Instituto de Ciencias Tecnológicas

ciisa

Instituto de Ciencias Tecnológicas

De la Universidad San Sebastián

[DESARROLLO BACKEND]

[Sección 71]

[EVALUACIÓN DE UNIDAD 3]

Integrantes

Gonzalo Meneses
David Robles
Gabriel Sanhueza
Álvaro Vargas

Docente

Sebastián Cabezas

Introducción

Este proyecto tiene como objetivo desarrollar una landing page moderna y atractiva para Terrasol Parcelas, una empresa inmobiliaria especializada en la venta de parcelas en el centro sur de Chile. La landing page debe ser responsive, accesible y fácil de usar, permitiendo a los clientes conocer los servicios de Terrasol Parcelas, visualizar las parcelas disponibles y contactarse con la empresa.

Tecnologías utilizadas (PHP, JavaScript, Bootstrap).

- **HTML5:** El lenguaje base para la estructura de la página web.
- **CSS3:** Para el diseño visual y la adaptación a diferentes dispositivos.
- **JavaScript:** Para la interactividad y la gestión de datos dinámicos.
- **Bootstrap:** Un framework CSS para simplificar el desarrollo y mejorar la consistencia.
- **API:** Para la integración con un sistema de gestión de contenido (CMS) o un backend para obtener datos dinámicos.
- **Php:** para el consumo de APIs.

Estructura general del proyecto.

```
└── 🗁Evaluacion3_FrontEnd_Ciisarros
    └── 🗁componentes
        └── card.php
        └── casa_en_parcela.php
        └── contacto.php
        └── footer.php
        └── header.php
        └── inicio.php
        └── nosotros.php
        └── parcelas.php
        └── preguntas.php
        └── solo_terreno.php
        └── testimonios.php
    └── 🗁css
        └── styles.css
    └── debug.log
    └── 🗁dummy_data
        └── testimonials.json
    └── Eval_U3_CIISARROS_Manual.pdf
    └── fiveserver.config.js
    └── 🗁functions
        └── funciones.php
        └── procesar_formulario.php
    └── 🗁img
        └── bailey-anselme-Bkp3gLygyeA-unsplash.jpg
        └── bailey-anselme-Bkp3gLygyeA-unsplash.webp
        └── illiya-vjestica-W5FdAcHp7l8-unsplash.bak.jpg
        └── illiya-vjestica-W5FdAcHp7l8-unsplash.bak.webp
        └── jack-b-aLxqnaKgS9A-unsplash.jpg
        └── jack-b-aLxqnaKgS9A-unsplash.webp
        └── mina-rad-qFSQFSmfZkA-unsplash.jpg
        └── mina-rad-qFSQFSmfZkA-unsplash.webp
        └── peter-muscutt-pkg77ZtBCmg-unsplash.jpg
```

```
        └── peter-muscutt-pkg77ZtBCmg-unsplash.webp
        └── screenshots
            └── conexion-sql.png
            └── contactanos-sindata.jpg
            └── contactanos-validacion-recaptcha.jpg
            └── contactanos.jpg
            └── footer-ontop.png
            └── header-nosotros.jpg
            └── parcelas.jpg
            └── puerto.index.php.jpg
            └── script-testimoniales.png
            └── teestimoniales.png
            └── testimonios-preguntas.jpg
        └── testimonials
            └── testimonial-1.png
            └── testimonial-2.png
            └── testimonial-3.png
            └── testimonial-4.png
            └── testimonial-5.png
            └── testimonial-6.png
        └── tomasz-filipek-CUWC-6MRcNg-unsplash.jpg
        └── tomasz-filipek-CUWC-6MRcNg-unsplash.webp
        └── werner-sevenster-JuP0ZG0UNi0-unsplash.jpg
        └── werner-sevenster-JuP0ZG0UNi0-unsplash.webp
    └── index.php
    └── js
        └── script.js
    └── README.md
    └── sql
        └── insert_data.sql
        └── schema_user_tables_data_B.sql
```

Convenciones de Nombres

Variables

- camelCase para nombrar variables.
- Nombres descriptivos de las variables.
- Ejemplo: $nombre_cliente, $precio_parcela.

Funciones

- camelCase para nombrar funciones en JavaScript.
- snake_case para funciones en PHP. Ejemplo: $nombre_cliente, $precio_parcela.

Clases

- Usar PascalCase para nombrar clases.

Estilo de Código

- Espacios para la indentación en PHP y JavaScript.
- Consistencia en todo el archivo.

Espacios y Comillas

- Espacios alrededor de operadores (=, +, -, etc.).
- Comillas dobles para atributos HTML.
- Comillas simples para cadenas en PHP.
- Comillas dobles para cadenas en JavaScript.

Comentarios

- Utilizar comentarios para explicar el propósito de secciones del código o de líneas de código complejas. Ejemplo: // Función para obtener datos de la API.
- Comentar sobre el propósito de las funciones.

Análisis del código

HTML

Semántica y accesibilidad

- Utiliza etiquetas HTML semánticas como header, section, article, nav, footer.
- Incluye títulos (<h1> para el título principal, potencialmente <h2> para subtítulos).
- Define el idioma del documento con lang="es".

Estructura y jerarquía del contenido

- Divide el contenido en secciones utilizando section con identificadores (id) para una mejor estructura.
- Incluye un encabezado (header) y un pie de página (footer).

Buenas prácticas para etiquetas y atributos

- Utiliza charset="UTF-8" para la codificación de caracteres.
- Incluye el viewport meta para el diseño responsivo.
- Carga Bootstrap como biblioteca externa.

CSS

Organización de Estilos

- Uso de: root para definir variables globales de CSS, facilitando la gestión de temas y colores.
- Importación de fuentes al inicio del archivo CSS para asegurar que las fuentes estén disponibles al renderizar el contenido.

Metodologías de CSS (BEM, OOCSS, SMACSS)

- Aplicación de la metodología BEM (Block Element Modifier) para nombrar clases de manera consistente y predecible.
- Separación clara de estilos base, layouts, módulos y estados, siguiendo principios de SMACSS.

Principios de Diseño Responsivo

- Uso de unidades relativas como rem y em para mejorar la escalabilidad y accesibilidad.
- Implementación de imágenes de fondo responsivas utilizando propiedades como background-size, background-repeat, y background-position.
- Aplicación de media queries para ajustar los estilos según el tamaño de la pantalla.

Documentación y Comentarios

- Inclusión de comentarios en el código CSS para explicar bloques de código y su propósito, facilitando la colaboración y el mantenimiento.

Buenas prácticas

- **Importación de fuentes externas:** El código utiliza @import para cargar fuentes de Google Fonts, lo que evita incluir las fuentes directamente en el código CSS y optimiza la carga de la página.
- **Restablecimiento de box-sizing:** Se establece box-sizing: border-box en el elemento html, lo que es una práctica recomendada para un comportamiento más predecible del modelo de caja.

JavaScript

Separación de Código en Funciones

Funciones separadas para manejar eventos y lógica específica, mejorando la legibilidad y mantenibilidad del código.

Manejo de Evento

Uso de addEventListener para asignar eventos, siguiendo las mejores prácticas de manejo de eventos, evitando la sobrescritura de otros manejadores de eventos.

Validación de Formularios

Implementación de validación personalizada de formularios utilizando clases de Bootstrap y eventos de JavaScript.

Manejo de Elementos Dinámicos

- Funciones para mostrar y ocultar elementos basados en la interacción del usuario, como el botón de "volver al inicio" que aparece al hacer scroll.
- Organización y Modularización del Código

Encapsulamiento con IIFE

Uso de una Immediately Invoked Function Expression (IIFE) para evitar la contaminación del ámbito global y encapsular la lógica de validación de formularios.

Funciones Específicas y Reutilizables

División del código en funciones específicas como la gestión del tema oscuro/claro, la validación del formulario y el manejo de eventos de desplazamiento y clic, lo que mejora la claridad y la mantenibilidad.

Uso de Bibliotecas Externas

Aunque comentado, el código muestra la intención de usar jQuery para manejar eventos, lo que indica una buena práctica en el manejo de dependencias externas.

Patrones de Diseño y Principios SOLID

- **Principio de Responsabilidad Única (SRP):** Cada función tiene una única responsabilidad clara, como la gestión del tema oscuro/claro, la validación del formulario y el manejo de eventos de desplazamiento y clic.
- **Principio de Abierto/Cerrado (OCP):** El código está preparado para ser extendido sin modificar el existente, como el manejo de temas, que puede ampliarse sin cambiar el código actual.

Uso de Local Storage

Uso de localStorage para almacenar y recuperar configuraciones del usuario, mejorando la experiencia del usuario.

Estilo de Código

- Consistente uso de la indentación y el espaciado, lo que mejora la legibilidad del código.
- Declaración de Variables con const y let:
- Uso de const y let en lugar de var para declarar variables, alineándose con las mejores prácticas modernas de JavaScript.
- Consistente uso de comillas dobles para cadenas de texto.
- Inclusión de comentarios que explican el propósito de los bloques de código y las acciones que realizan.
- Minimización del acceso al DOM almacenando referencias a elementos en variables locales.

Funciones PHP

Organización y Modularización del Código

La lógica para obtener datos de un endpoint mediante un token está encapsulada en una función getEndpointByToken. Esto hace que el código sea reutilizable y fácil de mantener.

Manejo de Errores

El código verifica si curl_exec devuelve false, lo que indica un error en la solicitud. En caso de error, se devuelve un mensaje descriptivo con curl_error($ch), lo cual es crucial para el diagnóstico de problemas.

Configuración de cURL

- La configuración de cURL es explícita y fácil de seguir. Los curl_setopt para establecer los encabezados HTTP y asegurar que la respuesta se devuelva como una cadena son buenas prácticas.
- Uso de curl_close($ch) para cerrar la sesión de cURL y liberar recursos. Esto es esencial para evitar fugas de memoria y otros problemas relacionados con la gestión de recursos.

Autenticación con Token:

- La inclusión del token de autenticación en los encabezados HTTP muestra una correcta implementación de seguridad para la solicitud a un endpoint protegido.
- Uso Eficiente de Recursos.
- Legibilidad y Mantenibilidad del Código.

Comentarios Explicativos:

El código incluye comentarios que explican cada paso importante del proceso, desde la configuración de la solicitud hasta la verificación de errores. Esto mejora la legibilidad y facilita el mantenimiento del código.

Buenas Prácticas de Seguridad

Pasar el token de autenticación en los encabezados HTTP (Authorization: Bearer) es una práctica segura y estándar para la autenticación en APIs RESTful

Componentes

- **Uso de clases de Bootstrap:** El código utiliza clases de Bootstrap para el diseño de la página, como card, col-md-3, row, etc. Esto permite un desarrollo más rápido y consistente con el framework.
- **Estructura semántica:** Se utilizan elementos HTML semánticos como header, section, article, aside, etc. para definir las diferentes secciones de la página.
- **Imágenes responsivas:** Se utiliza la etiqueta <picture> con srcset para ofrecer imágenes en diferentes formatos según el dispositivo del usuario, mejorando la performance.
- **Formulario con validación:** El formulario de contacto utiliza atributos HTML5 como required y la clase needs-validation para la validación básica del lado del cliente.
- **Uso de Bootstrap:** Se hace uso de las clases de Bootstrap para el diseño y la disposición de los elementos, lo que facilita la creación de un diseño responsive y atractivo.
- **Organización en Contenedores**: Se utilizan contenedores ( ) para agrupar y organizar secciones específicas de la página, lo que mejora la estructura y legibilidad del código.
- **Uso de Imágenes Optimizadas:** Se emplea la etiqueta <picture> para proporcionar versiones optimizadas de las imágenes, utilizando formatos WebP cuando están disponibles, lo que ayuda a mejorar el rendimiento y la velocidad de carga de la página.
- **Validación de Formularios**: Se implementa la validación de formularios utilizando las clases de Bootstrap (needs-validation, was-validated) y el atributo required, lo que garantiza una mejor experiencia de usuario al evitar el envío de formularios incompletos.
- **Accesibilidad:** Se utilizan etiquetas semánticas (<h1>, <h2>, <p>, <button>) y atributos alt en las imágenes para mejorar la accesibilidad y la indexación de los motores de búsqueda.
- **Reutilización de Código**: Se utiliza PHP para incluir contenido dinámico en la página, lo que facilita la reutilización del código y la gestión de contenido dinámico.