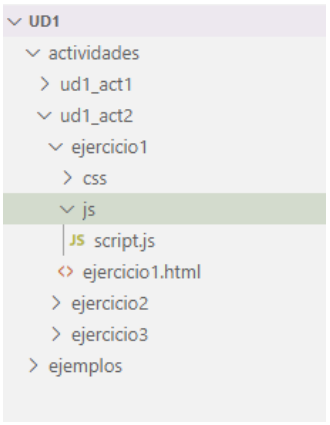
UD1 ACT2. Funciones y objetos

Crea (si no la tienes ya) una carpeta que llamarás **`ud1`**, dentro habrá una llamada **`actividades`** dentro habrá otra llamada **`ud1_act2`**. 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 Moodle de la clase previamente a la fecha límite de entrega.

1. Crea un script que pida al usuario su nombre y apellidos y muestre:
    - El tamaño del nombre más los apellidos (sin contar espacios).
    - La cadena en minúsculas y en mayúsculas.
    - Que divida el nombre y los apellidos y los muestre en 3 líneas, donde ponga:
        - Nombre: *Nombre introducido*
        - Apellido 1: *Apellido 1*
        - Apellido 2: *Apellido 2*
    - Una propuesta de nombre de usuario, compuesto por el nombre, la inicial del primer apellido y la inicial del segundo: *ej. Para Javier Gonzalez Pisano sería javiergp.* Implementa esta funcionalidad mediante una función anónima.
    - Una propuesta de nombre de usuario compuesto por las dos primeras letras del nombre y de los dos apellidos: *ej. jagopi.* Implementa esta funcionalidad mediante una función flecha.

[AMPLIACIÓN] ¿Cómo se podría hacer para generar una contraseña que fuese el nombre pero separando cada letra con el número que indica su posición?

Ejemplo: `console.log(creaPassword("Javier"));` // imprime "`J0a1v2i3e4r5`"

2. Implementa la siguiente función:

Función `infoCumple()`	
Recibe	dia: Día de tu cumpleaños (numérico) mes: Mes de tu cumpleaños (numérico)
Devuelve	Objeto con dos propiedades: `dias`: Días que faltan para tu cumpleaños `diaSemana`: Día de la semana de tu cumpleaños (cadena)

Pruébala con fechas límite (ayer, hoy, mañana…). También puedes utilizar el método `toLocaleString()` de `Date` para obtener el día de la semana

[AMPLIACIÓN] Que reciba la fecha de nacimiento en lugar del día y mes del cumpleaños

3. Implementa la siguiente función:

Función `numerosCorrectos()`	
Recibe	Un array de números que se espera que estén ordenados. Ejemplo: [1,2,3] ó [1,3,4,5,6,7,8,9]
Devuelve	Un número que es el índice del primer elemento fuera de orden dentro de dicho array. Por ejemplo, si recibe el array [1,2,3,7,4] debería devolver el número 3 (es la posición donde hay un elemento fuera de orden Si todos los elementos están en orden se devuelve un -1

Prueba la función con varios casos de prueba diversos. Recuerda probar posiciones límites

4. Implementa la siguiente función:

Función `tiposEnArray()`	
Recibe	Un array de elementos de distintos tipos Ejemplo: `tiposEnArray[true,1,"Casa",function(){})`
Devuelve	Otro array con el mismo número de elementos del array que recibe que contiene los tipos de cada elemento del array original En el caso del ejemplo: `[boolean, number, string, function]`

5. Define un array **`paises`** que contenga un listado de países
   Implementa funciones que permitan:
   - Mostrar todos los elementos del array separados por un salto de línea usando `for..of` (consola)
   - Mostrar los elementos del array en sentido inverso separados por un salto de línea usando **`foreach`**
   - Mostrar los elementos del array alfabéticamente separados por un salto de línea usando una sola sentencia
   - Añadir un elemento al comienzo del array
   - Añadir un elemento al final del array
   - Borrar un elemento al comienzo del array (indicar cuál es)
   - Borrar un elemento al final del array (indicar cuál es)

6. Queremos almacenar usando una tabla los resultados obtenidos en las elecciones en Villaconejos, teniendo en cuenta:
   - Ha habido 5 sedes para votar (Ayuntamiento, Polideportivo, Instituto, Mercado y Colegio)
   - Se han presentado 4 partidos (Puede que Villaconejos (PV), Obreros de Villaconejos (OV), Villaconejos Por el Si (VpSI), Unión Progreso y Villaconejos (UPV).
   - Para simular los votos, se pueden generar aleatoriamente los votos correspondientes a cada partido (entre 5 y 10 votos).

   A continuación se mostrará por consola una tabla con todos los colegios electorales y partidos, así como sus votos asociados.

[AMPLIACIÓN] Calcular el número total de votos por partido y por sede

7. Desarrolla un script con una función **`crearPersona()`** que pueda recibir los siguientes datos:
    a. Nombre y apellidos
    b. Nombre, apellidos y edad
        i. De no recibir la edad esta tendrá por defecto el valor 0.
    c. Nombre, apellidos, edad y un número indeterminado de formas de contacto (números de teléfono, email, etc.)

Posteriormente deberá mostrar por consola un texto en el que se indique toda la información del usuario. Usa forEach para recorrer las formas de contacto. Prueba a usar una función anónima y una función flecha dentro de forEach

8. Desarrolla un script con la clase **`Persona`** con los siguientes elementos:

Clase **`Persona`**	
Atributos (con sus setter y getter asociados)	`Nombre` (cadena)
	`Apellidos` (cadena)
Métodos	[Constructor] `Persona(nombre, apellidos)` Si no le paso los parámetros por defecto "Sin nombre" y "Sin apellidos"
	`comer([cadena] platos)` Recibe un array de cadenas que representa los platos que va a comer. Devuelve un mensaje concatenando el nombre, los apellidos y los platos que está comiendo. Si no recibe parámetros, estará comiendo bocata de chorizo

Crea un par de objetos de tipo `Persona` y prueba sus métodos con diversos casos de prueba.

9. Crea una clase `Progenitor` y una clase `Sucesor` que hereden de `Persona` (usa un archivo separado para cada clase):

Clase **`Sucesor`** (hereda de **`Persona`**)	
Atributos	`tieneMoto` (boleano)
Métodos	[Constructor] `Sucesor(nombre, apellidos,tieneMoto)` Si no le paso la moto por defecto no tiene.
	`desplazarse(puntoA,puntoB)` Devuelve una cadena que indica como va del puntoA al punto B (en moto o andando)

Clase **Progenitor** (hereda de **Persona**)	
Atributos	`tieneCoche` (boleano)
Métodos	[Constructor] `Progenitor(nombre, apellidos,tieneMoto)` Si no le paso el parámetro por defecto no tiene.
	`desplazarse(puntoA,puntoB)` Devuelve una cadena que indica como va del puntoA al punto B (en coche o andando)
	`comer` [redefinido] Como buen progenitor comerá huevos, su mensaje será "Estoy comiendo huevos"

Crea un progenitor y un sucesor y prueba a mostrar su coche/moto y sus métodos comer y cenar

10. Crea una clase **`Familia`**:

Clase `Familia`	
Atributos (públicos)	`domicilio (cadena)`
	`Renta`
	`[Persona] miembros`
Métodos	`llamadaAComer()` Recorre todos los miembros invocando a `comer( )`
	`vacaciones(puntoA, puntoB)` Recorre todos los miembros invocando a `desplazarse(puntoA,puntoB)`.

Crea un objeto de la clase Familia

- Añade las personas creados en el punto anterior a la misma.
- Invoca a los métodos `llamadaAComer()` y `vacaciones()`

En el HTML enlaza sólo al script con el código del programa principal

En cada JS importa las clases que necesites con la sentencia `import`

Exporta las clases para que puedan ser usadas con la sentencia `export default`

11. Crea cuatro objetos para guardar información de libros (almacena cada uno en una constante) con los siguientes campos:
    - **`Nombre`**: cadena
    - **`Color`**: cadena
    - **`Autor`**: cadena
    - **`nPaginas`**: entero
    - **`Editorial`**: cadena
    - **`Forrado`**: booleano
    - **`coverURL`**: cadena
    - **`comprar()`**: Función. Mostrará por consola el mensaje: "Libro de aventuras del autor XXX comprado"

- Usa tres de las constantes para crear un **`array`** que represente una biblioteca
- Crea las siguientes funciones:
    - **`checkPages`**: Función flecha que devuelve true si el número de páginas que se le pasa es mayor que 150.
        - Usa en este caso la desestructuración de objetos, de forma que le pasamos el libro entero pero la función recibe las páginas
    - **`checkLibro`**: Función flecha a la que se le pasa la biblioteca y un libro y devuelve true si la biblioteca contiene ese título.
        - Pruébalo con un libro de la biblioteca y con el que has dejado fuera.

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.