

**Programación Web**

**Trabajo Practico 4**

**Nombre y apellido del alumno: Matias Hernan Braga**

**Turno: noche**

**Año: 2022**

1. ***¿Qué es javascript? ¿para qué capa se utiliza generalmente y cuales son sus principales características?***

Es un lenguaje de secuencias de comandos que te permite crear contenido de actualización dinámica, controlar multimedia, animar imágenes, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, entre otras cosas.

Javascript se encuentra en la capa de comportamiento y sus principales características son:

**Imperativo y estructurado**

JavaScript es compatible con gran parte de la estructura de programación de C (por ejemplo, sentencias if, bucles for, sentencias switch, etc.). Con una salvedad, en parte: en C, el ámbito de las variables alcanza al bloque en el cual fueron definidas; sin embargo JavaScript no es compatible con esto, puesto que el ámbito de las variables es el de la función en la cual fueron declaradas. Esto cambia con la versión de ECMAScript 2015, ya que añade compatibilidad con block scoping por medio de la palabra clave let. Como en C, JavaScript hace distinción entre expresiones y sentencias. Una diferencia sintáctica con respecto a C es la inserción automática de punto y coma, es decir, en JavaScript los puntos y coma que finalizan una sentencia pueden ser omitidos.

**Dinámicos**

**Tipado dinámico**

Como en la mayoría de lenguajes de scripting, el tipo está asociado al valor, no a la variable. Por ejemplo, una variable x en un momento dado puede estar ligada a un número y más adelante, religada a una cadena. JavaScript es compatible con varias formas de comprobar el tipo de un objeto, incluyendo duck typing.25​ Una forma de saberlo es por medio de la palabra clave typeof.

**Objetual**

JavaScript está formado casi en su totalidad por objetos. Los objetos en JavaScript son arrays asociativos, mejorados con la inclusión de prototipos (ver más adelante). Los nombres de las propiedades de los objetos son claves de tipo cadena: obj.x = 10 y obj['x'] = 10 son equivalentes, siendo azúcar sintáctico la notación con punto. Las propiedades y sus valores pueden ser creados, cambiados o eliminados en tiempo de ejecución. La mayoría de propiedades de un objeto (y aquellas que son incluidas por la cadena de la herencia prototípica) pueden ser enumeradas por medio de la instrucción de bucle for... in. JavaScript tiene un pequeño número de objetos predefinidos como son Function y Date.

Evaluación en tiempo de ejecución

JavaScript incluye la función eval que permite evaluar expresiones expresadas como cadenas en tiempo de ejecución. Por ello se recomienda que eval sea utilizado con precaución y que se opte por utilizar la función JSON.parse() en la medida de lo posible, pues puede resultar mucho más segura.

**Funcional**

**Funciones de primera clase**

A las funciones se les suele llamar ciudadanos de primera clase; son objetos en sí mismos. Como tal, poseen propiedades y métodos, como .call() y .bind().26​ Una función anidada es una función definida dentro de otra. Esta es creada cada vez que la función externa es invocada. Además, cada función creada forma una clausura; es el resultado de evaluar un ámbito conteniendo en una o más variables dependientes de otro ámbito externo, incluyendo constantes, variables locales y argumentos de la función externa llamante. El resultado de la evaluación de dicha clausura forma parte del estado interno de cada objeto función, incluso después de que la función exterior concluya su evaluación.

**Prototípico**

**Prototipos**

JavaScript usa prototipos en vez de clases para el uso de herencia.28​ Es posible llegar a emular muchas de las características que proporcionan las clases en lenguajes orientados a objetos tradicionales por medio de prototipos en JavaScript.29​

Funciones como constructores de objetos

Las funciones también se comportan como constructores. Prefijar una llamada a la función con la palabra clave new crear una nueva instancia de un prototipo, que heredan propiedades y métodos del constructor (incluidas las propiedades del prototipo de Object).30​ ECMAScript 5 ofrece el método Object.create, permitiendo la creación explícita de una instancia sin tener que heredar automáticamente del prototipo de Object (en entornos antiguos puede aparecer el prototipo del objeto creado como null).​ La propiedad prototype del constructor determina el objeto usado para el prototipo interno de los nuevos objetos creados. Se pueden añadir nuevos métodos modificando el prototipo del objeto usado como constructor. Constructores predefinidos en JavaScript, como Array u Object, también tienen prototipos que pueden ser modificados. Aunque esto sea posible se considera una mala práctica modificar el prototipo de Object ya que la mayoría de los objetos en Javascript heredan los métodos y propiedades del objeto prototype, objetos los cuales pueden esperar que estos no hayan sido modificados.3

**Otras características**

**Entorno de ejecución**

JavaScript normalmente depende del entorno en el que se ejecute (por ejemplo, en un navegador web) para ofrecer objetos y métodos por los que los scripts pueden interactuar con el "mundo exterior". De hecho, depende del entorno para ser capaz de proporcionar la capacidad de incluir o importar scripts (por ejemplo, en HTML por medio del tag <script>). (Esto no es una característica del lenguaje, pero es común en la mayoría de las implementaciones de JavaScript.)

**Funciones variádicas**

Un número indefinido de parámetros pueden ser pasados a la función. La función puede acceder a ellos a través de los parámetros o también a través del objeto local arguments. Las funciones variádicas también pueden ser creadas usando el método .apply().

**Funciones como métodos**

A diferencia de muchos lenguajes orientados a objetos, no hay distinción entre la definición de función y la definición de método. Más bien, la distinción se produce durante la llamada a la función; una función puede ser llamada como un método. Cuando una función es llamada como un método de un objeto, la palabra clave this, que es una variable local a la función, representa al objeto que invocó dicha función.

**Arrays y la definición literal de objetos**

Al igual que muchos lenguajes de script, arrays y objetos (arrays asociativos en otros idiomas) pueden ser creados con una sintaxis abreviada. De hecho, estos literales forman la base del formato de datos JSON.

**Expresiones regulares**

JavaScript también es compatible con expresiones regulares de una manera similar a Perl, que proporcionan una sintaxis concisa y poderosa para la manipulación de texto que es más sofisticado que las funciones incorporadas a los objetos de tipo string.

**Extensiones específicas del fabricante**

JavaScript se encuentra oficialmente bajo la organización de Mozilla Foundation, y periódicamente se añaden nuevas características del lenguaje. Sin embargo, sólo algunos motores JavaScript son compatibles con estas características:

* Las propiedades get y set (también compatibles con WebKit, Opera,34​ ActionScript y Rhino).
* Cláusulas catch condicionales.
* Protocolo iterador adoptado de Python.
* Corrutinas también adoptadas de Python.
* Generación de listas y expresiones por comprensión también adoptado de Python.
* Establecer el ámbito a bloque a través de la palabra clave let.
* Desestructuración de arrays y objetos (forma limita de emparejamiento de patrones).
* Expresiones concretas en funciones (function(args) expr).
* ECMAScript para XML (E4X), una extensión que añade compatibilidad nativa XML a ECMAScript.

1. ***Mencione las diferentes formas en la cual podemos incluir código javascript en nuestro html, realice un ejemplo.***

* **Agregar JavaScript directamente a un archivo HTML**

La primera forma de insertar JavaScript en HTML es directa. Puedes hacerlo utilizando la etiqueta <script> </script> que debe envolver todo el código JS que escribas.

Se puede agregar el código JS:

* entre las etiquetas <head>
* entre las etiquetas <body>

Dependiendo de dónde agregues el código JavaScript en tu archivo HTML, la carga será diferente. Por lo general se recomienda agregarlo en la sección <head> para que permanezca separado del contenido de tu archivo HTML. Pero colocarlo dentro de <body> puede ayudar a mejorar la velocidad de carga, ya que el contenido del sitio web se cargará más rápido, y solo después de eso se procesará el JavaScript.

Para este ejemplo, echemos un vistazo al siguiente archivo HTML que debe mostrar la hora actual:

<!DOCTYPE html>

<html lang="en-US">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<script>JAVASCRIPT IS USUALLY PLACED HERE</script>

<title>Time right now is: </title>

</head>

<body>

<script>JAVASCRIPT CAN ALSO GO HERE</script>

</body>

</html>

En este momento, el código anterior no contiene JavaScript y, por lo tanto, no puede mostrar la hora. Podemos agregar el siguiente código para asegurarnos de que muestre la hora correcta:

var time = new Date();

console.log(time.getHours() + ":" + time.getMinutes() + ":" + time.getSeconds());

Envolveremos este código con las etiquetas <script> y </script> y lo colocaremos en la cabecera (head) del código HTML para garantizar que cada vez que se cargue la página, se genere una alerta que muestre la hora actual al usuario. Así es como se verá el archivo HTML después de agregar el código:

<!DOCTYPE html>

<html lang="en-US">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Time right now is: </title>

<script>

var time = new Date();

console.log(time.getHours() + ":" + time.getMinutes() + ":" + time.getSeconds());

</script>

</head>

<body>

</body>

</html>

Si quieres mostrar la hora dentro del cuerpo (body) de la página, deberás incluir la script dentro de las etiquetas <body> de la página HTML. Así es como se verá el código cuando lo hagas:

<!DOCTYPE html>

<html lang="en-US">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Time right now is: </title>

</head>

<body>

<script>

let d = new Date();

document.body.innerHTML = "<h1>Time right now is: " + d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()

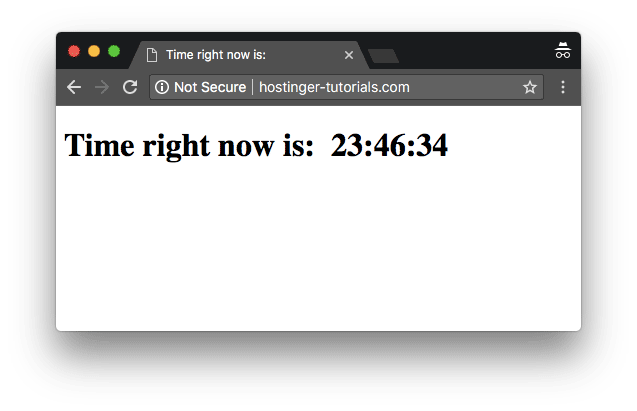
"</h1>"

</script>

</body>

</html>

Así es como se vería el resultado final:



* **Agregar código JavaScript a un archivo separado**

A veces, insertar JavaScript en HTML directamente no parece ser la mejor opción. Sobre todo porque algunos scripts de JS deben utilizarse en varias páginas, por lo tanto, es mejor mantener el código JavaScript en archivos separados. Es por esto que la forma más aceptable de incluir JavaScript en HTML es importando archivos externos. Se puede hacer referencia a estos archivos desde adentro de los documentos HTML, al igual que hacemos referencia a los documentos CSS. Algunos de los beneficios de agregar código JS en archivos separados son:

* Cuando los códigos HTML y JavaScript están separados, se cumple el principio de diseño de separación y hace que todo sea mucho más sostenible y reutilizable.
* La legibilidad y el mantenimiento del código es mucho más fácil.
* Los archivos JavaScript en caché mejoran el rendimiento general del sitio web al disminuir el tiempo que tardan las páginas en cargarse.

Podemos hacer referencia al archivo JavaScript desde HTML así:

<!DOCTYPE html>

<html lang="en-US">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Time right now:</title>

</head>

<body>

</body>

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

</html>

El contenido del archivo myscript.js será:

let d = new Date();

document.body.innerHTML = "<h1>Time right now is: " + d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()</h1>"

1. ***Realice una calculadora en javascript utilizando controles de html***

<input type="text" id="n11" placeholder="n1">

<input type="text" id="n22" placeholder="n2">

<select id="op">

<option value="+">+</option>

<option value="-">-</option>

<option value="\*">\*</option>

<option value="/">/</option>

</select>

<button id="calcular2">Calcular</button>

<output id="r2"></output>

<script>

document.querySelector('#calcular2').addEventListener('click', () =>{

const n1 = parseInt(document.querySelector('#n11').value);

const n2 = parseInt(document.querySelector('#n22').value);

const op2 = document.querySelector('#op').value;

let r;

if(op2 == '+'){

r = n1 + n2;

}else if( op2 == '-'){

r = n1 - n2;

}else if(op2 == '\*'){

r = n1 \* n2;

}else if(op2 == '/'){

r = n1 / n2;

}

document.querySelector('#r2').innerHTML = r;

});

</script>