

Práctica

Introducción

¡Hola nuevamente! 

En nuestra clase práctica de hoy, nos pondremos manos a la obra con el fin de afianzar lo aprendido de una manera divertida y dinámica.

¿Recuerdas las clases anteriores? Pues hoy haremos que todas ellas confluyan en ejercicios prácticos y reales. Cada vez que encontremos un problema, iremos descubriendo poco a poco cómo encajan esas piezas para resolverlo.

¡Buena suerte! 

Ejercicio 1

En este ejercicio, te desafiamos a crear un archivo HTML y utilizar etiquetas `<div>` para diseñar una estructura con diferentes estilos.

Sigue los siguientes pasos para completar el ejercicio:

1. Abrir Visual Studio Code y crear un archivo HTML (aparte de tu sitio web en el que vienes trabajando), como por ejemplo: *"ejercicio-1.html"*.
2. Dentro del archivo HTML, crear la estructura básica de un documento HTML agregando las etiquetas `<html>`, `<head>` y `<body>`.
3. Dentro del `<body>`, agregar tres etiquetas `<div>` y anidarlas una dentro de la otra para crear una estructura en capas.
4. Crear un nuevo archivo CSS (por ejemplo, *"estilos.css"*) y enlazarlo con el archivo HTML utilizando la etiqueta `<link>`.
5. Aplicar los siguientes estilos CSS al *"div del medio"* utilizando estas propiedades:


- **margin:** 30px
- **padding:** 20px
- **width:** 400px (ancho)
- **height:** 200px (alto)

6. Agregar un color de fondo diferente a cada uno de los tres “divs” utilizando la propiedad “background-color”. Asegúrate de que cada uno tenga un color de fondo diferente.

7. Lograr que el “div interno” ocupe todo el contenido del “div del medio” y que el borde del “div externo” coincida con el **margin** de 30px del “div del medio”.


En la estructura final deberías ver como los div están anidados y centrados uno dentro del otro y, al revisar el *Box Model* del “div del medio”, deberías poder observar como el **padding** y el **margin** coinciden con la distancia de separación entre los “divs”.

 [Ver video](#)

 *Para hacer coincidir el borde de “div externo” con el margen del “div del medio”, vamos a tener que darle un ancho (width) y alto (height) al “div externo”. Para calcular las medidas correspondientes a ese ancho y a ese alto debemos: sumar los valores del padding y margin del “div del medio” y multiplicar el resultado por dos. Luego, tenemos que sumar ese resultado tanto al ancho como al alto del “div del medio” para obtener las medidas para el “div externo”.*

8. Guardar los cambios de tu archivo HTML.

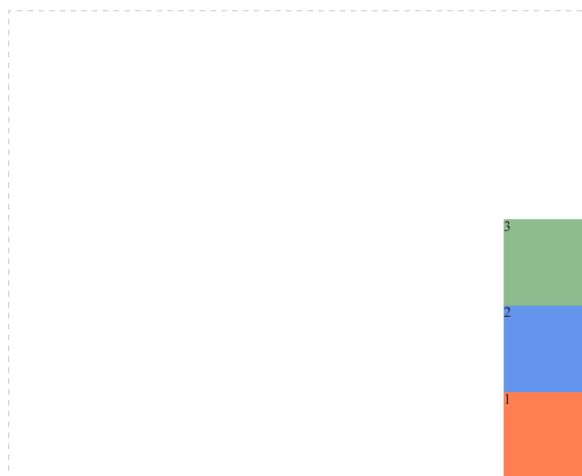
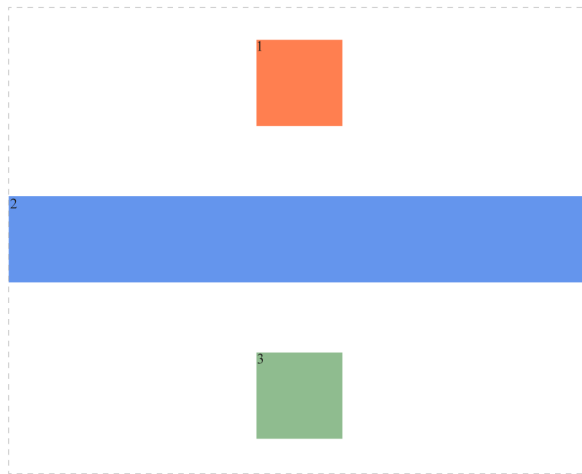
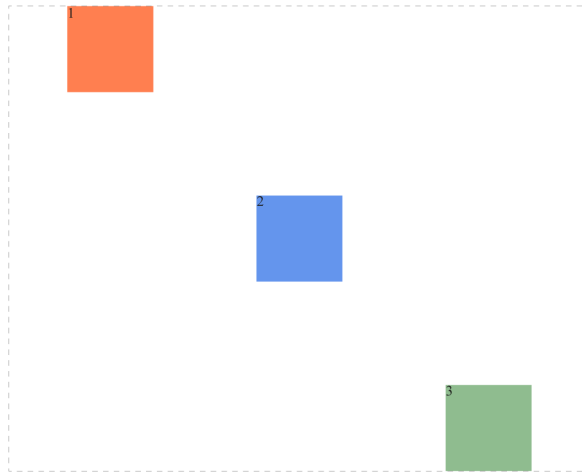
9. Compartir con tus compañeros cómo se visualiza el archivo que acabas de crear.

 *Recuerda que el objetivo de este ejercicio es crear un archivo HTML con una estructura de divs anidados y aplicar estilos para lograr los efectos visuales deseados.*

¡Muy bien! Avancemos al próximo ejercicio.


Ejercicio 2

En este ejercicio, te desafiamos a crear un archivo HTML y utilizar *contenedores flexbox* para lograr replicar las figuras de las imágenes dadas:



1. Crear un nuevo archivo HTML (como por ejemplo: “ejercicio-2.html”) con la estructura básica de un documento HTML agregando las etiquetas <html>, <head> y <body>.

2. Dentro del `<body>`, agregar tres etiquetas `<div>` que servirán como contenedores flexbox.
3. Crear un nuevo archivo CSS (por ejemplo, "estilos.css") y enlazarlo con el archivo HTML utilizando la etiqueta `<link>`.
4. Aplicar estilos CSS a los contenedores utilizando la propiedad `border` con el valor "darkkhaki" para darles un borde.
5. Para cada uno de los contenedores, establecer un `background-color` (color de fondo) utilizando los siguientes valores:
 - Caja 1: coral
 - Caja 2: cornflowerblue
 - Caja 3: darkseagreen
6. Dentro de cada uno de los contenedores, en tu archivo HTML, agregar otras tres etiquetas `<div>` para representar los elementos internos.
7. Guardar el archivo HTML y CSS.
8. Abrir el archivo en tu navegador web para ver los contenedores flexbox con los colores y la estructura requerida.
9. Compartir con tus compañeros cómo se visualiza el archivo que acabas de crear.

 **Recuerda que el objetivo de este ejercicio es que utilices contenedores flexbox para organizar los elementos de tu archivo HTML e imitar las formas de los ejemplos brindados.**

¿Todo bien hasta aquí? Continuemos.


Ejercicio 3

En este ejercicio, te desafiamos a crear una página que cambie el color de fondo según el ancho de la ventana del navegador. Utilizaremos "media queries" para cambiar el color cuando la ventana tenga menos de 600px de ancho y también cuando tenga menos de 400px de ancho.

Sigue los siguientes pasos para completar el ejercicio:

1. Crear un nuevo archivo HTML (como por ejemplo, "ejercicio-4.html").

2. Crear una estructura básica de HTML agregando las etiquetas `<html>`, `<head>` y `<body>`.
3. Crear un nuevo archivo CSS (por ejemplo, "estilos.css") y enlazarlo con el archivo HTML utilizando la etiqueta `<link>`.
4. Agregar estilos básicos al cuerpo de la página, como un `background-color` (color de fondo) predeterminado.
5. Dentro de tu archivo CSS, crear dos "media queries":
 - Una *media query* para agregar un estilo para cambiar el color de fondo cuando la ventana tenga un ancho menor o igual a 600px.
 - Otra *media query* para agregar otro estilo para cambiar el color de fondo cuando la ventana tenga un ancho menor o igual a 400px.
6. Guardar los cambios de tus archivos HTML y CSS.
7. Abrir el archivo HTML en tu navegador web y probar el cambio de color de fondo al redimensionar la ventana del navegador.
8. Compartir con tus compañeros cómo se visualiza el archivo que acabas de crear.

 *No olvides que el objetivo de este ejercicio es que puedas utilizar "media queries" para cambiar el color de fondo según el ancho de la ventana del navegador.*


¡Ahora pasemos al próximo ejercicio!

Ejercicio 4

En este ejercicio, te desafiamos a *escribir una función que reciba por parámetro un string y devuelva el mismo string con las palabras invertidas*. Por ejemplo, si recibimos el string "Hola mundo", la función deberá devolver "aloh odnum".

Sigue los siguientes pasos para completar el ejercicio:

1. Crear un nuevo archivo JavaScript, como por ejemplo: "ejercicio-4.js".
2. Configurar la estructura básica de JavaScript agregando la declaración de una función con el nombre que desees.
3. Dentro de la función, definir un parámetro para recibir el *string* que se desea invertir; y crear una variable para almacenar el resultado final.

4. Utilizar un método o técnica para separar el string en un *array* de palabras individuales. Puedes investigar métodos como el *split()* para lograr este paso.
 5. Iterar sobre el *array* (arreglo) de palabras de forma inversa por medio de un bucle (puedes ayudarte con otro arreglo para almacenar las letras en orden inverso).
 6. Utilizar un método o técnica para unir las palabras invertidas en un solo *string* nuevamente (pista: empieza con "j" y termina con "oin").
 7. Utilizar *console.log()* para imprimir el *string* resultante por la consola.
 8. Guardar tu archivo JavaScript.
 9. Ejecutar el script y pasar diferentes strings como argumento a la función para comprobar si las palabras están siendo invertidas correctamente.
-  *Ten presente que el objetivo de este ejercicio es escribir una función que invierta las palabras de un string.*


Ejercicio 5

En este ejercicio, te desafiamos a escribir una función que tome dos números como parámetros y devuelva un objeto con las operaciones matemáticas realizadas entre ellos (suma, resta, multiplicación y división).

A continuación, te presentamos los pasos que debes seguir para completar el ejercicio:

1. Crear un nuevo archivo JavaScript, como por ejemplo: *"ejercicio-5.js"*.
2. Configurar la estructura básica de JavaScript agregando la declaración de una función con el nombre que desees.
3. Dentro de la *función*, definir dos parámetros para recibir los números que se utilizarán en las operaciones.
4. Crear un *objeto* vacío para almacenar los resultados de las operaciones.
5. Utilizar *operadores aritméticos* para realizar las operaciones matemáticas requeridas (suma, resta, multiplicación y división) entre los dos números recibidos como parámetros.

6. Asignar cada resultado al correspondiente atributo del objeto creado en el paso anterior.
7. Utilizar `console.log()` para imprimir el objeto resultado por la consola.
8. Guardar tu archivo JavaScript.
9. Ejecutar el script y pasar diferentes pares de números como argumentos a la función para comprobar si las operaciones están siendo realizadas correctamente y si el objeto resultado es impreso adecuadamente por la consola.

 **Recuerda que el objetivo de este ejercicio es escribir una función que realice diferentes operaciones matemáticas entre dos números y que devuelva un objeto con los resultados.**

Ejercicio 6

En este ejercicio, te desafiamos a crear un arreglo de objetos y una función que transforme esos objetos en mensajes de presentación.


Sigue los siguientes pasos para completar el ejercicio:

1. Crear un nuevo archivo JavaScript, como por ejemplo: `"ejercicio-6.js"`.
2. Definir un *array* (arreglo) llamado "personas" que contenga al menos 3 objetos. Cada objeto debe tener las propiedades "nombre", "edad" y "ciudad". Puedes agregar valores ficticios a estas propiedades por ahora.
3. Declarar una *función* llamada `"crearMensajesDePresentacion"` que reciba por parámetro el arreglo de objetos "personas".
4. Dentro de la función, crear un nuevo *array* (arreglo) vacío llamado `"mensajes"`.
5. Utilizar un *bucle*, como "for" o "forEach", para recorrer el arreglo "personas".
6. En cada iteración, acceder a las propiedades "nombre", "edad" y "ciudad" de cada objeto y crear un mensaje de presentación utilizando el siguiente formato: `"Mi nombre es [nombre], tengo [edad] años y vivo en [ciudad]."`
7. Agregar cada mensaje al arreglo "mensajes" utilizando el *método* `"push"`.
8. Al final de la función, utilizar la palabra clave "return" seguida del arreglo "mensajes" para devolverlo como resultado.
9. Fuera de la función, declara una constante que y luego asignar el resultado de la invocación de `"crearMensajesDePresentacion()"`, luego utilizar un *bucle* o

método de iteración para recorrer el arreglo de mensajes y mostrar cada mensaje por consola.

10. Guardar el archivo y ejecutarlo en un entorno que permita la ejecución de JavaScript, como el navegador web o una plataforma de desarrollo.

11. Abrir la consola de desarrollador de tu entorno para ver los mensajes de presentación impresos.

 *No olvides que el objetivo de este ejercicio es crear un arreglo de objetos, transformar esos objetos en mensajes de presentación utilizando una función y mostrar los mensajes resultantes por consola.*

Este mismo ejercicio **puede resolverse en una sola línea de código usando dos métodos propios de los arrays**. ¿Te animas a descubrir en grupo cómo hacerlo?.

Ejercicio 7

En este ejercicio, te desafiamos a crear una lista numerada con elementos `` que contengan botones "subir" y "bajar" y un `` con el nombre de un país.

Sigue los siguientes pasos para completar el ejercicio:

- 1.** Crear un nuevo archivo HTML y nómbralo como desees, por ejemplo, "ejercicio.html".
- 2.** Definir la estructura básica de un archivo HTML, incluyendo las etiquetas `<html>`, `<head>` y `<body>`.
- 3.** Dentro del cuerpo (`<body>`) del archivo HTML, crear una etiqueta `` para representar la lista numerada.
- 4.** Dentro de la etiqueta ``, crear cinco elementos ``, uno por cada país que desees incluir en la lista.
- 5.** Para cada elemento ``, agregar tres elementos hijos.
 - El primer hijo debe ser un botón (`<button>`) con el texto "subir".
 - El segundo hijo debe ser un span (``) que contenga el nombre de un país.
 - El tercer hijo debe ser otro botón (`<button>`) con el texto "bajar".
- 6.** Utilizar JavaScript para seleccionar todos los botones "subir" y "bajar" del documento.

7. Recorrer los botones utilizando un bucle o método de iteración, como "forEach".
8. Para cada botón "bajar", asignar una función al evento "click" que permita bajar una posición al elemento contenedor.
9. Para cada botón "subir", asignar una función al evento "click" que permita subir una posición al elemento contenedor.
10. Guardar el archivo y abrirlo en un navegador web.
11. Verificar que la lista numerada y los botones "subir" y "bajar" se muestren correctamente. Haz clic en los botones "subir" y "bajar" para comprobar si se produce el desplazamiento de los elementos dentro de la lista numerada.

💡 *Ten presente que el objetivo de este ejercicio es crear una lista numerada con elementos que contengan botones "subir" y "bajar", y asignar funciones a los botones para permitir el desplazamiento de los elementos.*

Ejercicio 8

En este último ejercicio, te desafiamos a crear un formulario para obtener información de los empleados de una empresa.


Sigue los siguientes pasos para completar el ejercicio:

💡 *Si te encuentras con algún elemento que no conoces puedes consultar el siguiente enlace: [🔗 Documentación de MDN.](#)*

1. Crear un formulario con *dos campos*: correo electrónico, edad y un botón de "submit". El formulario tiene que estar centrado en la página y sus inputs ordenados y separados verticalmente (pista: puedes usar div internos).
2. Agregar dos "labels" con el atributo "for" correspondiente para asociarlos a los dos inputs.
3. Validar el correo con una expresión regular que permita las siguiente entradas: *nacho-ff@mail.com.ar*, *ñandu@mail.museum.com*, *elias95@mail.com*. (Utiliza el evento "input" sobre los inputs para validarlos).
4. Validar que la edad sea un número mayor a 0.
5. Si algún dato no es válido debe aparecer un mensaje con letra roja pequeña debajo de cada input. (No debe moverse el resto de los elementos HTML cuando

aparezca el mensaje, tiene que reservarse un espacio para que pueda aparecer el mensaje sin causar que las distancias entre los inputs varien cuando aparezca el mensaje de error).

¡Fin del último ejercicio!

 Si aún dispones de tiempo y quieres continuar poniendo a prueba tus habilidades, puedes continuar con los siguientes puntos extras del ejercicio 8 para mejorar y profundizar tus conocimientos.

6. Agregar un campo “select” que permita elegir entre tres tipos de opciones de empleado: “Vendedor”, “Encargado de turno” y “Supervisor general”.

- Si se elige Vendedor: tendrán que aparecer los siguientes dos inputs: cantidad de ventas (debe ser mayor a 0), monto total vendido (debe ser mayor a cero y permitir hasta dos números decimales).
- Si se elige Encargado de turno: debe aparecer un input de cantidad de empleados a cargo (el número no puede ser menor a 5).
- Si se elige Supervisor general: debe aparecer un input de cantidad de locales administrados (el número no puede ser menor a 2) y otro input de text-area donde pida una descripción del desempeño general (debe completarse con al menos 5 palabras).
- Agregar los labels correspondientes.
- Comprobar que cuando cambia el select deben desaparecer y aparecer los campos de acuerdo al tipo de empleado elegido.

7. Agregar un elemento “fieldset” para agrupar dos inputs radio junto con sus labels correspondientes para que los empleados elijan entre dos turnos “mañana” o “tarde”.

8. Agregar un “datalist” para que los empleados completen cual es su color favorito: amarillo, azul, rojo, verde, naranja, violeta, blanco o negro.

9. Mostrar los mensajes de error para cada input si no cumplen las condiciones de validez.

10. Verificar que el botón de “submit” quede deshabilitado mientras que existan inputs no válidos.

- Prueba completar todos los campos, que el botón se habilite y luego volver a invalidar algún campo para comprobar que funcione correctamente.
- Prueba cambiar de tipo de empleado y volver a comprobar que el botón submit se habilite y deshabilite correctamente.

11. Usar las pseudo-clases :valid e :invalid para modificar la apariencia de los inputs.

12. Crear una tabla donde las columnas sean los campos del formulario y que cada vez que se presiona el botón de submit se agregue una fila con la información correspondiente (recuerda usar el método `preventDefault()`).
13. Si se vuelve a ingresar información con el mismo correo en lugar de crear una fila nueva, se debe modificar la fila que corresponde a dicho correo con la información nueva.
14. Que los elementos `"td"` que tienen el correo automáticamente tengan la propiedad `contenteditable="true"` para poder modificar el correo, luego comprobar que todos los puntos anteriores sigan cumpliéndose.

¡Y listo!

Fin de la clase

¡Bien hecho, desarrolladores web! 🎉 Hemos llegado al final de nuestra clase práctica de hoy, en la que nos centramos en aplicar conceptos utilizando HTML, CSS y JavaScript.

A lo largo de la clase, enfrentamos problemas prácticos y aprendimos cómo aplicar los conceptos de desarrollo web para resolverlos. Pudimos ver en tiempo real cómo estos tres lenguajes trabajan juntos para resolver indicaciones e interacciones.

Espero que hayas disfrutado y aprendido mucho en esta clase. Recuerda que la práctica y la experiencia son fundamentales para mejorar tus habilidades en desarrollo web. No dudes en volver a revisar el contenido si necesitas repasar algún concepto o técnica como también experimentar en tu proyecto de sitio web.

En la próxima clase, veremos cómo agregar distintos formatos multimedia a tu sitio y también vincularemos distintas páginas al menú que has creado anteriormente.

Gracias por tu esfuerzo y dedicación. ¡Hasta la próxima clase!

¿Nos cuentas cómo te fue hoy?

 [Responder encuesta](#)

En la próxima clase, veremos cómo agregar distintos formatos multimedia a tu sitio y también vincularemos distintas páginas al menú que has creado anteriormente.

Gracias por tu esfuerzo y dedicación. ¡Hasta la próxima clase!