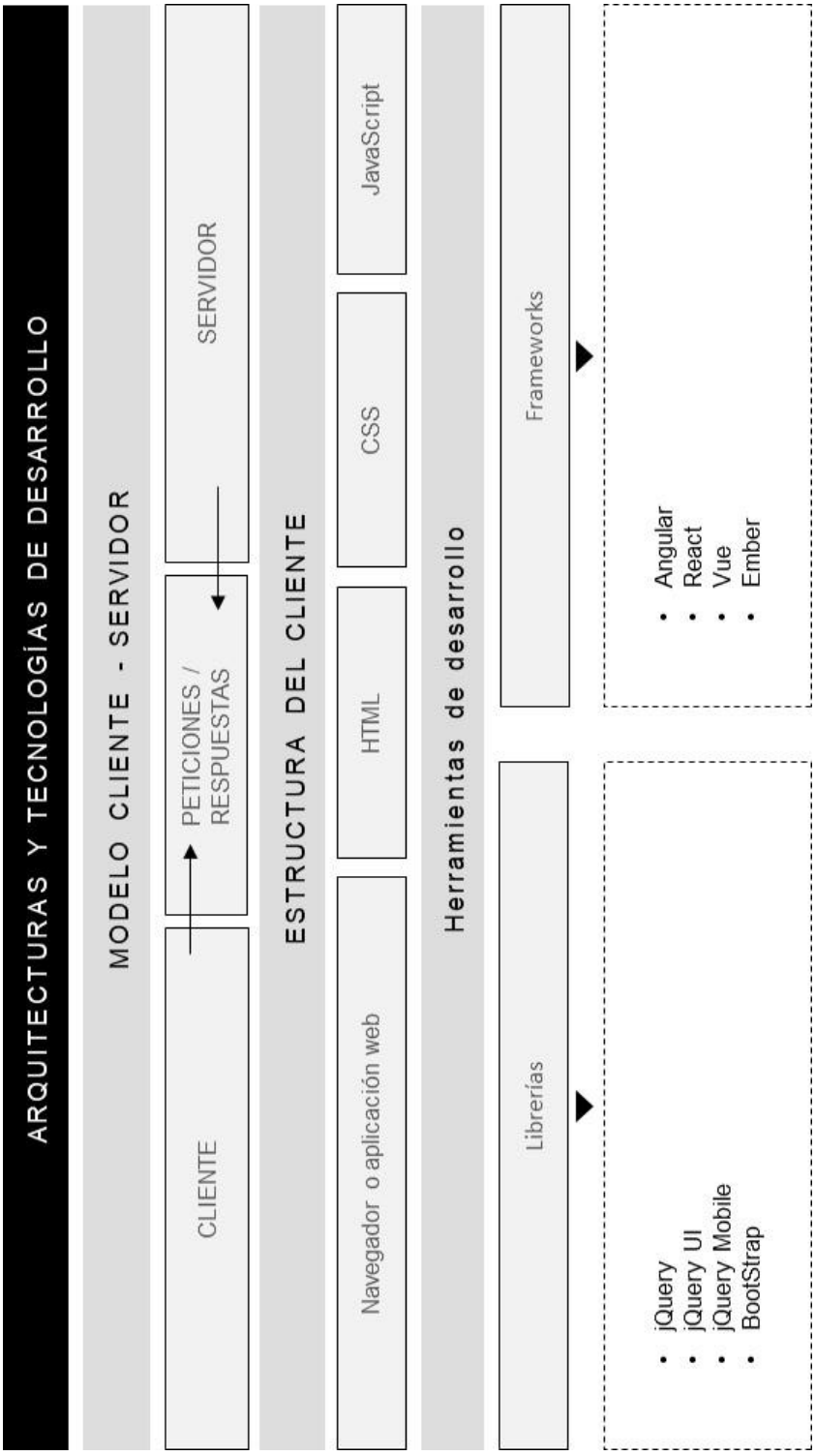
Desarrollo Web en Entorno Cliente

Arquitecturas y tecnologías de desarrollo web en el entorno cliente

Índice

Esquema


ARQUITECTURAS Y TECNOLOGÍAS DE DESARROLLO
MODELO CLIENTE - SERVIDOR
CLIENTE
PETICIONES / RESPUESTAS
SERVIDOR
ESTRUCTURA DEL CLIENTE
Navegador o aplicación web
HTML
CSS
JavaScript
Herramientas de desarrollo
Librerías
jQuery
jQuery UI
jQuery Mobile
BootStrap
Frameworks
Angular
React
Vue
Ember

Material de estudio

1.1. Introducción y objetivos

En el transcurso de este tema se realizará una introducción al mundo del desarrollo web en entorno cliente, analizando sus características y componentes principales. Además, se verán las herramientas, frameworks y librerías más importantes.

Durante el desarrollo de este tema se deben conseguir los siguientes objetivos:

- Conocer las funciones de un navegador, sus elementos y componentes principales.
- Comprender qué es la web y entender las áreas que afectan al desarrollo de los sites.
- Conocer qué es realmente el desarrollo web: en que se basa, en que áreas se centra, etc.
- Conocer los lenguajes y tecnologías de programación en entorno cliente.
- Saber cuál es la forma de insertar nuestros archivos JS en HTML y conocer los Frameworks y librerías existentes en el mercado actual.

1.2. Mecanismos de ejecución de código en un navegador web

La web es un universo paralelo de información accesible de forma permanente y global. Está formada por un conjunto innumerable de recursos que conforman lo que conocemos como Internet.

A la hora de desarrollar aplicaciones web, debemos tener muy en cuenta la distribución de elementos y la función de cada uno de ellos:

- El cliente: solicita la información.
- El servidor: responde a esa solicitud.

Un cliente es todo proceso que reclama servicios de otro, por lo que sus funciones principales las podemos resumir en:

- Administrar la interfaz e interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos y recibir/formatear los resultados del servidor.

El navegador es el software cliente más utilizado de todos los que tenemos disponibles. Actualmente, el mercado se divide en muchos navegadores diferentes, pero podemos destacar los cinco navegadores más populares:

- Microsoft Edge: software propietario.
- Chrome: software de código abierto.
- Firefox: software de código abierto.
- Safari: software de código abierto en parte.
- Opera: software propietario.

La función principal de un navegador es solicitar al servidor los recursos elegidos por el usuario y mostrarlos. Estos recursos son documentos HTML, imágenes y otros archivos, que se solicitan por medio del uso de las llamadas URI (Uniform Resource Identifier o identificador uniforme de recurso), consisten en una cadena de caracteres que identifican los recursos de una red.

Cómo nuestro navegador interpreta y muestra los recursos es algo que depende de una serie de especificaciones establecidas por el Consorcio W3C (World Wide Web Consortium), organización de estándares de Internet.

El motor de renderización del navegador web muestra el contenido solicitado en la pantalla. De forma predeterminada, el motor de renderización puede mostrar

imágenes y documentos HTML y XML, y también otros tipos mediante el uso de complementos (o extensiones).

Ciertas páginas HTML contienen código embebido para la provisión de funcionalidades al usuario creadas mediante JavaScript (estándar ECMAScript). El intérprete de Javascript será el encargado de analizar y ejecutar dicho código.

La existencia de módulos de interpretación de código difiere de un navegador a otro, por lo que podrían existir sistemas intérpretes de otros lenguajes, como applets de Java, AJAX o ActionScript.

Para acceder más fácilmente a los contenidos definidos en un documento HTML los navegadores web suelen incluir un módulo (parser) que permite cargar en memoria una representación en árbol del DOM de la página, dotando de rapidez de acceso a ciertos elementos de la página.

Los navegadores tienen un sistema de persistencia de datos que funciona como almacén de diferentes tipos de datos, relacionados con el almacenamiento de historiales de navegación y el mantenimiento de sesiones de usuario en disco, así como la administración de certificados de seguridad y de las cookies.

1.3. Tecnologías y lenguajes de programación web en entorno cliente

Los lenguajes de programación del entorno de cliente son aquellos que se ejecutan en el navegador web, es decir, en el lado del cliente dentro de una arquitectura cliente-servidor. Vamos a analizar los más importantes actualmente.

HTML

Es una mejora de un lenguaje antiguo. el SGML (Standard Generalized Markup Language), utilizado para referirnos, tanto al tipo de documento, como al lenguaje de marcas o Hyper Text Markup Language, lenguaje que no necesita ser compilado, sino que es interpretado a medida que se avanza en el documento (independientemente de que haya o no errores, porque continuará con la interpretación). Actualmente se encuentra en la versión 5, siendo el estándar de la industria.

XML

Es un lenguaje de etiquetado extensible, muy simple, pero con unas reglas de codificación muy estrictas, similar a HTML (también deriva de SGML). Su objetivo principal es describir datos para su transferencia eficiente y no para mostrarlos.

XHTML

Es muy similar a HTML, porque es una adaptación de HTML al lenguaje XML (es descendiente de éste). Actualmente, los procesos de estandarización de HTML y XHTML siguen procesos paralelos.

CSS (Cascade Style Sheets)

Hojas de Estilo en Cascada, que sirven para separar el formato que se quiere dar a la web de la estructura e instrucciones de esta, definiendo atributos de una sola vez y haciendo referencia a los mismos en el documento HTML.

JavaScript

Es un lenguaje de programación de scripting (interpretado), orientado a objetos, débilmente tipado y con características dinámicas. Se utiliza principalmente su forma del lado del cliente mediante un intérprete en el navegador web (siendo actualmente el estándar de la industria), aunque existe una forma de JavaScript del lado del servidor (nodeJS, actualmente en auge). Posee las especificaciones ECMAScript e ISO/IEC 16262.

AJAX

Este lenguaje de programación, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML) se utiliza para el desarrollo de aplicaciones interactivas o RIA (Rich Internet Applications), las cuales se ejecutan en el navegador del cliente mientras se mantiene la comunicación asíncrona con el servidor en segundo plano, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax utiliza técnicas válidas para múltiples plataformas, sistemas operativos y navegadores, y está basado en estándares abiertos como JS y DOM. Es una combinación de cuatro tecnologías ya existentes:

- XHTML/HTML + CSS para el diseño.
- DOM (lenguaje scripting) para mostrar e interactuar dinámicamente con la información presentada.
- Objeto XMLHttpRequest para intercambiar datos de forma asíncrona con el servidor.
- XML para utilizar transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

SASS

El acrónimo SASS significa hoja de estilo sintácticamente impresionante (Syntactically Awesome Style Sheet) fue lanzado en 2006 y frecuentemente se hace referencia a el como una extensión de CSS, que lo potencia y simplifica.

SASS es muy personalizable y permite el uso de variables, bucles, operaciones matemáticas, funciones de importación.

TypeScript

TypeScript es un lenguaje de programación fuertemente tipado que se basa en JavaScript. Actualmente está en un gran auge y es utilizado por frameworks como React y Angular.

El código TypeScript se convierte en JavaScript, que se ejecuta en cualquier lugar donde se ejecute JavaScript: En un navegador, en Node.js, etc. TypeScript añade sintaxis adicional a JavaScript para permitir una integración más estrecha con el IDE detectando los errores en tiempo de programación.

1.4. Integración del código HTML y Javascript. Herramientas de desarrollo

JavaScript fue creado por Brendan Eich. Vio la luz en el año 1.995 con el nombre de LiveScript, y posteriormente fue nombrado JavaScript. Nació como un lenguaje sencillo destinado a añadir algunas características interactivas a las páginas web, ya que HTML sólo permitía crear páginas estáticas, donde se podía mostrar textos con estilos, pero se necesitaba, además, interactuar con los usuarios.

Actualmente encontramos JavaScript en Internet en aplicaciones como:

- Correo y chat.
- Buscadores de información.
- Relojes, contadores de visitas y fechas.
- Calculadoras, validadores (acceso a intranets, formularios y documentación).
- Detectores de navegadores e idiomas.

¿Pero cómo insertamos el código JavaScript en nuestras páginas web? Existen tres formas de incluir JavaScript en HTML, según nuestras necesidades y las de la aplicación. Pero siempre es recomendable que utilicemos la que nos resulte más cómoda, es decir, aquélla que nos proporcione control sobre las cargas de archivos JS, sobre HTML o cualquier CMS (Gestores de Contenidos) del mercado.

1. JavaScript en los elementos de HTML

Si bien es la forma de incluir JS menos utilizada, no requiere nada de código ya que sólo tenemos que codificar la acción que deseemos que se ejecute en un lugar determinado dentro de nuestro código HTML:

```
<span onclick="alert('¡¡Hola Usuario!!')">Haz click aquí</span>
```

Ejemplo de acción JS en una tag típica de HTML.

2. JavaScript en el propio documento HTML

Para introducir JS en cualquier parte de nuestro código HTML se recomienda que se haga entre las tags <head> y </head>, ya que es lo primero que se carga al procesar la web. Evidentemente, antes de escribir el código JS debemos abrir las etiquetas <script>. En el siguiente ejemplo vemos cómo funciona:

```
<html>
	<head>
		<title>JS en HTML - 2</title>
		<script type="text/javascript">
			alert('¡¡Hola Usuario!!');
		</script>
	</head>
	<body>
		Bienvenido a JavaScript, Usuario
	</body>
</html>
```

Ejemplo de carga de un JS embebido en el propio documento.

3. JavaScript en un fichero externo

Sin duda, se trata de la mejor opción de todas, ya que disponemos del código en diferentes páginas llamando únicamente a un sólo archivo o varios, según la aplicación o site que estemos desarrollando.

Aunque la tónica general puede no ser la normal en desarrollo, es muy recomendable separar cada archivo JS para cada acción que queramos realizar, al igual que se debería hacer con los archivos CSS. El problema viene a la hora de realizar SEO (Search Engine Optimization, es decir, la optimización de la página para obtener mejores resultados en los buscadores), ya que esta acción puede perjudicarnos al tener múltiples archivos repartidos en carpetas JS que se dedican a cosas diferentes. Es una de las luchas más controvertidas que tenemos en la actualidad: desarrolladores VS SEOs:

- Por comodidad, lo mejor sería tener un solo archivo con extensión .js y separar acciones de diferentes partes del código o site mediante comentarios.
- Por ética profesional, lo mejor sería separar las acciones que desarrollemos en tantos archivos con extensión .js como necesitemos, independientemente del posible perjuicio SEO.

Para ello, utilizaremos de nuevo la etiqueta <script>, que podemos repetir para insertar diferentes archivos JS en un mismo sitio por medio del atributo src, que apunta la url del archivo JS que se quiere enlazar.
Observa cómo funciona:

```
<html>
	<head>
		<title>JS externo</title>
		<script type="text/javascript" src="acciones_1.js"></script>
		<script type="text/javascript" src="acciones_2.js"></script>
	</head>
	<body>
		Bienvenido a JavaScript
	</body>
</html>
```

Ejemplo de carga de un JS externo

Peculiaridades de los JS externos

Muchas veces el error y el problema viene a la hora referenciar el chero origen .js, ya que dependemos de la localización física de ese fichero. Lo normal es que el archivo esté en el mismo directorio que el HTML, aunque en los CMS actuales no es así: pueden estar dentro de cualquier carpeta de un plugin, en la propia raíz...

Si es así, los podemos enlazar a otros cheros de JavaScript localizados en directorios diferentes de nuestro servidor o dominio, lo que nos obligará a referenciar de forma absoluta el fichero, de modo que la ruta será algo parecida al siguiente ejemplo:

```
<script type="text/javascript" src="http://www.midominio.es/archivo.js">
</script>
<script type="text/javascript" src="../js/archivo.js"></script>
```

El fichero archivo.js se encuentra en el directorio anterior (../) al actual, dentro de la carpeta js/. Así subimos o bajamos dentro del árbol de directorios hasta encontrar la carpeta donde esté el archivo a cargar.

¿Qué pasa cuando el navegador no soporta JavaScriprt?

Aunque no es lo normal, hay ocasiones en las que un navegador no dispone del soporte de JS o puede que el usuario lo haya bloqueado creyendo que navega de forma más segura. Cuando esto ocurre, podemos insertar la etiqueta <noscript>, que nos permite introducir un aviso si se detecta que JS está bloqueado y mostrar su interior. Esta etiqueta se suele colocar al principio del <body>.

Veamos cómo funciona:

```
<html>
	<head>
		<title>
			Introducción de código js en las etiquetas script
		</title>
		<script type="text/javascript" src="script.js"></script>
	</head>
```

```
	<body>
		<noscript>
			Usuario, tu navegador no tiene activado Javascript
		</nocript>
		Bienvenido a JavaScript
	</body>
</html>
```

¿Y con XHTML?

Si estamos trabajando con XHTML, la sintaxis para insertar un código de JS es un poco diferente:

```
<script type="text/javascript">
	<!--//--><![CDATA[//><!--
		// código de JavaScript a continuación
	//--><!]]>
</script>
```

Los analizadores de XHTML son intolerantes a los elementos que no comienzan por < y a las tags HTML que no comienzan por &, que son caracteres utilizados por JS, y la razón por la que debemos encapsular las instrucciones de JS en lo que se conoce como sección CDATA.

Herramientas de desarrollo. Visual Studio Code.

Durante el desarrollo del curso usaremos Visual Studio Code como IDE principal durante las clases y videos.

Visual Studio Code es un editor de código fuente ligero pero eficaz que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Esta desarrollado en Typescript, JavaScript y css. Implementa soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Además, incluye compatibilidad integrada con JavaScript, TypeScript y Node.js. Cuenta con un amplio ecosistema de extensiones para otros

lenguajes y entornos de ejecución (como C++, C#, Java, Python, Go, .NET, PHP, SQL, etc.).

Las extensiones son complementos que personalizan y mejoran Visual Studio Code, incluyendo la configuración adicional, las características o el uso de las herramientas existentes.

1.5. Frameworks y librerías específicos

Librerías en JavaScript

En JavaScript, la manera de reutilizar código es usando una biblioteca, es decir, un archivo de JS que contiene muchas funciones precocinadas creadas por un programador que las hace disponibles para todo el mundo.

Al incluir el código de JS en cualquiera de nuestros sites, cargamos en el <HEAD> el archivo o archivos que vamos a necesitar para, posteriormente, programar o utilizar las bibliotecas que necesitemos, de la misma forma que se podría cargar un CSS.

Hay miles de bibliotecas de JS que se pueden importar para una web y muchos son los aspectos en que éstas nos pueden ayudar, si pensamos que una web se compone de la UI (HTML y CSS), interactividad (JS + DOM) y datos (que a menudo traemos por medio de JS):

- Manipulación del DOM (Modelo de Objetos del Documento) y eventos del DOM.
- AJAX / recuperación de datos.
- Efectos y animación.
- Plantillas de HTML, diseño de página, widgets de interfaz de usuario.
- Gráficas, tablas, modelado de datos, rutas y navegación.
- Accesibilidad, soporte para múltiples navegadores y soporte móvil.

Una de las partes más difíciles del desarrollo web es decidir qué bibliotecas utilizar y en qué versión. No hay una respuesta correcta: debes conocer las opciones y después tomar una decisión informada; ésta es la solución más efectiva.

Veamos algunos ejemplos de bibliotecas JS muy conocidas en el mercado actual:

- jQuery
- jQuery UI.
- jQuery Mobile.
- Bootstrap
- Sizzle.
- Qunit.
- modernizr.

```
<!DOCTYPE html>
<html lang=”es”>
	<head>
		<title>Hola mundo con jQuery</title>
		<script type="text/javascript"
		src="http://ajax.microsoft.com/ajax/jquery/jquery-
		1.4.4.min.js"></script>
		<script type="text/javascript">
			$(document).ready(function() {
			$('#hola').text("Hola, Mundo!! Soy jQuery!!"); });
		</script>
	</head>
	<body>
		<div id="hola"></div>
	</body>
</html>
```

Ejemplo típico JS: Hola Mundo con una función creada desde 0.

Frameworks

Hay algunas bibliotecas que se encargan de todo, desde la recuperación de datos hasta la manipulación del DOM y widgets elegantes en la interfaz de usuario. Si se utiliza uno de estos frameworks, lógicamente, estarás incluyendo mucho JS en la página. Para algunos desarrolladores, estos frameworks se encargan de demasiadas cosas, por lo que prefieren usar bibliotecas más pequeñas con tareas más enfocadas, cuando no hay que realizar proyectos de gran envergadura.

Veamos algún ejemplo de los Frameworks más importantes actualmente.

- Angular: Angular es un framework web que permite a los desarrolladores crear aplicaciones rápidas y fiables. Mantenido por un equipo dedicado en Google, Angular proporciona un amplio conjunto de herramientas, APIs y bibliotecas para simplificar y agilizar el flujo de trabajo de desarrollo. Angular es nativa de Typescript.
- React: React es una biblioteca para construir interfaces de usuario. React no es un framework — ni siquiera se limita a la web. React es utilizado con otras bibliotecas para renderizar en ciertos entornos. Por ejemplo, React Native puede usarse para desarrollar aplicaciones móviles; React 360 permite crear aplicaciones de realidad virtual; además de otras posibilidades. En React se puede utilizar JS o Typescript.
- Vue: Vue (pronunciado /vjuː/, como view) es un framework JavaScript para crear interfaces de usuario. Se construye sobre HTML, CSS y JavaScript estándar y proporciona un modelo de programación declarativo basado en componentes que ayuda a desarrollar de forma eficiente interfaces de usuario de cualquier complejidad.
- Ember.js: Ember es un framework JavaScript productivo y probado para crear aplicaciones web modernas. Incluye todo lo necesario para crear interfaces de usuario completas que funcionen en cualquier dispositivo.
- nodeJS: Node es un entorno de ejecución de JavaScript gratuito, de código abierto y multiplataforma que permite a los desarrolladores escribir herramientas de línea de comandos y scripts del lado del servidor fuera de un navegador.

1.6. Referencias bibliográficas

Azaustre, C. (2021). *Aprendiendo JavaScript: Desde cero hasta ECMAScript 6+.* Independently published.

Gómez, M. R. (2021). *Curso de desarrollo Web. HTML, CSS y JavaScript.* ANAYA MULTIMEDIA.

Microsoft. (15 de 07 de 2024). *TypeScript official web site*. Obtenido de https://www.typescriptlang.org/

A fondo

BrowserStack

https://www.browserstack.com/

Esta herramienta permite comprobar la funcionalidad de una página web en más de 700 navegadores, a través de dispositivos físicos para la mayoría de las pruebas, además de los emuladores para Android. BrowserStack se vale de poderosos scripts que buscan y corrigen errores en tiempo real, incluso para páginas web que aún se encuentren en fase de desarrollo.

Validador W3C

https://validator.w3.org/

Software libre creado por el W3C para ayudar a los desarrolladores web a los documentos HTML o XHTML.

Validador W3C CSS

http://jigsaw.w3.org/css-validator/

Software libre creado por el W3C para ayudar a los diseñadores y desarrolladores web a validar Hojas de Estilo en Cascada (CSS).

MDN Web Docs

https://developer.mozilla.org/es/

Portal web de la fundación Mozzila con gran cantidad de documentación, cursos y ejemplo de desarrollo web (HTML, CSS, JavaScript, etc).

Entrenamientos

Entrenamiento 1

- Utiliza la plataforma https://www.browserstack.com/live para analizar el comportamiento de una web de tu elección en los diferentes navegadores y dispositivos.
- Desarrollo paso a paso:
    - Escoge una pagina web a tu elección.
    - Comprueba su comportamiento en los principales navegadores (Chrome, Safari, Firefox, Opera).
    - Repite el proceso simulando dispositivos móviles y viendo su comportamiento responsive.
- Solución: https://www.youtube.com/watch?v=lO1qS2mctN0

Entrenamiento 2

- Desarrollar una página web que disponga de un botón que cada vez que sea presionado escriba un mensaje en la consola.
- Desarrollo paso a paso
    - Generar el documento HTML
    - Definir hoja de estilos
    - Configurar los estilos de la página.
    - Insertar el botón.
    - Capturar el evento.
    - Mostrar por pantalla el mensaje.
- Solución: https://github.com/Anuar-UNIR/DWEC_2024_2025/tree/main/Tema%201/Entrenamiento%202

Entrenamiento 3

- Desarrollar una página web que disponga de un botón que cada vez que sea presionado ejecute una función de un fichero JS externo.
- Desarrollo paso a paso
    - Generar el documento HTML
    - Definir hoja de estilos
    - Configurar los estilos de la página.
    - Insertar el botón.
    - Definir una función en un fichero JS y que muestre un mensaje por consola.
    - Capturar el evento.
- Solución: https://github.com/Anuar-UNIR/DWEC_2024_2025/tree/main/Tema%201/Entrenamiento%203

Entrenamiento 4

- Desarrollar una página web en la que se importe el uso de una librería externa como BootStrap y se use algún elemento sencillo.
- Desarrollo paso a paso:
    - Generar el documento HTML
    - Definir hoja de estilos propios.
    - Importar los estilos y scripts de Bootstrap
    - Configurar los estilos de la página.
    - Insertar y configurar algún elemento de BootStrap.
- Solución: https://github.com/Anuar-UNIR/DWEC_2024_2025/tree/main/Tema%201/Entrenamiento%204

Entrenamiento 5

- Desarrollar una pagina web libre, con uso de librerías (por ejemplo, Bootstrap, o material), CSS.
- Desarrollo paso a paso:
    - Generar el documento HTML
    - Definir hoja de estilos.
    - Definir fichero de scripts (opcional)
    - Configurar los estilos de la página.
    - Diseñar y definir los elementos que se deseen.
- Solución: Es un simple ejemplo, el proyecto es libre para cada alumno. https://github.com/Anuar-UNIR/DWEC_2024_2025/tree/main/Tema%201/Entrenamiento%205