4. Trabajar con ventanas.

Trabajar con ventanas en el desarrollo web implica gestionar la creación y la interacción entre múltiples ventanas del navegador. Esta técnica se utiliza para mejorar la funcionalidad y la experiencia del usuario al permitir la apertura de contenido en ventanas separadas, lo que puede ser útil para aplicaciones que requieren comparaciones de información o trabajo simultáneo en diferentes documentos.

Actividad 13

Crea una página web sencilla que utilice un iframe para incrustar un video de YouTube y otro iframe para incrustar un mapa de Google.

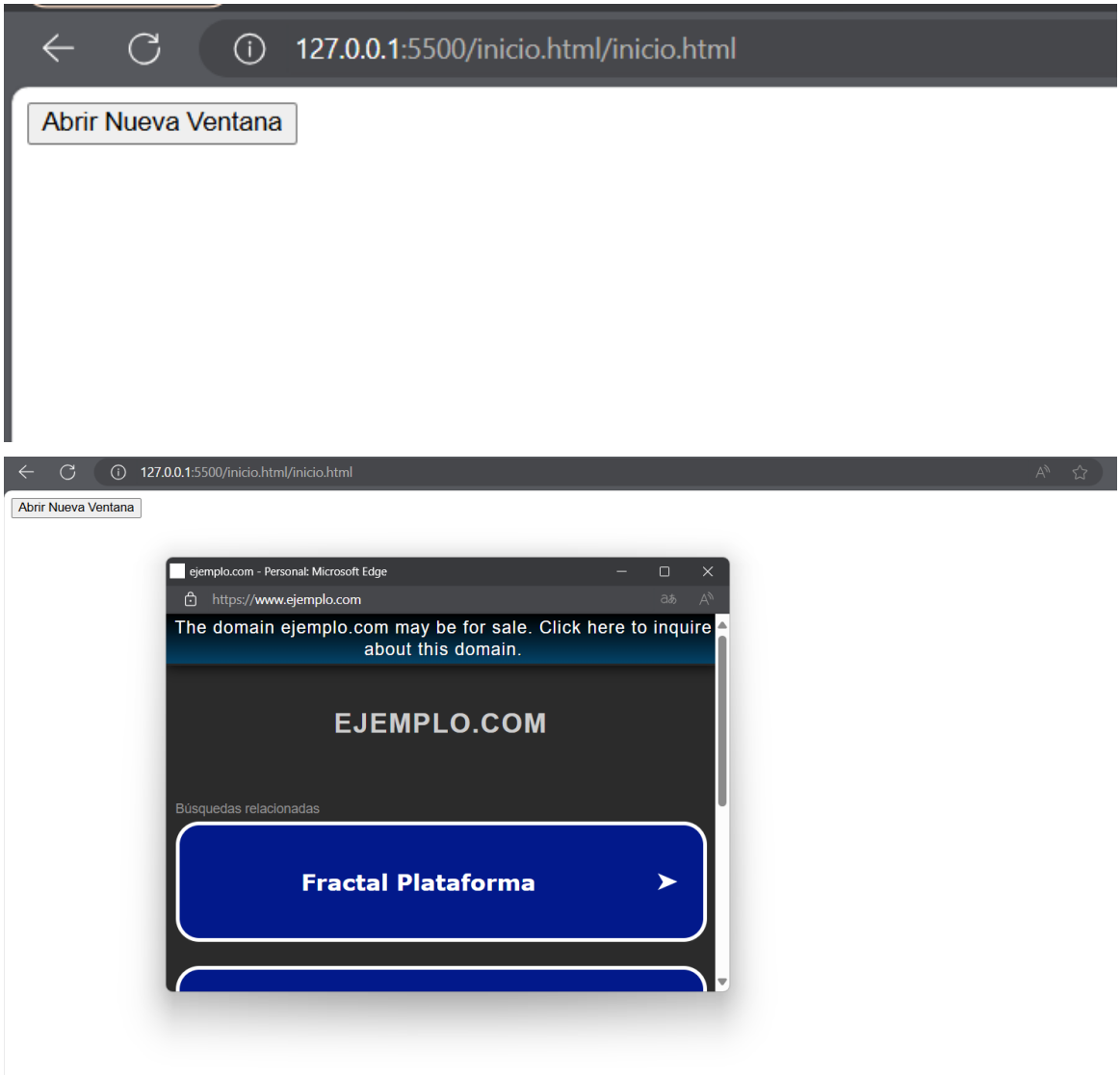
4.1. Creación de varias ventanas.

La creación de varias ventanas en el navegador puede realizarse utilizando JavaScript, permitiendo abrir nuevas ventanas o pestañas con contenido específico. Esto es particularmente útil en aplicaciones web donde se necesita presentar información adicional sin recargar la ventana principal.

Un ejemplo de código para abrir una nueva ventana es el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Gestión de Ventanas</title>
  <script>
    function abrirVentana() {
      window.open("https://www.ejemplo.com", "_blank", "width=600,height=400");
    }
  </script>
</head>
<body>
  <button onclick="abrirVentana()">Abrir Nueva Ventana</button>
</body>
</html>
```

En este ejemplo, al hacer clic en el botón "Abrir Nueva Ventana", se abre una nueva ventana del navegador que carga la URL "https://www.ejemplo.com" con las dimensiones especificadas.



Pie de página: Resultado del código anterior renderizado en un navegador web.

Además de abrir nuevas ventanas o pestañas, es posible controlar diversos aspectos de estas ventanas como la posición en la pantalla, la habilitación o deshabilitación de barras de herramientas, menús y barras de desplazamiento. Esto permite una personalización más precisa de la experiencia del usuario.

A continuación, se expone un ejemplo avanzado de creación de una ventana:

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Gestión Avanzada de Ventanas</title>
  <script>
    function abrirVentana() {
      const opciones = "width=800,height=600,left=200,top=100,toolbar=no,menubar=no,scrollbars=yes";
      window.open("https://www.ejemplo.com", "_blank", opciones);
    }
  </script>
</head>
<body>
  <button onclick="abrirVentana()">Abrir Ventana Personalizada</button>
</body>
</html>
```

En este ejemplo, además del tamaño de la ventana, se especifica su posición y se deshabilitan la barra de herramientas y el menú, mientras que se habilitan las barras de desplazamiento.

Pie de página: Resultado del código anterior renderizado en un navegador web.

4.2. Interactividad entre varias ventanas.

La interactividad entre varias ventanas es importante cuando se necesita comunicación entre ellas. Esto puede lograrse mediante el uso del objeto window y su método postMessage, que permite el envío de mensajes entre ventanas de manera segura.

Un ejemplo de código para la comunicación entre ventanas es el siguiente:

Ventana principal (index.html):

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ventana Principal</title>
  <script>
    let nuevaVentana;

    function abrirVentana() {
      nuevaVentana = window.open("ventana_secundaria.html", "_blank", "width=600,height=400");
    }

    function enviarMensaje() {
      if (nuevaVentana) {
        nuevaVentana.postMessage("Hola desde la ventana principal", "*");
      }
    }

    window.addEventListener("message", (event) => {
      if (event.origin === "http://tudominio.com") {
        alert("Mensaje recibido: " + event.data);
      }
    });
  </script>
</head>
<body>
  <button onclick="abrirVentana()">Abrir Nueva Ventana</button>
  <button onclick="enviarMensaje()">Enviar Mensaje</button>
</body>
</html>
```

Ventana secundaria (ventana_secundaria.html):

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ventana Secundaria</title>
  <script>
    window.addEventListener("message", (event) => {
      if (event.origin === "http://tudominio.com") {
        alert("Mensaje recibido: " + event.data);
        event.source.postMessage("Hola desde la ventana secundaria", event.origin);
      }
    });
  </script>
</head>
<body>
  <h1>Ventana Secundaria</h1>
</body>
</html>
```

En este ejemplo, la ventana principal puede abrir una nueva ventana secundaria y enviarle mensajes. La ventana secundaria también puede responder a estos mensajes. La comunicación se asegura especificando el origen del mensaje, lo que ayuda a mantener la seguridad.

La interactividad entre ventanas no se limita al intercambio de mensajes mediante postMessage. También es posible compartir información a través del almacenamiento local (localStorage) y las cookies, aunque con ciertas limitaciones de seguridad y alcance.

A continuación, se expone un ejemplo del uso de localStorage para compartir información entre ventanas:

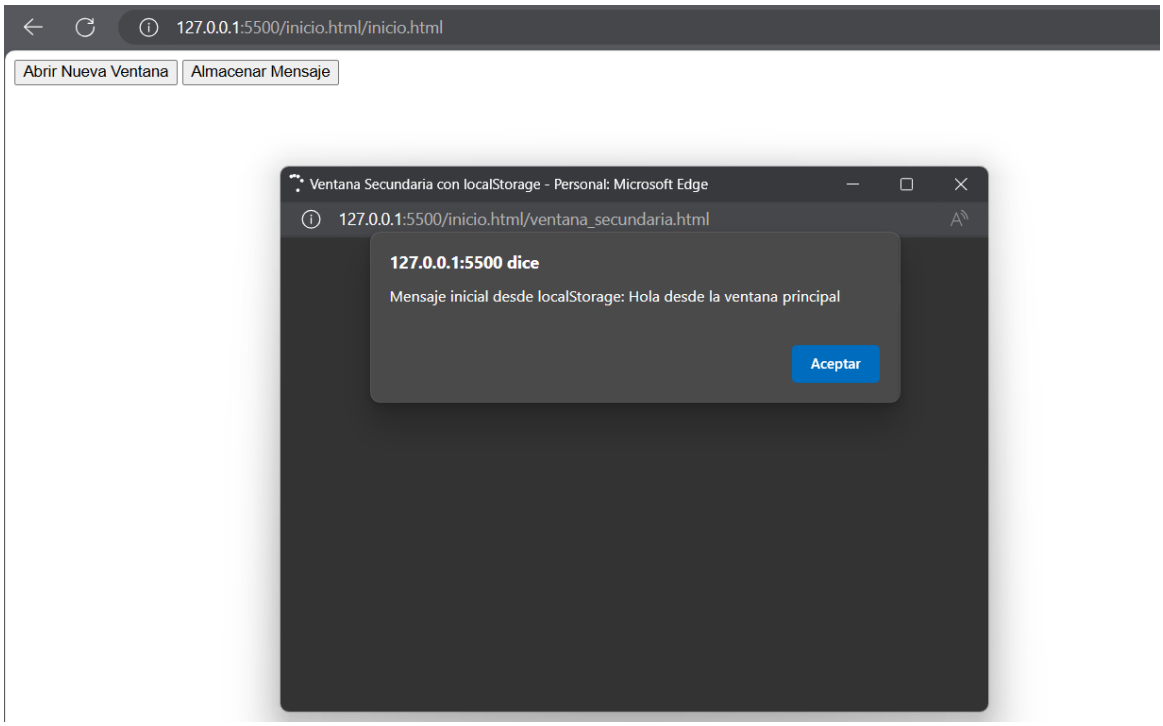
Ventana principal (index.html):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ventana Principal con localStorage</title>
  <script>
    let nuevaVentana;

    function abrirVentana() {
      nuevaVentana = window.open("ventana_secundaria.html", "_blank", "width=600,height=400");
    }

    function almacenarMensaje() {
      localStorage.setItem("mensaje", "Hola desde la ventana principal");
    }
  </script>
</head>
<body>
  <button onclick="abrirVentana()">Abrir Nueva Ventana</button>
  <button onclick="almacenarMensaje()">Almacenar Mensaje</button>
</body>
</html>
```

Ventana secundaria (ventana_secundaria.html):

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ventana Secundaria con localStorage</title>
  <script>
    window.addEventListener("storage", (event) => {
      if (event.key === "mensaje") {
        alert("Mensaje recibido desde localStorage: " + event.newValue);
      }
    });

    window.onload = () => {
      const mensaje = localStorage.getItem("mensaje");
      if (mensaje) {
        alert("Mensaje inicial desde localStorage: " + mensaje);
      }
    };
  </script>
</head>
<body>
  <h1>Ventana Secundaria</h1>
</body>
</html>
```

En este ejemplo, el mensaje almacenado en localStorage en la ventana principal se puede leer en la ventana secundaria tanto al cargar la página como a través del evento storage.



Pie de página: Resultado del código anterior renderizado en un navegador web.

Actividad

14

Crea una página web con un botón que, al hacer clic, abra una nueva ventana del navegador con una URL específica y dimensiones personalizadas. Asegúrate de que la nueva ventana tenga la barra de herramientas deshabilitada y las barras de desplazamiento habilitadas.

Pasos:

1. Crea una estructura básica de HTML.

2. Añade un botón que ejecute una función JavaScript al ser clicado.

3. Implementa la función JavaScript para abrir una nueva ventana con las características especificadas.

La siguiente tabla desglosa cada parte del código utilizado en los ejemplos proporcionados, explicando su propósito y funcionalidad.

	Parte del código	Descripción
Creación de ventanas	<button onclick="abrirVentana() ">Abrir Nueva Ventana</button>	Botón en HTML que, al ser clicado, ejecuta la función abrirVentana para abrir una nueva ventana.
	function abrirVentana() { window.open("https://www.ejemplo.com", "_blank", "width=600,height=400"); }	Función JavaScript que abre una nueva ventana con la URL especificada y con las dimensiones indicadas.

Creación de ventanas avanzado	const opciones = "width=800,height=600,left=200,top=100,toolbar=no,menubar=no,scrollbars=yes"; window.open("https://www.ejemplo.com", "_blank", opciones);	Define opciones avanzadas para la ventana, incluyendo tamaño, posición y elementos de la interfaz deshabilitados o habilitados.
Comunicación entre ventanas	nuevaVentana.postMessage("Hola desde la ventana principal", "*");	Envía un mensaje desde la ventana principal a la nueva ventana abierta.
	window.addEventListener("message", (event) => { if (event.origin === "http://tudominio.com") { alert("Mensaje recibido: " + event.data); } });	Escucha mensajes enviados a la ventana secundaria y muestra una alerta con el contenido del mensaje recibido.
Uso de localStorage	localStorage.setItem("mensaje", "Hola desde la ventana principal");	Almacena un mensaje en localStorage desde la ventana principal.
	const mensaje = localStorage.getItem("mensaje"); if (mensaje) { alert("Mensaje inicial desde localStorage: " + mensaje); }	Recupera y muestra un mensaje almacenado en localStorage cuando la ventana secundaria se carga.
	window.addEventListener("storage", (event) => { if (event.key === "mensaje") { alert("Mensaje recibido desde localStorage: " + event.newValue); } });	Escucha cambios en localStorage y muestra una alerta cuando el mensaje es actualizado.

5. Otros efectos.

En la creación de páginas web modernas, no solo las imágenes y los textos son esenciales para atraer y mantener la atención de los usuarios. Existen otros efectos que pueden ser implementados para enriquecer la experiencia visual y funcional de un sitio web. Estos efectos pueden lograrse utilizando diversas tecnologías y técnicas, como HTML, CSS y la manipulación de capas, cada una aportando sus propias ventajas y posibilidades. A continuación, se exploran diferentes tipos de efectos que pueden integrarse en el diseño web para crear interfaces más interactivas y atractivas.

5.1. Efectos con HTML.

Los efectos con HTML se logran utilizando las capacidades propias del lenguaje para estructurar y presentar contenido de manera interactiva y visualmente atractiva. Un ejemplo común es el uso de elementos multimedia, como videos y audios, que pueden ser integrados directamente en el código HTML para enriquecer la experiencia del usuario. Además, las etiquetas <canvas> y <svg> permiten la creación de gráficos dinámicos y animaciones, ofreciendo posibilidades avanzadas sin la necesidad de plugins externos.

La etiqueta <canvas> se utiliza para dibujar gráficos en tiempo real a través de JavaScript, permitiendo la creación de juegos, gráficos y otras animaciones interactivas. Por otro lado, <svg> permite la inclusión de gráficos vectoriales que son escalables sin pérdida de calidad, ideales para logotipos y diagramas.

```
<canvas id="miCanvas" width="500" height="500"></canvas>
<script>
  var canvas = document.getElementById('miCanvas');
  var context = canvas.getContext('2d');
  context.fillStyle = '#FF0000';
  context.fillRect(50, 50, 150, 150);
</script>
```

Este código crea un rectángulo rojo en un elemento <canvas>.

5.2. Efectos con CSS.

Los efectos de CSS pueden variar desde simples cambios de color hasta animaciones complejas. Las transiciones y animaciones CSS permiten cambiar gradualmente las propiedades de los elementos, como el tamaño, el color y la posición, ofreciendo una experiencia más dinámica y atractiva.

Las transiciones CSS permiten cambios suaves entre dos estados de un elemento. Por ejemplo, cambiar el color de un botón cuando se pasa el ratón por encima.

```
.boton {
    background-color: green;
    transition: background-color 0.5s ease;
  }
  .boton:hover {
    background-color: red;
  }
```

Las animaciones CSS utilizan la propiedad @keyframes para definir puntos clave de una animación, permitiendo crear efectos detallados.

```css
@keyframes desvanecer {
  from { opacity: 1; }
  to { opacity: 0; }
}
.animacion {
  animation: desvanecer 3s infinite;
}
```

Actividad

15

Crea una página web con un botón que, al hacer clic, dibuje un rectángulo rojo en un elemento <canvas>. Además, aplica una transición CSS a otro botón para que cambie de color cuando se pasa el ratón por encima.

Además de las transiciones y animaciones, CSS ofrece una variedad de otros efectos que puedes implementar. Aquí algunos ejemplos:

Transformaciones CSS:

Con transformaciones CSS puedes alterar la forma, tamaño, posición y orientación de un elemento, incluyendo rotaciones, escalado, sesgado y desplazamientos. Por ejemplo:

```css
/* Seleccionamos el elemento con la clase 'elemento' */
.elemento {
    /* La propiedad 'transform' permite aplicar transformaciones al elemento */

    /* ROTACIÓN */
    /* 'rotate(45deg)' rota el elemento 45 grados en sentido horario */
    transform: rotate(45deg);

    /* ESCALADO */
    /* 'scale(2)' aumenta el tamaño del elemento al doble en ambos ejes (ancho y alto) */
    transform: scale(2);

    /* TRASLACIÓN */
    /* 'translate(50px, 100px)' mueve el elemento 50px a la derecha y 100px hacia abajo */
    transform: translate(50px, 100px);

    /* SESGADO */
    /* 'skewX(20deg)' inclina el elemento 20 grados en el eje X */
    transform: skewX(20deg);
}

/* También puedes combinar múltiples transformaciones en una sola propiedad 'transform' */
.elemento-combinado {
    /* Primero rota el elemento 5 grados, luego lo escala al doble de su tamaño, y finalmente lo
    traslada 25px a la derecha y 150px hacia abajo */
    transform: rotate(5deg) scale(2) translate(25px, 150px);
}
```

Sombras CSS:

Utilizando las propiedades box-shadow y text-shadow, puedes añadir sombras a los elementos y texto para crear una sensación de profundidad y resaltarlos en la página. Por ejemplo:

```css
/* Seleccionamos el elemento con la clase 'caja' */
.caja {
    /* La propiedad 'box-shadow' permite aplicar sombras a las cajas */

    /* '10px 10px 5px 0 rgba(0, 0, 0, 0.3)' aplica una sombra difuminada de color negro
    con un desplazamiento de 10px a la derecha y 10px hacia abajo, un desenfoque de 5px y sin expansión */
    box-shadow: 10px 10px 5px 0 rgba(0, 0, 0, 0.3);
}

/* También puedes aplicar múltiples sombras a una misma caja */
.caja-multiple {
    /* Primero aplica una sombra difuminada de color negro con un desplazamiento de 10px a la derecha y
    10px hacia abajo, un desenfoque de 5px y sin expansión, y luego aplica una sombra interna difuminada de color rojo con un desplazamiento de
    box-shadow: 10px 10px 5px 0 rgba(0, 0, 0, 0.3), inset 5px 5px 10px 0 rgba(255, 0, 0, 0.5);
}

/* Seleccionamos el elemento con la clase 'texto' */
.texto {
    /* La propiedad 'text-shadow' permite aplicar sombras al texto */

    /* '2px 2px 2px rgba(0, 0, 0, 0.3)' aplica una sombra difuminada de color negro con un desplazamiento de
    2px a la derecha y 2px hacia abajo, y un desenfoque de 2px */
    text-shadow: 2px 2px 2px rgba(0, 0, 0, 0.3);
}
```

Gradientes CSS:

Los gradientes te permiten mostrar una transición suave entre dos o más colores, pudiendo ser lineales o radiales. Por ejemplo:

```css
/* Seleccionamos el elemento con la clase 'elemento' */
.elemento {
    /* TRANSFORMACIONES */
    /* 'rotate(45deg)' rota el elemento 45 grados en sentido horario */
    transform: rotate(45deg);
    /* 'scale(2)' aumenta el tamaño del elemento al doble en ambos ejes (ancho y alto) */
    transform: scale(2);
    /* 'translate(50px, 100px)' mueve el elemento 50px a la derecha y 100px hacia abajo */
    transform: translate(50px, 100px);
    /* 'skewX(20deg)' inclina el elemento 20 grados en el eje X */
    transform: skewX(20deg);

    /* SOMBRAS */
    /* '10px 10px 5px 0 rgba(0, 0, 0, 0.3)' aplica una sombra difuminada de color negro con un
    desplazamiento de 10px a la derecha y 10px hacia abajo, un desenfoque de 5px y sin expansión */
    box-shadow: 10px 10px 5px 0 rgba(0, 0, 0, 0.3);

    /* GRADIENTES */
    /* 'linear-gradient(blue, pink)' crea un gradiente lineal que va de azul a rosa de arriba a abajo */
    background: linear-gradient(blue, pink);
}
```

Filtros CSS:

Los filtros CSS te permiten aplicar efectos visuales a los elementos, tales como desenfoque, brillo, contraste, saturación, entre otros. Por ejemplo:

```css
/* Seleccionamos el elemento con la clase 'elemento' */
.elemento {
/* FILTROS */
    /* 'blur(5px)' aplica un desenfoque gaussiano al elemento */
    filter:blur(5px);
    /* 'brightness(0.4)' reduce el brillo del elemento al 40% */
    filter:brightness(0.4);
    /* 'contrast(200%)' duplica el contraste del elemento */
    filter:contrast(200%);
    /* 'grayscale(50%)' convierte el 50% del color del elemento a escala de grises */
    filter:grayscale(50%);
    /* 'hue-rotate(90deg)' rota el matiz de los colores del elemento 90 grados */
    filter:hue-rotate(90deg);
    /* 'invert(75%)' invierte el 75% de los colores del elemento */
    filter:invert(75%);
    /* 'opacity(25%)' reduce la opacidad del elemento al 25% */
    filter:opacity(25%);
    /* 'saturate(30%)' reduce la saturación del color del elemento al 30% */
    filter:saturate(30%);
    /* 'sepia(60%)' convierte el 60% del color del elemento a sepia */
    filter:sepia(60%);
}
```

Modos de mezcla CSS:

Estos modos determinan cómo se combinan los colores de elementos superpuestos. Por ejemplo:

```css
/* Seleccionamos el elemento con la clase 'elemento' */
.elemento {
/* MODOS DE MEZCLA */
    /* 'multiply' multiplica los colores de la capa superior con los de la capa inferior */
    mix-blend-mode: multiply;
    /* 'screen' invierte los colores de la capa superior y los multiplica con los de la capa inferior */
    mix-blend-mode: screen;
    /* 'overlay' combina Multiply y Screen, oscureciendo o aclarando la capa inferior dependiendo de sus colores */
    mix-blend-mode: overlay;
    /* 'darken' conserva los colores más oscuros de las dos capas */
    mix-blend-mode: darken;
    /* 'lighten' conserva los colores más claros de las dos capas */
    mix-blend-mode: lighten;
    /* 'color-dodge' aclara los colores de la capa inferior dependiendo de los colores de la capa superior */
    mix-blend-mode: color-dodge;
    /* 'color-burn' oscurece los colores de la capa inferior dependiendo de los colores de la capa superior */
    mix-blend-mode: color-burn;
    /* 'hard-light' es el resultado de multiply si el color superior es más oscuro, o screen si el color superior es más claro */
    mix-blend-mode: hard-light;
    /* 'soft-light' oscurece o aclara los colores, dependiendo de los colores de la capa superior */
    mix-blend-mode: soft-light;
    /* 'difference' resta los colores de la capa superior de los de la capa inferior */
    mix-blend-mode: difference;
    /* 'exclusion' crea un efecto similar a 'difference', pero con menor contraste */
    mix-blend-mode: exclusion;
    /* 'hue' crea una mezcla con la luminosidad y la saturación de la capa inferior, y el tono de la capa superior */
    mix-blend-mode: hue;
    /* 'saturation' crea una mezcla con la luminosidad y el tono de la capa inferior, y la saturación de la capa superior */
    mix-blend-mode: saturation;
    /* 'color' crea una mezcla con la luminosidad de la capa inferior, y el tono y la saturación de la capa superior */
    mix-blend-mode: color;
    /* 'luminosity' crea una mezcla con el tono y la saturación de la capa inferior, y la luminosidad de la capa superior */
    mix-blend-mode: luminosity;
}
```

Variables CSS:

También conocidas como propiedades personalizadas, las variables CSS te permiten almacenar valores específicos para reutilizarlos en todo el documento CSS. Por ejemplo:

```css
/* Seleccionamos el elemento con la clase 'elemento' */
.elemento {
 /* VARIABLES CSS */
    /* '--color-principal: red;' define una variable llamada
    'color-principal' con el valor 'red' */
    --color-principal: red;
    /* 'color: var(--color-principal);' aplica el color
    almacenado en la variable 'color-principal' al elemento */
    color: var(--color-principal);
}
```

A continuación, se expone un ejemplo de cómo se pueden combinar todas estas técnicas en un documento HTML:

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        /* Definimos una variable CSS */
        :root {
            --color-principal: #ff6347;
        }

        /* Seleccionamos el elemento con la clase 'elemento' */
        .elemento {
            /* Aplicamos la variable CSS */
            background-color: var(--color-principal);

            /* Aplicamos una transformación */
            transform: rotate(20deg);

            /* Aplicamos una sombra */
            box-shadow: 10px 10px 5px 0 rgba(0, 0, 0, 0.3);

            /* Aplicamos un gradiente */
            background: linear-gradient(to right, red, orange, yellow, green, blue, indigo, violet);

            /* Aplicamos un filtro */
            filter: blur(2px);

            /* Aplicamos un modo de mezcla */
            mix-blend-mode: multiply;

            /* Otros estilos para hacer visible el elemento */
            width: 200px;
            height: 200px;
            margin: 50px;
            padding: 50px;
            text-align: center;
            color: white;
        }
    </style>
</head>
<body>
    <!-- Creamos un elemento con la clase 'elemento' -->
    <div class="elemento">Hola, mundo!</div>
</body>
</html>
```

Pie de página: Código anterior renderizado en un navegador web.

5.3. Efectos con capas.

Los efectos con capas se logran combinando HTML y CSS para manipular la disposición y visibilidad de diferentes elementos en la página. Las capas permiten superponer contenido, creando efectos de profundidad y mejorando la interacción visual. Utilizando propiedades como position, z-index, y opacity, es posible controlar la visibilidad y el orden de apilamiento de los elementos. Por ejemplo:

```
<div class="capa1">Capa 1</div>
<div class="capa2">Capa 2</div>
<style>
  .capa1, .capa2 {
    position: absolute;
    top: 50px;
    left: 50px;
    width: 100px;
    height: 100px;
  }
  .capa1 {
    background-color: green;
    z-index: 1;
  }
  .capa2 {
    background-color: red;
    z-index: 2;
    opacity: 0.5;
  }
</style>
```

En este ejemplo, la capa roja (capa2) se superpone sobre la capa verde (capa1) debido a su mayor z-index y se muestra semi-transparente gracias a la propiedad opacity.

A continuación, se presenta un código HTML que incluye una combinación de imágenes, efectos de sustitución, animaciones de texto, y un menú de navegación, todos optimizados para mejorar la experiencia del usuario y la interacción con el sitio web. La temática de este ejemplo es sobre una página de autocaravanas.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Autocaravanas y Naturaleza</title>
    <style>
        /* Estilos generales para el cuerpo del documento */
        body {
            font-family: Arial, sans-serif;
            background-color: #E8F5E9;
            color: #2E7D32;
            margin: 0;
            padding: 0;
        }
        /* Estilos para el encabezado */
        header {
            background-color: #388E3C;
            color: white;
            text-align: center;
            padding: 1em 0;
        }
        /* Estilos para el menú de navegación */
        nav {
            background-color: #2E7D32;
            overflow: hidden;
        }
        nav a {
            float: left;
            display: block;
            color: white;
            text-align: center;
            padding: 14px 16px;
            text-decoration: none;
        }
```

```
36          nav a:hover {
37              background-color: #45a049;
38          }
39          /* Contenedor principal */
40          .container {
41              width: 80%;
42              margin: auto;
43              padding: 1em 0;
44          }
45          /* Estilos para la sección hero */
46          .hero {
47              position: relative;
48              text-align: center;
49              color: white;
50          }
51          .hero img {
52              width: 100%;
53              height: auto;
54          }
55          .hero-text {
56              position: absolute;
57              top: 50%;
58              left: 50%;
59              transform: translate(-50%, -50%);
60              font-size: 2em;
61              background-color: rgba(56, 142, 60, 0.7);
62              padding: 0.5em 1em;
63              border-radius: 10px;
64          }
```

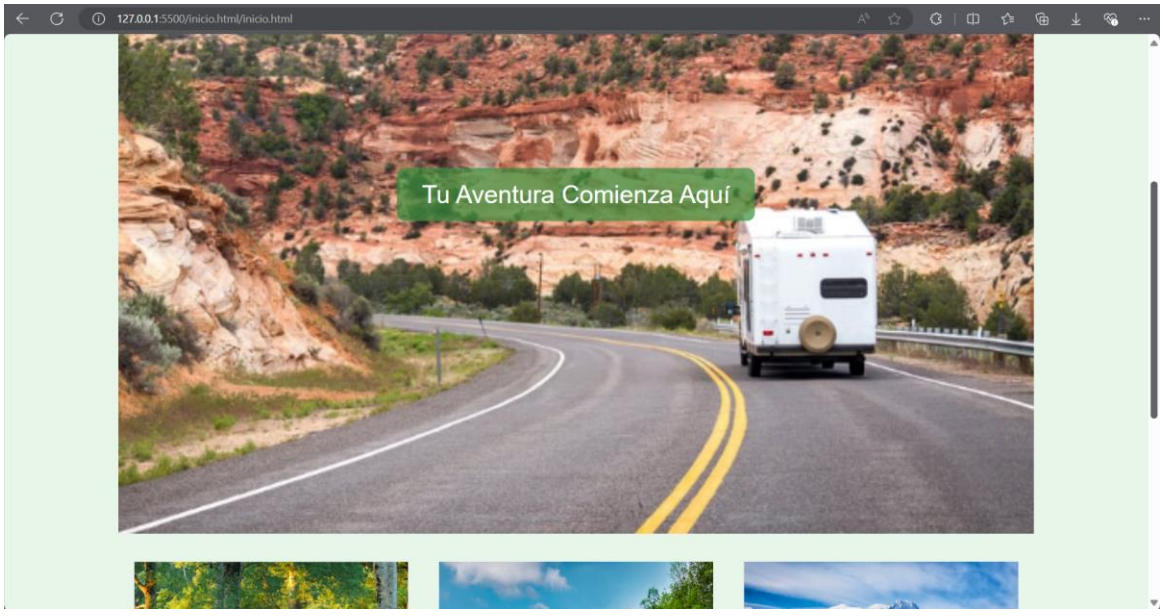
```
/* Estilos para la galería de imágenes */
.gallery {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-around;
    margin: 2em 0;
}
.gallery img {
    width: 30%;
    margin-bottom: 1em;
    transition: transform 0.3s ease;
}
.gallery img:hover {
    transform: scale(1.1);
}
/* Estilos para la sección de efectos de texto */
.text-effects {
    text-align: center;
    margin: 2em 0;
}
.fade-text {
    opacity: 0;
    animation: fadeIn 3s forwards;
}
@keyframes fadeIn {
    to {
        opacity: 1;
    }
}
```

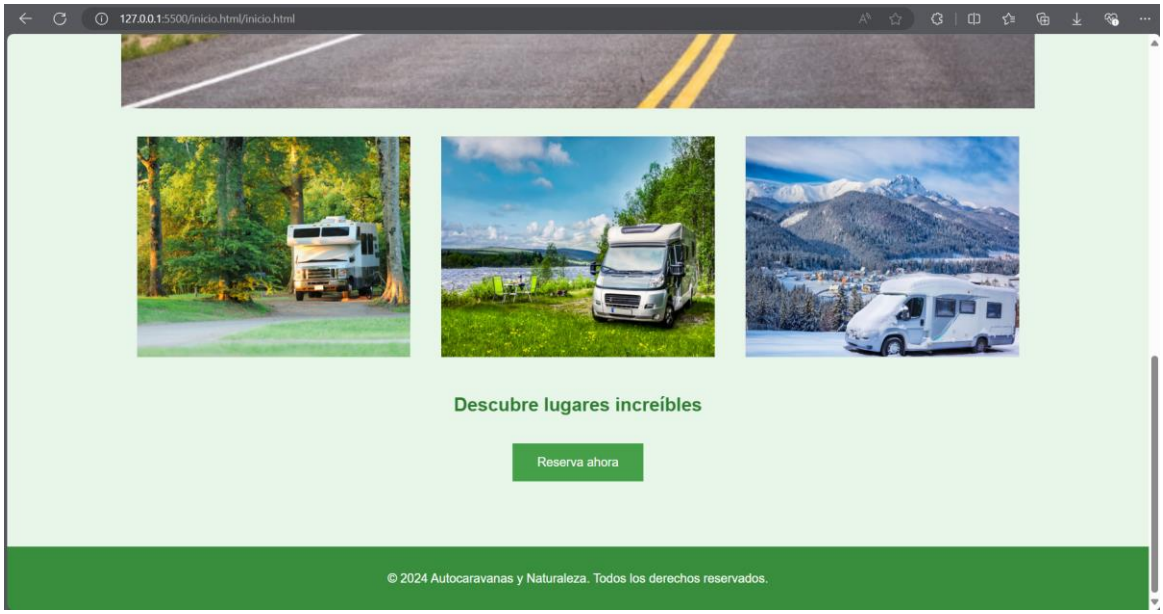
```
        /* Estilos para el contenedor del botón */
        .button-container {
            text-align: center;
            margin: 2em 0;
        }
        .button {
            background-color: #4CAF50;
            border: none;
            color: white;
            padding: 15px 32px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            margin: 4px 2px;
            cursor: pointer;
            transition: background-color 0.3s ease;
        }
        .button:hover {
            background-color: #45a049;
        }
        /* Estilos para el pie de página */
        footer {
            background-color: #388E3C;
            color: white;
            text-align: center;
            padding: 1em 0;
            position: relative;
            width: 100%;
            margin-top: 2em;
        }
    </style>
</head>
```

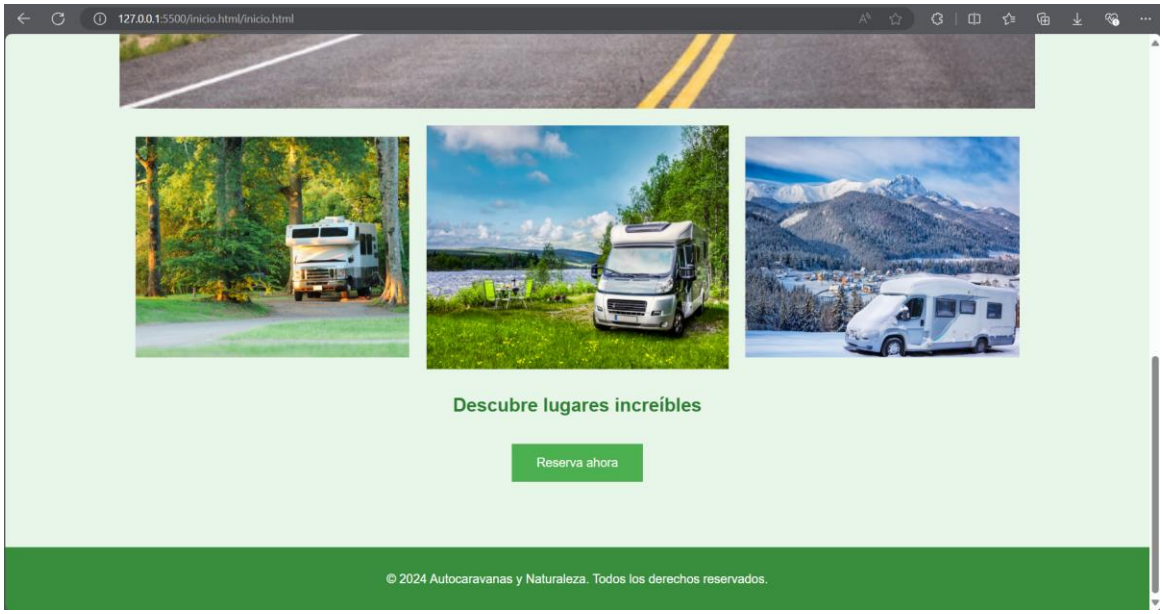
```
<body>
    <!-- Encabezado de la página -->
    <header>
        <h1>Explora el mundo con nuestras autocaravanas</h1>
    </header>
    <!-- Menú de navegación -->
    <nav>
        <a href="#home">Inicio</a>
        <a href="#services">Servicios</a>
        <a href="#contact">Contacto</a>
    </nav>
    <!-- Contenedor principal -->
    <div class="container">
        <!-- Sección hero con imagen y texto superpuesto -->
        <div class="hero">
            <img src="https://example.com/hero-image.jpg" alt="Autocaravana en la naturaleza">
            <div class="hero-text">Tu Aventura Comienza Aquí</div>
        </div>
        <!-- Galería de imágenes -->
        <section class="gallery">
            <img src="https://example.com/image1.jpg" alt="Autocaravana en el bosque">
            <img src="https://example.com/image2.jpg" alt="Autocaravana junto al lago">
            <img src="https://example.com/image3.jpg" alt="Autocaravana en la montaña">
        </section>
        <!-- Sección de efectos de texto -->
        <section class="text-effects">
            <h2 class="fade-text">Descubre lugares increíbles</h2>
        </section>
        <!-- Contenedor del botón de reserva -->
        <div class="button-container">
            <button class="button">Reserva ahora</button>
        </div>
    </div>
```

```
    <!-- Pie de página -->
    <footer>
        <p>&copy; 2024 Autocaravanas y Naturaleza. Todos los derechos reservados.</p>
    </footer>
    <!-- Script para animar el texto -->
    <script>
        document.addEventListener('DOMContentLoaded', function () {
            document.querySelector('.fade-text').style.opacity = 1;
        });
    </script>
</body>
</html>
```


127.0.0.1:5500/inicio.html/inicio.html
Explora el mundo con nuestras autocaravanas
Inicio
Servicios
Contacto
Tu Aventura Comienza Aquí


127.0.0.1:5500/inicio.html/inicio.html
Tu Aventura Comienza Aquí


127.0.0.1:5500/inicio.html/inicio.html
Descubre lugares increíbles
Reserva ahora
© 2024 Autocaravanas y Naturaleza. Todos los derechos reservados.


127.0.0.1:5500/inicio.html/inicio.html
Descubre lugares increíbles
Reserva ahora
© 2024 Autocaravanas y Naturaleza. Todos los derechos reservados.

6. Prueba de autoevaluación.

¿Cuál es una de las técnicas mencionadas para optimizar imágenes en una página web?

a) Usar solo imágenes PNG

b) Usar formatos modernos como WebP

c) Usar imágenes de alta resolución sin compresión

¿Qué atributo HTML se puede utilizar para implementar la carga diferida de imágenes?

a) srcset

b) sizes

c) loading="lazy"

¿Cuál es una ventaja de usar la librería Animate.css?

a) Facilita la creación de animaciones personalizadas mediante JavaScript

b) Ofrece una colección de animaciones predefinidas que se aplican fácilmente con clases CSS

c) Permite la edición en vivo de animaciones en el navegador

¿Qué propiedad CSS se utiliza para crear una animación que cambia el color del texto suavemente?

a) transform

b) transition

c) animation

¿Cuál de las siguientes es una técnica recomendada para asegurarse de que los efectos no perjudiquen la accesibilidad y la legibilidad?

a) Evitar el uso de cualquier animación

b) Utilizar animaciones intensas y colores llamativos

c) Mantener un contraste adecuado entre el texto y el fondo

Las imágenes deben ser _____ para asegurar que se adapten a diferentes tamaños de pantalla sin perder calidad.

La técnica de _____ permite retrasar la carga de imágenes no visibles hasta que el usuario se desplaza hacia ellas.

Para mejorar el rendimiento de la página, es importante utilizar _____ para comprimir las imágenes sin pérdida significativa de calidad.

La propiedad CSS @keyframes se utiliza para definir puntos clave de una _____.

GSAP es una librería avanzada para crear _____ complejas y personalizadas.

Pruebas y verificación en páginas web