¿Qué diferencia a Javascript de cualquier otro lenguaje de programación?
JavaScript es el único lenguaje de programción que puede ser interpretado directamente por un navegador web. Además se caracteriza porque es un lenguaje orientado a objetos. Otra de las razones que lo hacen diferente es que la mayoría de las aplicaciones se han desarrollado con este lenguaje por lo que si quieres ser un desarrollador Web tu camino pasa por aprender este lenguaje de programación. Por último destacas que es uno de los lenguajes más potentes en cuanto a la automatización de procesos se refiere.

Entre los beneficios que presenta JavaScript se pueden encontrar lo siguientes:

- Es un lenguaje fácil de aprender y utilizar.
- Cuenta con una gran cantidad de librerías y frameworks.
- Permite trabajar con el formato de texto tipo JSON y contectarse a BBDD.
- Interecatividad en tiempo real.
- Detección y respuesta de eventos.
- Creación de formularios interactivos.
- Animación y efectos visuales para los sitios web.

Si quereis obtener más información sobre lo que es JavaScript y sus ventajas podeís consultar los siguientes enlaces:

https://blog.hubspot.es/website/que-es-javascript
https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript
https://blog.hubspot.es/website/ventajas-y-desventajas-de-javascript
https://ceeivalencia.emprenemjunts.es/?op=8&n=28660

¿Cuáles son algunos tipos de datos JS?
Algunos de los tipos de datos en JavaScript son Number (número entero), String (cadena de texto), Boolean (puede contener del valor True o False), Null (representa la ausencia de un valor intencionalemente), Undefined (representa una variable no delcarada o a la que no se le ha asignado un valor), etc.

Acontinuacion muestro ejemplos de estos tipos de datos:

```
numeroNumber = 4
otroNumeroNumber = 5.8
cadenaString = "Esto es una cadena o string"
valorBoolean = true
valorBoleanDos = false
variableSinValor = Null
var variableSinDefinir; // esto dará Undefined
```

Por otro lado tenemos algunos tipos de datos algo más complejos como son los objetos (contiene key/value pairs), en Python se llaman diccionarios. Después tenemos los arrrays que son los equivalente a las listas en Python. Lo siguiente es un ejemplo para cada uno de estos tipos de dato en JavaScript, tened en cuenta que los arrays son colecciones de datos

que pueden contener cualquier tipo de dato aunque lo común es almacenar datos del mismo tipo:

mi_array = [1, 4, 6, "Matias", 8, true, "Pedro"]
mi_objeto = {"nombre": "Lurdes", "edad": 16, "localidad": "Madrid"}

En los siguientes enlaces podéis obtener más información al respecto:

https://blog.hubspot.es/website/tipos-de-datos-javascript
https://developer.mozilla.org/es/docs/Web/JavaScript/Data_structures
https://es.javascript.info/types

¿Cuáles son las tres funciones de String en JS?

Algunas de las funciones para trabajar y modificar cadenas o String son:

- length() : sirve para conocer la longitud de una String.
- slice(indiceDondeEmpezamos, índiceHastaDondeTerminamos): sirve para obtener una porción de la String. Se le pasan dos parámetros númericos correspondientes al ínidice de donde empezamos a coger la cadena y el segundo hasta donde queremos cogerlo. Termian de coger la cadena en este segundo índice menos 1.
- replace(parámetroUno, parámetroDos) : sirve para remplazar algun elemento de la cadena por otro. Hay que pasarle dos parámetros, el primero la parte de la cadena que queremos remplazar y el segundo con lo que queremos sustituir.
- toUpper() : para cambiar las letras de la cadena a mayúsculas.
- toLower() : para cambiar las letras de la cadena a minúsculas.

La sintaxis de estas se aplica con un "." y despues el nombre de la función seguido de los paréntesis para hacer la llamada a ésta excepto en la función length. Voy a poner algunos ejemplos para enteder estas funciones con "console.log(loQueSeImprime)" para ver el resultado de cada función:

cadenaString = "El perro de mi vecino es muy simpático"

```
console.log(cadenaString.length)
console.log(cadenaString.slice(3, 8))
console.log(cadenaString.replace("perro", "gato"))
console.log(cadenaString.toUpperCase())
console.log(cadenaString.toLowerCase())
```

Resultado:
38
perro
El gato de mi vecino es muy simpático

EL PERRO DE MI VECINO ES MUY SIMPÁTICO
el perro de mi vecino es muy simpático

Encontraréis más información sobre estas funciones y otras en los siguientes enlaces:
https://desarrolloweb.com/articulos/objetos-string-javascript.html
https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Useful_string_methods

¿Qué es un condicional?

Si los programas fueran simplemente líneas de código que se ejecutan una tras otra estaríamos muy limitados en cuanto a la flexibilidad por ello todos los lenguajes de programación incluyen la posibilidad de ejecutar código sujeto a condiciones.

Se deben tener en cuenta los siguientes aspectos para componer un condicional:

- La indentación: se trata del nivel o tabulación al que se ha escrito el código e indica si una o varias líneas de código se encuentran enlazadas con el condicional.
- La sintaxsis: en JavaScript se utilizan las palabras: if, else if y else. Las dos primeras llevan asociadas una o varias condiciones y el “else” sirve para ejecutar el código que esté relacionado a éste si todas las condiciones asociadas anteriores no se han cumplido. Después de las palabras "if" y "else if" se escriben los paréntesis para poner la condición y después dentro de las llaves "{}" el código que queremos ejecutar si la condición/ones dan true. Sin embargo en el “else” escribiremos “esle{}” y dentro de la llaves el código a ejecutar.
- Operadores lógicos and y or: && y ||. El “&&” corresponde al "and" de Python y sirve para enlazar varias condiciones. Para que el resultado sea true todas tienen que dar como resultado true. Con el operador “||” que corresponde al "or" de Python con que una de las condiciones sea true basta para que el resultado sea true.. A continuación, expongo ejemplos de varios casos posibles y sus resultados utilizando los operadores lógicos, la primera línea es la línea de código ejecutada y la segunda el resultado obtenido:

console.log(true && true)

True

console.log(true && false)

False

console.log(true || false)

True

console.log(false || false)

False

A continuación, muestro un par de ejemplos para comprender como funcionan los condicionales utilizando JavaScript:

```
a = 55
b = 67
if (b > a){
    console.log('b es más grande que a!')
}
```

Resultado:

b es más grande que a!

En este caso el resultado de la condición es true porque el número contenido en la variable b es mayor que el número en la variable a, por lo tanto, la línea del “console.log…” la cual está relacionada a ese “if” gracias a la indentación (tabulación) se ejecuta.

```
a = 47
b = 30
if (b > a){
   console.log('b es más grande que a!')
} else {
   console.log('a es más grande que b! ')
}
```

Resultado:

a es más grande que b!

En este caso el resultado de la condición del “if” es false y esto hace que el código dentro de ese condicional no se ejecute, pero al haber un “else” el programa ejecuta la línea contenida en el mismo.

Tenéis disponible una documentación más completa en los siguientes enlaces:
https://developer.mozilla.org/es/docs/Learn/JavaScript/Building_blocks/conditionals
https://makeitrealcamp.gitbook.io/javascript-book/condicionales
https://lenguajejs.com/fundamentos/estructuras-de-control/condicionales-if-else/

¿Qué es un operador ternario?

El operador ternario se utiliza para simplificar la instrucción de un "if". Contiene tres operandos, el primero la condición, el segundo es lo que devuelve si la condición es true y el tercero lo devuelve si la condición es false. La sintáxis es la siguiente:

condición ? expr1 : expr2

Los siguientes son ejemplos sencillos de operador ternario:

```
console.log("La Cuota es de:  " + (true? "2.00€" : "10.00€"));
```
Resultado:
La Cuota es de: 2.00€

```
console.log("La Cuota es de:  " + (false? "2.00€" : "10.00€"));
```
Resultado:
La Cuota es de: 10.00€

```
var einsteinLives = Math.PI > 4 ? "Si" : "No";
console.log(einsteinLives)
```
Resultado:
No

```
var einsteinLives = Math.PI < 4 ? "Si" : "No";
console.log(einsteinLives)
```
Resultado:
Si

Tenéis más información sobre operadores ternarios en los siguientes enlaces:
https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Conditional_operator
https://lenguajejs.com/fundamentos/estructuras-de-control/operador-ternario/
https://keepcoding.io/blog/operadores-ternarios-en-javascript/

¿Cuál es la diferencia entre una declaración de función y una expresión de función?

Las declaraciones de funciones se crean directamente con la palabra clave "function" seguido del nombre de la función, los paréntesis con o sin parámetros y las llaves conteniendo el código a ejecutar en la función. Las expresiones de función se crean asignado el código de la función a una variable con el símbolo "=".

Ejemplo declarción de función:

```
function suma(a, b) {
        return a + b
}
console.log(suma(2,3))
```

Resultado:
5

Ejemplo expresión de función:

```
var sumaEx = function(a, b) {
   return a + b
}

console.log(sumaEx(4,5))
```

Resultado:
9

Para entender otra diferencia entre las declaraciones y las epxresiones de función necesitamos hablar del hoisting. El hoisting lo que permite es usar variables y funciones antes de que se hayan declarado. JavaScript funciona declarando todas las variables y funciones primero antes de asignar y ejecutar cualquier otro valor. Por ello podriamos hacer lo siguiente con una declaración de función pero no nos lo permitiría con una expresión de función:

```
// declaración de funcion:
console.log(suma(2,3))
function suma(a, b) {
        return a + b
}
```

Resultado:
5

Sin embargo lo siguiente nos daría un error:

```
// expresión de función:
console.log(sumaEx(4,5))
var sumaEx = function(a, b) {
   return a + b
}
```

Resultado:
TypeError: sumaEx is not a function

Podeís consultar más ejemplos e información sobre la declaración y la expresión de funciones en los siguientes links:

https://blog.koalite.com/2011/10/javascript-diferencias-entre-declaracion-de-funcion-y-expresion-con-funcion/
https://www.youtube.com/watch?v=jFzE36QYKD0

¿Qué es la palabra clave "this" en JS?

La palabra clave "this" se utiliza en multiples lenguajes de programación pero voy a explicaros como de que se trata y como funciona en JavaScript.

This en JavaScript es una palabra clave muy utilizada dentro de funciones y clases, pues tiene un valor flexible. *This* significa *esto* en español y, como su nombre indica, hace referencia al objeto en cuestión. Es decir, si estamos creando cualquier función, la palabra clave *this* se usará para representar o llamar al objeto que dicha función está modificando.
A continuación, os expongo un ejemplo con una variable llamada *ejemplo*:

```
const ejemplo = {
color: rojo,
func: function ( ) {
return this.color;
},
};
```

Esta variable tiene dos propiedades, *function* y *color.* Aquí, *this* hace referencia a la constante *ejemplo*, pues es el objeto al que pertenece en este contexto. Entonces, al hacer que la función nos devuelva la propiedad *this.color*, el programa entiende que lo que hará realmente es devolvernos la propiedad *ejemplo.color.*

```
console.log (ejemplo.func());
```
Resultado: rojo

Os dejo algunos recursos para ampliar y profundizar en el funcionamiento de la palabra "this":
https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/this
https://www.freecodecamp.org/espanol/news/la-guia-completa-sobre-this-en-javascript/