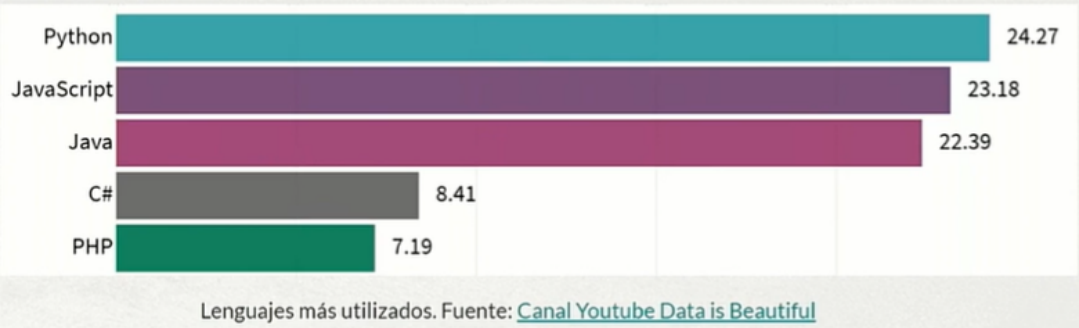
JAVASCRIT- JS

1. ¿Qué es?

JavaScript o **JS** es el lenguaje de programación que se usa para añadir características interactivas a un sitio web, es decir hacer una página **web dinámica**..



JavaScript es regulado desde 1997 por el estándar **ECMAScript** que se encarga de regir como debe ser interpretado y cómo funciona el lenguaje JavaScript,

2. Frontend y Backend

Cuando accedemos a un servidor web para pedirle una página web nos la descargamos a nuestro navegador pero en el servidor también se queda parte del funcionamiento y tratamiento de la página, por eso decimos que podemos hablar de la página web del lado del cliente que es la página descargada en nuestro navegador y con el que el cliente interactúa y ciertas acciones del cliente o usuario puede que obliguen a realizar ciertas acciones en el servidor (base de datos, etc.).

Según lo anterior distinguimos dos partes:

- **Frontend** es la parte de un sitio web que interactúa con los usuarios, es decir que está del lado del cliente en el navegador. Los lenguajes usados para interactuar con el cliente son HTML, CSS, ***Javascript***, JQuery, etc.
- **Backend**: es la parte que se conecta con el servidor y con la base de datos, etc. Esta parte no interactúa con el cliente. Los lenguajes más usados son **PHP**, Python, Javascript, Java, Ruby, .Net, etc.

3. Que podemos hacer con Javascript

Algunas de las cosas que podemos hacer con Javascript son las siguientes:

- Crear alertas.
- Añadir un juego,
- Eventos que ocurren cuando los botones son presionados
- Eventos que ocurren cuando los datos son introducidos en los formularios
- Cambiar el html y css consiguiendo efectos de estilo dinámicos.
- Crear animaciones, carruseles de imágenes o menús desplegables.
- Abrir una ventana pop-up cada cierto tiempo.
- Visualizar un reloj digital en la página con cuenta regresiva.
- Deshabilitar el botón derecho del ratón.
- Etc.

Probar JS: hay una página muy interesante donde podemos seguir un manual de JS y probar ejemplos y código que es https://www.w3schools.com/js/default.asp)

4. Conceptos básicos de programación con JavaScript

Para empezar a trabajar con JavaScript necesitamos usar un editor de código. Hay muchos editores de código como Visual Studio Code, Atom, Sublime, Notepad ++, etc. Nosotros vamos a usar el Visual Studio Code que es uno de los más usados. Nos lo descargamos y lo instalamos.

Para empezar a probar cosas con JavaScript nos creamos una carpeta llamada Pruebas JavaScript y dentro creando carpetas para los ejercicios que vamos a hacer. Creamos un primer documento llamado **ejemplo1.html**

Agregar Javascript a la página HTML

Al igual que CSS, para agregar JavaScript a una página web lo podemos de varias maneras:

1. En el Head: entre las etiquetas **<script>** …… **</script>**

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
        alert("Probamos javascript")
    </script>
</head>
<body>

</body>
</html>
```

La sentencia **alert("")** nos muestra un aviso o alerta por pantalla
Ejecutamos la página y nos muestra la alerta de JavaScript.

2. Dentro del **body** de la página, o incluso dentro de cualquier etiqueta **html**. Podemos cortar y pegar el código anterior dentro del **body** y lo vemos. Dentro del body se aconseja poner el javascript al final.
3. En un archivo independiente. Esta opción es la más habitual sobre todo cuando tenemos bastante código JavaScript que es lo normal en casi todas las páginas dinámicas. En este caso creamos un archivo nuevo con el nombre que queramos y extensión **js** y desde la página **html** lo enlazamos con el siguiente código. La línea de código se puede poner en el head o también se suele poner al final antes del cierre de la etiqueta **body**.

En la página html:

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

<script src="app.js"></script>
</body>
</html>
```

En el archivo app.js:

```
alert("Esto es una alerta para probar")
```

Nota: no dejar espacios ni caracteres especiales en los nombres de los archivos. Si nuestro nombre es compuesto es bueno utilizar la primera palabra en minúscula y la primera letra de la segunda palabra en mayúscula. Por ejemplo. miCodigo.js, sumarNumeros.js, carritoCompra.js, etc. (CAMELCASE)

Comentarios en JavaScript y palabras reservadas

Para poner comentarios ponemos dos barras //

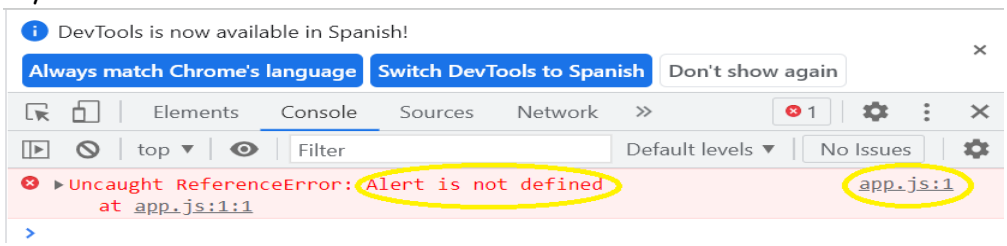
```
// Esto es un comentario para explicar algo
alert("Esto es una alerta para probar")
```

Como en todos los lenguajes, hay palabras reservadas ya que las usa el propio lenguaje como por ejemplo la palabra "**alert**" ya que es una función de JavaScript, pero si podemos usar "alerta", gracias a que nosotros utilizamos castellano probablemente no utilicemos palabras reservadas. También hay que tener en cuenta que JavaScript distingue entre mayúsculas y minúsculas, por lo que la palabra "Alert" no funcionará ya que es "**alert**".

La consola y la depuración de errores

Para empezar a aprender javascript y también para ver cualquier error de sintaxis, tenemos que acceder a la consola del navegador. Para ello tenemos que picar clic derecho/Inspeccionar y en el menú superior picamos en Console y cada vez que carguemos o actualicemos la página podemos ver lo que se ejecuta internamente y

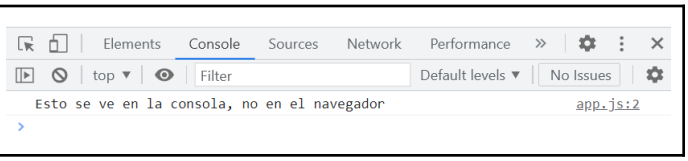
también los errores. En este caso vemos un error que nos dice que Alert no está definido en JavaScript y nos indica el archivo y la línea donde se encuentra el error:



Según lo visto siempre debemos de ir probando nuestro código en la consola del navegador para ver si funciona o ver los errores.

Para probar parte de nuestro programa podemos usar la sentencia **console.log()**, por ejemplo para ver el contenido de una variable, de un array, etc. Lo vemos a continuación

```
alert("Esto es una alerta para probar")
console.log("Esto se ve en la consola, no en la página")
```



5. document.write ("")

Además de la utilidad de **console.log()** y de la de **alert()**, podemos hacer que la salida de información se escriba en la propia página web. Para ello usamos la utilidad **document.write("")** y dentro de comillas escribimos el texto con las etiquetas html que queramos.

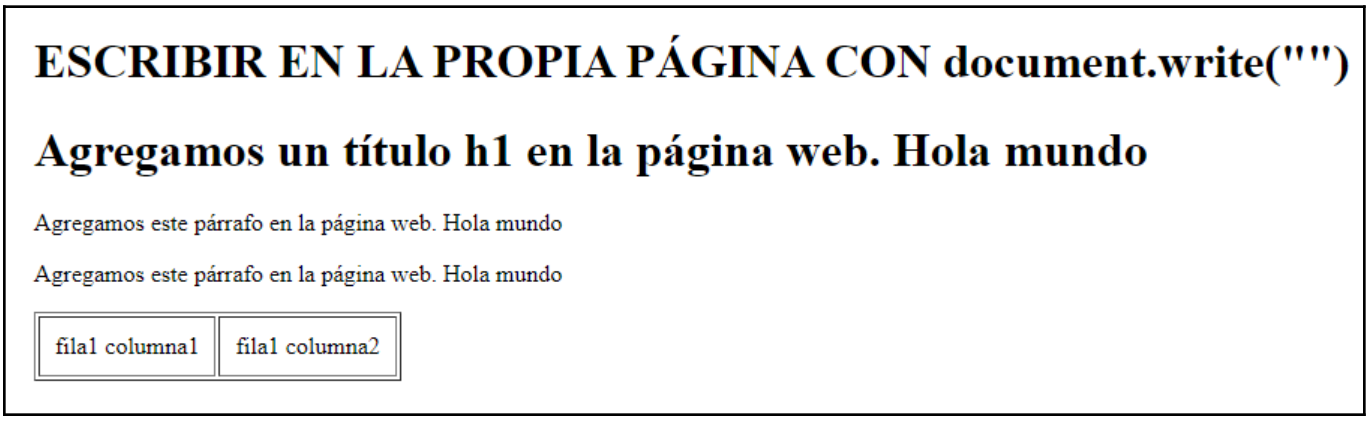
Por ejemplo:

En la página html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
<h1>ESCRIBIR EN LA PROPIA PÁGINA CON   document.write("")</h1>
<script src="ejemplo.js"></script>
</body>
</html>
```

En la página javascript

```
//Escribimos el documento html con document.write
document.write("<h1>Agregamos un título h1 en la página web. Hola mundo</h1>");
document.write("<p>Agregamos este párrafo en la página web. Hola mundo</p>");
document.write("<p>Agregamos este párrafo en la página web. Hola mundo</p>");
document.write("<table border=1 cellpadding=10><tr><td>fila1 columna1</td><td>fila1 columna2</td></tr></table>")
```



6. Tipos de datos

Vamos a trabajar con tres tipos de datos:

- **String**: son datos tipo texto. Se ponen entre comillas.
- **Numéricos**: son números con los que se puede hacer operaciones matemáticas con ellos. Si es decimal ponemos un punto (no una coma). No hace falta diferenciar entre entero o decimal. No se pone comillas.
- **Booleanos**: sólo admite dos valores: verdadero o falso. (true o false). No se pone comillas
- **Undefined**: que la variable no está definida. Además, hay que definirla antes de usarla.
- **Null**: la variable sí está definida pero no tiene valor.

```
1  alert("Estos son datos tipo texto que se muestran como alerta en el navegador")
2  console.log("estos también son datos tipo texto que se ven solo en consola")
3  console.log(15) //esto es un dato numérico
4  console.log(true) //esto es un dato boleano con valor verdadero
5  console.log(variable) //esto nos da variable no definina
```

```
estos también son datos tipo texto que se ven solo en consola
15
true
Uncaught ReferenceError: variable is not defined
    at app.js:5:13
```

7. Variables

Es un espacio de memoria donde guardamos los datos que usemos (string, números, booleanos, etc.). Se llama variable porque su valor puede cambiar. Por ejemplo, la variable "sueldo" puede tener valores 1000, 1200, 1500, etc. O la variable "nombre" puede tener el valor Unai, Javier, Nagore, etc.

Para declarar una variable podemos hacerlo de tres maneras:

var: para usar variables las cuales se puede cambiar su valor. Pueden ser redefinidas. Se aconseja no usarla.

let: también para variables que se puede cambiar su valor. La ventaja es que se pueden declarar variables con el mismo nombre siempre que estén cada una en un bloque distinto if, while, etc.

const: en este caso se usa cuando no se va a cambiar el valor del contenido.

Ejemplo:
Declaramos la variable llamada **nombreUsuario** y con él = le asignamos el valor "Fernan" y luego la mostramos en la consola. Lo mismo con la edad que es numérica

```
let nombreUsuario ="Fernan";
let edadUsuario = "22";
console.log(nombreUsuario);
console.log(edadUsuario);
```

8. Concatenación

Podemos unir texto con variables. Para ello usamos el signo **+**. (Pero si lo usamos con dos números lo que haría es la operación matemática de suma)

```
9   console.log("Tu nombre es:" + nombreUsuario + "y tu edad es de " + edadUsuario + " años")
```

9. Operadores

Operadores aritméticos de suma, resta, multiplicación, división y módulo son: **+ - * / %**

Operadores relacionales son igual, menor que, mayor que, diferente: **== < <= > >= !=**

Operadores lógicos son "y", "o" y la negación: **&& || !**

10. Estructuras de control

Condicionales (if): son aquellas en las que preguntamos por una condición. Si se cumple la condición hacemos una cosa y si no se cumple hacemos otra

Consulta 1 condición:

```
if (condicion) {
    // bloque verdadero
}else {
    // bloque falso
}
```

Consulta 2 o más condiciones:

```
if (condicion1) {
    //bloque verdadero condicion1
} else if(condicion2){
    //bloque verdadero condicion2
}
else {
    //bloque falso condicion1 y 2
}
```

prompt(" "): nos permite solicitar un dato (string, numérico, etc) por pantalla, que guardamos en una variable para luego poder usarla. Por ejemplo solicitamos la nota de "Matemáticas" y si la nota es menor que 5 le decimos que está suspendido y si no aprobado.

```
nombreUsuario = prompt("Introduce tu nombre")
notaMate = prompt("Introduce la nota de mate")
if (notaMate < 5){
    alert("Jopelas " + nombreUsuario + ", has suspendido")
}else {
    alert("Genial " + nombreUsuario + ", has aprobado")
}
```

parseInt() Hay que tener en cuenta que cuando ponemos prompt, el valor introducido siempre es tipo texto, aunque pongamos un número. Podemos verlo poniendo esto en la consola y ver las comillas:

```
> notaMate
<- '4.99'
```

Por lo que podríamos hacer operaciones lógicas(> < =), pero no matemáticas. Por ejemplo, se piden dos notas, una del reto y otra del examen y el resultado final se suma y se le muestra al usuario. Vemos que no funciona a no ser que convirtamos los números con **parseInt()**

No nos funciona el programa ¿Por qué?: por que las notas introducidas se consideran como texto, por lo que hay pasarlas a numérico poniendo **parseInt()**

```
nombreUsuario = prompt("Introduce tu nombre")
notaReto = prompt("Introduce la nota del reto")
notaExamen = prompt("Introduce la nota del examen")
notaTotal = notaReto + notaExamen
if (notaTotal < 5){
    alert("Jopelas " + nombreUsuario + ", has suspendido"
}else {
    alert("Genial " + nombreUsuario + ", has aprobado")
}
```

Pegar aquí el ejercicio bien hecho usando **document.write()**

```
//Pedir nota de mate e indicar si ha suspendido o aprobado
//Solicitamos la nota al profe
let nombreUsuario = prompt("Introduce tu nombre:")
let notaReto = parseInt(prompt("Introduce la nota del reto de Lenguaje de Marcas:"))
let notaExamen = parseInt(prompt("Introduce la nota del examen de Lenguaje de Marcas:"))
let notaFinal = (notaReto + notaExamen)/2


//Consultamos si ha aprobado o no
if (notaFinal < 5) {
    document.write("Jopelas " + nombreUsuario + ", has suspendido con un " + notaFinal)
} else{
    document.write("Genial " + nombreUsuario + ", has aprobado con un " + notaFinal)
}
```

Comprobamos ahora en la consola que no tiene comillas al ser tipo número:

```
> notaReto
<- 3
> notaExamen
<- 2
```

EjercicioIF1: Haz un caso en el que se le pida al usuario su nombre y un número del 1 al 10 por pantalla y si el número introducido es 6 que le diga que ha acertado y sino que le diga que ha fallado, que lo intente de nuevo. (pista: cuidado con el operador que debes usar)

```
//El usuario tiene que acertar un número del 1 al 10

alert("Juego en el que tienes que adivinar un número del 1 al 10")
let nombreUsuario = prompt("Introduce tu nombre")
let numeroUsuario = parseInt(prompt("Introduce el número que crees que es : "))

if (numeroUsuario == 6) {
    document.write("GENIAL!!, " +nombreUsuario+ " has acertado el número que era el 6")
}else{
    document.write("OHHH!!, lo siento " +nombreUsuario+ " no has acertado el número, sigue intentándolo")
}
```

EjercicioIF2: que aparezca un menú en pantalla donde nos pida entre sumar, restar o multiplicar. El usuario elige una opción y luego le pide el valor de los dos números. Dependiendo de la opción que elija, que nos dé el resultado y si no ha elegido una opción válida que nos lo diga.
Nota: si queremos escribir varias líneas en el prompt o alert podemos hacerlo de dos formas:
`texto` :Ponemos las comillas simples inclinadas o…

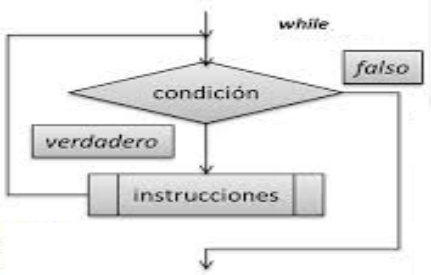
```
//programa para elegir que operación matemática hacer
let eligeMenu = prompt(`
        Elige una opción:
        1: Sumar
        2: Restar
        3: Multiplicar
`)
//Sigue haciendo tú la lógica del programa
```

\n:ponemos saltos de línea

```
//Programa para elegir que operación matemática hacer
let eligeMenu = prompt("Elige una opción:\n1: Sumar\n2: Restar\n3: Multiplicar")
//Sigue haciendo tú la lógica del programa
```

EjercicioIF3: se le pide a un usuario su nombre y un número entero y luego se le dice si el número introducido es par o impar o cero.

Bucle While(): es un bucle que se ejecuta mientras se cumpla una condición y en cuando deja de cumplirse sale del bucle.




```
while (condicion){
    // Bloque verdadero
}
// Ya no se cumple la condicion
```

EjercicioWHILE1: en el siguiente ejemplo nos pide introducir un número del 1 al 10 y hasta que no lo introduce no sigue con el programa.

```
let numero = parseInt(prompt("Deber introducir un número del 1 al 10: "))
while (numero == 0 || numero > 10){
    console.log("No has introducido lo que te pide. Intentalo otra vez")
    numero = parseInt(prompt("Introduce otra vez el número del 1 al 10: "))
}
console.log("Ahora si has introducido lo que te pide. Podemos seguir")
```

Otro ejemplo parecido: que el usuario introduzca un número por pantalla entre 1 y 10. Y hasta que no acierte el número que nosotros queramos poner que le siga pidiendo más números hasta que acierte.

```
1  //Bucle que controla que se introduce un nº del 1 al 10
2  let numeroAdivinar = 6
3  let numeroUsuario = parseInt(prompt("Intenta adivinar un nº 1 al 10:"))
4
5  while (numeroUsuario != numeroAdivinar) {
6      numeroUsuario = parseInt(prompt("No has adivinado, introduce otro nº del 1 al 10:"))
7  }
8  document.write("Genial, has adivinado el número que era " + numeroAdivinar)
```

EjercicioWHILE2: Se quiere mostrar en la página la tabla de multiplicar de un número que se le pide al usuario por pantalla (document.write)

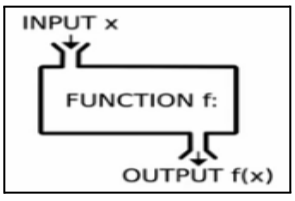
TABLA DE MULTIPLICAR DE UN NÚMERO

8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80

EjercicioWHILE3: seleccionamos un número del 1 al 100. El usuario tiene que adivinarlo, se le van pidiendo números hasta que lo adivine. Si no lo adivina se le da la ayuda de decirle si el número que ha introducido es mayor o menor que el número a adivinar y continua. Cuando lo adivine se le dice "GENIAL, LO HAS ADIVINADO, era el número XXX!!!!!"

---

11. Funciones function()



Una función consiste en la realización de una o varias tareas de forma separada al programa. Su utilidad es no repetir código, por ejemplo, si en varios sitios de mi programa necesito sumar el valor de mis artículos de un carrito de compra, lo que hago es crear la función suma que sume los valores y luego llamar a la función tantas veces como queramos para que sume. Es decir que nos simplifica la introducción de tareas repetitivas.

Una función puede tener algún valor de entrada o no, pero siempre tienen algún resultado o algún valor de salida. Por ejemplo, si tenemos una función que va a realizar una suma, le podemos pasar los sumandos como valores de entrada o no pasárselos ya que estarán dentro de la función, pero la función siempre realizará el cálculo de la suma que se puede mostrar dentro de la función sin devolver ningún valor de salida o devolver su valor como salida para seguir realizando operaciones, en este caso usaremos **return** (luego se ve).

Si creamos variables dentro de una función sólo son visibles dentro de la función. Por el contrario, si hemos creado variables fuera de una función sí se pueden utilizar dentro de la función.

JavaScript y cualquier lenguaje de programación tienen funciones ya definidas, pero el usuario puede crear otras funciones. En el ejemplo siguiente vemos la sintaxis para crear una función genérica y vemos varios ejemplos de funciones.

Sintaxis de una función:

```
function nombreFuncion(parametros){
    //sentencias de la funcion
}

nombreFuncion(parametros) //llamamos a la funcion
```

Función sin parámetros:

```
function saludar(){
    console.log("Hola mundo")
}

saludar()
```

Función con un parámetro.

```
function saludar(nombreUsuario){
    console.log("Bienvenido a nuestra web " + nombreUsuario)
}

saludar("Fernan")
```

También le puedo pasar una variable como parámetro de entrada.

```
function saludar(nombre){
    console.log("Hola "+ nombre)
}
let nombre = "Fernan"
saludar(nombre)
```

Función suma pasando dos variables como parámetros

```
function sumar(n1,n2){
    let resultado = n1 + n2
    console.log("El sesultado de la suma es: " + resultado)
}

numeroUno = parseInt(prompt("Introduce el primer número: "))
numeroDos = parseInt(prompt("Introduce el segudno número: "))
sumar(numeroUno,numeroDos)
```

La función se puede poner en cualquier lugar del código ya que no se ejecuta hasta que no sea llamada, aunque se recomienda ponerlas antes de ser invocadas.

Return

En los ejercicios anteriores la función no nos devolvía ningún valor, ya que el resultado de la función se indicaba dentro de la función. Pero en muchas ocasiones puede que la función nos devuelva un valor para seguir operando con dicho valor, o simplemente mostrarlo, pero esto ahora lo hacemos fuera de la función. Para que la función devuelva un valor debemos usar la sentencia **return**.

```
function cuadrado(n1){
    return  n1 * n1
}

console.log (cuadrado(5))
```

EJERCICIOS CON FUNCIONES

EjercicioFUNCION1: realiza una función con parámetros que calcule y muestre en la página web el perímetro de un rectángulo cuyos lados se piden por pantalla. Puedes usar return o no. (Voluntario para casa)

EjercicioFUNCION2: realiza una función que pida por pantalla dos números y que devuelva un mensaje que escriba en la página web cual es el número mayor, por ejemplo "El número mayor es el X". Puedes usar return o no. SI

EjercicioFUNCION3: realiza el ejercicio anterior pero ahora introduciendo 3 números. Puedes usar return o no. (Voluntario para casa)

EjercicioFUNCION4: realiza una función en la cual el usuario introduce números por pantalla y se le va preguntando si quiere introducir otro número o no (confirm("")). Esos números se van guardando y cuando no quiera introducir más nos tiene que mostrar cuál es el número mayor introducido. Puedes usar return o no. (Voluntario para casa)

EjercicioFUNCION5: realiza una función en la cual se pida por pantalla el precio de un producto y nos tiene que mostrar en la página o como una alerta el valor del IVA(21%) y el precio total del producto. SI `variable.toFixed(2)` sirve para en una variable numérica con decimales indicar cuántos decimales queremos mostrar

isNaN() = es una función de JavaScript que comprueba si un valor **"is Not a Number" - "no es un número"**, dando true si no es un número y false si lo es.
!isNaN() = en este caso es al revés, valida si un valor **"not(is Not a Number)" ó "es un número"**. **Ya sabemos que la admiración ! es excepto o lo contrario.**
	`console.log(isNaN(10)); //False ya que 10 es un número`
	`console.log(isNaN("hola")); //True ya que es string`
	`console.log(isNaN("10")); //True ya que es string al tener ""`
	
	`let variable = "25";`
	`if (!isNaN(variable)) { //False ya que es string`
	`  //Acciones a realizar`
	`}`
	
	`let sueldo = parseInt(prompt("Introduce tu sueldo")) ;`
	`while (isNaN(sueldo)) { //True o false dependiendo del valor`
	`  //Acciones a realizar`
	`}`

Puedes mejorar el ejercicio5 anterior para que introduzca un precio válido, es decir que sea un número y que sea positivo.	`// Verificar si la entrada es válida`
	`if (!isNaN(precioArticulo) && precioArticulo > 0) {`
	`  calcularIVA(precioArticulo); // Llamamos a la función`
	`} else {`
	`  console.log("Por favor, introduce un número válido.");`
	`}`

El mismo ejercicio5, pero ahora un bucle para que hasta que no introduzca un número válido no seguimos con el programa.	`//Verificar que el precio es válido`
	`let precio = parseFloat(prompt("Introduce el precio del producto:"))`
	`while (isNaN(precio) || precio<0) {`
	`  precio = parseFloat(prompt("No es un precio válido, introduce de nuevo:"))`
	`}`
	`//Sigue el programa`

Eventos que llaman a funciones: **onload "function()"** y **onclick="function()"**

Un evento es un suceso que ocurre en la página web, como por ejemplo picar un botón, pasar el ratón sobre un menú o una imagen, cargar la página, etc. En muchas ocasiones nos interesa programar que cuando se produzca un evento que ocurra algo, es decir que se ejecute una función. Dos ejemplos de estos eventos son:

- Cuando se carga la página: el evento es "**onload**" o
- Cuando se hace click en algún elemento como un botón: el evento es "**onclick**".

Lo vemos con dos ejemplos:

onload="funcion()": función que se ejecuta automáticamente cuando se carga la página. Se suele poner en la etiqueta body.
`<!DOCTYPE html>`
`<html>`
`<head> </head>`
`<body onload="saludar()">`
`  <h1>Hello World!</h1>`
	onclick="funcion()": función que se ejecuta al hacer click en el elemento donde lo pongamos como por ej un botón.
	`<!DOCTYPE html>`
	`<html>`
	`<head> </head>`
	`<body>`
	`  <h1>Hello World!</h1>`
	`  <input type="button" value="Saludo" onclick="saludar()">`

</body> </html>	</body> </html>

En el archivo JS
function **saludar**() {
  alert("Hola mundo, la página se ha cargado");
}
	En el archivo JS
	function **saludar**() {
	  alert("Hola mundo, se ha pulsado el botón");
	}

Funciones flecha

Las funciones "flecha" (arrow functions) es otra forma muy usada y concisa de escribir funciones. Las funciones "flecha" se guardan en variables, por ello hay que declararlas con **cons** o **let**. Tienen la siguiente sintaxis básica:

Función flecha:

```
const funcionX = (parámetro1, parámetro2, ...) => {
  // Cuerpo de la función
  // Una o más instrucciones
  return valor; //Opcional si devuelve un valor
}

funcionX(valorParam2,valorParam2,...)
```

Función no flecha:

```
function nombreFuncion(parametros){
    //sentencias de la funcion
}

nombreFuncion(parametros) //llamamos a la funcion
```

Hay diferencias entre usar funciones flecha o normales, pero se escapan al alcance de este curso de introducción, una de las diferencias es que en funciones pequeñas que solo tienen una instrucción no es necesario poner llaves. El resto es igual que una función normal, puede tener parámetros o no, pueden devolver un valor (return) o no.

```
// Función normal
function suma(a, b) {
    return a + b;
}

// Función flecha
const suma = (a, b) => a + b;
```

EJEMPLOS DE FUNCIONES FLECHA:

Vamos a crear una página sencilla html que llama a un archivo javascript para ir probando los siguientes casos

```
const saludar = () =>
    console.log("Hola!");

saludar();
```

```
const saludar = () => {
    console.log("Hola!");
}
saludar();
```

Sin parámetros:

En el primer ejemplo debido a que solo tiene una sentencia no hace falta poner llaves **{ }**

En este segundo vemos que aunque solo tiene una sentencia, podemos también poner las llaves.

A partir de ahora lo hacemos con llaves ya que se visualiza y entiende mejor, además si hay varias sentencias habría que ponerlas.

```
const cuadrado = (x) => {
  return x * x;
}
console.log(cuadrado(4));
```

Con un solo parámetro creamos la función recibiendo el parámetro x. La función realiza la operación de x*x y devuelve el resultado con return ya que el resultado se usa fuera de la función..
Llamamos a la función para visualizar su resultado en la consola.

```
const multiplicar = (a, b) => {
    return a * b;
}
console.log(multiplicar(3, 4));
```

Con múltiples parámetros
El ejemplo es igual que el anterior, pero en este caso pasamos dos parámetros.

```
const suma = (a, b) => {
    console.log("Sumando " +  a + " y " + b);
    return a + b;
  };

  console.log(suma(5, 7));
```

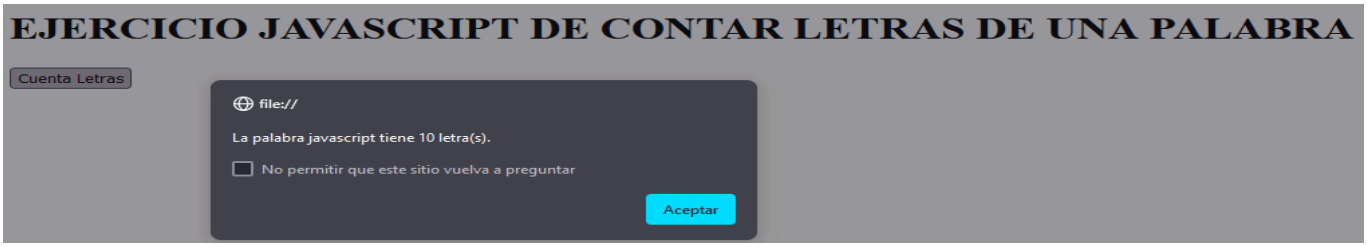
Con un cuerpo de varias líneas
Al tener varias líneas requiere obligatoriamente llaves { } y return si deseas devolver un valor

EjercicioFLECHA1 (ya se hizo sin funciones, opcional para vosotros, no lo hacemos)
Realiza una página html con un botón que si lo pulsamos llama a una función flecha en la cual pida al usuario a través de prompt lo siguiente:

a. La operación que desea realizar (1-Sumar, 2-Restar, 3-Multiplicar, 4-Dividir),
b. Le pide el primer número.
c. Le pide el segundo número.
d. Le da el resultado de la operación elegida.

https://drive.google.com/drive/folders/1tuHRJBbFckNxWI0y4ioV3IW7XALJtA1L?usp=sharing

EjercicioFLECHA2a:
Realiza una página html con un botón que si lo pulsamos llama a una función flecha en la cual pida al usuario una palabra o frase y lo que debe de hacer es contar los caracteres de dicha palabra y lo muestra por pantalla.
Ayuda: `let longitud = variable.length;  //permite obtener la longitud de caracteres del contenido de la variable, es decir que nos da un número (caracteres)que lo podemos guardar en otra variable (longitud)`



En el caso de que NO quisiéramos contar los espacios en blanco de una palabra o frase reemplazaríamos el contenido de la variable quitando los espacios de la siguiente forma (no hacer):

```
longitudSinEspacios = variable.replace(/[^a-zA-Z]/g, "").length
```

EjercicioFLECHA2b: *Contraseña segura con 8 caracteres o más*

El ejercicio anterior lo puedes mejorar suponiendo que la palabra debe tener al menos 8 caracteres (como si fuese una contraseña), y mientras no cumpla al menos esa condición se le sigue pidiendo la palabra, y una vez que la cumpla se le muestre el mensaje "*Contraseña correcta, la palabra "XXXX " tiene Y caracteres*".

https://drive.google.com/drive/folders/1k6dqwyjrA8ZsG3PZoFgCJ5uzZ21ZpUw0?usp=sharing

EjercicioFLECHA2c: *Contraseña compleja*

Seguimos con el ejercicio anterior lo puedes mejorar suponiendo además de tener al menos 8 caracteres, debe de tener mayúsculas, minúsculas y números, y mientras no cumpla al menos esa condición se le sigue pidiendo la palabra, y una vez que la cumpla se le muestre el mensaje "*Contraseña correcta, la contraseña es segura*".

Para comprobar si el contenido de una variable(password) tiene minúsculas, mayúsculas o números usamos las siguientes sentencias que nos devuelve true si lo cumple y false si no lo cumple

```
let minusculas = /[a-z]/.test(password); //Nos devuelve true si tiene alguna minúscula
let mayusculas = /[A-Z]/.test(password); //Nos devuelve true si tiene alguna mayúscula
let numeros = /[0-9]/.test(password); //Nos devuelve true si tiene algún número
```

12. ENTRADA DE DATOS. INPUT EN LUGAR DE PROMPT (DOM)

Hasta ahora hemos visto que si el usuario quería introducir un dato, este se le pedía a través de una caja de texto en pantalla usando **prompt**. Pero en realidad en las páginas web los datos se suelen introducir a través de un **input** (como si fuese un formulario). Por lo que ahora debemos de aprender cómo coger ese dato del input y usarlo en nuestro programa o incluso en el apartado siguiente veremos como dejar un dato en otro input vacío.

Podemos seleccionar cualquier elemento o caja html, o en caso de que sea un elemento de un formulario podemos seleccionar el elemento o su contenido. Para ello debemos de etiquetar en el html el elemento con un identificador que puede ser un id o class. Luego podemos seleccionar dicho elemento o su contenido de la siguiente manera:

- Para seleccionar el elemento o caja (no su contenido) sería:
  ```
  let elemento = document.getElementById("identificadorID")
  ```
- En cambio si es un elemento de un formulario, como un input, podemos seleccionar su contenido o valor:
  ```
  let elemento = document.getElementById("identificadorID").value
  ```

Una vez seleccionado el contenido o valor lo introducimos en una variable y trabajamos con la variable.

EjercicioInput1: ejercicio anterior de contraseñas complejas, pero con un input

Realizamos el ejercicio anterior, pero ahora el usuario introduce la password en un input, además no hacemos un bucle while, ya que no le pedimos por pantalla, ya que lo introduce en un input, por lo que hacemos un if.

https://drive.google.com/drive/folders/1XywWYjD3T_CX02gMsUenb_9PuqDpCt9N?usp=sharing

13. SALIDA DE RESULTADOS EN LA PÁGINA (DOM)

Para mostrar la salida o resultado al usuario podemos hacerlo de varias formas:

- Con una **alerta** que ya hemos visto.
- Con **document.write**, pero se nos carga al final de la página o en ocasiones en una página nueva.
- **Dentro de una celda input**. En este caso introducimos la variable que tenga el resultado en el valor de la celda.

```
elemento = document.getElementById("identificadorID").value
```

- **Dentro de cualquier elemento de la página**, como un párrafo, etc. En este caso en el html debemos de haber creado ese elemento o caja vacía (por ej. un párrafo `<p id="resultado"></p>` y podemos usar dos formas:

```
let elemento = document.getElementById("identificadorID") //Lo seleccionamos
elemento.textContent = resultado;  //Una forma para sacar el resultado
elemento.innerHTML = resultado;   //Otra forma
```

Para mostrar un mensaje que se va acumulando, por ejemplo cada vez que pasa por un valor de un array que coja un valor y lo guarde y se vaya acumulando a las veces que ha pasado antes sería:

```
mensaje = "";  //Creamos la variable del mensaje vacío
    for (let index = 0; index < nombreArray.length; index++) {
        if (nombreArray[index] > 10) {
           //Hay que poner += para que se acumule el mensaje cada vez que pasa por
            el array, sino sería solo =
           mensaje += nombreArray[index] + ", ";//Guarda y acumula la información
        }
    }
    resultado.innerHTML = mensaje; //Para mostrar el mensaje en un <p>
```

EjercicioInput2

Tenemos un input tipo numérico donde un usuario introduce una edad y el programa clasifica la persona en varias categorías que son niño/a (1-12), adolescente(13-18), joven (19-29), adulto(30-69), anciano/a (69…), el resto de edades nos muestra una alerta con un mensaje. Según la edad introducida se muestra en la página, en la caja con el **id="resultado"** el mensaje correspondiente.

https://drive.google.com/drive/folders/1qq1KrH2pzw6nXVGmQzZTyfud0wvrJDWH?usp=sharing

EjercicioInput3 (voluntario, no se hace en clase)

Tenemos una página igual que la anterior, pero con dos inputs donde el usuario tiene que introducir en el primer input su posible contraseña y en el otro input tiene que volver a introducirla. Además de que se cumpla las condiciones de contraseña compleja, se pide que si las contraseñas no son iguales aparece una alerta indicándolo, si no es compleja que aparezca otra alerta indicándolo, y si todo es correcto nos escribe abajo en la página un texto diciendo "Contraseñas válidas"

https://drive.google.com/drive/folders/11cswGsq7oA_tn9NSjzAm6RjJPPR_PE__?usp=sharing

EjercicioInput4 (voluntario para practicar, no se hace en clase)

Realiza una página web con tres inputs donde se introducen tres números. Debajo tenemos un botón que al picar nos calcula cual es elmayor de ellos y nos lo muestra en la página o en otro input o en una alerta, como queráis.

14. Arrays

Un array es una lista de elementos que tienen algo en común, por ejemplo, una lista de animales, una lista de productos del carrito de compra de un usuario, etc. Cada elemento del array se identifica con un número índice (0,1.,2…) que se usa para saber la posición del elemento y para obtener dicho elemento. El tamaño total del array se puede ver con el atributo "**length**" que es el nº de elementos que tiene (no el índice).

```
var nombre_array = [valor1, valor2, ..., valorN];
```

índice = **[0]** **[1]** .... **[n-1]** **(length = n)**

```
let frutas =["pera","manzana","naranja","platano"]
console.log(frutas)
console.log(frutas[2])
console.log(frutas.length)
```

```
▼ Array(4) [ "pera", "manzana", "naranja", "platano" ]
    0: "pera"
    1: "manzana"
    2: "naranja"
    3: "platano"
    length: 4
  ▶ <prototype>: Array []
naranja
4
```

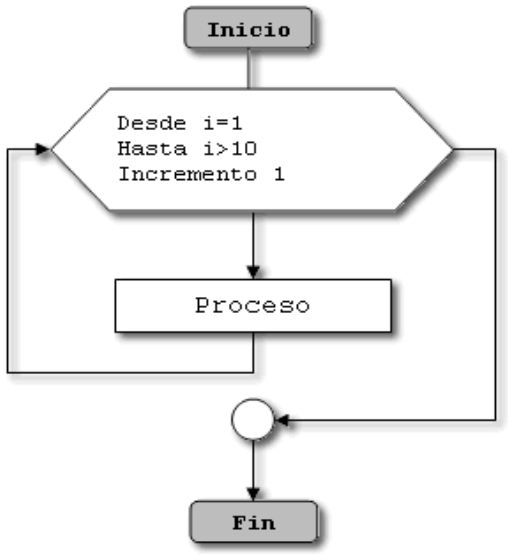
Lo normal es que un array tenga muchos elementos, por lo que si quisiéramos obtener muchos o todos los elementos habría que recorrer el array desde el principio al final. Para recorrer el array de forma secuencial del inicio al final existe la estructura **for()**.

Bucle for(): En todos los lenguajes de programación existe el bucle **for()** que, desde un valor inicial hasta uno final, permite realizar las acciones que queramos. Pero cada vez que pasemos por el bucle debemos de pasar al siguiente elemento de la secuencia incrementando o en su caso disminuyendo la posición en alguna cantidad (normalmente en 1) hasta que llegue al valor final en el cual saldrá del bucle.

Los bucles **for()** se utilizan para recorrer un array de elementos del primero al último, o al que queramos. Genéricamente sería así:

```
for(variable = valorInicial; valorFinal; incremento){
    // acciones a realizar
}
```

El diagrama de flujo sería:



```
let frutas =["pera","manzana","naranja","platano"]
console.log(frutas)
console.log(frutas[2])
console.log(frutas.length)

for (let i = 0; i < frutas.length; i++) {
    document.write(frutas[i] + "<br>");
}
```

```
▶ Array(4) [ "pera", "manzana", "naranja", "platano" ]
naranja
4
```

pera
manzana
naranja
platano

EjercicioARRAY1
Crear un array llamado meses con el nombre de los doce meses del año. Mostrar en la página web, en la misma línea separados por comas los nombres de los 12 meses.
EjercicioARRAY2
Crea un array de los números 0,1,2,3,4,5,6,7,8,9,10,11,12, y muestra por pantalla solo los números pares.
EjercicioARRAY3
Suma todos los valores del array anterior y al final nos diga "La suma total del array es: XXX".

Modificar un array. PUSH, POP, SHIFT, UNSHIFT

array.push(): lo que hace es agregar un nuevo elemento al final del array.

```
1  const frutas = ["manzana", "pera", "naranja"]
2  console.log(frutas)
3  frutas.push("uva")
4  console.log(frutas)
```

```
▶ (3) ['manzana', 'pera', 'naranja']
▶ (4) ['manzana', 'pera', 'naranja', 'uva']
```

array.unshift(): también agrega un nuevo elemento, pero al principio del array.

```
1  const frutas = ["manzana", "pera", "naranja"]
2  console.log(frutas)
3  frutas.unshift("fresa")
4  console.log(frutas)
```

```
▶ (3) ['manzana', 'pera', 'naranja']
▶ (4) ['fresa', 'manzana', 'pera', 'naranja']
```

array.pop(): elimina el último elemento del array y lo podemos ver si lo guardamos en una variable

```
1  const frutas = ["manzana", "pera", "naranja"]
2  console.log(frutas)
3  const frutaeliminada = frutas.pop()
4  console.log(frutas)
5  console.log(frutaeliminada)
```

```
▶ (3) ['manzana', 'pera', 'naranja']
▶ (2) ['manzana', 'pera']
naranja
```

array.shift(): elimina el primer elemento del array y también lo podemos ver si queremos.

```
const frutas = ["manzana", "pera", "naranja"]
console.log(frutas)
const frutaeliminada = frutas.shift()
console.log(frutas)
console.log(frutaeliminada)
```

```
▸ (3) ['manzana', 'pera', 'naranja']
▸ (2) ['pera', 'naranja']
manzana
```

Buscar un elemento en un array indexOf

Tenemos dos opciones:

- Buscar el elemento: **array.includes("elemento")**: en este caso buscamos por el nombre del elemento y nos devuelve true si lo encuentra y false si no lo hace.

```
let frutas = ["manzana", "pera", "plátano", "uva"];
console.log(frutas.includes("pera")); // true
console.log(frutas.includes("naranja")); // false
```

- Buscar su índice: **array.indexOf("elemento"):** busca el elemento en el array y si lo encuentra nos devuelve su índice, y si no lo encuentra devuelve como índice el valor -1. (también podríamos haber encontrado su índice recorriendo todo el array, pero de esta forma es más rápido). Usaremos esta opción para cuando queramos eliminar un elemento de un array.

```
let frutas = ["manzana", "pera", "plátano", "uva"];
let indice = frutas.indexOf("plátano"); // Devuelve 2
let noExiste = frutas.indexOf("naranja"); // Devuelve -1
```

EjercicioARRAY4a: Queremos hacer un array de colores, para ello tenemos un input donde ponemos un color y un botón al lado para agregarlo al array y así continuamente. Luego debajo tenemos otro input con otro botón para ver si existe un color en el array y por último debajo otro input con otro botón para saber la posición concreta de un color. En estos dos últimos casos nos muestra debajo en una celda el resultado.
https://drive.google.com/drive/folders/1da0QRulJ5xB6BzxI3hGA3NQrP3HrB-Ig?usp=sharing

Eliminar un elemento en una posición intermedia. array.splice(indice,1)

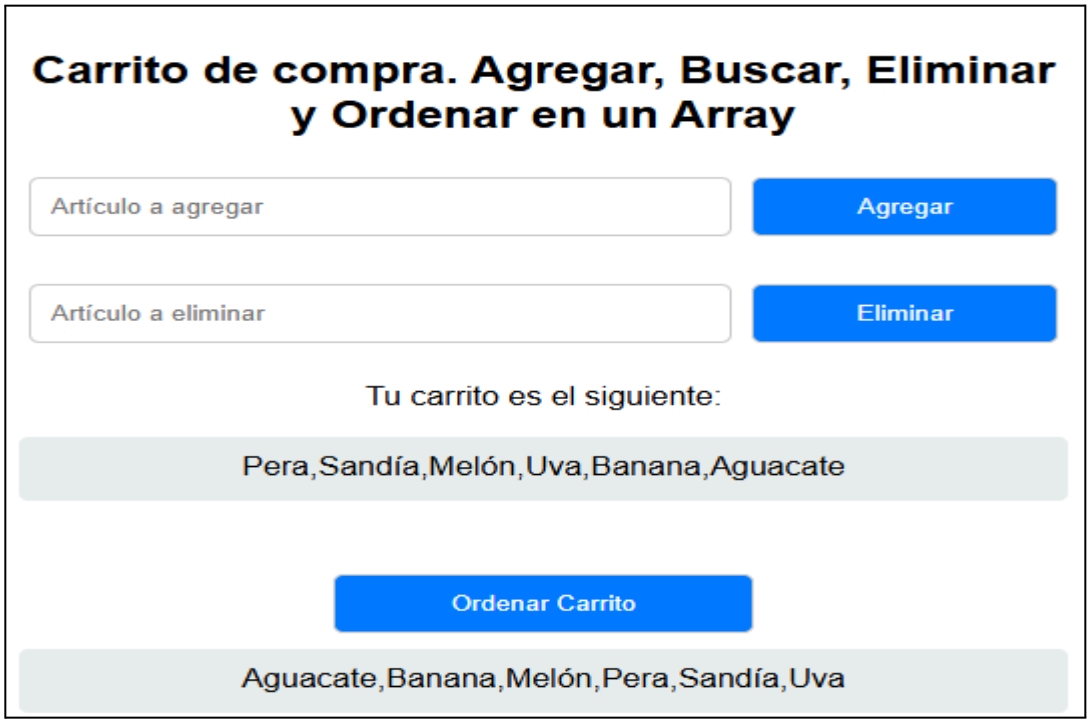
Hemos visto antes cómo eliminar el primer o último elemento del array, pero para eliminar un elemento determinado, primero hay que saber su índice que podemos buscar con **indexOf** y luego eliminarlo con **splice(indice,1)** que lo que hace es eliminar el elemento del array con dicho índice y el 1 significa que se elimina 1 elemento a partir de dicho índice, se decir en este caso se elimina solo ese elemento.

EjercicioARRAY4b: En el ejercicio anterior podríamos agregar un input y un botón para introducir un color y si existe eliminarlo de la lista.

EjercicioARRAY5: Crear un carrito de compra de un usuario:

- Tenemos un input donde un usuario puede ir agregando productos a un carrito de compras. Al pulsar en el botón "Agregar" el producto se agrega y a la vez se muestra abajo (párrafo <p>) todo el carrito. Si el producto lo deja en blanco le da una alerta y si introduce un artículo que ya existe le da otra alerta y no lo agrega.
- También le damos la posibilidad de eliminar algún artículo del carrito de compra, si el artículo no existe le da una alerta y si sí existe lo eliminamos y nos actualiza el carrito en la parte inferior.
- Por último nos permite mostrar el carrito ordenado alfabéticamente de A-Z y nos lo muestra en la parte inferior.

https://drive.google.com/drive/folders/1_zSVly_xmkQI4xi58TG4FR1fzGP180-p?usp=sharing



EjercicioARRAY6: Alumnos de Lenguaje de Marcas y sus notas (voluntario para practicar)

Tenemos una página con dos inputs, en uno se introduce el nombre de un/a alumno/a y en otro se introduce su nota de Lenguaje de Marcas (tiene que ser de 0 a 10). Tenemos varios botones.

- Uno para introducir los dos valores anteriores,
- Otro para consultar todas las notas de todos/as los/as alumnos/as

- Otro para consultar los/as alumnos/as aprobados/as,
- Otro para consultar los/as alumnos/as que han sacado 9 o más.
- Otro para calcular la nota media,
- Etc.

https://drive.google.com/drive/folders/1zqyrNaHlQ89dZqGibez-692XHGMCXsFi?usp=sharing

NO VEMOS MÁS, EL RESTO PARA EL AÑO QUE VIENE EN SEGUNDO

Arrays asociativos (objetos)

Hemos visto los arrays indexados, en los cuales cada valor del array se asocia a un índice (0,1,2…). Pero hay muchos más tipos de arrays que pueden contener datos más complejos como los arrays asociativos u objetos que en vez de un índice que se asocie a cada elemento, existe una key que se asocie a dicho elemento. Esta key no tiene que ser un número, puede ser una cadena de texto, etc.

En JavaScript, los arrays asociativos no son estructuras de datos nativas (en php sí). Sin embargo, podemos simularlos utilizando objetos, ya que estos permiten asociar claves (keys) a valores del array. A diferencia de los arrays normales que tienen índices numéricos, los objetos permiten usar claves personalizadas (como cadenas de texto).

Ejemplos:

let **notasMarcas** = { alumno1: 8.5, alumno2: 3.8, alumno3: 6.75, alumno4: 9.1, alumno5: 4.2 };
let **telefonos** = { Juan: "123-456-789", María: "987-654-321", Pedro: "555-555-555" };
let **loteria** = { 12345: "Primer premio: 1 millón", 67890: "Segundo premio: 500 mil", 11223: "Tercer premio: 250 mil" };
let **persona** = { nombre: "Juan", edad: 30, profesión: "Ingeniero" };
let **libro** = { título: "El Principito", autor: "Antoine de Saint-Exupéry", año: 1943, género: "Ficción" };
Nota: no hace falta poner comillas a las keys a no ser que haya espacios.

Acceso a los valores del array. Se usa las keys:

console.log(**array[key]**); console.log(**notasMarcas[alumno2]**);
alert("Has ganado un premio de" + **loteria[12345**]);

Búsqueda de un valor de array. Se usa las keys:

if (**notasMarcas[alumno3]**) { //buscamos en el array si existe el elemento con la key alumno3
}

EjercicioARRAY6: Lotería de Navidad

Se quiere hacer un ejercicio en html, css y javascript donde pida por pantalla un número de lotería, lo consulte en un array asociativo(objeto) que tenga 5 premios y si está premiado nos lo diga con un alerta, y si no está premiado que también lo diga con alerta
https://drive.google.com/drive/folders/1TwNTQD9Yot0fYe5Y-PPIw0RbKNUKxr2l?usp=sharing

15. OBJETOS (NO VER)

Muchos lenguajes de programación están orientados a objetos. Un objeto es algo que tiene unas propiedades y cada propiedad puede tener un valor.

Ejemplo de un objeto puede ser un gato. Sus propiedades pueden ser "nombre", "duerme", "edad", "enemigos", etc. Esas propiedades pueden ser tipo texto, número, booleano, array, etc.

Los objetos se parecen a los arrays, pero no tienen índice como los arrays, sino que tienen propiedades.

La sintaxis es:

```
const nombreObjeto = {
    propiedad1: valor,
    propiedad2: valor,
    propiedad3: valor,
}
```

Un objeto gato:

```
//creamos el objeto gato
const gato = {
    nombre: 'Valiente',
    duerme: true,
    edad: 10,
    enemigos: ["agua", "perros"]
}
//para acceder al objeto y a los valores
console.log(gato)
console.log(gato.nombre)
console.log(gato.enemigos[0])
```

CRUD (create, read, update, delete) Propiedades de un objeto

CRUD permite crear, leer, actualizar o eliminar propiedades de un objeto y su valor.

```
//creamos el objeto gato
const gato = {
    nombre: 'Valiente',
    duerme: true,
    edad: 10,
    enemigos: ["agua", "perros"]
}
console.log(gato)
//para agregar una propiedad con su valor
gato.color = "azul"
//para leer un valor de una propiedad
console.log(gato)
console.log(gato.nombre)
console.log(gato.enemigos[0])
//para modificar una valor de una propiedad
gato.edad = 5
//para eliminar una propiedad ponemos delete
delete gato.duerme

console.log(gato)
```

```
▸ {nombre: 'Valiente', duerme: true, edad: 10, enemigos: Array(2)}
▸ {nombre: 'Valiente', duerme: true, edad: 10, enemigos: Array(2), color: 'azul'}
Valiente
agua
▸ {nombre: 'Valiente', edad: 5, enemigos: Array(2), color: 'azul'}
```

16. Objeto Date:

JavaScript tiene muchos objetos ya definidos como por ejemplo el de la fecha actual. Podemos crear un objeto utilizando el objeto fecha y trabajar con él. No hay que poner las llaves ya que es un objeto creado a través de una función.

```
let fecha =  Date();
console.log(fecha)
```

17. Objeto Math (librería):

Esta librería tiene muchos objetos para realizar operaciones matemáticas. Si escribimos **Match.** nos enseña todas las opciones que podemos hacer. Por ejemplo, podemos buscar el valor de PI, calcular el máximo precio o mínimo entre varios números, redondear, generar números aleatorios, etc.

```
const numeroPi = Math.PI
console.log(numeroPi)

let maximo = Math.max(4,5,8,6,1,2,4,11, -8,95,6,78)
console.log(maximo)

console.log(Math.min(4,5,8,6,1,2,4,11, -8,95,6,78))
console.log(Math.round(4.5))
console.log(Math.floor(4.9999))
```

```
3.141592653589793
95
-8
5
4
```

```
//Número aleatorio entre 0(lo incluye) y 1(sin incluir)
let numeroAleatorio = Math.random();
document.write(numeroAleatorio)
```

random() saca un número entre 0 y 1, sin incluir el 1

```
0.6180805777937661
```

Para generar un número aleatorio entre un mínimo y un máximo haríamos lo siguiente:

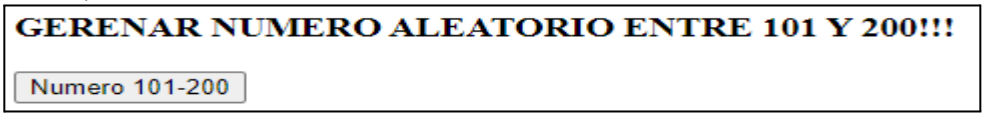
```
//Número aleatorio entre un número1 y un número2 (Fórmula Genérica)
let numeroAleatorio = Math.floor(Math.random() * (max - min + 1)) + min;
```

```
//Por ejemplo entre el 5 y 20 sería
let numeroAleatorio2 = Math.floor(Math.random() * 16) + 5;

document.write(numeroAleatorio2)
```

EJERCICIOS NÚMEROS ALEATORIOS

ejerRANDOM1: "Generar número 101-200":
Si picamos en un botón, se genera un número aleatorio entre 101 y 200 y se muestra con una alerta (o en un input o debajo en un párrafo).



ejerRANDOM2: "Generar número entero entre un mínimo y un máximo":
Tenemos dos inputs donde un usuario introduce el número inferior y otro input con el nº superior (el nº inferior no puede ser superior al inferior) y un botón que al pulsarlo nos genera un número aleatorio entre esos dos y nos muestra en otro input inferior que está deshabilitado.
https://drive.google.com/drive/folders/13SDRkDPQLb49MBpb9xXvGRBt7A0_zIin?usp=sharing

ejerRANDOM3: Igual que el ejercicio anterior, pero ahora el número debe de ser par.

ejerDADOS

Es un juego en el que se van a tirar dos dados. El jugador tiene que apostar qué cara va a salir en cada dado y para ello realiza una apuesta en €. Si el usuario acierta un dado se le duplica su apuesta, si acierta los dos dados se triplica y si no acierta pierde su apuesta. Tener en cuenta que la apuesta del usuario tiene que ser un valor posible de un dado y que tiene que apostar un valor numérico positivo.
https://drive.google.com/drive/folders/11jsMwWB6Z7hp8LdPDcM2nyoPTS8E0u8v?usp=sharing

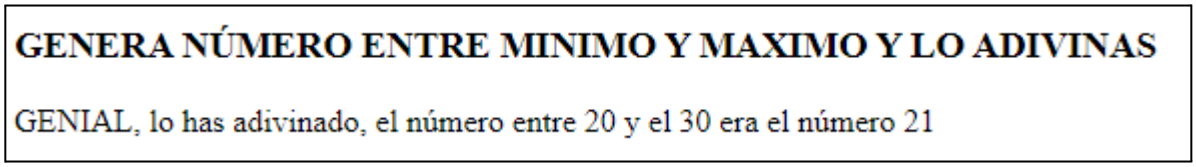
ejerRULETA: No hacemos

ejerEthazi "Grupos Ethazi": El profesor os va a dar un ejercicio (html, css, js) que contiene un array de todos los alumnos/as de clase que son 25. En la página tenemos dos botones, uno para "Mostrar alumnos/as" y otro para "Hacer grupos". Se pide:
https://drive.google.com/drive/folders/1HGyKimrj2gYZs1kOMbSTU3SV5yrRsU_8?usp=sharing

a. Si picamos en el botón "Mostrar alumnos/as" nos muestra los/as 25 alumnos/as sin engrupar.

b. Si picamos en el botón de "Elegir grupos", nos aparecerán de forma aleatoria todos los grupos.

"**Adivinar número**": ME HE QUEDADO AQUIIII
Se le pregunta al usuario entre qué número mínimo y máximo quiere generar un nº aleatorio para luego intentar adivinarlo. A continuación se le va pidiendo un número hasta que lo adivine o diciéndole si "el número introducido es mayor" o "es menor que el número a adivinar". Cuando adivine le le dice "HAS ADIVINADO EL NÚMERO QUE ES X". Se podría mejorar mostrándonos cuántas veces ha necesitado para adivinarlo. "HAS ADIVINADO EL NÚMERO QUE ES XX AL DE ZZ INTENTOS"



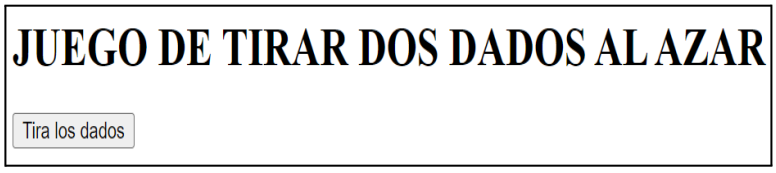
1. **"Notas de clase"**: El profesor os va a dar un ejercicio con dos archivos (html, js) que contiene un array de todos los alumnos/as de clase. Se pide:
   c. Se debe generar de forma aleatoria una nota para la asignatura de Lenguaje de Marcas para cada alumno/a que se guardará en otro array.
   d. Mostrar en la página el nombre de cada alumno/a con su nota.

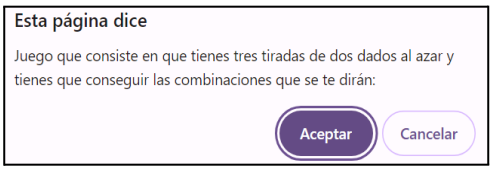
ALUMNOS/AS DE CLASE Y SUS NOTAS DE LM

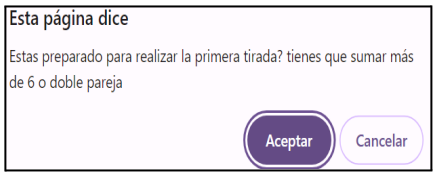
Koldo ha sacado un 9
Roberto ha sacado un 1
Iker ha sacado un 5
Marius ha sacado un 7
Daniela ha sacado un 5
Galder ha sacado un 3
Joseba ha sacado un 8
Ibai ha sacado un 5
Jose ha sacado un 2
AsierH ha sacado un 8
Endika ha sacado un 6
Skarleth ha sacado un 6
Gorka ha sacado un 7
Julen ha sacado un 7
Gabriel ha sacado un 4
Nicolas ha sacado un 3
Mikel ha sacado un 7
Ibai ha sacado un 4
AsierS ha sacado un 5
Keiner ha sacado un 4
Gontzal ha sacado un 9
Karla ha sacado un 3
Ekain ha sacado un 8

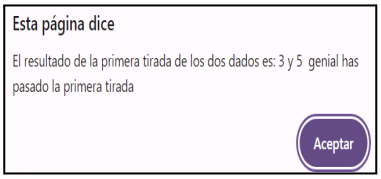
e. Se quieren hacer dos supuestos distintos (independientes) (voluntario):
   a. Lo mismo que antes pero solo para los/as alumnos/as aprobados/as
   b. Se pide al usuario por pantalla la nota mínima a partir de la cual se quiere sacar por pantalla a todos los alumnos que hayan sacado más de esa nota.

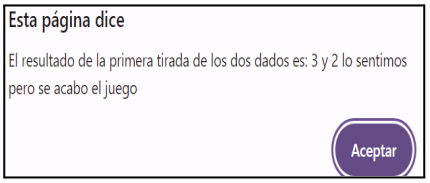
2. **"Juego de dados":** Tenemos un botón(imagen1) que si picamos nos informa del funcionamiento del juego (imagen2) que consiste en que tenemos tres tiradas de dados y hay que sacar las combinaciones que nos va diciendo. Aceptamos y nos informa de la primera tirada (imagen3). Si sacamos más de 6 o doble pareja nos dice (imagen4) que hemos pasado la primera tirada y que vamos a por la segunda tirada, sino (imagen5) y se acaba el juego. En la segunda tirada lo mismo pero hay que sumar más de 8 o doble pareja (imagen6), y en la tercera tirada hay que sacar más de 10 o doble pareja y si ganamos nos da el premio de "HAS GANADO 0,5 PUNTOS MÁS EN JAVASCRIPT"

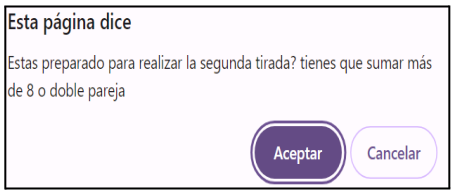






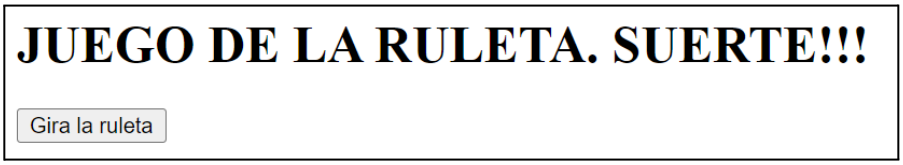


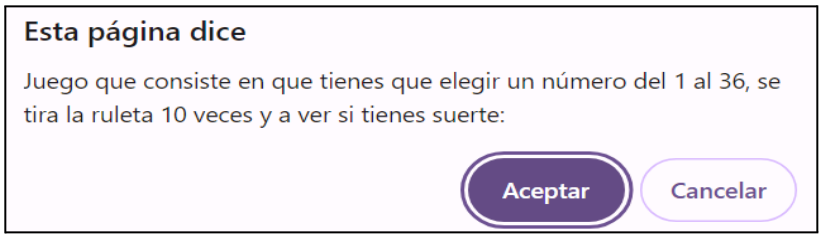


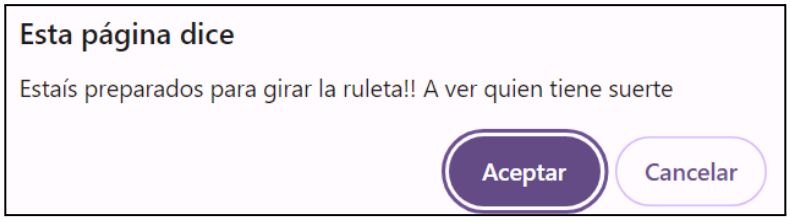

Esta página dice
Estas preparado para realizar la segunda tirada? tienes que sumar más de 8 o doble pareja
Aceptar
Cancelar

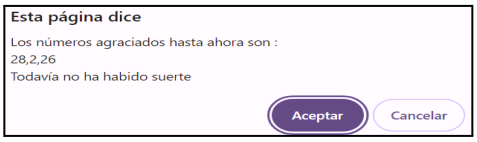
3. **"La ruleta":** Es una variedad del juego de la ruleta de los casinos en el cual se apuesta a un número entre 1 y 36. Tenemos un botón que si picamos en él empieza el juego. (imagen1)
   - Primero nos explica como funciona el juego (imagen2)
   - Luego, suponiendo que hay dos jugadores les pide su nombre y el número que apuesta cada uno, no pueden apostar los dos jugadores al mismo número.
   - Luego avisa que va a empezar el sorteo. (imagen3)
   - Se lanza la ruleta y sale un número y lo muestra en pantalla y dice si algún jugador ha acertado y se acaba el juego. Si no ha acertado genera otro número y los va mostrando junto con los números anteriores (imagen4). Así hasta generar los 10 números o que alguno de los jugadores o los dos hayan ganado. (imagen5). Se gana cuando sale tu número, cuando sale el nº de la suerte (7) y cuando sale la niña bonita (15)

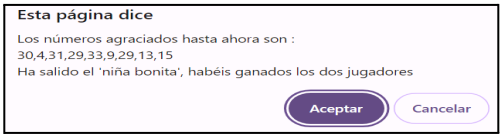
.











Ejercicio "La ruleta II"

Igual que el anterior pero los números que introducen los usuarios deben de estar entre 1 y 36 y los generados en la ruleta no se pueden repetir.

EJERCICIOS DE REPASO

notasAlumnos.js (1punto)

En una página web se van pidiendo por pantalla el nombre y nota de cada alumno de una asignatura que se quiere guardar en dos arrays distintos (nombresAlumnos y notasAlumnos). Cada vez que introducimos un nombre y nota nos dice que si queremos seguir. Por ejemplo:

Array1	Iñaki	Marta	Xabi	Hugo	.....	Unai	Roberto	Ana	María
Array2	6,85	4,56	5,65	2,50	.....	0,56	9,95	9,75	6,5

En la página se muestra un botón para obtener el nombre y la nota de los alumnos que han aprobado
Otro botón para los que han suspendido.
Otro botón para los que han sacado más de un 7

.

asirEdad.js (1punto)

Programa que pide por pantalla que ciclo de fp está haciendo y su edad. Si estudia asir y es mayor de edad se muestra el resto de la página, sino se le dice que no cumple con los requisitos y no puede seguir y por lo tanto no se le muestra la página.

agenciaViajes.js (1punto)

Una agencia de viajes de Bilbao vende viajes a distintos destinos y según el mismo tiene un descuento. Además si el mes de viaje es julio o agosto tiene un suplemento según la tabla siguiente que aparece en la página html.

Destino	Descuento
Mallorca	5%
Tenerife	8%
Península Ibérica	—-

Mes	Suplemento
Julio	+200
Agosto	+300
Otro mes	—

Se le pide por pantalla el destino y el mes del viaje (de enero a diciembre) y se muestra el la página el presupuesto del viaje teniendo en cuenta el descuento y suplemento posible