TPO Desarrollo Web HTML, CSS y JavaScript

Codo a Codo – Comisión 23529

Proyecto Randart: Página web que ofrece arte y coleccionismo al azar, dando a todos los usuarios la igual oportunidad de exhibir y vender sus obras en la plataforma.

Repositorio: https://github.com/nrmattar/randart

Link de la web: https://randart.netlify.app/

Integrantes:

- Ruiz Mattar, Nahuel - DNI 33193110 (Representante)
- Esposito, Maximiliano Roberto - DNI 31224315
- López Becerra, Analía - DNI 35270596
- Ponce, Emanuel David - DNI 38644043

OBJETIVO DEL PROTYECTO

El objetivo principal de Randart es proporcionar una plataforma en línea en la que los artistas y coleccionistas puedan exhibir sus obras y los usuarios puedan comprarlas o intercambiarlas de manera aleatoria. Además, se busca fomentar la creatividad y la cultura artística en línea.

PLANIFICACIÓN

El proyecto Randart ha sido desarrollado con la metodología ágil, lo que nos ha brindado una mayor flexibilidad y adaptabilidad a los cambios durante el proceso de desarrollo. Partimos de la idea de crear una plataforma web enfocada en el arte y a medida que avanzábamos en las clases, fuimos proponiendo nuevos contenidos y aplicando lo aprendido para mejorar constantemente el proyecto.

Para asegurar una gestión efectiva del proyecto, hemos establecido un cronograma de trabajo y hemos asignado tareas específicas a cada miembro del equipo

ORGANIZACIÓN

El equipo de desarrollo de Randart está formado por cuatro miembros, cada uno con habilidades y conocimientos que complementan el proyecto. Para asegurar una comunicación constante y efectiva, hemos establecido un chat grupal donde discutimos las ideas y al menos dos reuniones semanales a través de Zoom. Esto nos permite colaborar de manera eficiente y garantizar una entrega exitosa del proyecto.

A medida que surgían nuevas ideas, las discutíamos en conjunto y las incorporábamos en un esquema detallado que incluía una lista de temas pendientes. Posteriormente, distribuíamos estos temas según las destrezas y conocimientos de cada miembro del equipo. Para facilitar el desarrollo individual, cada miembro tenía su propia rama (Branch) en GitHub, donde trabajaba en el tema asignado. Una vez finalizado, el representante del equipo se encargaba de unificar los cambios y asegurar la coherencia y calidad del proyecto.

Esta metodología de trabajo nos ha permitido mantener un flujo de trabajo eficiente, maximizando la productividad y asegurando la calidad del proyecto Randart.

DESCRIPCIÓN Y CONTENIDO DEL PROYECTO

Páginas

La web tiene 4 páginas principales accesibles desde su menú de navegación y una página extra la cual esta linkeada al detalle de cada obra desde la galería.

- Inicio (index.html): Introducción de la marca y visión aleatoria con reproducción automática de las obras principales
    - Diseño: randart.css y banner.css
    - Scripts: banner.js y menu.js

- Galería (galería.html): exposición de obras de arte ******* (articulo.html)
    - Diseño: randart.css y articulos.css
    - Scripts: script.js, getarticulo.js

- Directorio (directorio.html): guía de direcciones de exposiciones y galerías físicas
    - Diseño: randart.css y articulos.css
    - Scripts:

- Contacto (formulario.html): formulario de registro validado mediante JavaScript
    - Diseño: randart.css y formulario.css
    - Scripts: formulario.js

Estilos

Se desarrollaron en 4 archivos .css

Randart.css: diseño principal de toda la web aplicado a todos los html. Estos estilos son utilizados para crear una estructura, tipografía, iconos y diseño responsivo en una página web.

Estructura:

- → Todos los elementos tienen un margen de 0.
- → La etiqueta h1 seguida de una etiqueta p tiene un relleno de 0vh en la parte superior, 0vw en los lados y 2vh en la parte inferior.
- → El menú de navegación está fijo en la parte superior.
- → El contenido principal tiene una altura mínima del 70% de la altura visible de la pantalla y un relleno de 0vh en la parte superior, 2vw en los lados y 2vh en la parte inferior.
- → El menú tiene enlaces con un fondo gris y un margen de 0 en la parte superior y derecha.
- → El pie de página tiene un relleno de 3vh en la parte superior, 3vw en los lados y 1.5vw en la parte inferior.
- → La sección "Seguinos" tiene una lista sin viñetas y un margen de 0 a la izquierda.
- → Los enlaces tienen un borde inferior transparente y un color de texto de #585858.
- → Los iconos en los enlaces tienen un tamaño de 2em y un margen de 0.5em a la derecha.

Tipografía:

- → El tamaño de fuente para los encabezados h1 es de 2em.
- → El tamaño de fuente para los párrafos es de 1em.
- → El tamaño de fuente para los enlaces es de 1em.
- → El tamaño de fuente para los elementos del menú es de 1em.

Diseño responsivo:

- → En dispositivos extra pequeños (600px y menos), se oculta el menú de navegación.
- → En dispositivos extra pequeños (600px y menos), se ajusta la altura de la sección "Presentación" a 0vh.
- → En dispositivos extra pequeños (600px y menos), se ajusta el margen superior del contenido principal a 15vh.

→ En dispositivos extra pequeños (600px y menos), se cambia la dirección de la fila del encabezado y se ajusta el espacio entre filas.

→ En dispositivos extra pequeños (600px y menos), se cambia la dirección de la columna del pie de página y se ajusta el espacio entre columnas.

→ En dispositivos extra pequeños (600px y menos), se cambia la disposición de la lista de elementos del directorio a una sola columna.

Medios:

→ Para dispositivos pequeños (600px y más), se aplica un fondo verde a los elementos con la clase "example".

→ Para dispositivos medianos (768px y más), se aplica un fondo azul a los elementos con la clase "example".

→ Para dispositivos grandes (992px y más), se aplica un fondo naranja a los elementos con la clase "example".

→ Para dispositivos extra grandes (1200px y más), se aplica un estilo específico al menú de navegación.

Iconos:

→ Los enlaces con la dirección "#menu" tienen un icono antes y después del texto.

→ El icono antes del texto tiene una imagen de fondo con una línea horizontal.

→ El icono después del texto tiene una imagen de fondo con tres líneas horizontales.

Javascript

Se desarrollaron en 4 archivos .JS

1. banner.js se emplearon varias funciones de validaciones y verificaciones diferentes para que el formulario reaccione, de manera que se pueda actualizar automáticamente al agregarse imágenes al archivo galería.html.
2. Funciones:

- El código comienza ejecutando el evento "DOMContentLoaded,"(lo que significa que se ejecutará una vez que el documento HTML esté completamente cargado en el navegador).
- Luego, se obtienen referencias a dos elementos HTML en el documento: ".gallery" y ".gallery-indicators," los cuáles son contenedores para las imágenes y los indicadores de la galería.
- Se inicializan las variables, currentImageIndex (para rastrear la imagen actual) e images (que se usará para almacenar las imágenes cargadas).

- La función changeImage(index) se define para cambiar la imagen que se muestra en la galería. Oculta la imagen actual y muestra la imagen en el índice index, además de actualizar los indicadores.
- La función updateIndicators() se utiliza para actualizar los indicadores de la galería, resaltando el indicador correspondiente a la imagen actual.
- Luego, el código realiza una solicitud fetch("/templates/galeria.html"), para cargar el contenido de "galeria.html," que probablemente contiene las imágenes y otros elementos relacionados con la galería.
- Cuando se completa la solicitud, el contenido se inserta en un elemento temporal (tempDiv) para facilitar la manipulación.
- Se limpia el contenedor actual de imágenes (galleryContainer) para asegurarse de que esté vacío.
- Se buscan todas las imágenes en el contenido de "galeria.html" y se almacenan en la variable images. Luego, se agregan estas imágenes al contenedor de la galería.
- En el código también agrega un evento clic en el contenedor de imágenes para redirigir al usuario a la página de la galería "galeria.html."

1. Por último, se configura un temporizador para cambiar automáticamente la imagen cada 5 segundos. Esto se hace usando setInterval, que llama a la función changeImage para mostrar la siguiente imagen en la galería.getarticulos.js se emplearon varias funciones de validaciones y verificaciones diferentes para que el formulario reaccione según los datos ingresados por el usuario, y a su vez, guie a este en una correcta forma de insertar la información requerida para proceder con él envió del mismo, con la información más veraz posible.
2. Menu.js se emplearon varias funciones de validaciones y verificaciones diferentes para que el formulario reaccione según los datos ingresados por el usuario, y a su vez, guie a este en una correcta forma de insertar la información requerida para proceder con él envió del mismo, con la información más veraz posible.

3. Formulario.js se emplearon varias funciones de validaciones y verificaciones diferentes para que el formulario reaccione según los datos ingresados por el usuario, y a su vez, guie a este en una correcta forma de insertar la información requerida para proceder con el envió del mismo, con la información más veraz posible.
   En primera instancia se escribió más de una expresión de Javascript (document.getElement ById("nombre").value.trim();)) que se utiliza para acceder al elemento especifico y que permita trabajar con la información registrada por el usuario, con el fin de que a posteriori se le apliquen las verificaciones pertinentes según los criterios y reglas ya pensados de ante mano.

- **`document.getElementById("nombre"):`** Selecciona en este caso el elemento HTML cuyo atributo(id), en este caso, es “nombre”. Teniendo a **`document`**: como el objeto global del documento en sí mismo, **`getElementById`**: como el método para seleccionar el elemento buscado (de donde se toma el dato)
- **`.value`**: Con esta propiedad se busca que lo ingresado o no por el usuario, sea captado para su posterior análisis contractando con las reglas que se escribieron pensando en el tipo de formulario que se desea obtener.

- `.trim()`: `.trim()` este método se utiliza para eliminar los espacios en blanco al principio y al final.

Validaciones para los distintos campos que se deben completar en el formulario

Validación de los campos/input en blanco:

```
if (nombre === ""....) { alert ("Debe llenar todos los datos del formulario"); return false;  }
```

- `If` Es una declaración que contiene las condiciones que se deben cumplir.
- `nombre ===` Esta parte de la condición como así otras que están escrita de la misma manera, pero con otra etiqueta diferente a "nombre" buscan la verificación de un campo especifico.
- `Alert` si la condición en el paso anterior se cumple en esta condición se muestra el mensaje pertinente que se va a revelar al usuario para que revea el input e ingrese la información requerida para proseguir.
- `return false` Si se activa el paso anterior "alert" evita que el formulario se envié y si nos e completan todos los campos obligatorios

Validación de los distintos campos del formulario con las reglas y características que debe tener la información ingresada por parte del usuario:

Ej.

```
var nombreTest = /^([A-Za-zÑñÁáÉéÍíÓóÚú]+['\-]{0,1}[A-Za-zÑñÁáÉéÍíÓóÚú]+)(\s+([A-Za-zÑñÁáÉéÍíÓóÚú]+['\-]{0,1}[A-Za-zÑñÁáÉéÍíÓóÚú]+))*$/.test(nombre); if (nombreTest === false) { alert("Ingrese su primer nombre correctamente"); return false; }
```

- `^`: Con esto se inicia la cadena
- `[A-Za-zÑñÁáÉéÍíÓóÚú]` Permite la incorporación de las letras mayúsculas y minúsculas que estén en el abecedario, así como también las Ñ y distintas vocales con acentos.
- `+` Permite la información incorporada pueda tener mas de una palabra dentro del campo (en este caso, el campo nombre) separadas por un espacio
- `['\-]` Esta expresión permite incorporar apóstrofes y guiones al nombre
- `{0,1}` Esta expresión permite las veces que puede repetirse un elemento o sea 0 o 1 vez
- `(nombreTest === false)` Esta expresion busca activar la alerta pertinente en caso de no cumplirse las condiciones previamente mencionadas
- `$`: Indica el final de la cadena.

Ej.

```
var direccionTest = /^[a-zA-Z0-9À-ÿ\s\-,.#]+$/i.test(direccion);
```

- ✓ **`[a-zA-Z0-9À-ÿ\s\-,.#]+`** es una expresión que puntualiza como espera que se ingrese una dirección teniendo caracteres diferentes al campo antes expuesto para el campo "nombre". Permitiendo números y caracteres especiales como el numeral entre otros
- ✓ **`À-ÿ`** Permite la información incorporada pueda tener palabras acentuadas en otros idiomas
- ✓ **`\s`** Esta expresión permite espacio y tabulaciones
- ✓ **`\-,.#`** Esta expresión permite la incorporación de guion, punto, como y numeral en el campo
- ✓ **`$/i`** Esta expresión busca cerrarla expresión regular pero la i buscar hacerla insensible a las letras mayúsculas y minúsculas.

Ej.

```
var telefonoTest = /^\d{7,}$/g.test(telefono);
```

- ✓ **`\d`** esta expresión permite cualquier digito numérico del 0 al 9
- ✓ **`{7,}`** es una cuantificación de que al menos debe haber 7 número o más ingresados
- ✓ **`g`** que no se detenga en la primera coincidencia

ej.

```
var UsuarioTest = /^[a-zA-Z0-9_-]{4,20}$/.test(Usuario);
```

- ✓ **`[a-zA-Z0-9_-]`** Esta expresión permite cualquier combinación con letra mayúscula o minúscula del abecedario para conformar el nombre del usuario, así como también Guion bajo, guion del medio y números,
- ✓ **`{4,20}`** Es una cuantificación restringe a que el nombre de usuario que se debe generar debe tener entre 4 y 20 letras de los valores antes mencionados

ej.

```
var ContraseñaTest = /^(?=.*\d)(?=.*[a-z])(?=.*[A-
Z])(?=.*[\W_]).{8,20}$/.test(Contraseña);
```

- ✓ **`(?=.*\d)`**Es un patron de búsqueda que si hay algún digito y el asterisco permite cualquier carácter antes del digito.
- ✓ **`(?=.*[a-z])`**Es patron busca las letras minúsculas del abcedario en la contraseña
- ✓ **`(?=.*[A-Z])`**Esta expresión es igual a la anterior pero con las letras mayúsculas.
- ✓ **`(?=.*[\W_]`**En este caso se busca verificar si hay un carácter especial o subrayado en la cadena.

EJ.

```
var mailTest = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/.test(mail);
```

- ✓ `@` Al poner en la expresión después del (+) el arroba se busca puntualizar una separación donde este carácter este en medio de las palabras ingresadas
- ✓ `\.` Aquí se busca que igual al caso anterior con el arroba la separación entre ambas palabras debe darse con un punto en el medio, ambos caracteres deben estar de manera obligatoria en el dato ingresado para la validación positiva del campo.
- ✓ `[a-zA-Z]{2,}` Aquí se pone en claro que el dominio debe tener al menos 2 letras, generalmente las paginas oficiales de los países tienden a tener esa característica "AR" "RU" "ES", etc.

- ✓ En definitiva las condiciones en esta expresión buscan condicionar la formación de la contraseña ingresada por el usuario a las siguientes características:
  - ✓ `Contiene al menor un digito`
  - ✓ `Contiene al menos una letra minúscula`
  - ✓ `Contiene al menos una letra mayúscula`
  - ✓ `Contiene al menos un carácter especial o subrayado`
  - ✓ `Tiene una longitud entre 8 y 20 caracteres.`