

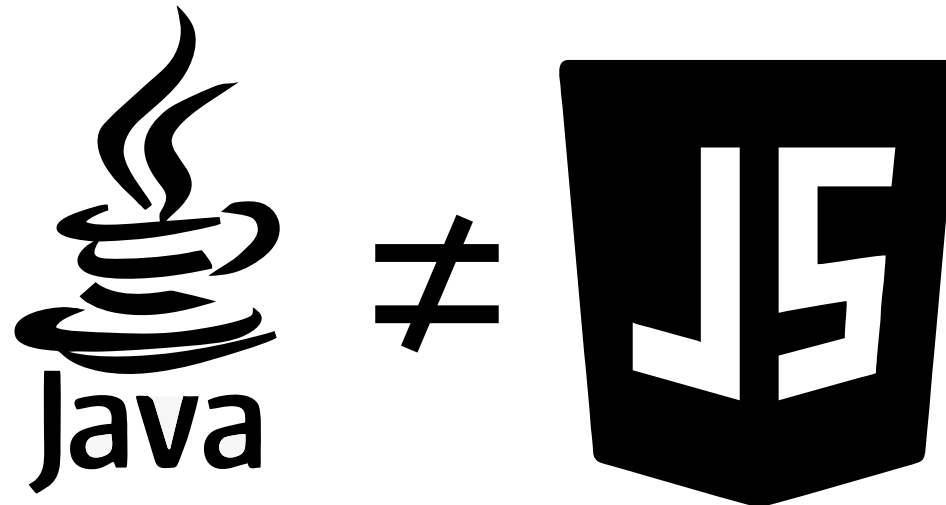


clase n°9

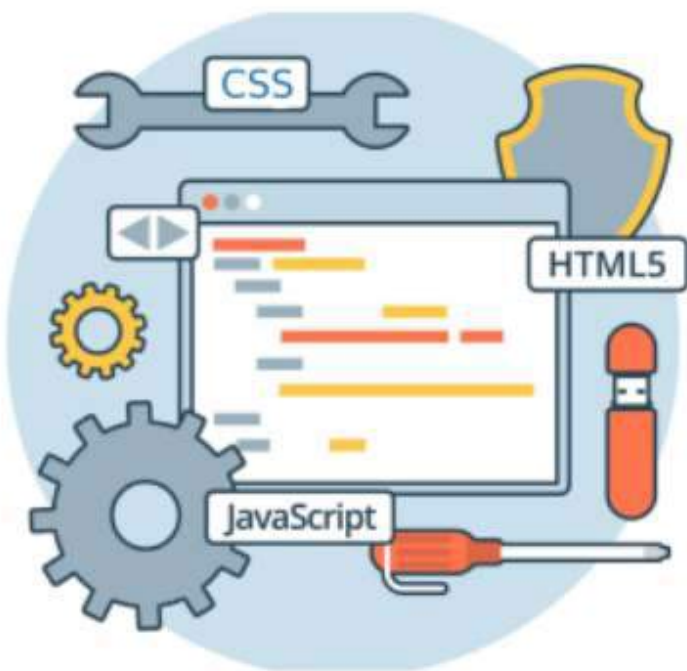
JavaScript

<animate a/> revolución*
programar digital_

se introdujo en 1995 como una forma de agregar programas a páginas web en el navegador Netscape Navigator. En su momento fue una idea novedosa. En los primeros días de la World Wide Web, HTML era bastante simple, y bastante fácil de aprender casi todo lo que se necesitaba saber para agrupar páginas web.



JavaScript consiste en un lenguaje de programación interpretado, que habitualmente se utiliza en sitios web para ejecutar acciones en el lado del cliente, estando **embebido** en el código fuente de la página web. Un sistema **embebido** (también conocido como “empotrado”, “incrustado” o “integrado”) es un sistema de computación diseñado para realizar funciones específicas.



JavaScript permite, en una página web, crear elementos como cuadros de diálogo, recoger información entrada por el usuario y mandarla al servidor para ser procesada.

Este lenguaje pasó varias etapas hasta poder convertirse en lo que hoy es y tener un estándar global.

observaciones

- ⚙ JavaScript (que no debe confundirse con Java)
- ⚙ JavaScript es un lenguaje que distingue entre mayúsculas y minúsculas. Esto significa que el lenguaje considera que las mayúsculas son diferentes de sus equivalentes en minúsculas.
- ⚙ Las palabras clave en JavaScript son minúsculas
- ⚙ Los archivos JavaScript por sí solos no pueden ejecutarse directamente desde el navegador; Es necesario incrustarlos en un documento HTML.

¿conocés la sintaxis de javascript?



```
1 let lenguaje = 'JavaScript'
2 let company = {
3   name: 'EDteam',
4   slogan: 'Nunca te detengas',
5   founded: 2015
6 }
7 console.log(company.name)
8 // 'EDteam'
9 const getMajorNumber = (a,b) => {
10   if (a > b) { return a }
11   else { return b }
12 }
13 getMajorNumber(4,6)
14 // 6
```

Las variables se declaran con **let** (no hay que indicar el tipo de dato)

Los objetos encierran entre llaves parejas con el formato **propiedad:valor** separadas por comas.

console.log() imprime en consola la expresión entre los paréntesis.

Para obtener el valor de una propiedad de un objeto se usa **objeto.propiedad**

Condicionales (**if.else**)


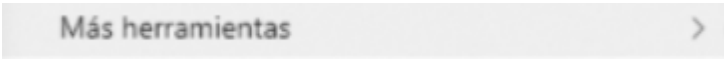
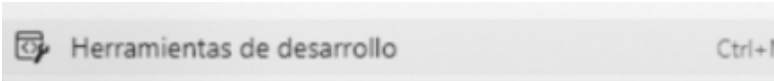
Comentarios (**líneas 8 y 14**)

Ejecución de la **función**

Definición de función (se recomienda usar constantes con **const**)



A diferencia de Python, los saltos de línea e indentación no son parte de la sintaxis.

- ✳ Ingresamos al navegador y ponemos una pagina en blanco 
- ✳ En la barra de direcciones ponemos: 
- ✳ Luego entramos a las herramientas de desarrolladores 
- ✳ Dentro de la consola, empecemos a conocer los primeros pasos
- ✳ Básicamente la consola nos permite comunicarnos con el navegador, a través de JavaScript

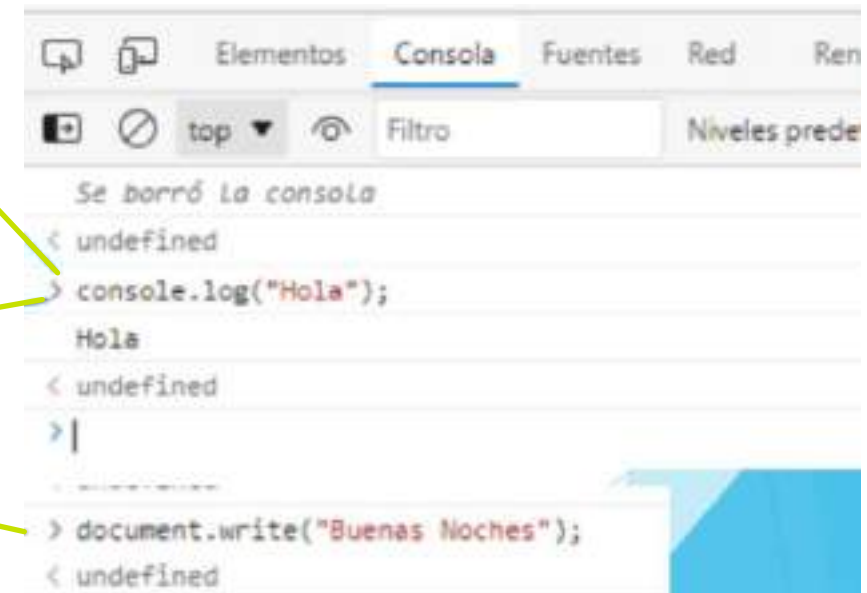
Nos permite trabajar básicamente en la consola, esto el usuario no lo va a ver.

Si quiero mostrar un texto por pantalla, o alterar un definido en html.

`log()` Muestra un mensaje en la consola web.

Consola muéstrame lo que tienes entre paréntesis.

`document.write ("TEXTO")`



JS. podemos añadir características interactivas a tu sitio web, (por ejemplo, juegos, eventos que ocurren cuando los botones son presionados o los datos son introducidos en los formularios, efectos de estilo dinámicos, animación, y mucho más)

Todos los navegadores web modernos y todos los demás entornos de JavaScript admiten la escritura de mensajes en una consola utilizando un conjunto de métodos de registro. El más común de estos métodos es `console.log()`.

La función `console.log()` se utiliza principalmente para fines de depuración. Entramos a la consola y tipiamos: `console.log("Hello");` presionamos enter.

Tipeamos y probamos **`var foo = "bar";`**
`console.log(foo)`

Si desea registrar dos o más valores, simplemente sepárelos con comas.

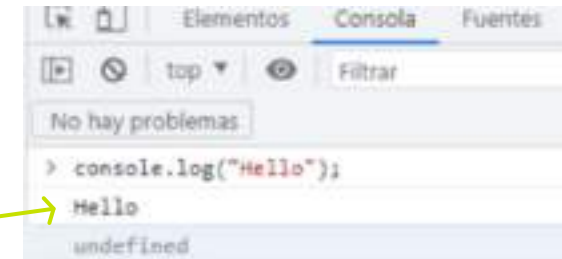
`var greet = "Hello", who = "World";`
`console.log("%s, %s!", greet, who);`

Que pasa si borramos en el `console.log "%s, %s!",`

limpiar la consola

Cuando hay un exceso de mensajes, podemos limpiar la consola de nuestro navegador simplemente haciendo clic con el botón derecho y seleccionando `clear console`.

También lo podemos hacer a través de código tecleando **`clear()`**.



variable a la que no se le ha asignado valor, o no se ha declarado

```
> var greet = "Hello", who = "World";  
    console.log("%s, %s!", greet, who);  
Hello, World!  
← undefined
```

```
> var greet = "Hello", who = "World";  
    console.log(greet, who);  
Hello World  
← undefined
```

Los objetos son, una de las características menos entendidas en JavaScript, dado que su implementación tiene algunas diferencias importantes con muchos lenguajes de programación más tradicionales.

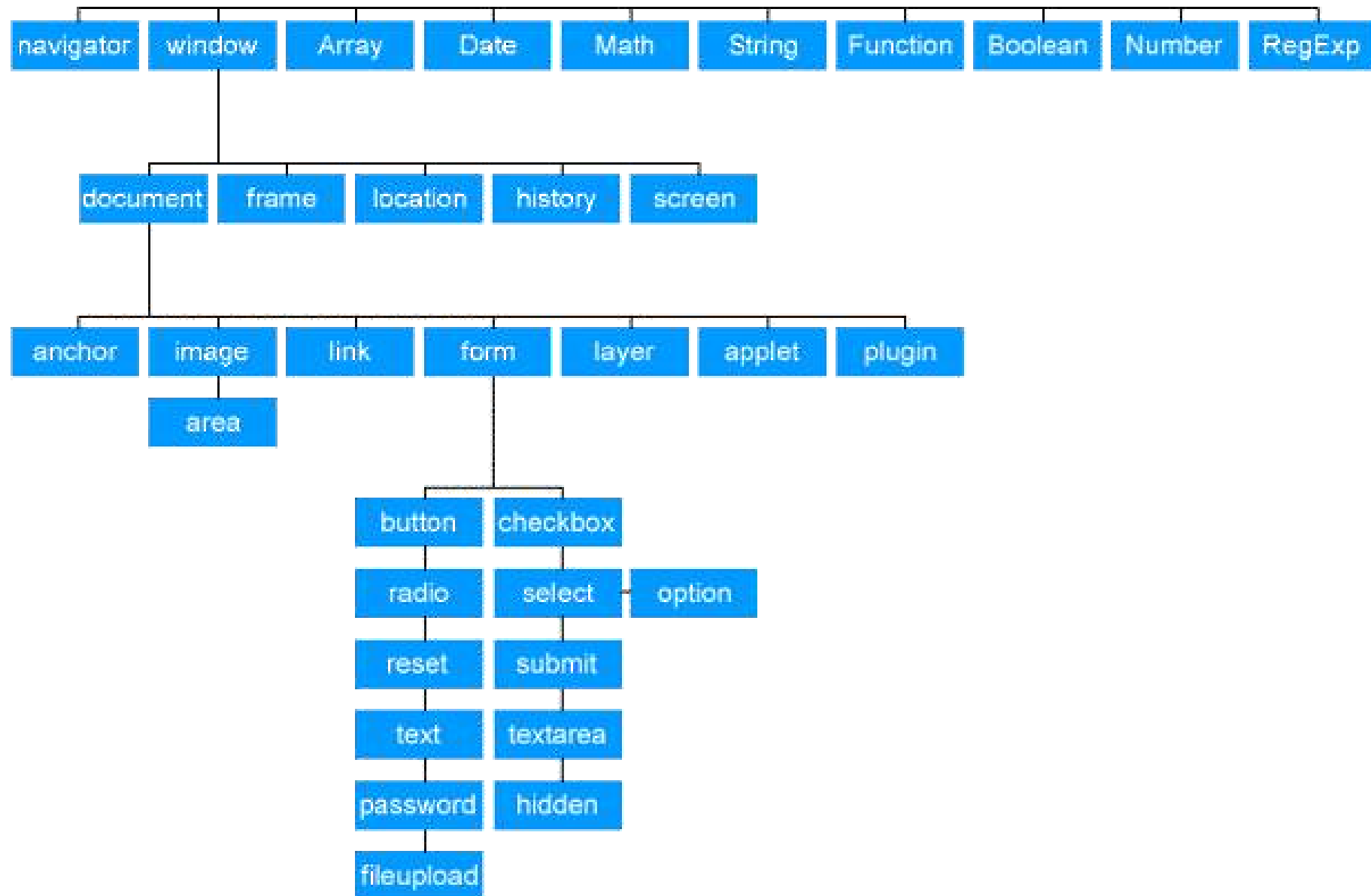
Un **objeto** en **JavaScript** es un contenedor de propiedades, donde una propiedad tiene un nombre y un valor. El nombre de una propiedad **puede** ser una cadena de caracteres, incluso una vacía. El valor de la propiedad puede ser cualquier valor que podamos utilizar en **JavaScript**, excepto undefined.

Recordemos que window, el objeto global en JavaScript, tiene sus propiedades y métodos. La función que venimos usando para mostrar mensajes por pantalla, alert, es un método de window. Cuando escribimos alert('Hola'); en realidad estamos invocando **window.alert('Hola');**

Es decir, alert y window.alert son lo mismo para el navegador. El hecho de que no sea necesario escribir window cuando escribimos alert obedece a que si se invoca una función que no ha sido definida de otra manera, se considera que es un método del objeto global, es decir, un método de window.

En estas jerarquía podemos pensar en objetos predefinidos de JavaScript (como Math, Date, String, RegExp, etc., incluido el objeto window). También podemos pensar en el objeto document que contiene toda la información relativa a la estructura del documento HTML, lo que llamamos el DOM. No obstante, debemos diferenciar entre la jerarquía del DOM, teniendo en cuenta que el DOM existe de forma independiente a JavaScript, y la jerarquía de objetos JavaScript, aunque guarden cierta similitud organizativa. Podemos ver una página web como una colección de objetos. Por ejemplo, para JavaScript un formulario es un objeto, una imagen es un objeto, etc. Los objetos tienen propiedades, métodos y eventos asociados.

JERARQUÍA DE OBJETOS JAVASCRIPT



Los objetos se organizan conforme a una jerarquía de forma que heredan métodos o propiedades de sus objetos padre, e incluso el nombre de un objeto se crea a partir de sus objetos padre.

Todo documento HTML dispone de los siguientes objetos en la jerarquía de objetos JavaScript:

- ✱ **navigator:** tiene propiedades relacionadas con el nombre y la versión del navegador, protocolos de transferencia permitidos por el navegador (mime types) y sobre plugins instalados.
- ✱ **window:** considerado habitualmente el objeto global o de máximo nivel. Tiene propiedades relacionadas con la ventana del navegador. En caso de uso de frames (“subventanas”) hay un objeto window por cada “ventana hija” que exista.
- ✱ **document:** tiene propiedades relacionadas con el documento como título, links, formularios, etc.
- ✱ **location:** tiene propiedades relacionadas con la URL actual.
- ✱ **history:** tiene propiedades relacionadas con URLs previamente visitadas



Además de estos objetos en el documento HTML existirán más objetos según su contenido: objetos imágenes, objetos link, objetos formulario, etc

como se nombrar los objetos en la jerarquía JavaScript

Para nombrar los objetos en la jerarquía de objetos JavaScript debemos tener en cuenta las siguientes reglas:

- 1) El nombre de un objeto que desciende de otro en la jerarquía JavaScript incluye el nombre de los objetos padre de la jerarquía. Por ejemplo el objeto document podemos nombrarlo como window.document. Un objeto hijo es a su vez un objeto y una propiedad del objeto padre. document es a su vez un objeto y una propiedad de window.
- 2) Dado que todos los objetos que podemos usar en el código descienden de window, normalmente omitiremos el uso de window a la hora de nombrar un objeto. Por eso escribiremos por ejemplo document.body en lugar de window.document.body, aunque ambas formas son válidas.
- 3) JavaScript organiza de forma automática ciertos objetos de naturaleza “múltiple” en arrays. Por ejemplo, una página web puede contener varios formularios. JavaScript, de forma automática, crea un array de objetos cuyo nombre es forms, siendo el primer formulario el de índice 0 y sucesivamente el 1, 2, etc. representan los siguientes formularios que aparezcan en el documento HTML por orden de aparición. Nos podemos referir a un formulario como window.document.forms[0], o más frecuentemente: document.forms[0]

entre los arrays de objetos que crea JavaScript automáticamente tenemos:

forms: array con todos los formularios existentes en el documento HTML.

elements: array para cada formulario con los objetos que conforman dicho formulario (esto comprende objetos text, button, checkbox, hidden, radio, textarea, etc.). Si el primer formulario de una web tiene 3 input de tipo texto nos podemos referir al último de ellos como forms[0].elements[2]

images: array con todas las imágenes existentes en el documento HTML.

links: array con todos los links (tag HTML a) existentes en el documento HTML.



Para comprobar cómo podemos acceder a los objetos en la jerarquía de JavaScript escribe este código y comprueba los resultados (hemos incluido la ruta de dos imágenes, cámbiala si es necesario):

```
<script type="text/javascript">
function ejemplo() {
window.document.images[0].style.border = 'solid blue 10px';
document.images[1].style.border = 'solid red 10px';
document.links[0].style.color = 'grey';
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue; margin:20px;" id="pulsador" onclick="ejemplo()"><a href="#"> Probar</a> </div>


</body>
```

los objetos son una colección de propiedades

Para construir objetos podemos hacerlo de dos maneras:

- ⚙️ Objetos declarativos o literales: podemos crear objetos sin necesidad de un constructor o instanciar una clase, para esto solo declaramos el objeto y sus propiedades
- ⚙️ Objetos contruidos: JavaScript es un lenguaje libre de clases, pero tenemos el keyword new, el cual nos permite crear un nuevo objeto, de esta manera podemos utilizar una función que cumpla el rol del constructor.

```
const camilo = {  
  nombre: 'Camilo',  
  edad: 22,  
  sexo: 'masculino',  
  pasatiempos: ['patinar', 'bailar'],  
  hablar: function(){  
    return `hola soy ${this.nombre}, y tengo ${this.edad} años`;  
  }  
}  
  
console.log(camilo);
```

```
function Persona(nombre, edad, sexo, pasatiempos) {  
  this.nombre = nombre;  
  this.edad = edad;  
  this.sexo = sexo;  
  this.pasatiempos = pasatiempos;  
  this.hablar = function() {  
    return `hola soy ${this.nombre}, y tengo ${this.edad} años`;  
  };  
}  
  
const camilo = new Persona('camilo', 22, 'masculino', ['patinar', 'bailar']);  
  
console.log(camilo)
```



JavaScript no almacena el contenido de las propiedades dentro de los objetos, este solo guarda el nombre de las propiedades, con referencias a donde están almacenados los valores.

revolución*
digital_