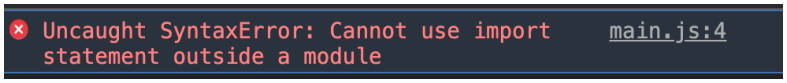
HINTS

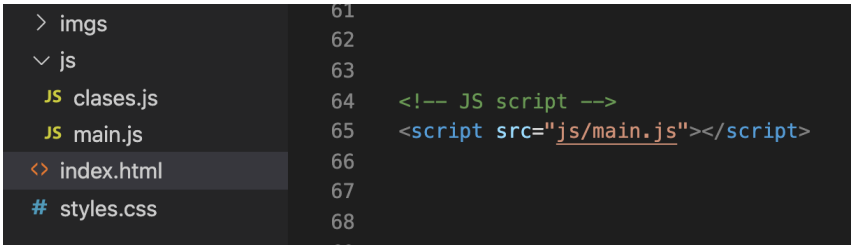
TYPE = "MODULE"

Con ES6 se introduce el concepto de módulos, el cual nos permite integrar código de distintos archivos JavaScript, para utilizar todo nuestro código de manera uniforme. Si bien esta herramienta es muy útil, nos puede causar frustración si al usarla nos encontramos con el siguiente error:



Este error se puede producir por un número de razones, donde la más común de todas es que en nuestro HTML no hemos definido el atributo módulo, de la referencia a nuestro script. ¿A qué nos referimos con esto?

Supongamos que nuestro Proyecto tiene la siguiente estructura:



Al lado izquierdo, vemos la estructura del Proyecto, que contiene una carpeta donde almacenamos nuestros archivos de JavaScript, una carpeta de imágenes (*imgs*), y los archivos HTML y CSS. A la derecha, vemos en nuestro HTML la referencia hacia el archivo *main.js* ubicado en la carpeta *js*.

Si tenemos nuestra referencia de esta forma, nos encontraremos con el error que mencionamos previamente, dado que es en esta etiqueta script donde debemos definir que el archivo *main.js* es de tipo "módulo". Esto se hace de la siguiente forma:

Al especificar el atributo **type="module"**, lograremos solucionar el error visto al inicio y podremos desarrollar usando módulos sin ningún problema.

CREACIÓN DE OBJETOS LITERALES

Los objetos literales son un conjunto de propiedades compuestas de una llave y un valor, y separados por dos puntos (:).

Para crearlos utilizamos las llaves ({ }), y dentro de ellas, las propiedades separadas por una coma.

```
let auto = {
    marca: 'Suzuki',
    modelo: 808,
    nuevo: true
}
```

En el ejemplo se aprecian tres propiedades: la llave “marca”, con valor “Suzuki”; la llave “modelo”, con valor “808”; y la llave “nuevo”, con valor “true”.

LA PROPIEDAD PROTOTIPO PARA DEFINIR MÉTODOS

JavaScript es un lenguaje basado en prototipos, por lo tanto, cada vez que se crea un objeto, se le añade una propiedad por defecto llamada “prototype”.

La propiedad “prototype” es un objeto que contiene una propiedad “constructor”. Al llamar a un objeto con la palabra reservada “new”, los objetos recién creados heredarán todas las propiedades del prototipo.

Veamos un ejemplo.

Primero crearemos un prototipo de un objeto, usando la función constructora.

```
function Auto (marca, modelo, nuevo){
    this.marca = marca;
    this.modelo = modelo;
    this.nuevo = nuevo;

    this.encender = function(){
        return "Se enciende el auto "+this.marca;
    }
}
```

Instanciamos dos autos, e imprimimos en nuestra consola.

```
var suzuki = new Auto("Suzuki", 808, true);
var toyota = new Auto("Toyota", 44, false);
```

```
▶ Auto {marca: 'Suzuki', modelo: 808, nuevo: true, encender: f}
▶ Auto {marca: 'Toyota', modelo: 44, nuevo: false, encender: f}
```

Añadimos una propiedad al primer auto, e imprimimos nuevamente.

```
suzuki.numPuertas = 4;

console.log(suzuki.numPuertas);
console.log(toyota.numPuertas);
```

```
4
undefined
```

Como podemos observar, el segundo auto nos retorna “undefined”, pues estamos agregando la propiedad al objeto y no al prototipo (function Auto).

Para agregar propiedades al prototipo, basta con escribir “prototype” al nombre del objeto, seguido del nombre del atributo o método.

```
Auto.prototype.color;

suzuki.color = "Rojo";
toyota.color = "Negro";


console.log(suzuki.color);
console.log(toyota.color);
```

```
Rojo
Negro
```