'Comillas simples.';
var dobl = "Comillas dobles."
```

Cadenas de caracteres (string)

Escapando caracteres: escapar caracteres significa que les hacemos algo para asegurarnos que sean reconocidos como texto, y no parte del código. En JavaScript, colocamos una barra invertida justo antes del caracter. Intenta ésto:

```
var bigmouth = 'I\'ve got no right to take my place...';
```

Concatenar significa: "unir". Para unir cadenas en JavaScript el símbolo de más (+), el mismo operador que usamos para sumar números, pero en este contexto hace algo diferente.

Vamos a probar un ejemplo en nuestra consola. El resultado de este código es una variable llamada joined, que contiene el valor: "Hello, how are you?" ("Hola, cómo estas?").

```
var multiple = one + one + one + one + two; multiple;
```

Cadenas de caracteres (string)

Concatenar significa: "unir". Para unir cadenas en JavaScript el símbolo de más (+), el mismo operador que usamos para sumar números, pero en este contexto hace algo diferente.

Vamos a probar un ejemplo en nuestra consola. El resultado de este código es una variable llamada joined, que contiene el valor: "Hello, how are you?" ("Hola, cómo estas?").

```
var one = 'Hello, ';
var two = 'how are you?';
var joined = one + two;
joined;
```

En la última instancia del código, unimos dos strings, pero lo puedes hacer con cuantas desees (incluso mezclando strings con variables), mientras que incluyas el símbolo de + entre ellas. Prueba esto:

```
var multiple = one + one + one + one + two;
multiple;

var response = one + 'I am fine — ' + two;
response
```

Cadenas de caracteres (string)

Entonces, ¿qué sucede cuando intentamos agregar (o concatenar) un string y un número? Vamos a probar en la consola:

```
'Front ' + 242;
```

- Podrías esperar que diera un error, pero funciona a la perfección. Tratar de representar un string como un número no tiene sentido, pero representar un número como string si que lo tiene, así que el navegador convierte el número en una string y las muestra juntas.

- Incluso puedes hacer esto con dos números — puedes forar un número para que se convierta en una string envolviéndolo entre comillas. Prueba lo siguiente (estamos utilizando el operador typeof para verificar si la variable es un número o una cadena):

```
var myDate = '19' + '67';
typeof myDate;
```

Cadenas de caracteres (string)

Si tienes una variable numérica, que deseas convertir en una string, pero no cambiar de otra forma, o una variable string, que deseas convertir a número, pero no cambiarla de otra forma, puedes usar las siguientes construcciones:

El objeto [Number] convertirá cualquier cosa que se le pase en un número, si puede. Intenta lo siguiente:

```
var myString = '123';
var myNum = Number(myString);
typeof myNum;
```

Por otra parte, cada número tiene un método llamado toString() que convertirá el equivalente en una string. Prueba esto:

```
var myNum = 123;
var myString = myNum.toString();
typeof myString;
```

Booleanos

Los booleanos son valores verdadero/falso pueden tener dos valores, true o false. Estos, generalmente se utilizan para probar una condición, después de lo cual se ejecuta el código según corresponda. Así, por ejemplo, un caso simple sería:

```
let iAmAlive = true;

Let test = 6 < 3;
```

Arreglos

Un arreglo es un objeto único que contiene múltiples valores encerrados entre corchetes y separados por comas. Una vez que se definen estos arreglos, puedes acceder a cada valor por su ubicación dentro del arreglo.

Prueba estas líneas: Intenta ingresar las siguientes líneas en tu consola:

```
let myNameArray = ['Chris', 'Bob', 'Jim'];
let myNumberArray = [10, 15, 40];

myNameArray[0]; // debería devolver 'Chris'
myNumberArray[2]; // debe devolver 40
```

Los corchetes especifican un valor de índice correspondiente a la posición del valor que deseas devolver. Posiblemente hayas notado que los arreglos en JavaScript tienen índice cero: el primer elemento está en el índice 0.

Objetos

En programación, un objeto es una estructura de código que modela un objeto de la vida real. Puedes tener un objeto simple que represente una caja y contenga información sobre su ancho, largo y alto, o podrías tener un objeto que represente a una persona y contenga datos sobre su nombre, estatura, peso, qué idioma habla, cómo saludarlo, y más.

Intenta ingresar la siguiente línea en tu consola:

```
let dog = { name : 'Spot', breed : 'Dalmatian' };
```

Para recuperar la información almacenada en el objeto, puedes utilizar la siguiente sintaxis:

```
dog.name
```

Tipado dinámico

JavaScript es un "lenguaje tipado dinámicamente", lo cual significa que, a diferencia de otros lenguajes, no es necesario especificar qué tipo de datos contendrá una variable (números, cadenas, arreglos, etc.).

Por ejemplo, si declaras una variable y le das un valor entre comillas, el navegador trata a la variable como una cadena (string):

```
let myString = 'Hello';
let myNumber = '500';
typeof myNumber;
myNumber = 500;
typeof myNumber;
```

Para recuperar la información almacenada en el objeto, puedes utilizar la siguiente sintaxis:

```
dog.name
```

Intenta ingresar las cuatro líneas anteriores en tu consola una por una y ve cuáles son los resultados. Notarás que estamos usando un operador especial llamado `typeof` — esto devuelve el tipo de datos de la variable que escribes después.

Constantes

Muchos lenguajes de programación tienen el concepto de una constante — un valor que, una vez declarado no se puede cambiar. Hay muchas razones por las que querrías hacer esto, desde la seguridad (si un script de un tercero cambia dichos valores, podría causar problemas) hasta la depuración y la comprensión del código (es más difícil cambiar accidentalmente valores que no se deben cambiar y estropear cosas claras).

En los primeros días de JavaScript, las constantes no existían. En JavaScript moderno, tenemos la palabra clave const, que nos permite almacenar valores que nunca se pueden cambiar:

```
const daysInWeek = 7;

const hoursInDay = 24;

const daysInWeek = 7;

daysInWeek = 8;
```

const funciona exactamente de la misma manera que let, excepto que a const no le puedes dar un nuevo valor. Por eso la segunda línea arroja un error:

¿PREGUNTAS?

¡MUCHAS GRACIAS!

OPINA Y VALORA ESTA CLASE