

## Actividades

1. Documenta qué devuelve cada conversión de actividad\_casting.js (1 punto)

```
<script>
// Funcion de flecha autoejecutable
(() => {

    const conversions = [];

    // String to Number ////////////////////////////////////////

    // Devuelve 123, convierte la cadena en enteros
    conversions.push(Number("123"));

    //Devuelve Nan porque 123abc no se puede onvertir al tener caracteres
    conversions.push(Number("123abc"));

    //Devuelve 0, al no haber valor se interpreta como 0
    conversions.push(Number(""));

    //Devuelve 0, ya que tambien ignora los espacios en blanco
    conversions.push(Number(" "));

    // Boolean to Number ////////////////////////////////////////

    // Devuelve 1, true se interpreta como 1
    conversions.push(Number(true));

    // Devuelve 0, false se interpreta como 0
    conversions.push(Number(false));

});
```

```
// Null and Undefined to Number ////////////////////////////////////////  
  
// Devuelve 0, null se interpreta como ausencia de valor, osea, 0.  
conversions.push(Number(null));  
  
// Devuelve Nan, significa "No es un numero" a pesar de ser tipo Number  
conversions.push(Number(undefined));  
  
// String to Boolean ////////////////////////////////////////  
  
// Devuelve false, ya que 0 es false  
conversions.push(Boolean(""));  
  
// Se considera True, al tener espacio Boolean no lo ignora a diferencia de Number (trim)  
conversions.push(Boolean(" "));  
  
// Devuelve true, porque toda cadena no vacia se considera verdadero al pasar a Boolean  
conversions.push(Boolean("false"));  
  
// Number to String ////////////////////////////////////////  
  
// Devuelve la cadena 123  
conversions.push(String(123));  
  
// Devuelve una cadena representando 0  
conversions.push(String(0));  
  
// Devuelve una cadena NaN  
conversions.push(String(NaN));  
  
// Devuelve la cadena Infinity  
conversions.push(String(Infinity));
```

```
// Boolean to String //////////////////////////////////////  
  
// Devuelve true  
conversions.push(String(true));  
  
// Devuelve false  
conversions.push(String(false));  
  
// Null and Undefined to String //////////////////////////////////////  
  
// Devuelve null como cadena  
conversions.push(String(null));  
  
// Devuelve undefined como cadena  
conversions.push(String(undefined));  
  
// Number to Boolean //////////////////////////////////////  
  
// Devuelve false, al ser 0 un valor falsy  
conversions.push(Boolean(0));  
  
// Cualquier numero diferente a 0 es true  
conversions.push(Boolean(1));  
  
// Nan se considera tambien false  
conversions.push(Boolean(NaN));  
  
// Al no ser 0 ni Nan se considera true  
conversions.push(Boolean(Infinity));
```

```
// Implicit Coercion in Operations //////////////////////////////////////  
  
// Devuelve 51 porque concatena y no suma  
conversions.push("5" + 1);  
  
// Por el signo - tiene sentido numerico devuelve 4, ya que convierte 5 en numero  
conversions.push("5" - 1);  
  
// Convierte 5 a numero, devuelve 10 ya que * tiene sentido numero  
conversions.push("5" * 2);  
  
// Convierte 5 a numero por el signo /, resultado 2.5  
conversions.push("5" / 2);  
  
// % es aritmetico, por lo que convierte 5 a num y devuelve 1 al ser resto  
conversions.push("5" % 2);
```

```
// Coercion with == ////////////////////////////////////////  
  
// Devuelve true, al ser los operadores iguales  
conversions.push(5 == "5");  
  
// Devuelve true ya que 0 se convierte a false  
conversions.push(false == 0);  
  
// Devuelve true, son una excepcion especial  
conversions.push(null == undefined);  
  
// Devuelve true, ya que el array vacio se convierte "" --> 0 y false pasa a 0  
// 0 == 0 es true  
conversions.push([] == false);  
  
// Devuelve true, ya que el array vacio pasa de [] -> "" -> 0 y "" -> 0  
// 0 == 0 es true  
conversions.push([] == "");  
  
// El array devuelve la cadena "1, 2" y al ser igual a la otra cadena devuelve true  
conversions.push([1, 2] == "1,2");
```

```
// Coercion with === ////////////////////////////////////////  
  
// Devuelve false ya que uno es tipo numerico y otro de tipo cadena a pesar de ser iguales  
conversions.push(5 === "5");  
  
// Devuelve false ya que uno es tipo booleano y otro numerico  
conversions.push(false === 0);  
  
// Aqui devuelve false, ya que no hay coercion y son tipos distintos  
conversions.push(null === undefined);  
  
// Devuelve False, al ser el array objeto y false boolean  
conversions.push([] === false);  
  
// Devuelve False, al ser el array un objeto y "" string  
conversions.push([] === "");  
  
// Devuelve False, al ser el array un objeto y "1,2" un string  
conversions.push([1, 2] === "1,2");  
  
// Imprime la info  
console.log(conversions);  
})();  
</script>
```

2. Escribe un programa donde realices la prueba y documentación de 10 métodos del objeto String.

```
<script>

let cadena = "Hola ovni";

// 1. Funcion at() : Devuelve un caracter en una posicion especifica, acepta negativos

console.log("Funcion at(): ");
console.log("Primer caracter: " + cadena.at(0));
console.log("Ultimo carater: " + cadena.at(-1));

// 2. Funcion charAt() : Devuelve el caracter en un indice dado, no acepta negativos

console.log("Funcion charAt(): ");
console.log("Primer caracter: " + cadena.charAt(0));

// 3. Funcion concat() : Une dos o mas caracteres y devuelve una cadena
let color = "verde";

console.log("Funcion concat(): ");
console.log(cadena.concat( " ", color, " aspero"));

// 4. Funcion endsWith() : Verifica si la cadena termina con un valor específico.

console.log("Funcion endsWith(): ");
console.log( cadena.endsWith("verde")); // Falso
console.log( cadena.endsWith("ovni")); // Verdadero

// 5. Funcion includes() : Comprueba si la cadena contiene un valor / Sensible a may y min

console.log("Funcion includes(): ");
console.log( cadena.includes("Hola")); // Verdadero
console.log( cadena.includes("casa")); // Falso
```

```
// 6. Funcion indexOf() : Devuelve la posición de la primera aparición de un valor.

console.log("Funcion indexOf(): ");
console.log( cadena.indexOf("i"));
console.log( cadena.indexOf("v"));

// 7. Funcion length() : Devuelve la longitud de una cadena

console.log("Funcion length(): ");
console.log( cadena.length);

// 8. Funcion padEnd() : Rellena la cadena al final hasta una longitud específica.

console.log("Funcion padEnd(): ");
console.log( cadena.padEnd(cadena.length + 1, "."));

// 9. Funcion repeat() : Repite la cadena N veces.

console.log("Funcion repeat(): ");
console.log( cadena.repeat(3));

// 10 Funcion substring() : Extrae caracteres entre dos índices.

console.log("Funcion substring(): ");
console.log(cadena.substring(0, 4)); //Hola
```

3. Escribe una función que reciba como argumento una frase y devuelva la misma frase, pero con la inicial de cada palabra en mayúsculas:

Por ejemplo, `actividad4("te estas pasando juanan")` debería devolver la frase: Te Estas Pasando Juanan.

```
<script>
function actividad3(frase) {
    let caracteres = "";
    let esMayuscula = true; // Verdadero porque la primera letra lo es

    // Recorre la longitud de la frase
    for (let i = 0; i < frase.length; i++) {

        // Guarda el caracter segun la posicion de la frase
        let caracter = frase.charAt(i);

        // Comprueba si el caracter es un espacio
        if(caracter === " ") {

            // Guarda el caracter en la nueva frase de caracteres
            caracteres += caracter;

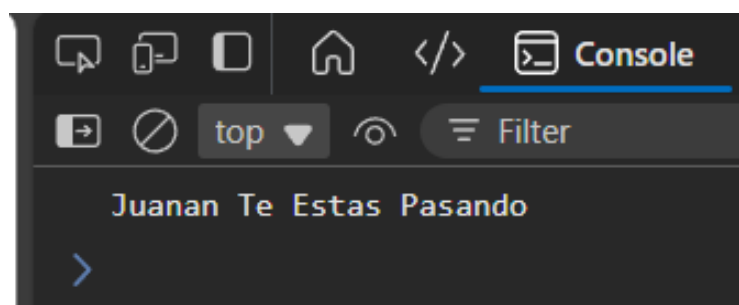
            // Cambia para que el siguiente caracter sea mayuscula
            esMayuscula = true;

            // Si es verdadero el caracter cambia a mayuscula y cambia a false
        } else if (esMayuscula ) {
            caracteres += caracter.toUpperCase();
            esMayuscula = false;

            //Controla que las demas letras excepto las primeras sean minusculas
        } else {
            caracteres += caracter.toLowerCase();
        }
    }

    // Devuelve la nueva frase reconstruida.
    return caracteres;
}

console.log(actividad3("juanan te estas pasando"));
</script>
```



4. Crea una función que descodifique un código de producto. El código contiene **tres partes** separadas por un guion (-):

- **Dos caracteres** → tipo de cliente:
  - a. CP : Cliente particular

- b. CE : Empresa
- **Dos dígitos** → ámbito:
  - c. 10: Local
  - d. 11: Autonómico
  - e. 12: Nacional
  - f. 20: Internacional
- **Una cifra** → número de años de antigüedad del cliente (0–9)
- Si el código es erróneo, la función debe **informar del tipo de error o errores**.

Por ejemplo, actividad5("CP-12-3") debería devolver Cliente particular nacional con 3 años de antigüedad.

```
<script>
function actividad4(codigo_producto) {
    let errores = "";
    let hayError = false;

    //Obtenemos la posición del primer guion y último, como separador
    let primer_guion = codigo_producto.indexOf("-");
    let ultimo_guion = codigo_producto.lastIndexOf("-");

    // Primer error, si no se incluye el separador "-"
    if(primer_guion == -1 || ultimo_guion == -1) {
        errores += "- El separador debe ser \"\\-\".\\n";
    }

    // Se extrae cada parte con substring
    let tipo = codigo_producto.substring(0, primer_guion);
    let ambito = codigo_producto.substring(primer_guion + 1, ultimo_guion);
    let antigüedad = codigo_producto.substring(ultimo_guion + 1);

    // ///////////////////////////////////
    let tipoCliente = "";

    // Comprueba el tipo de cliente
    if(tipo == "CP" ) tipoCliente = "Cliente particular";
    else if(tipo == "CE") tipoCliente = "Empresa";
    else errores += "- Tipo de cliente invalido, usar \"CP\" o \"CE\".\\n";
}
```

```
//Comprueba el tipo de ambito
switch (Number(ambito)) {
    case 10: tipoAmbito = "local"; break;
    case 11: tipoAmbito = "autonomico"; break;
    case 12: tipoAmbito = "nacional"; break;
    case 20: tipoAmbito = "internacional"; break;
    default: errores += "- Ambito no valido, usar 11, 12, 13, 20.\n"; break;
}

if(isNaN(Number(antiguedad))) errores += "- La antiguedad debe ser numerica.";

if(errores) return errores;
else return tipoCliente.concat(" ", tipoAmbito, " con ", antiguedad, " años de antiguedad." );
}

console.log(actividad4("CP-12-3"));
```

5. Diseña una función que sea capaz de encontrar los caracteres comunes entre dos palabras. La función recibe como argumentos dos palabras o frases y devuelve una cadena con los caracteres que haya en común entre ambas.
- Se desechan los espacios en blanco.
  - No se tiene en cuenta el caso (mayúsculas o minúsculas).

Por ejemplo, si llamamos a la función `actividad6("Ciudad","Cuando")` el resultado será "cud".

```
<script>
function actividad5(palabra1, palabra2) {
    let palabraNueva = "";

    //Le quitamos espacio y las mayusculas.
    palabra1 = palabra1.toLowerCase().trim();
    palabra2 = palabra2.toLowerCase().trim();

    // Esta palabra se va a consumir.
    palabra2V2 = palabra2;

    //Recorre la longitud de la primera palabra.
    for (let i = 0; i < palabra1.length; i++) {
        // Se obtiene el primer caracter de la palabra 1
        let caracter = palabra1.charAt(i);

        //Se obtiene la posición del primer caracter
        indice = palabra2V2.indexOf(caracter);

        // Solo si se encuentra la posición del caracter continua el flujo
        if(indice !== -1) {
            //Se añade la letra a la nueva palabra
            palabraNueva += caracter;

            //Se elimina el caracter usado, tomando todo lo anterior y después del índice, exceptuando la letra usada.
            palabra2V2 = palabra2V2.substring(0, indice) + palabra2V2.substring(indice + 1);
        }
    }
    return palabraNueva;
}

//Se llama a la función e imprime la nueva palabra formada.
console.log(actividad5("Ciudad", "Cuando"));
console.log(actividad5("bac", "abcd"));
</script>
```

Activar Windows  
Ve a Configuración para activar

6. Escribir una función JavaScript que acepte como argumento una cadena de caracteres y reúna los espacios repetidos en un solo espacio. Una especie de compresor de espacios.

Por ejemplo, tenemos la llamada `actividad7("JavaScript es muy fácil")`, deberá devolver "JavaScript es muy fácil".

```
<script>

function actividad6(texto) {
    // Se guarda el nuevo texto sin espacios duplicados
    let resultado = "";
    // Controlara el flujo de los espacios en el texto
    let hayEspacioAnterior = false;

    // Recorre el texto desde el inicio al final
    for (let i = 0; i < texto.length; i++) {
        // Se obtiene el caracter segun la posicion del bucle
        let caracter = texto.charAt(i);

        // Verifica si no es un espacio el caracter
        if(caracter !== " ") {
            // Se construye el texto sin espacios
            resultado += caracter;

            // al entrar aqui, ya no hay espacio anterior, por lo que cambia a false
            hayEspacioAnterior = false;
        } else if(!hayEspacioAnterior) {
            //Si el caracter actual es un espacio y el anterior no, se añade solo un espacio
            resultado += " ";
            //se marca que el ultimo caracter fue un espacio para ignorarlo en la siguiente
            hayEspacioAnterior = true;
        }
    }

    return resultado;
}

console.log(actividad6("JavaScript es muy fácil"));
```

