Trabajo de Investigación de JAVA SCRIPT

Ana Lucia Orozco Serrano

Universidad CENFOTEC

Fundamentos de Programación Web

Francisco Jiménez Bonilla

Fecha: Septiembre del 20

Tabla de contenido

Desarrollo de las preguntas

1. **¿Escriba la historia del lenguaje Java Script?**

1. Los Inicios (1995)

JavaScript fue creado en 1995 por Brendan Eich mientras trabajaba para Netscape Communications. El objetivo de Netscape era crear un lenguaje de scripting que pudiera ejecutarse en el navegador, haciendo las páginas web más interactivas. Originalmente, el lenguaje se llamó Mocha, luego pasó a llamarse LiveScript, y finalmente, fue bautizado como JavaScript para aprovechar el éxito de Java (aunque ambos lenguajes son muy diferentes).

Netscape incluyó JavaScript en su navegador Netscape Navigator 2.0, permitiendo a los desarrolladores web crear páginas con interacción dinámica, algo innovador en ese momento.

Fundador de Java Script es Brendan Eich

2. Estándar ECMAScript (1997)

En 1996, Microsoft vio el potencial de JavaScript y desarrolló su propia versión para su navegador Internet Explorer bajo el nombre de JScript. Esto creó incompatibilidades entre navegadores, lo que llevó a la necesidad de un estándar.

En 1997, el lenguaje fue formalizado bajo el nombre de ECMAScript por ECMA International, un organismo de estandarización. La especificación ECMAScript sirvió como base para todas las implementaciones de JavaScript en diferentes navegadores.

3. La Evolución Lenta (1997-2005)

Durante finales de los 90 y principios de los 2000, JavaScript fue percibido principalmente como un lenguaje para pequeños scripts y validaciones en formularios, y muchos lo consideraban limitado. Los desarrolladores a menudo enfrentaban problemas de compatibilidad entre navegadores y el lenguaje no era tan popular para aplicaciones grandes.

Sin embargo, en 2005 ocurrió un cambio importante cuando Jesse James Garrett acuñó el término AJAX (Asynchronous JavaScript and XML). Esto permitió la creación de aplicaciones web más rápidas y dinámicas al cargar datos en segundo plano, sin necesidad de recargar la página completa.

Jesse James Garrett

4. El Renacimiento de JavaScript (2009)

El verdadero renacimiento de JavaScript ocurrió en 2009 con la llegada de Node.js, una plataforma que permitió usar JavaScript en el lado del servidor, lo que revolucionó la manera en que los desarrolladores usaban el lenguaje. Con Node.js, JavaScript dejó de ser exclusivo del navegador y comenzó a utilizarse para construir servidores y aplicaciones de backend.

Además, durante este tiempo, surgieron importantes bibliotecas y frameworks como jQuery, que facilitó el trabajo con JavaScript, y posteriormente React, Angular y Vue, que ayudaron a construir interfaces de usuario más complejas.

5. ECMAScript 6 y Más Allá (2015-presente)

La versión ECMAScript 6 o ES6, lanzada en 2015, introdujo características fundamentales que mejoraron drásticamente el lenguaje, como las clases, promesas, y el uso de let y const en lugar de var.

Desde entonces, JavaScript ha seguido evolucionando rápidamente con versiones anuales de ECMAScript, añadiendo nuevas capacidades al lenguaje y mejorando su rendimiento y versatilidad tanto en el navegador como en el servidor.

6. JavaScript Hoy en Día

Hoy, JavaScript es uno de los lenguajes más usados en el desarrollo de software. Es crucial para el desarrollo web moderno y es parte del llamado stack full-stack, que permite a los desarrolladores trabajar tanto en el front-end como en el back-end con el mismo lenguaje.

Además, el ecosistema de JavaScript continúa creciendo con nuevas herramientas y frameworks, y su uso se ha expandido a áreas como aplicaciones móviles (con React Native) y el desarrollo de videojuegos.

2. ¿Por qué se debe aprender Java Script?

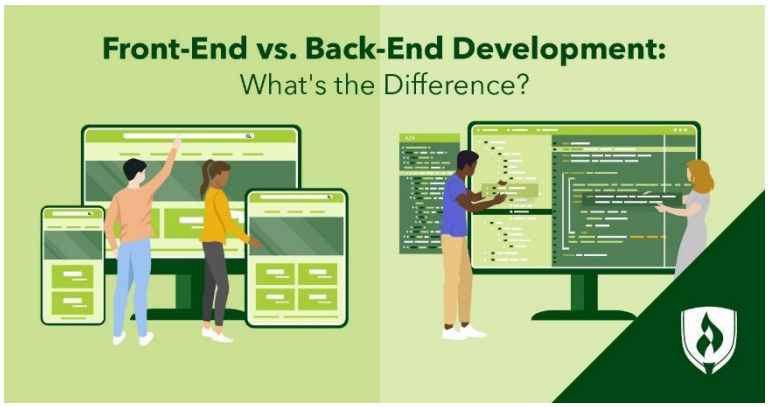
Aprender JavaScript es esencial por varias razones, tanto para quienes están comenzando en el desarrollo como para desarrolladores experimentados. Aquí te doy algunos de los motivos más importantes:

Lenguaje fundamental en el desarrollo web

JavaScript es el único lenguaje de programación que puede ejecutarse de manera nativa en todos los navegadores web, lo que lo convierte en el lenguaje principal para el desarrollo front-end. Si deseas construir sitios web interactivos, dinámicos y modernos, es indispensable conocer JavaScript.

Versatilidad

Con la llegada de Node.js, JavaScript se convirtió en un lenguaje full-stack, lo que significa que puede ser utilizado tanto para el front-end como para el back-end de una aplicación. Esto permite a los desarrolladores trabajar en ambas partes de un proyecto usando un solo lenguaje, lo que ahorra tiempo y facilita la gestión del código.



Alta demanda laboral

JavaScript es uno de los lenguajes de programación más populares y utilizados en el mundo, lo que se traduce en múltiples oportunidades laborales. La mayoría de las empresas, desde startups hasta grandes corporaciones, requieren desarrolladores que dominen JavaScript para sus aplicaciones web y móviles.

Ecosistema rico en herramientas y frameworks

JavaScript tiene un ecosistema muy amplio, con bibliotecas y frameworks como React, Angular, Vue, y jQuery. Estos frameworks hacen que el desarrollo de interfaces de usuario sea más eficiente, rápido y escalable, lo cual facilita la creación de proyectos complejos.

Desarrollo de aplicaciones móviles y de escritorio

Además del desarrollo web, JavaScript se utiliza en el desarrollo de aplicaciones móviles (con React Native y Ionic) y aplicaciones de escritorio (con Electron). Esto lo convierte en un lenguaje multipropósito que te permite abordar diferentes tipos de proyectos.

Comunidad y recursos abundantes

JavaScript cuenta con una gran comunidad de desarrolladores y una vasta cantidad de recursos educativos, lo que facilita el aprendizaje continuo. Hay muchos tutoriales, foros, y recursos en línea que pueden ayudarte a mejorar tus habilidades y resolver problemas de manera rápida.

3. ¿Cuál es la relación entre HTML y Java Script?

La relación entre HTML y JavaScript es fundamental en el desarrollo web, ya que ambos lenguajes trabajan en conjunto para crear sitios interactivos y dinámicos. A continuación, te explico la relación entre ambos:

HTML: Estructura

HTML (HyperText Markup Language) es un lenguaje de marcado que se utiliza para crear la estructura básica de una página web. Define los elementos que componen el contenido de la página, como títulos, párrafos, imágenes, formularios y enlaces, organizando todo en una jerarquía de etiquetas.

JavaScript: Comportamiento e Interactividad

JavaScript es un lenguaje de programación que se utiliza para agregar interactividad y dinamismo a una página web. Mientras que HTML define la estructura, JavaScript permite modificar el contenido, responder a eventos del usuario (como clics o desplazamientos), y hacer que la página sea más dinámica.

Interacción entre HTML y JavaScript

JavaScript puede manipular y cambiar los elementos HTML en tiempo real. Esto se hace a través del DOM (Document Object Model), que es una representación en árbol de la estructura HTML de la página. JavaScript puede acceder al DOM para modificar el contenido de la página sin necesidad de recargarla.

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

Usar Bootstrap en el desarrollo de sitios y aplicaciones web en JavaScript trae una serie de beneficios clave que facilitan tanto el diseño como la implementación de funcionalidades dinámicas. Aquí tienes un resumen con las principales ventajas:

1. Diseño responsivo

Bootstrap facilita la creación de sitios web que se adapten automáticamente a diferentes tamaños de pantalla, gracias a su sistema de rejilla (grid system). Esto garantiza que tu sitio o aplicación sea accesible en dispositivos móviles, tabletas y computadoras de escritorio sin tener que diseñar diferentes versiones.

Beneficio: Ahorra tiempo al evitar la creación de múltiples versiones del mismo sitio y asegura una experiencia consistente en todos los dispositivos.

2. Componentes reutilizables

Bootstrap ofrece una amplia variedad de componentes predefinidos como menús de navegación, botones, formularios, modales y carruseles, que se integran fácilmente con el diseño de la página y su lógica de JavaScript. Estos componentes pueden ser personalizados y se adaptan a las necesidades de tu proyecto.

Beneficio: Reduce el tiempo de desarrollo al usar componentes listos para ser implementados y personalizables.

3. Integración con JavaScript y jQuery

Bootstrap incluye plugins JavaScript que permiten agregar fácilmente interactividad a la página, como menús desplegables, tooltips y ventanas modales. Estos plugins funcionan bien con jQuery, lo que facilita la creación de comportamientos dinámicos sin escribir mucho código JavaScript personalizado.

Beneficio: Simplifica la adición de elementos interactivos, mejorando la experiencia del usuario sin necesidad de codificar desde cero.

4. Consistencia en el diseño

Bootstrap sigue principios de diseño coherente y moderno, lo que asegura que todos los elementos de la interfaz de usuario sean uniformes. Esto proporciona una experiencia visual atractiva y profesional para los usuarios sin requerir un esfuerzo significativo en la creación de estilos personalizados.

Beneficio: Mantiene un aspecto visual uniforme y moderno, garantizando una experiencia de usuario fluida y profesional.

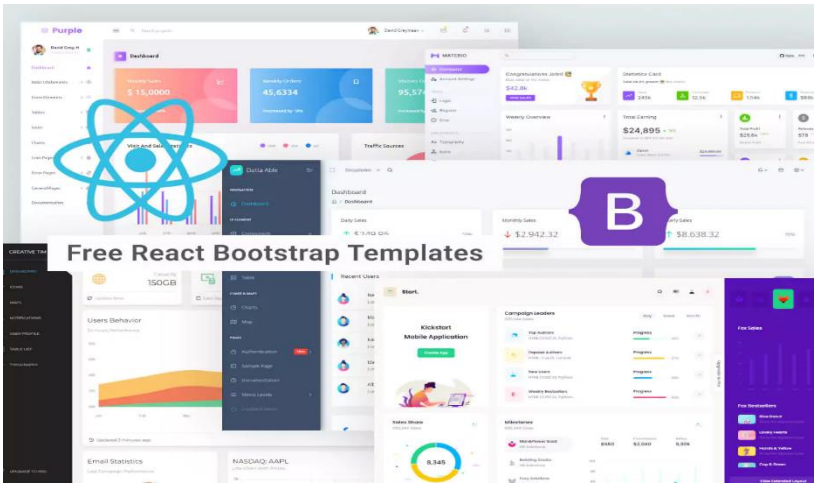
5. Ahorro de tiempo

Bootstrap ofrece un conjunto de clases y estilos predefinidos, lo que permite a los desarrolladores centrarse en la lógica de la aplicación y reducir la cantidad de CSS y JavaScript que necesitan escribir. Esto acelera significativamente el proceso de desarrollo, especialmente para aplicaciones grandes o complejas.

Beneficio: Acelera el desarrollo al proporcionar una base sólida para el diseño y la funcionalidad.

6. Fácil de aprender

Bootstrap es fácil de usar tanto para principiantes como para desarrolladores avanzados. Su extensa documentación y gran comunidad de soporte hacen que su curva de aprendizaje sea suave y que cualquier problema pueda resolverse rápidamente con la ayuda de recursos en línea. Beneficio: Facilita la adopción rápida del framework, lo que permite a los equipos trabajar de manera más eficiente.



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

Semejanzas:

1. Ambos se utilizan en desarrollo web.
2. Interactúan con HTML para generar contenido dinámico.
3. Son lenguajes de código abierto.
4. Pueden conectarse a bases de datos (PHP en el servidor y JavaScript con Node.js en el servidor).

Diferencias:

1. **PHP** es un lenguaje del **lado del servidor**, mientras que **JavaScript** se ejecuta en el **lado del cliente** (aunque con Node.js puede estar en el servidor).
2. **PHP** genera HTML desde el servidor y envía la página al cliente. **JavaScript** puede modificar la página directamente en el navegador sin recargarla.
3. **PHP** es ideal para manejo de bases de datos y formularios. **JavaScript** se usa para hacer páginas interactivas y responder a eventos en tiempo real.
4. **PHP** no puede interactuar con la página una vez enviada, mientras que **JavaScript** puede seguir manipulándola dinámicamente.

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

Incrustado directamente en el HTML (Inline JavaScript):
Se escribe el código JavaScript dentro de una etiqueta <script> en el archivo HTML, generalmente dentro de la sección <head> o al final del <body>.

```
<html>
  <head>
    <script>
      alert("Hola, mundo!");
    </script>
  </head>
  <body>
    <!-- Contenido de la página -->
  </body>
</html>
```

Archivo JavaScript externo (External JavaScript):
El código JavaScript se coloca en un archivo separado con la extensión .js y se enlaza a la página HTML mediante la etiqueta <script src="archivo.js"></script>

```
<html>
  <head>
    <script src="script.js"></script>
  </head>
  <body>
    <!-- Contenido de la página -->
  </body>
</html>
```

Se puede agregar código JavaScript directamente a los eventos de los elementos HTML, como onclick, onmouseover, etc.

```
<button onclick="alert('¡Botón presionado!')">Haz clic</button>
```

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

Depuración: Permite a los desarrolladores inspeccionar el código, realizar pruebas y encontrar errores. Puedes utilizar métodos como console.log(), console.error(), console.warn(), y console.info() para imprimir mensajes y datos en la consola del navegador. Esto facilita la identificación de problemas en el código y el seguimiento del flujo de ejecución.

Inspección de datos: Facilita la visualización de variables, objetos y estructuras de datos en tiempo real. Puedes imprimir valores de variables, inspeccionar objetos y arrays, y ver cómo cambian durante la ejecución del código.

Interacción y pruebas: Permite ejecutar código JavaScript directamente en la consola del navegador para probar fragmentos de código, experimentar con funciones y ver resultados inmediatos. Esto es útil para prototipar y experimentar con nuevas ideas sin modificar el código fuente.

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

En JavaScript, las declaraciones var, let y const tienen diferentes características y comportamientos:

1. var:

   - Alcance (scope): El alcance de var es global o local a una función, dependiendo de dónde se declare. No respeta el bloque (como dentro de un if, for, etc.).
   - Re-declaración: Permite redeclarar la misma variable dentro del mismo alcance, lo que puede llevar a errores imprevistos.
   - Hoisting: Las variables declaradas con var se "elevan" al principio del ámbito, pero su valor no se inicializa hasta que la línea de la declaración se ejecuta.

```
console.log(x); // undefined (hoisting)
var x = 5;
```

2. **let:**

   - **Alcance (scope)**: Tiene un alcance **local al bloque**, es decir, solo está disponible dentro del bloque {} donde se declara.
   - **Re-declaración**: No permite redeclarar la misma variable dentro del mismo bloque.

   - **Hoisting**: También sufre "hoisting", pero a diferencia de var, no se puede acceder a la variable antes de que se declare (esto genera un error).

```
console.log(y); // ReferenceError: y is not defined
let y = 5;
```

3.const:

- **Alcance (scope)**: Igual que let, tiene un alcance **local al bloque**.
- **Re-declaración**: No permite redeclarar ni reasignar el valor. Una vez asignado, no se puede cambiar. Sin embargo, si el valor es un objeto o un array, las propiedades o elementos sí pueden modificarse.
- **Hoisting**: Como let, también se eleva pero no permite acceder a la variable antes de su declaración.

```
const z = 10;
z = 15; // TypeError: Assignment to constant variable.
```

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

En JavaScript, existen dos tipos de comentarios:

1. **Comentarios de una sola línea**: Se usan para comentarios breves y ocupan una sola línea. Para crearlos, se utiliza // seguido del comentario.

```
// Esto es un comentario de una sola línea
let x = 5; // También se puede usar al final de una línea de código
```

2. **Comentarios de múltiples líneas**: Son útiles para escribir comentarios largos que abarcan varias líneas o para deshabilitar grandes bloques de código. Se encierran entre /* y */.

```
/* Esto es un comentario
de múltiples líneas
que puede ocupar
varias líneas */
let y = 10;
```

10. ¿Qué es ECMAScript6? Explique claramente.

ECMAScript 6 (ES6), también conocido como **ECMAScript 2015**, es la sexta versión del estándar ECMAScript, que es el lenguaje base sobre el cual se construye **JavaScript**. Fue lanzado en 2015 y representa una actualización importante que introdujo nuevas características y mejoras significativas al lenguaje, haciéndolo más eficiente y moderno.

Características principales de ECMAScript 6:

Declaraciones let y const: Introducen un mejor control sobre el alcance (scope) de las variables. let es similar a var pero con alcance de bloque, mientras que const se usa para declarar constantes que no pueden ser reasignadas.

```
let x = 10;
const y = 20;
```

Funciones Flecha (Arrow Functions): Una sintaxis más concisa para escribir funciones. Además, no tienen su propio valor de this, lo que evita problemas al referenciar el contexto dentro de funciones.

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

Clases: Introducen la sintaxis de clases, lo que facilita la creación de objetos y herencia en JavaScript, similar a otros lenguajes de programación orientados a objetos.

```
class Persona {
  constructor(nombre) {
    this.nombre = nombre;
  }
  saludar() {
    console.log(`Hola, soy ${this.nombre}`);
  }
}
```

Plantillas de cadenas de texto (Template literals): Permiten incluir variables y expresiones dentro de cadenas de texto mediante el uso de backticks (`) y ${}.

```
const nombre = 'Juan';
console.log(`Hola, ${nombre}!`);
```

Destructuración: Permite extraer valores de arrays u objetos y asignarlos a variables de manera más concisa.

```
const [a, b] = [1, 2];
const {nombre, edad} = {nombre: 'Ana', edad: 30};
```

Parámetros por defecto: Facilita la asignación de valores predeterminados a los parámetros de las funciones.

```
function saludar(nombre = 'Amigo') {
  console.log(`Hola, ${nombre}`);
}
```

Promesas (Promises): Introducen una forma de manejar operaciones asíncronas, evitando el "callback hell" y facilitando la gestión de procesos que dependen de otros.

```
const promesa = new Promise((resolve, reject) => {
  // código asíncrono
});
```

Módulos (Modules): Permiten dividir el código en archivos reutilizables, lo que mejora la organización y el mantenimiento del código. Se usa export para exportar funciones o variables y import para importarlas

```
export const nombre = 'Juan';
import { nombre } from './modulo.js';
```

Conclusión

Para concluir, JavaScript ha experimentado una evolución significativa desde sus inicios en 1995 hasta convertirse en un pilar fundamental del desarrollo web moderno. Su historia abarca momentos clave, como la creación por Brendan Eich, la estandarización a través de ECMAScript, y su renacimiento con la aparición de Node.js y frameworks como React y Angular. Hoy en día, JavaScript es esencial no solo en el desarrollo del lado del cliente, sino también en el servidor, gracias a su versatilidad y su amplio ecosistema.

Aprender JavaScript es crucial para cualquier desarrollador que quiera construir aplicaciones web interactivas y dinámicas, debido a su papel central en la estructura de la web. Con su capacidad para integrarse con HTML, manejar dinámicamente el DOM, y su compatibilidad con herramientas como Bootstrap, es un lenguaje poderoso para crear experiencias web completas. Además, el continuo crecimiento de JavaScript, impulsado por actualizaciones como ECMAScript 6 y más allá, garantiza su relevancia en el futuro, siendo una habilidad valiosa en el competitivo mercado laboral tecnológico.

Por último, la relación entre JavaScript y otros lenguajes como PHP, aunque complementaria, resalta las diferencias entre el código que se ejecuta del lado del servidor y el cliente, siendo JavaScript la opción preferida para manejar la interactividad en tiempo real en el navegador.

Lista de Referencias

- *Flanagan, D. (2020). JavaScript: The definitive guide (7th ed.). O'Reilly Media.*
- *Eich, B. (2008). Brendan Eich and the birth of JavaScript. In M. Erwin (Ed.), Coders at work:Reflections on the craft of programming (pp. 35-60). Apress.*
- Garrett, J. J. (2005). Ajax: A new approach to web applications. Adaptive Path.Retrieved from https://adaptivepath.org/
- Flanagan, D. (2020). JavaScript: The definitive guide (7th ed.). O'Reilly Media.
- Freeman, E., & Robson, E. (2020). Head first JavaScript programming. O'Reilly Media.
- Crockford, D. (2008). JavaScript: The good parts. O'Reilly Media.
- Duckett, J. (2014). HTML & CSS: Design and build websites & JavaScript & jQuery: Interactive front-end web development. Wiley
- Flanagan, D. (2020). JavaScript: The definitive guide (7th ed.). O'Reilly Media.
- Nixon, R. (2018). Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (5th ed.). O'Reilly Media.
- Welling, L., & Thomson, L. (2016). PHP and MySQL web development (5th ed.). Addison-Wesley Professional.