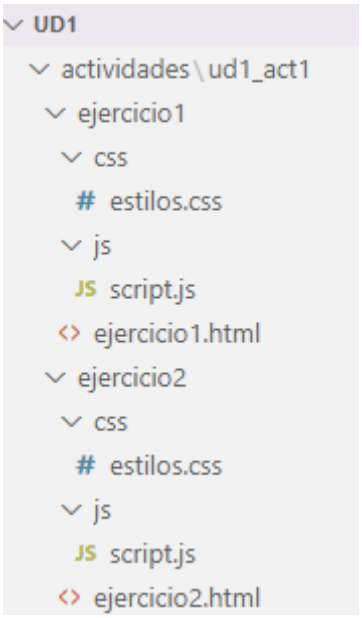
UD1 ACT1. Introducción a JS

Crea (si no la tienes ya) una carpeta que llamarás **`ud1`**, dentro habrá una llamada **`actividades`** dentro habrá otra llamada **`ud1_act1`**. Dentro de la misma, cada ejercicio irá ubicado en una carpeta que llamarás **`ejercicio1`**, **`ejercicio2`**, etc… respectivamente. En los nombres de las carpetas de los ejercicios no uses espacios ni mayúsculas. Dentro de cada ejercicio:

- Los archivos Javascript irán ubicados en la carpeta **`js`**
- Los estilos, de habrerlos, irán ubicados en la carpeta **`css`**.
- Las imágenes, de haberlas, irán ubicadas en la carpeta **`img`**.



El código debería entregarse convenientemente formateado y comentado si fuera necesario. Todas las páginas deberían tener el nombre del ejercicio y la actividad en el título.

La entrega de la actividad se realizará vía GitHub, aunque deberá constar como entregada en el aula virtual (Moodle) previamente a la fecha límite de entrega.

1. Crea un script en el que crees 4 variables (2 cadenas y 2 números), con los siguientes valores respectivamente:
    - Tu nombre
    - Tus apellidos
    - Tu año de nacimiento
    - El dinero que tienes en el banco

Muestra en un mensaje por consola:

- Los tipos de datos de las cuatro variables
- Tu nombre y apellidos separados por un salto de línea y a continuación el dinero que tendrás dentro de dos años (calculado a partir del dinero que tienes ahora mismo más 10.000)

A continuación cambia el programa para que los datos sean introducidos en tiempo real por el usuario. ¿Los tipos de datos son iguales? Realiza los cambios necesarios para que los mensajes se muestren correctamente.

2. Vamos a simular un juego de "*Piedra, papel o tijera*". Crea dos variables **`jugador1`** y **`jugador2`** que representen las jugadas del jugador 1 y jugador 2 (cadena que puede ser "*Piedra*", "*Papel*" o "*Tijera*"). Muestra el resultado teniendo en cuenta que:

    - Piedra gana sobre Tijera, Tijera gana sobre Papel, Papel gana sobre Piedra
    - El programa tendrá 3 posibles salidas (mensaje por consola):
        - El ganador es jugador1
        - El ganador es jugador2
        - Es un empate

3. Realiza una función **`sumaDigitos`** que reciba un número (entero) y devuelva la suma de todos sus dígitos sin usar cadenas. La función debería devolver la suma de los dígitos del número. Por ejemplo:
    - `sumaDigitos(318) => 12`
    - `sumaDigitos (-314569) => 28`

4. Refactoriza el código del ejercicio 2 para crear una función que reciba como parámetros las jugadas de los dos jugadores y devuelva el resultado del combate

Función `piedraPapelTijera()`	
Recibe	jugada1: Jugada del jugador 1 (cadena) jugada2: Jugada del jugador 2 (cadena)
Devuelve	Cadena con el resultado de la partida

5. Refactoriza el código del ejercicio 4 para que la función reciba un objeto literal con las jugadas de ambos jugadores.

Función **`piedraPapelTijera ()`**	
Recibe	Objeto con las jugadas de ambos jugadores
Devuelve	Cadena con el resultado de la partida

[AMPLIACIÓN] Reescribe la lógica de la aplicación para que esté en un objeto como el siguiente, donde se indica la jugada ganadora en cada caso

```
let jugadas = {
    Piedra: "Tijera",
    Tijera: "Papel",
    Papel: "Piedra"
}
```

6. Crea una función **`valorABilletes`** que reciba un número real que representa una cantidad en euros y que devuelva un array de 7 enteros representando el número mínimo de billetes de cada tipo necesario para pagar dicha cantidad, ordenadas por valor decreciente.
    - Posición 1: Número de billetes de 500 euros
    - Posición 2: Número de billetes de 200 euros
    - Posición 3: Número de billetes de 100 euros
    - Posición 4: Número de billetes de 50 euros
    - Posición 5: Número de billetes de 20 euros
    - Posición 6: Número de billetes de 10 euros
    - Posición 6: Número de billetes de 5 euros

Ejemplo: **`valorABilletes (787)`** devuelve

1	1	0	1	1	1	1

787 euros son 1 x 500 euros, 1 x 200 euros, 1 x 50 euros, 1 x 20 euros, 1 x 10 euros, 1 x 5 euros.

Prueba la función en un script que pida al usuario una cantidad y devuelva el número de billetes en formato amigable. El script se repetirá hasta que el usuario introduzca la palabra “FIN” en lugar de un número real.

PISTAS
- El operador % te ayudará a saber cuánto te queda por cambiar
- Implementa y prueba en primer lugar la función **`valorABilletes.`** Una vez implementada y probada, implementa el cuerpo principal del script.

7. ¡Es una batalla de Pokémon! Tu tarea es calcular el daño que causaría un movimiento en particular usando la siguiente fórmula:

```
daño = 50 * (ataque / defensa) * efectividad
```

donde:

- ataque = tu poder de ataque
- defensa = la defensa del oponente
- efectividad = la efectividad del ataque según el enfrentamiento (ver explicación a continuación)

En cuanto a la efectividad, los ataques pueden ser súper efectivos, neutrales o poco efectivos según el enfrentamiento. Por ejemplo, el agua sería súper eficaz contra el fuego, pero no muy eficaz contra la hierba.

- Súper efectivo: 2x daño
- Neutral: 1x daño
- No muy efectivo: 0,5x daño

Para evitar que este desafío sea tedioso, solo habrá cuatro tipos: fuego, agua, hierba y electricidad. Aquí está la efectividad de cada enfrentamiento:

- fuego > hierba
- fuego < agua
- fuego = electricidad
- agua < hierba
- agua < electricidad
- hierba = electricidad

La función que debes implementar y que se llamará **`calculaImpacto`** recibirá:

- Tu tipo
- El tipo del oponente
- Tu poder de ataque
- La defensa del oponente

PISTA

Intenta codificar la efectividad de cada enfrentamiento en forma de objeto, del modo siguiente:

```
const efectividad = {
    fuego: {
      hierba: 2,
      agua: 0.5,
      electricidad: 1,
    }…
```

RESULTADOS DE APRENDIZAJE Y CRITERIOS DE EVALUACIÓN

Esta actividad contribuye a la consecución del RA1. *Genera interfaces gráficas de usuario mediante editores visuales utilizando las funcionalidades del editor y adaptando el código generado.*

Los criterios de evaluación que permiten valorar la consecución del resultado de aprendizaje son los siguientes:

1a) Se han analizado las herramientas y librerías disponibles para la generación de interfaces gráficos.
1b) Se ha creado un interfaz gráfico utilizando las herramientas de un editor visual.
1f) Se ha modificado el código generado por el editor visual.

En el Moodle de la asignatura se detalla la rúbrica de corrección para la actividad.