Universidad CENFOTEC
SOMOS LO QUE SABEMOS

Tarea #1

Fundamentos de programación web

Estudiantes

Alejandro Alvarado Lobo

Catherine Coto Vargas

Profesor

Francisco Jose Jimenez Bonilla

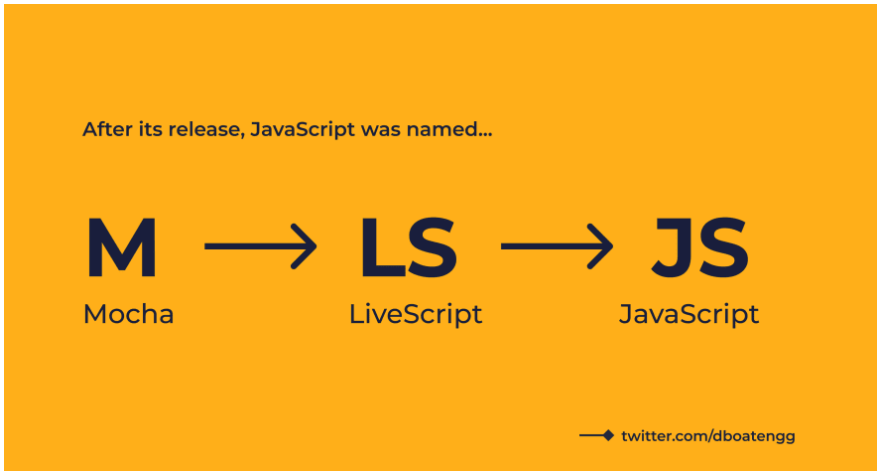
Segundo cuatrimestre
2024

1. ¿Escriba la historia del lenguaje Java Script?

Para hablar de JavaScript y su historia como lenguaje de programación, debemos remontarnos hacia el año 1995 donde Brendan Eich desarrolló este lenguaje que en su momento se llamó Mocha, luego pasó a LiveScript, y terminó por llamarse JavaScript. Aunque su nombre pudiera confundirse, no tiene nada que ver con Java en sí, sin embargo se dice que la similaridad de su nombre ayudó a darle popularidad a este nuevo lenguaje. JavaScript nace de la mano de Netscape Navigator 2.0 para dar interactividad a las páginas web de dicho navegador, el cuál estaba en medio de una competencia con Microsoft y su navegador Internet Explorer.

Ganando éxito, JavaScript buscó estandarizar su código y en 1997 junto a ECMA International, lanzó la estandarización ECMAScript, la cual es la base del lenguaje de programación. A partir de ese momento JavaScript ha evolucionado bastante junto a múltiples versiones de ECMAScript, dando nuevas características y mejoras.

Aun cuando JavaScript nació de Netscape, la caída del navegador en el año 2003 no hizo que el lenguaje de programación perdiera popularidad, al contrario con el paso del tiempo se ha afianzado en el mundo de la tecnología. Tiene una numerosa comunidad entregada a su aprendizaje y enseñanza a base de cursos, prácticas a través de todo el internet. Desde sus inicios ha evolucionado no solo para dar interactividad a las páginas, sino ser parte fundamental de las mismas, además del desarrollo aplicaciones web y móviles.



2. ¿Por qué se debe aprender Java Script?

JavaScript es un lenguaje de programación que debemos aprender ya que es fundamental para el desarrollo web del lado del cliente, además que con herramientas como Node.js se puede usar JS para el desarrollo del lado del servidor, lo cual nos permite crear aplicaciones web completas utilizando solo un lenguaje.

Por otra parte su numerosa comunidad y popularidad, hace que su demanda laboral genere una gran cantidad de oportunidades profesionales, las cuales nos impulsan a conocer no solo sobre JS sino también sobre otros conocimientos ligados a este.

Por último, su innovación constante le permite al lenguaje estar evolucionando cada vez más para ajustarse a las nuevas necesidades de los usuarios, tanto los programadores de las aplicaciones y páginas web.

3. ¿Cuál es la relación entre HTML y JavaScript?

HTML y JS son lenguajes muy diferentes entre sí, sin embargo se complementan de una gran manera.

Primero HTML es la estructura estándar para crear páginas Web. Se utiliza para marcar el contenido tal como imágenes, párrafos, incluso enlaces y elementos multimedia. Por otra parte JavaScript como comentamos en preguntas anteriores, es utilizado para que los elementos tomados del HTML puedan ser interactivos agregando funciones como: animaciones, botones, formularios, hasta efectos visuales. El contenido JavaScript puede ser agregado de varias formas al archivo HTML, como por ejemplo con las etiquetas <script>.

Teniendo esto en cuenta, se puede entender la profunda relación que estos 2 lenguajes tienen, dando una mayor y mejor experiencia los usuarios de las páginas web, tanto a las personas que crean dichas herramientas.

4. ¿En qué beneficia usar Bootstrap para sitios y aplicaciones web en JS?

Bootstrap ofrece una serie de herramientas y componentes para desarrollar sitios web, las cuales combinadas con JavaScript pueden ser complementadas y potenciarlas. Como por ejemplo:

1- Responsividad: Bootstrap está diseñado para ser totalmente receptivo lo que resulta en páginas web y aplicaciones que se adaptan en una variedad de dispositivos y pantallas, desde computadoras hasta móviles.

2- Bootstrap posee una gran gama de componentes como: botones, formularios, barras de navegación lo que hace que la programación con JavaScript resulte mucho más sencillo programarlas, incluso modificarlas a conveniencia.

3- Al igual que JavaScript tiene una gran comunidad activa de desarrolladores que contribuyen con actualizaciones y mejoras del sistema. Además de una gran cantidad de recursos disponibles para aprender y sacar provecho de esta gran biblioteca.

5. ¿Qué semejanza y diferencia tienen los lenguajes web PHP y Java Script?

PHP es un lenguaje de programación que principalmente es utilizado para desarrollar aplicaciones web del lado del servidor (back-end) y justo acá recae una de las diferencias más notables con JavaScript, que este último se ejecuta principalmente en el navegador web del cliente (front-end).

Por otra parte, otra diferencia podría ser su sintaxis; ya que, aunque los 2 lenguajes trabajan con variables y funciones, tienen diferencias significativas en términos generales de estructura.

Ahora bien, en la parte de las semejanzas cabe destacar que ambos son lenguajes de programación de uso general que se utilizan en gran una variedad de aplicaciones web, solamente que como mencionamos anteriormente se especializan en partes diferentes de la página o aplicación. Además que ambos trabajan junto con HTML y CSS para manipular la estructura y el estilo, incluso trabajar con componentes como formularios o botones.

```php
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de PHP</title>
</head>
<body>
    <h1>Ejemplo de PHP</h1>

    <?php
        // Esto es un comentario en PHP
        echo "¡Hola, mundo!";
    ?>
</body>
</html>
```

En la imagen anterior se puede confirmar lo diferente de la sintaxis de PHP, sin embargo la similitud con JS al estar trabajando en medio del HTML como comentamos.

6. ¿Cite 3 formas en que se puede agregar código JS en una página web?

Cada técnica tiene un propósito para cada una de las prácticas o formas en las que se puede agregar JS a los HTML. El método interno es muy utilizado para script de una sola página ya que sirve para mantener el

código limpio y organizado y el método externo es ideal, cuando se reutiliza el mismo script en varias páginas.

1. JavaScript (en línea): Usando el atributo onclick, para insertar el código JavaScript directamente dentro de las etiquetas HTML.

```html
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Inline</title>
</head>
<body>
    <button onclick="alert('Hola, Mundo!')">Haz clic aquí</button>
</body>
</html>
```

2. JavaScript Interno (en el mismo archivo HTML): Se escribe el código JavaScript dentro de la sección <script> dentro de un archivo HTML. Esto se hace comúnmente dentro del <head> o al final del <body>.

```html
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Interno</title>
    <script>
        function mostrarMensaje() {
            alert('Hola, Mundo!');
        }
    </script>
</head>
<body>
    <button onclick="mostrarMensaje()">Haz clic aquí</button>
</body>
</html>
```

3. JavaScript Externo (en un archivo diferente): el código JavaScript se puede escribir en un archivo diferente con la extensión.js y luego enlazarlo al archivo HTML mediante la etiqueta <script> con el atributo src.

Archivo HTML

```html
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Externo</title>
    <script src="script.js"></script>
</head>
<body>
    <button onclick="mostrarMensaje()">Haz clic aquí</button>
</body>
</html>
```

Archivo Script.js

```javascript
function mostrarMensaje() {
    alert('Hola, Mundo!');
}
```

7. ¿Cuál es la función principal de la consola en JS?

En JavaScript, la consola tiene varias funciones, pero la principal es proporcionar una interfaz para que los desarrolladores interactúen con JavaScript en un entorno de desarrollo.

Se utiliza principalmente para mostrar o encontrar errores en el código y que sea más fácil resolverlo.

8. ¿Cuál es la diferencia que existe en las declaraciones var, let y const en JS?

1. 'var' : Las variables declaradas con ´var´ tienen un ámbito de función. Esto significa que si se declaran dentro de una función, no son accesibles fuera de ella, pero si se declaran en cualquier otro bloque (como un if o un for), siguen siendo accesibles fuera de ese bloque.
Se puede re declarar una variable usando ´var´ dentro de un mismo ámbito sin errores.

```
function example() {
    console.log(x); // undefined
    var x = 5;
    console.log(x); // 5
}
example();

if (true) {
    var y = 10;
}
console.log(y); // 10
```

2. 'let': Las variables declaradas con ´let´ tienen un ámbito de bloque. Esto significa que solo son accesibles dentro del bloque en el que se declararon (por ejemplo, dentro de unas llaves {}).
Al usar ´let´ no es permitida la re declaración en un mismo ámbito.

```
function example() {
    // console.log(a); // ReferenceError: Cannot access 'a' before initialization
    let a = 5;
    console.log(a); // 5
}
example();

if (true) {
    let b = 10;
    console.log(b); // 10
}
// console.log(b); // ReferenceError: b is not defined
```

3. ‘const’ : Al igual que ‘let’, ‘const’ tiene un ámbito de bloque, la diferencia es que este tiene una asignación única ya que las variables declaradas con ‘const’ deben ser inicializadas al momento de la declaración y no se pueden reasignar después.

Si se declara un objeto o un arreglo con ‘const’ , no se puede reasignar el objeto o arreglo en sí, pero si se puede modificar sus propiedades internas.

```
const c = 5;
console.log(c); // 5
// c = 10; // TypeError: Assignment to constant variable.

const obj = { key: 'value' };
obj.key = 'newValue'; // Esto es permitido
console.log(obj.key); // 'newValue'

// obj = {}; // TypeError: Assignment to constant variable.
```

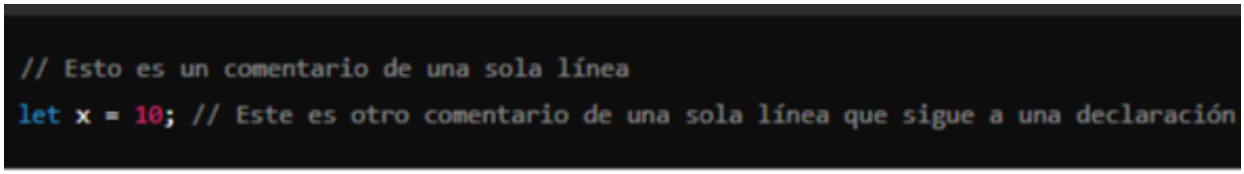
En conclusión para dejar un poco más claro lo de ‘var’, ‘let’ y ‘const’, el recomendable utilizar let y const en lugar de var ya que se evitan errores relacionados con la re declaración accidental de variables, const se utiliza principalmente para variables que no necesitan ser reasignadas, para que así el código sea más seguro.

9. ¿Explique los 2 tipos de comentarios que se pueden aplicar en JS?

En JavaScript se puede comentar de dos formas para hacer anotaciones en el código, la primer forma es comentarios de una sola línea y la segunda comentarios en múltiples líneas.

1. Comentarios de una sola línea

Estos comentarios se indican con las barras diagonales (‘//’). Se usa comúnmente para hacer anotaciones breves.



2. Comentarios de múltiples líneas

Estos comentarios comienzan con una barra diagonal y un asterisco (/*) y terminan con un asterisco y una barra diagonal (*/). Se utiliza para hacer anotaciones más extensas o para desactivar bloques de código más largos.

```
/*
  Esto es un comentario de múltiples líneas.
  Puedes escribir en varias líneas
  y todo este texto será ignorado por el intérprete de JavaScript.
*/
let y = 20;

/*
Este es otro comentario de múltiples líneas
que puede también estar en el mismo formato.
*/

// También puedes usarlo para desactivar bloques de código:
function example() {
  /*
  console.log('Esta línea está desactivada');
  console.log('Y esta también');
  */
  console.log('Esta línea se ejecutará');
}
```

10. ¿Qué es ECMAScript6? Explique claramente.

ECMAScript 6, también conocido como ECMAScript 2015 o ES6, es una versión estándar de JavaScript que se presentó en 2015. Para hacer el lenguaje más potente y fácil de usar, introdujo numerosas mejoras y características nuevas.

Como resultado, ECMAScript es un estándar creado para permitir que varios navegadores interactúen con varias páginas web. El estándar de lenguaje especifica las características y la implementación de un lenguaje de scripting.

JavaScript, ActionScript y JScript (Microsoft) son ejemplos de lenguajes de scripting para el lado del cliente del desarrollo web que se implementaron a partir de ECMAScript.

Novedades que incluyó ECMAScript6

Declaración de variables con ´let´ y ´const´: Permiten un mejor control del ámbito de las variables.

Funciones Flecha (Arrow Functions): Las funciones flecha proporcionan una sintaxis más corta para escribir funciones. Además, no tienen su propio ´this´, lo que significa que heredan el ´this´ del contexto donde fueron definidas.

```
const add = (a, b) => a + b;
```

Clases (classes):
Introducen una sintaxis más clara para trabajar con objetos y herencia.

```
class Persona {
    constructor(nombre, edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    saludar() {
        return `Hola, me llamo ${this.nombre}`;
    }
}
```

Módulos (modules):

Facilitan la modularización del código mediante importaciones y exportaciones.

```
// module.js
export const pi = 3.1416;

// main.js
import { pi } from './module.js';
```

Promesas (promises):

Maneja operaciones asíncronas, mejorando la legibilidad y manejabilidad del código asíncrono.

```
let promesa = new Promise((resolve, reject) => {
    // código asíncrono
    if (/* éxito */) {
        resolve("¡Éxito!");
    } else {
        reject("Error");
    }
});

promesa.then(result => {
    console.log(result);
}).catch(error => {
    console.log(error);
});
```

Conclusión

En conclusión, JavaScript se ha consolidado como un lenguaje esencial en la actualidad. Su amplia utilización abarca desde el desarrollo de páginas y aplicaciones web hasta la automatización de procesos y la creación de juegos. La versatilidad de JavaScript fomenta la creatividad y la experimentación con nuevos componentes y herramientas, haciendo que el proceso de aprendizaje sea dinámico e interactivo.