## **Informe sobre Fundamentos de Aplicaciones y Desarrollo Web**

## 

## 

## 

## 

## 

## 

**Nombre:** Juan Manuel Velasco Duque  
**Ficha:** 3144622  
**Competencia:** Evaluar requisitos de la solución de software de software de acuerdo con metodologías de análisis y estándares.  
**Instructor:** Arle Morales Ortiz

## 

## 

## **Introducción**

Este informe tiene como objetivo recopilar y sintetizar los temas abordados durante el curso de Fundamentos de Aplicaciones y Desarrollo Web. A lo largo del curso, se han explorado conceptos fundamentales que son esenciales para comprender el desarrollo web moderno, con un enfoque especial en JavaScript, estructuras de control, funciones y sistemas de control de versiones. Estos temas no solo son relevantes para la creación de aplicaciones web, sino que también sientan las bases para el aprendizaje de tecnologías más avanzadas.

## **Objetivos del Informe**

* Consolidar los conocimientos adquiridos durante el curso.
* Proporcionar una guía práctica para la aplicación de los conceptos vistos.
* Servir como material de referencia para futuros proyectos y estudios.

## **Importancia de los Temas Vistos**

Los temas cubiertos en este curso son fundamentales para cualquier persona que desee incursionar en el desarrollo web. Desde la lógica básica de programación hasta el uso de herramientas como Git y GitHub, estos conocimientos permiten crear aplicaciones web funcionales, eficientes y colaborativas. Además, el dominio de JavaScript es crucial, ya que es uno de los lenguajes más utilizados en el desarrollo frontend y backend.

## 

## 

## 

## **Desarrollo**

1. **Variables** Las variables son contenedores que almacenan datos que pueden cambiar durante la ejecución de un programa. En JavaScript, puedes declarar variables usando var, let o const.  
   * var: Declara una variable, opcionalmente inicializando con un valor. Tiene un alcance de función o global.
   * let: Declara una variable con alcance de bloque.
   * const: Declara una constante con alcance de bloque. Su valor no puede ser reasignado después de la inicialización.

**Ejemplo en JavaScript:**

var nombre = "Juan"; *// Alcance de función o global*

let edad = 30; *// Alcance de bloque*

const PI = 3.1416; *// Constante, no puede ser reasignada*

Las variables son esenciales para almacenar y manipular datos en un programa. Es importante elegir el tipo de declaración adecuado (var, let, const) según el alcance y la mutabilidad requerida.

1. **Operadores** Los operadores son símbolos que realizan operaciones en uno o más operadores. JavaScript tiene varios tipos de operadores:  
   * **Aritméticos:** +, -, \*, /, % (módulo), \*\* (exponenciación)
   * **Asignación:** =, +=, -=, \*=, /=, %=
   * **Comparación:** == (igual), != (no igual), === (igual estricto), !== (no igual estricto), >, <, >=, <=
   * **Lógicos:** && (AND), || (OR), ! (NOT)
   * **Ternario:** condición ? valor\_si\_verdadero : valor\_si\_falso

**Ejemplo en JavaScript:**

let x = 10;

let y = 5;

console.log(x + y); *// 15*

console.log(x > y); *// true*

console.log(x % y); *// 0*

Los operadores son fundamentales para realizar cálculos, comparaciones y operaciones lógicas en un programa.

1. **Operadores Lógicos** Los operadores lógicos son herramientas fundamentales en programación que permiten evaluar condiciones y tomar decisiones basadas en ellas. Los principales operadores lógicos son:  
   * AND (&&): Devuelve true si ambas condiciones son verdaderas.
   * OR (||): Devuelve true si al menos una de las condiciones es verdadera.
   * NOT (!): Invierte el valor de la condición (de true a false y viceversa).

**Ejemplo en JavaScript:**

let a = true;

let b = false;

console.log(a && b); *// false*

console.log(a || b); *// true*

console.log(!a); *// false*

Los operadores lógicos son esenciales para controlar el flujo de un programa. Por ejemplo, en una aplicación web, podrías usar un operador AND para verificar si un usuario ha iniciado sesión y tiene permisos de administrador antes de permitirle acceder a ciertas funciones. Además, son útiles para combinar múltiples condiciones en una sola expresión, lo que simplifica la lógica del código.

1. **Estructuras de Control** Las estructuras de control permiten controlar el flujo de ejecución de un programa. Las principales estructuras de control son:  
   * **Condicionales:** if, else if, else, switch
   * **Bucles:** for, while, do...while

**Ejemplo en JavaScript:** let edad = 18;

if (edad >= 18) {

console.log("Eres mayor de edad.");

} else {

console.log("Eres menor de edad.");

}

for (let i = 0; i < 5; i++) {

console.log("Iteración: " + i);

}

Las estructuras de control son fundamentales para tomar decisiones y realizar tareas repetitivas en un programa. Permiten crear lógica compleja y controlar el flujo de ejecución según las condiciones específicas.

1. **Fundamentos de Aplicaciones y Desarrollo Web** El desarrollo web se divide en dos áreas principales:  
   * **Frontend:** La parte visible de una aplicación web, construida con HTML, CSS y JavaScript.
   * **Backend:** La lógica del servidor, que maneja bases de datos, autenticación y procesamiento de datos.
   * **Base de datos:** Almacena y gestiona la información de la aplicación.

El frontend se encarga de la interfaz de usuario y la experiencia del usuario, mientras que el backend maneja la lógica del negocio y la interacción con la base de datos. Una base de datos bien diseñada es crucial para almacenar y recuperar información de manera eficiente. Por ejemplo, en una aplicación de comercio electrónico, el frontend mostraría los productos disponibles, mientras que el backend manejaría el procesamiento de pedidos y la interacción con la base de datos para actualizar el inventario.

1. **¿Qué es JavaScript (JS)?** JavaScript es un lenguaje de programación ampliamente utilizado en el desarrollo web. Permite agregar interactividad y dinamismo a las páginas web. Algunas características clave de JavaScript incluyen:  
   * Es interpretado y se ejecuta en el navegador.
   * Es compatible con todos los navegadores modernos.
   * Permite manipular el DOM (Document Object Model) para cambiar el contenido de una página web.

**Ejemplo básico:**

console.log("Hola, mundo!");

JavaScript es esencial para crear aplicaciones web dinámicas. Por ejemplo, puedes usar JavaScript para validar formularios, crear animaciones y cargar contenido dinámico sin recargar la página. Además, con frameworks como React y Angular, JavaScript se ha convertido en una herramienta poderosa para construir aplicaciones web complejas y escalables.

1. **Ejercicios IF** La estructura condicional if permite ejecutar un bloque de código si se cumple una condición específica.  
     
    **Ejemplo en JavaScript:**

let edad = 18;

if (edad >= 18) {

console.log("Eres mayor de edad.");

} else {

console.log("Eres menor de edad.");

}

La estructura if es fundamental para tomar decisiones en un programa. Puedes usarla para verificar condiciones y ejecutar diferentes bloques de código dependiendo del resultado. Por ejemplo, en una aplicación de reservas, podrías usar una estructura if para verificar si hay asientos disponibles antes de permitir que un usuario reserve un vuelo.

1. **Sistemas de Control de Versiones (Git y GitHub)** Git es un sistema de control de versiones que permite gestionar cambios en el código. GitHub es una plataforma que aloja repositorios Git y facilita la colaboración.  
     
    **Comandos básicos de Git:**
   * git init: Inicializa un repositorio.
   * git add <archivo>: Agrega cambios al área de preparación.
   * git commit -m "mensaje": Guarda los cambios con un mensaje descriptivo.
   * git push: Sube los cambios a un repositorio remoto.

Git es esencial para el desarrollo colaborativo. Permite a los desarrolladores trabajar en diferentes características simultáneamente sin interferir entre sí. GitHub facilita la colaboración al proporcionar una plataforma para compartir y revisar código. Por ejemplo, en un proyecto de equipo, cada desarrollador puede trabajar en una rama separada y luego fusionar sus cambios en la rama principal una vez que estén completos.

1. **Estructura de Condicional Múltiple** La estructura switch permite evaluar múltiples condiciones de manera más eficiente que varios if-else.  
     
    **Ejemplo en JavaScript:**

let dia = "Lunes";

switch (dia) {

case "Lunes":

console.log("Hoy es Lunes.");

break;

case "Martes":

console.log("Hoy es Martes.");

break;

default:

console.log("Día no reconocido.");

}

La estructura switch es útil cuando tienes múltiples condiciones que evaluar. Es más legible y fácil de mantener que varios if-else anidados. Por ejemplo, en una aplicación de calendario, podrías usar una estructura switch para mostrar diferentes mensajes dependiendo del día de la semana.

1. **Estructuras de Control (Decisiones)** Las estructuras de control de decisiones permiten ejecutar bloques de código basados en condiciones. Las más comunes son if, else if y else.  
     
    **Ejemplo en JavaScript:**

let nota = 85;

if (nota >= 90) {

console.log("Excelente");

} else if (nota >= 70) {

console.log("Aprobado");

} else {

console.log("Reprobado");

}

Las estructuras de control de decisiones son fundamentales para crear lógica en un programa. Permiten ejecutar diferentes bloques de código dependiendo de las condiciones específicas. Por ejemplo, en una aplicación de calificaciones, podrías usar una estructura if-else para determinar si un estudiante ha aprobado o reprobado un curso.

1. **Estructuras de Control (Ciclos y Bucles)** Los ciclos permiten repetir un bloque de código varias veces. Los más utilizados son for, while y do-while.  
     
    **Ejemplo en JavaScript:**

for (let i = 0; i < 5; i++) {

console.log("Iteración: " + i);

}

let j = 0;

while (j < 3) {

console.log("While: " + j);

j++;

}

Los ciclos son esenciales para realizar tareas repetitivas de manera eficiente. Por ejemplo, en una aplicación de lista de tareas, podrías usar un ciclo for para mostrar todas las tareas pendientes en la pantalla. Los ciclos también son útiles para procesar grandes cantidades de datos, como en el caso de una aplicación de análisis de datos.

1. **Guía Práctica de Funciones (JS)** Las funciones son bloques de código reutilizables que realizan una tarea específica.  
     
    **Ejemplo en JavaScript:**

function sumar(a, b) {

return a + b;

}

console.log(sumar(5, 3)); *// 8*

Las funciones permiten organizar el código en bloques reutilizables, lo que facilita la lectura y el mantenimiento del código. Por ejemplo, en una aplicación de calculadora, podrías crear una función para cada operación matemática (suma, resta, multiplicación, división) y luego llamar a estas funciones según sea necesario.

1. **Refuerzo de Temas y Conceptos** En esta sección se revisaron y consolidaron los conceptos clave vistos a lo largo del curso, incluyendo operadores lógicos, estructuras de control, funciones y sistemas de control de versiones. El objetivo fue reforzar la comprensión de estos temas a través de ejercicios prácticos que integran múltiples conceptos. Este refuerzo permite afianzar los conocimientos y asegurar que se puedan aplicar de manera efectiva en proyectos reales. Además, se destacó la importancia de combinar estos conceptos para resolver problemas complejos y desarrollar aplicaciones web funcionales y eficientes.
2. **Guía Práctica de Funciones Flecha (JS)** Las funciones flecha (arrow functions) son una sintaxis simplificada para definir funciones en JavaScript.  
     
    **Ejemplo en JavaScript:**

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

console.log(sumar(2, 3)); *// 5*

Las funciones flecha son especialmente útiles cuando se trabaja con funciones anónimas o callbacks. Por ejemplo, en una aplicación que utiliza el método map para transformar un array, podrías usar una función flecha para definir la lógica de transformación de manera concisa.

## **Conclusión**

Los temas abordados en este curso son fundamentales para el desarrollo de aplicaciones web. Desde operadores lógicos hasta el uso de Git y GitHub, cada concepto contribuye a la formación de un desarrollador web competente. Este informe resume los conocimientos adquiridos y proporciona ejemplos prácticos para su aplicación en proyectos futuros.

## **Referencias**

* Documentación oficial de JavaScript: [MDN Web Docs](https://developer.mozilla.org/es/)
* Guía oficial de Git: [Git Documentation](https://git-scm.com/doc)
* Tutoriales de GitHub: [GitHub Guides](https://guides.github.com/)