Facultad de Ciencias

Seminario FrontEnd

NOTAS DE CLASE

Semestre 2024 – 1

Autor

Marco Silva Huerta

Profesor:
Jesús Iván Saavedra Martínez

Ayudantes:
Adriana Hérnandez Gasca
Carlos López Rodríguez

29 de Agosto de 2023

Índice

1. Intrdoucción al desarrollo web **2**
1.1. Internet y la web . . . 2
1.2. Tipos de comunicación Cliente Servidor . . . 2

2. Intrdoucción al Frontend **3**

3. Introducción a HTML **3**

4. Introducción a CSS **4**

5. Introducción a JavaScript **4**
5.1. ¿Qué es JavaScript? . . . 4
5.2. Motores Javascript . . . 5
5.3. DOM y Javascript . . . 5
5.4. Elementos de JavaScript . . . 5

6. GIT **6**
6.1. Áreas de trabajo . . . 6
6.2. Ramas . . . 7
6.3. Convencion de Commits . . . 8

7. Angular **8**

8. Directivas en Angular **8**

9. Angular CRUD **8**

10. Angular Servives API **8**

11. TypeScript **8**

1. Intrdoucción al desarrollo web

1.1. Internet y la web

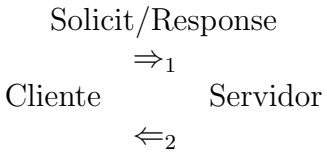
- El internet (o, también, la internet) es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP
- Ofrece una serie de servicios, siendo la World Wide Web (o Web o WWW) el de más éxito y conocido por todos.

La Web se desarrolló entre marzo de 1989 y diciembre de 1990 por el inglés Sir Tim Berners-Lee con la ayuda del belga Robert Cailliau mientras trabajaban en el CERN en Ginebra, Suiza. La Web nace a partir de la creación de los siguientes estándares:

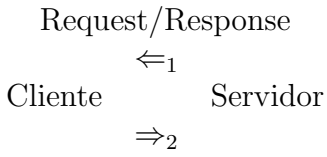
- **HTML** (Hypertext Markup Language), se utiliza para definir la estructura
- **URL** (Uniform Resource Locator), se utiliza para referenciar recursos en la Web de forma universal.
- **HTTP** (HyperText Transfer Protocol), es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

Cuando hablamos de Desarrollo Web hablamos necesariamente de **Aplicaciones Web** que es un posible producto resultante de esa tarea de desarrollo. Una aplicación web o, de forma abreviada, webapp, es una aplicación informática que un usuario puede utilizar accediendo a través de internet a un **servidor** web, y para ello hace uso de un **cliente** web o navegador.

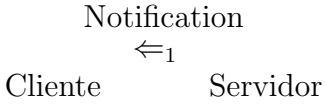
1.2. Tipos de comunicación Cliente Servidor



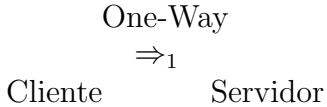
Funcionamiento: El cliente envía una solicitud al servidor, y el servidor responde con la información solicitada. Ejemplo: Un navegador web (cliente) solicita una página web al servidor web. El servidor procesa la solicitud y devuelve la página web al navegador.



Funcionamiento: El cliente hace una petición al servidor y recibe una respuesta. Ejemplo: Una aplicación de correo electrónico (cliente) envía una petición al servidor de correo para descargar nuevos mensajes. El servidor responde con los mensajes nuevos.



Funcionamiento: El servidor envía notificaciones o actualizaciones al cliente sin esperar una respuesta inmediata. Ejemplo: WhatsApp, puede enviar notificaciones al cliente cuando llega un nuevo mensaje, incluso si el cliente no ha solicitado activamente los nuevos mensajes.



Funcionamiento: Comunicación unidireccional, cliente envía una solicitud al servidor sin esperar una respuesta inmediata o sin necesidad de recibir una respuesta Ejemplo: En un sistema de telemetría de sensores, un sensor puede enviar datos al servidor para informar sobre las lecturas sin necesidad de que el servidor responda.

Frontend y Backend

En líneas generales, los desarrolladores frontend se encargan de diseñar y construir los elementos con los que el público tendrá contacto. Aquello incluye los botones, menús, páginas, enlaces, gráficos y otros componentes de una página o aplicación.

El backend consiste del servidor que provee la información que se solicita, la aplicación que se encarga de canalizarla y la base de datos que organiza la información. Por ejemplo, cuando un cliente busca zapatos en un sitio web, este interactúa con el frontend.

2. Intrdoucción al Frontend

El desarrollo frontend es el proceso de creación de componentes interactivos para el usuario final de una aplicación web. Las interfaces de usuario, los botones, los datos ingresados por el usuario, las páginas web y las funciones de experiencia del usuario (UX) son ejemplos de desarrollo frontend.

Objetivos

- Organizar y presentar la información, manteniéndola relevante, fácil de encontrar y acceder a ella.
- Lidiar con los múltiples dispositivos desde los cuales se pueden acceder a la información.

Ventajas del desarrollo web frontend

- Proporciona un desarrollo rápido gracias a todos los frameworks e innovaciones modernas disponibles.
- Las herramientas y técnicas son fáciles de aprender. La mayor parte del desarrollo frontend se limita a las tres tecnologías principales que son HTML, CSS y JavaScript.

Desventajas del desarrollo web frontend

- No importa cuán grande o pequeño sea un sitio web, la personalización es una parte esencial. Esto nos lleva a una base de código usualmente grande.
- En comparación con los lenguajes utilizados en el backend como PHP y Java, que existen desde hace bastante tiempo, JavaScript es bastante nuevo.
- Se lanzan versiones nuevas de bibliotecas y de los frameworks rápidamente. Lidiar con actualizaciones frecuentes es complicado, con cada nueva version, existe una mayor posibilidad de cometer errores y que funcionalidad previa no se encuentre soportada.

3. Introducción a HTML

Definiciones

- HTML es un lenguaje completamente independiente de la plataforma. Los navegadores web reciben las páginas HTML del servidor web o del almacenamiento local. Los documentos recibidos luego se convierten en páginas web multimedia.

- Existen varias versiones de HTML, la última versión es HTML 5

Ventajas

- Es ampliamente utilizado. Es compatible con todos los navegadores.
- Es fácil de aprender y utilizar.
- Es increíblemente liviano y se carga rápido.
- Es gratis. No es necesario comprar ningún software adicional.
- Se ejecuta en cualquier navegador o sistema operativo.
- Tiene una sintaxis laxa. Fácil de escribir y codificar incluso para principiantes en programación.

Desventajas

- Es un lenguaje estático, por lo que no puede producir ningún resultado dinámico.
- Crear la estructura de un documento HTML es complejo.
- Incluso un pequeño error a veces puede interrumpir todo el flujo de la página web.
- HTML no ofrece muchas funciones de seguridad.

4. Introducción a CSS

Definiciones

- CSS Hojas de Estilo en Cascada (Cascading Style Sheets).
- Lenguaje que **maneja el diseño y presentación** de las páginas web.
- Los estilos **se definen una vez y se aplican a múltiples** elementos en las páginas.

Sintaxis

h1	{ color:	blue;	font-size:	12px; }
⇑	⇑	⇑	⇑	⇑
Selector	Propiedad	Valor	Propiedad	Valor

5. Introducción a JavaScript

5.1. ¿Qué es JavaScript?

JavaScript es un lenguaje de programación interpretado basado en texto que nos permite agregar funcionalidad avanzada a cualquier páginaweb.

JavaScriptes increíblementeligero y se usa principalmenteparacrearscriptsen páginas web. También se utiliza para crear aplicacionesweb que interactúan conel cliente sin recargarla página cada vez.

5.2. Motores Javascript

Siemprequese ejecutaun programaJavaScript dentro del navegadorweb, el código es recibido por el motor del navegadory luego el motor lo ejecuta. El motor JavaScript tiene diferentes partes que nos ayudan en la ejecución del código.

- Parser
- AST
- MachineCodeConversion
- MachineCode

Fases de ejecución del códio JS

1. El parser convierte los archivos de código fuente en un árbol de sintaxis abstracta (AST).
2. El Árbol de sintaxis se transformaa bytecode: el intérprete de V8, Ignition, genera bytecode a partir del árbol de sintaxis (este paso existe desde 2017)
3. El bytecode genera código de máquina: el compilador de V8, TurboFan, genera un grafo a partir del bytecode, reemplazando secciones de bytecode con código máquina altamente optimizado.

5.3. DOM y Javascript

DOM significa *Document Object Model*. El DOM es multiplataforma, independiente del lenguaje y que opera mediante la construcción de una estructura de árbol a partir del contenido HTML

La API del DOMse utiliza para cambiarla interfaz que el usuariove con JavaScript. Esto se logra alterando dinamicamente HTMLy el CSS

5.4. Elementos de JavaScript

Variables de JavaScript

- Un identificador, el nombre de una variable, debe comenzar con una letra ASCII mayúscula o minúscula, un signo de dólar ($) o un guión bajo (_).
- Se puede usar números en un identificador, pero no como primer carácter.
- No puede incluir espacios.
- No se pueden utilizar palabras reservadas para los identificadores.

Variables de JavaScript

abstrac	arguments	await	boolean	break
byte	case	catch	char	class
const	continue	debugger	default	delate
do	double	else	enum	eval
export	extends	FALSE	final	finally
float	for	function	goto	if
implements	import	in	instanceof	int
interface	let	long	native	new
null	package	private	protected	public
return	short	static	super	switch
synchronized	this	throw	throws	transient
TRUE	try	typeof	var	void
volatile	while	with	yield	

Tipos de Datos JS

primitive
Boolean
Null
undefined
Number
String
Symbol

object
array
Object
Function
RegEx
Date

6. GIT

Git es un sistema de control de versiones distribuido y gratuito.

- Permite realizar un seguimiento de los cambios de código.
- Permite crear nuevas ramas para guardar diferentes versiones del código.
- Cada rama contiene un conjunto de nuevas características que se están desarrollando.

6.1. Áreas de trabajo

En Git, el proceso de trabajo se organiza en tres áreas clave: el directorio de trabajo (working directory), el área de preparación (staging area), y el repositorio (repository). Cada área cumple un papel específico en la gestión y seguimiento de los cambios en el código.

- Working directory
  - En esta área, se lleva a cabo todo el trabajo activo de creación, modificación, eliminación y organización de archivos y código.
  - Los cambios realizados en el working directory no se registran automáticamente en el repositorio, lo que brinda flexibilidad para experimentar y probar cambios antes de confirmarlos.

```
# Crear un nuevo archivo en el directorio de trabajo
touch nuevo_archivo.txt

# Modificar el archivo existente
nano nuevo_archivo.txt
```

- Staging area
  - En esta etapa, se seleccionan y preparan los cambios específicos del working directory que se desean incluir en el próximo commit.
  - La staging area actúa como un espacio intermedio donde se pueden revisar y ajustar los cambios antes de confirmarlos en el repositorio.

```
# Agregar cambios al area de preparacion
git add nuevo_archivo.txt
```

- Repository
  - Aquí es donde se almacenan permanentemente los cambios confirmados, incluyendo código, imágenes, logs, y demás.
  - Cada confirmación (commit) en el repositorio tiene un historial que permite rastrear el progreso del proyecto y revertir a versiones anteriores si es necesario.

```
# Confirmar los cambios en el repositorio
git commit -m "Actualizar nuevo archivo"
```

6.2. Ramas

En Git, las ramas son un concepto esencial que permite crear entornos de trabajo independientes dentro de un proyecto. Se pueden interpretar como solicitudes para establecer nuevos contextos o versiones del proyecto, cada una enfocada en el desarrollo de conjuntos específicos de características.

Cada rama representa una línea de desarrollo separada, con su propio conjunto de cambios y actualizaciones. Este enfoque brinda la flexibilidad de trabajar en nuevas funcionalidades o correcciones sin afectar directamente la rama principal del proyecto. La rama predeterminada en Git se denomina *master*, y es la línea base desde la cual se pueden ramificar otras versiones del código.

Flujo de trabajo

1. Crear la rama

```
git branch nueva-funcionalidad-rama
```

2. Cambiar a la nueva rama

```
git checkout nueva-funcionalidad-rama
```

3. Hacer modificaciones a los archivos de las ramas

4. Agregar y confirmar cambios en la nueva rama

```
git add .
git commit -m "Agregar nueva funcionalidad"
```

5. Volver a la rama principal (o de donde salio la rama)

```
git branch master
```

6. Integrar la nueva funcionalidad en la rama principal

```
git merge nueva-funcionalidad-rama
```

6.3. Convencion de Commits

Las convenciones de commits son prácticas que ayudan a estandarizar y estructurar los mensajes de confirmación (commits) en un repositorio Git. Estas convenciones facilitan la comprensión del historial del proyecto y la colaboración entre desarrolladores. Algunas de las convenciones comunes incluyen:

- Se utiliza un prefijo para indicar el propósito del commit.
  - feat: Nueva característica.

```
git commit -m "feat: Agregar funcionalidad de
   autenticacion"
```

  - fix: Corrección de un error.
  - docs: Cambios en la documentación.
  - style: Mejoras en el formato/código (sin cambios en la lógica).
  - chore: Tareas de mantenimiento, refactorización, etc.

7. Angular

8. Directivas en Angular

9. Angular CRUD

10. Angular Servives API

11. TypeScript