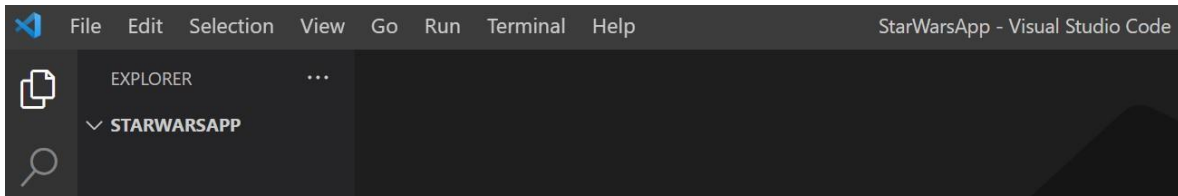


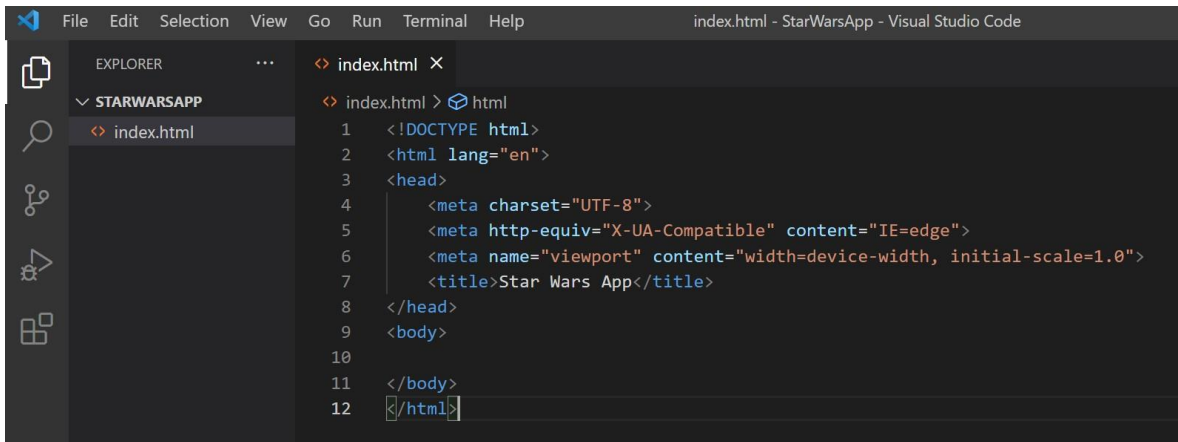
## STAR WARS APP

Empecemos

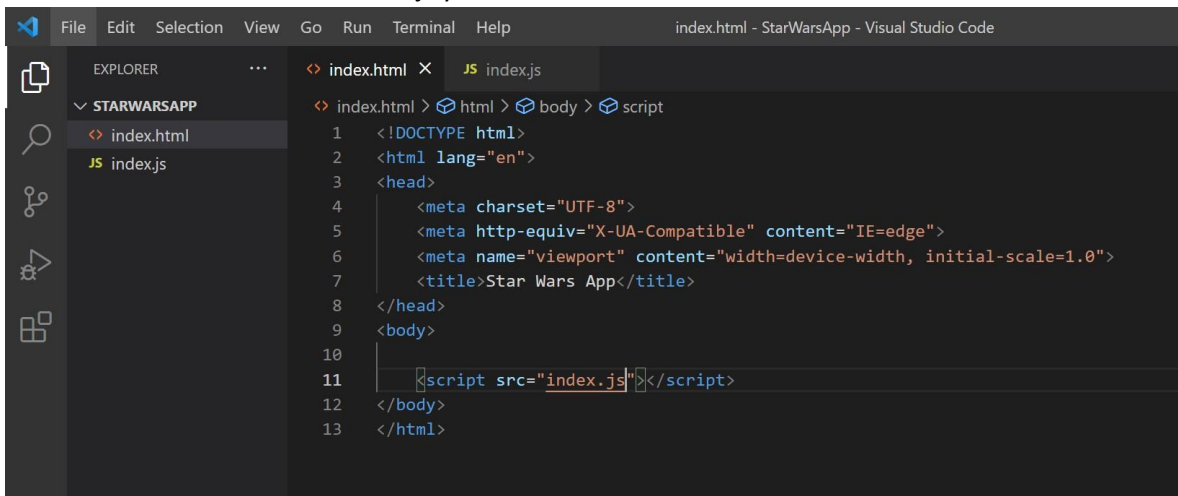
Abrir desde VSC (Visual Studio Code) nuestra carpeta de trabajo



Dentro de nuestra carpeta de trabajo creamos un archivo llamado index.html e insertamos una estructura base de html



Creamos un archivo llamado index.js y lo vinculamos a nuestro archivo index.html

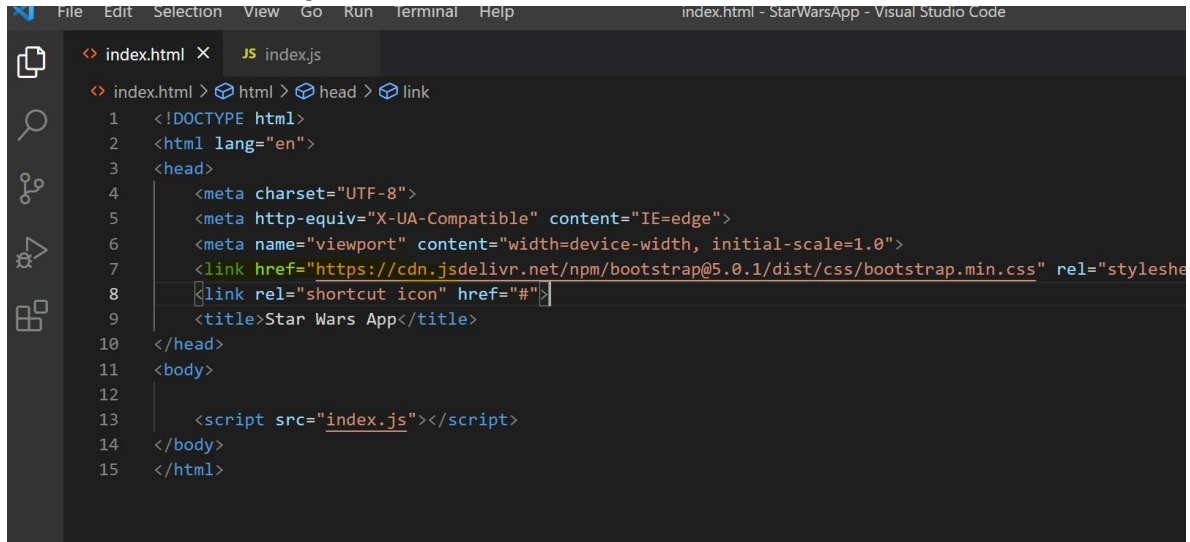


Utilizaremos los CDN de bootstrap. Estos CDN los copiamos y pegamos en el head de nuestro archivo index.html

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-+0n0xVW2eSR5OmGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOO17+AMvyTG2x" crossorigin="anonymous">
```

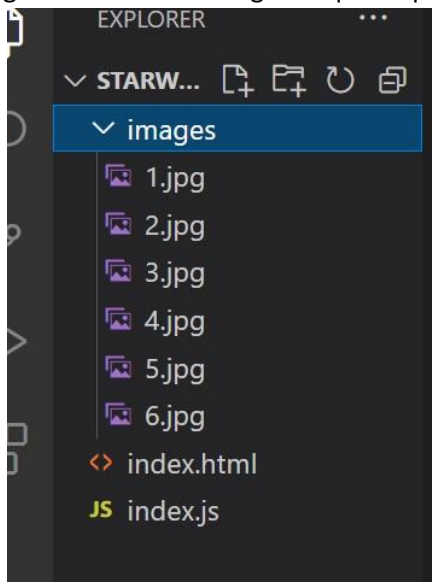
```
<link rel="shortcut icon" href="#">
```

Se debe visualizar de la siguiente manera



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet">
8   <link rel="shortcut icon" href="#">
9   <title>Star Wars App</title>
10 </head>
11 <body>
12
13   <script src="index.js"></script>
14 </body>
15 </html>
```

Dentro de nuestra estructura de trabajo creamos una carpeta llamada images y dentro de ella guardaremos las imágenes que requerimos

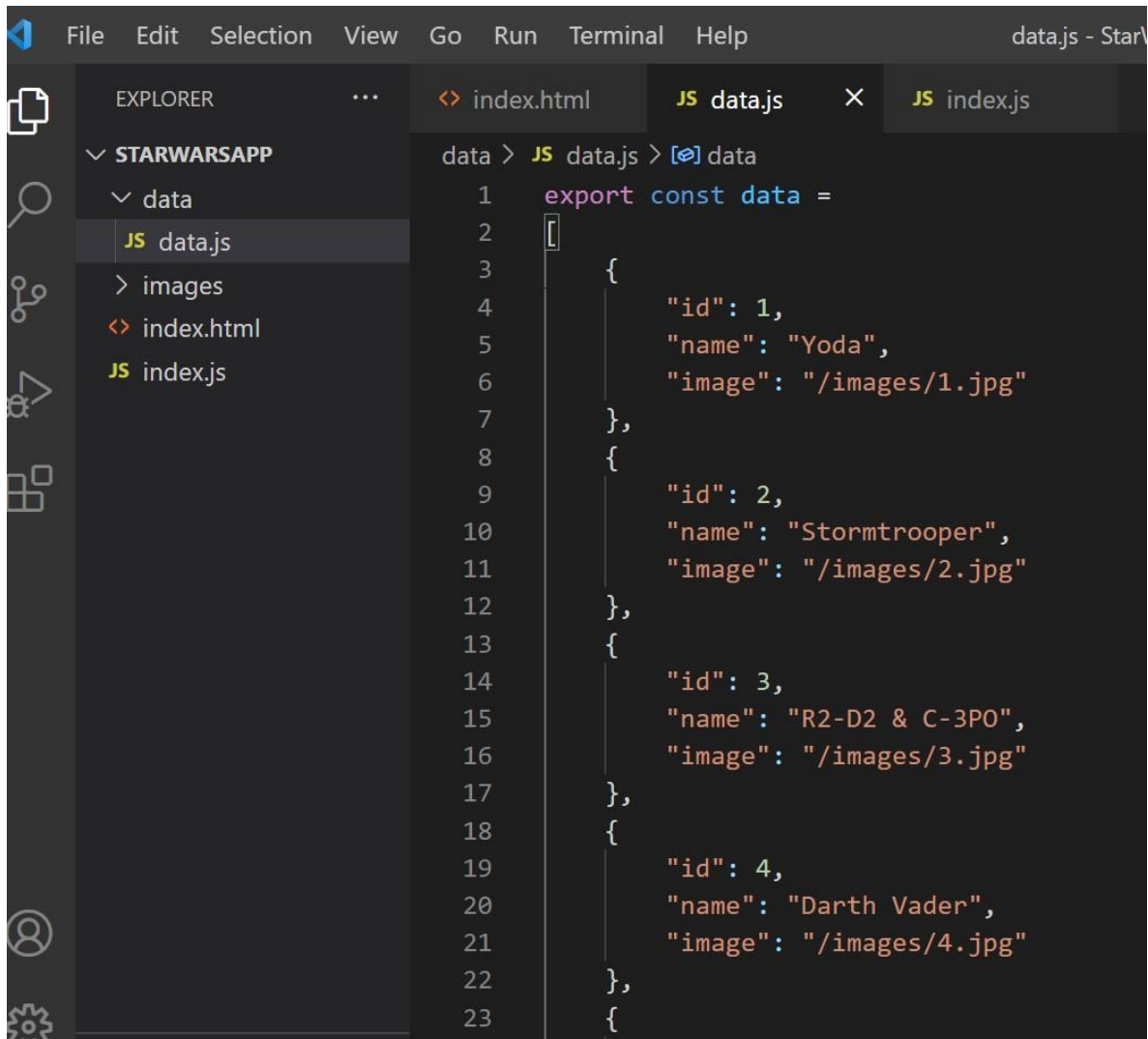


Dentro de nuestra estructura de trabajo creamos una carpeta llamada data y dentro de ella creamos un archivo llamado data.js el cual tendrá la siguiente estructura

```
export const data =
[
  {
    "id": 1,
    "name": "Yoda",
    "image": "/images/1.jpg"
  },
]
```

```
{
  "id": 2,
  "name": "Stormtrooper",
  "image": "/images/2.jpg"
},
{
  "id": 3,
  "name": "R2-D2 & C-3PO",
  "image": "/images/3.jpg"
},
{
  "id": 4,
  "name": "Darth Vader",
  "image": "/images/4.jpg"
},
{
  "id": 5,
  "name": "Obi-Wan Kenobi",
  "image": "/images/5.jpg"
}
,
{
  "id": 6,
  "name": "Chewbacca",
  "image": "/images/6.jpg"
}
]
```

Se debe visualizar de la siguiente manera



```
data > JS data.js > data
1  export const data =
2  [
3      {
4          "id": 1,
5          "name": "Yoda",
6          "image": "/images/1.jpg"
7      },
8      {
9          "id": 2,
10         "name": "Stormtrooper",
11         "image": "/images/2.jpg"
12     },
13     {
14         "id": 3,
15         "name": "R2-D2 & C-3PO",
16         "image": "/images/3.jpg"
17     },
18     {
19         "id": 4,
20         "name": "Darth Vader",
21         "image": "/images/4.jpg"
22     },
23     {
```

Esta data la debemos exportar e importarla en el archivo index.js

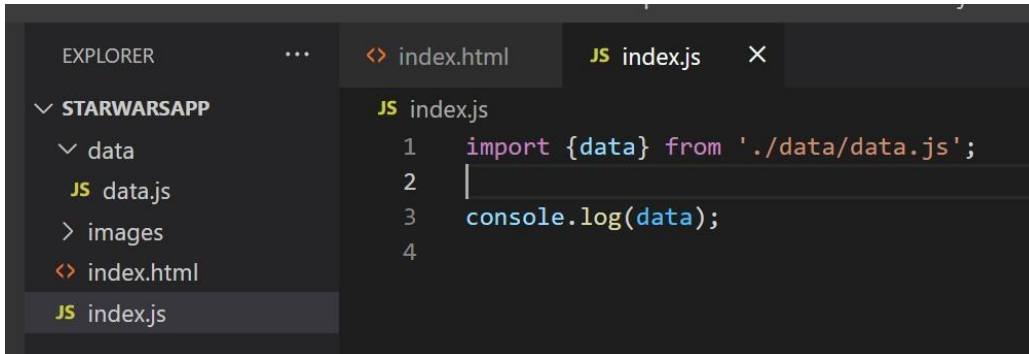
Realicemos lo siguiente

En nuestro archivo index.html relacionamos nuestro archivo data y tanto al archivo data como a index.js les indicamos type="module"



```
<body>
  <script src="index.js" type="module"></script>
  <script src="/data/data.js" type="module"></script>
</body>
</html>
```

Dentro del archivo index.js importaremos el archivo data y verificaremos por consola la obtención de la data



The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows a project named 'STARWARSAPP' with subfolders 'data' and 'images', and files 'data.js', 'index.html', and 'index.js'. The 'index.js' file is selected. The main editor shows the content of 'index.js' with the following code:

```
1 import {data} from './data/data.js';
2
3 console.log(data);
4
```

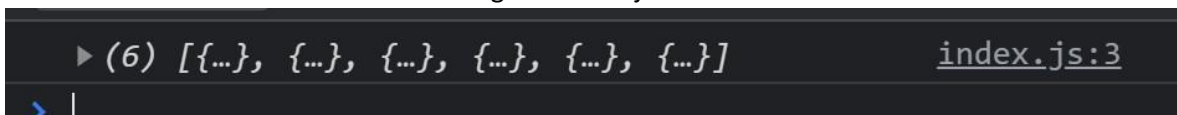
Ejecutamos



The screenshot shows the VS Code interface with the 'index.html' file open. A context menu is visible over the file, with options like 'Open with Live Server', 'Open to the Side', 'Open With...', 'Reveal in File Explorer', and 'Open in Integrated Terminal'. The 'index.html' file content is visible in the background:

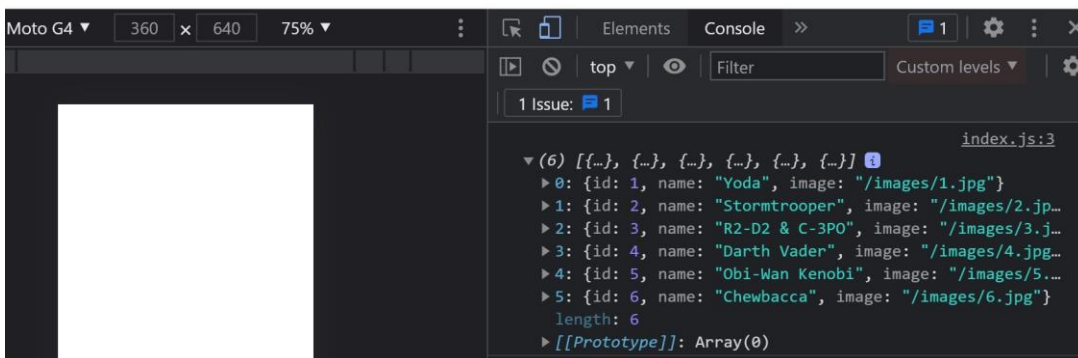
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
```

En la consola nos debe de mostrar el arreglo con 6 objetos



The screenshot shows the VS Code console with the following output:

```
(6) [{...}, {...}, {...}, {...}, {...}, {...}] index.js:3
```



The screenshot shows the VS Code console with the following output:

```
(6) [{...}, {...}, {...}, {...}, {...}, {...}] index.js:3
  ▶ 0: {id: 1, name: "Yoda", image: "/images/1.jpg"}
  ▶ 1: {id: 2, name: "Stormtrooper", image: "/images/2.jp...
  ▶ 2: {id: 3, name: "R2-D2 & C-3PO", image: "/images/3.j...
  ▶ 3: {id: 4, name: "Darth Vader", image: "/images/4.jpg...
  ▶ 4: {id: 5, name: "Obi-Wan Kenobi", image: "/images/5...
  ▶ 5: {id: 6, name: "Chewbacca", image: "/images/6.jpg"}
  length: 6
  ▶ [[Prototype]]: Array(0)
```

Ya que tenemos nuestra data, empecemos

Pintar data en pantalla

Dentro del archivo index.html crearemos una división con clase tipo contenedor. Dentro de esta división, creamos otra con un id llamado ítems. Debajo de esta división (no dentro) creamos un template



The screenshot shows the content of the 'index.html' file with the following HTML structure:

```
<div class="container">
  <h1>Star Wars</h1>
  <hr>
```

```

<div class="row" id="items"></div>

<template id="template-card">
  <div class="col-12 mb-2 col-md-4">
    <div class="card">
      <div class="card-body">
        <h5></h5>
        <img src="" alt="" class="img-thumbnail my-4"></img>
        <button type="button" class="btn btn-dark">
          Like
        </button><span></span>
        <label id="like">.</label>
      </div>
    </div>
  </div>
</template>
</div>

```

Se debe visualizar de la siguiente manera

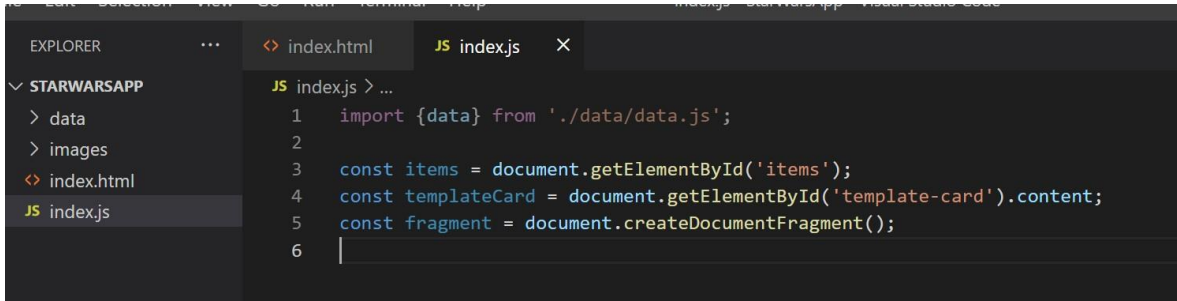
```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="UTF-8">
6    <meta http-equiv="X-UA-Compatible" content="IE=edge">
7    <meta name="viewport" content="width=device-width, initial-scale=1.0">
8    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css"
9          integrity="sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOO17+AMvyTG
10         <link rel="shortcut icon" href="#">
11    <title>Star Wars App</title>
12  </head>
13
14  <body>
15
16    <div class="container">
17      <h1>Star Wars</h1>
18      <hr>
19
20      <div class="row" id="items"></div>
21
22      <template id="template-card">
23        <div class="col-12 mb-2 col-md-4">
24          <div class="card">
25            <div class="card-body">
26              <h5></h5>
27              <img src="" alt="" class="img-thumbnail my-4"></img>
28              <button type="button" class="btn btn-dark">
29                Like
30              </button><span></span>

```

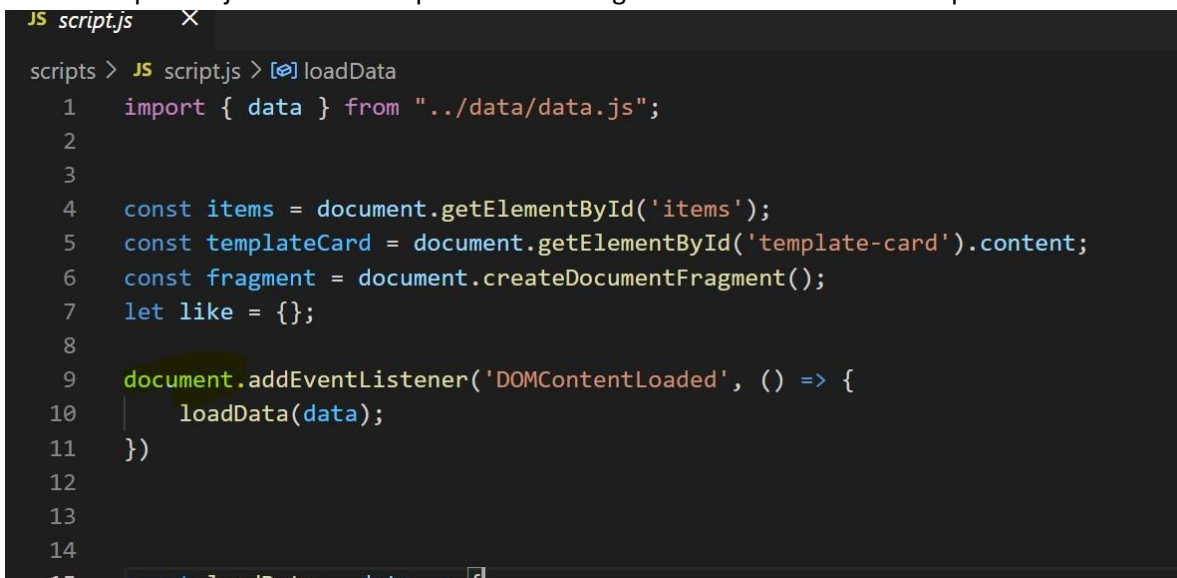
Nos ubicamos en el archivo index.js

Capturamos algunos elementos desde el html, tales como: la división con id ítems y el template con id template-card. Adicionalmente, añadiremos un fragmento



```
EXPLORER    ...    <> index.html    JS index.js    X
STARWARSAPP
  > data
  > images
  <> index.html
  JS index.js
JS index.js > ...
1  import {data} from './data/data.js';
2
3  const items = document.getElementById('items');
4  const templateCard = document.getElementById('template-card').content;
5  const fragment = document.createDocumentFragment();
6  |
```

Crearemos nuestra función para cargar la data y llamamos al DOMContentLoaded para al momento que se ejecute nuestra aplicación nos cargue la data en la estructura que definiremos



```
JS script.js    X
scripts > JS script.js > [loadData]
1  import { data } from "../data/data.js";
2
3
4  const items = document.getElementById('items');
5  const templateCard = document.getElementById('template-card').content;
6  const fragment = document.createDocumentFragment();
7  let like = {};
8
9  document.addEventListener('DOMContentLoaded', () => {
10 |   loadData(data);
11 | })
12
13
14
15
```

La función la llamaremos loadData y recibirá como parámetro la data que importamos desde el archivo data.js

Esta función recibirá la data y la debemos recorrer, lo haremos con un foreach. Cuando ingresemos al foreach y nos quede el objeto, lo desestructuraremos y asignamos a la estructura del template los atributos. Clonamos los nodos del template card, empezamos a adicionar los nodos al fragmento (uno tras otro) y luego mostramos el fragmento en la división con id ítems

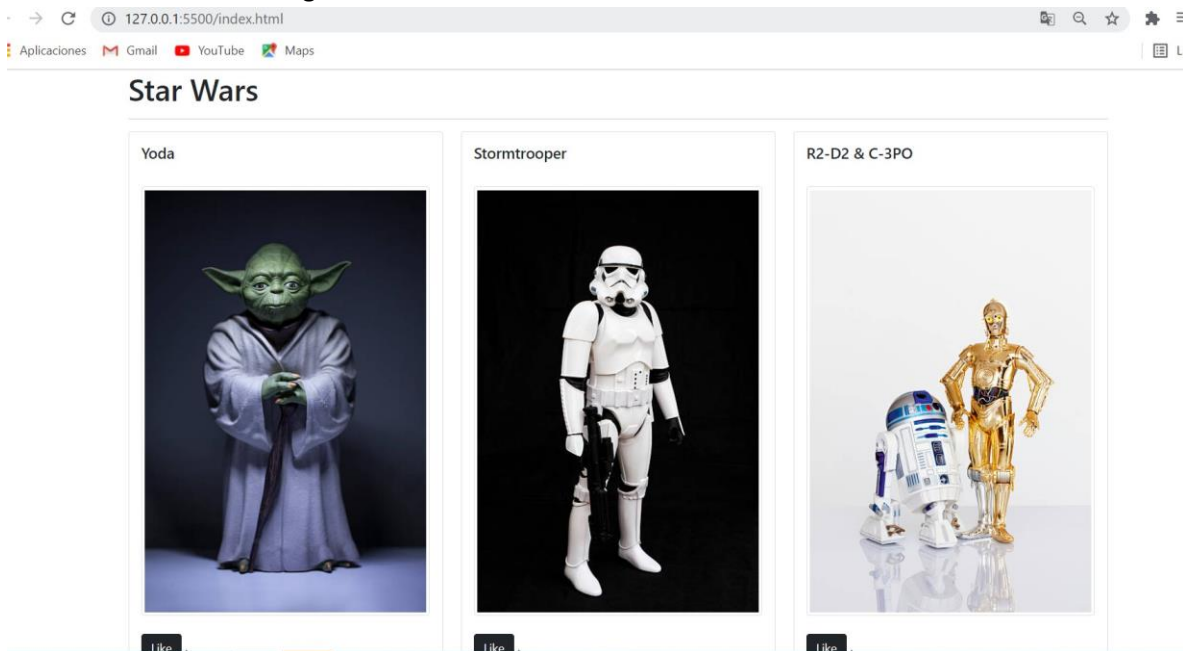


```

8  document.addEventListener('DOMContentLoaded', () => {
9      loadData(data);
10 })
11
12
13 const loadData = data => {
14     data.forEach(personaje => {
15         const { id, name, image } = personaje;
16         templateCard.querySelector('h5').textContent = name;
17         templateCard.querySelector('img').setAttribute('src', image);
18         templateCard.querySelector('.btn-dark').dataset.id = id;
19         const clone = templateCard.cloneNode(true);
20         fragment.appendChild(clone);
21     })
22     // appendChild agrega un nodo al final de la lista
23     items.appendChild(fragment);
24 }

```

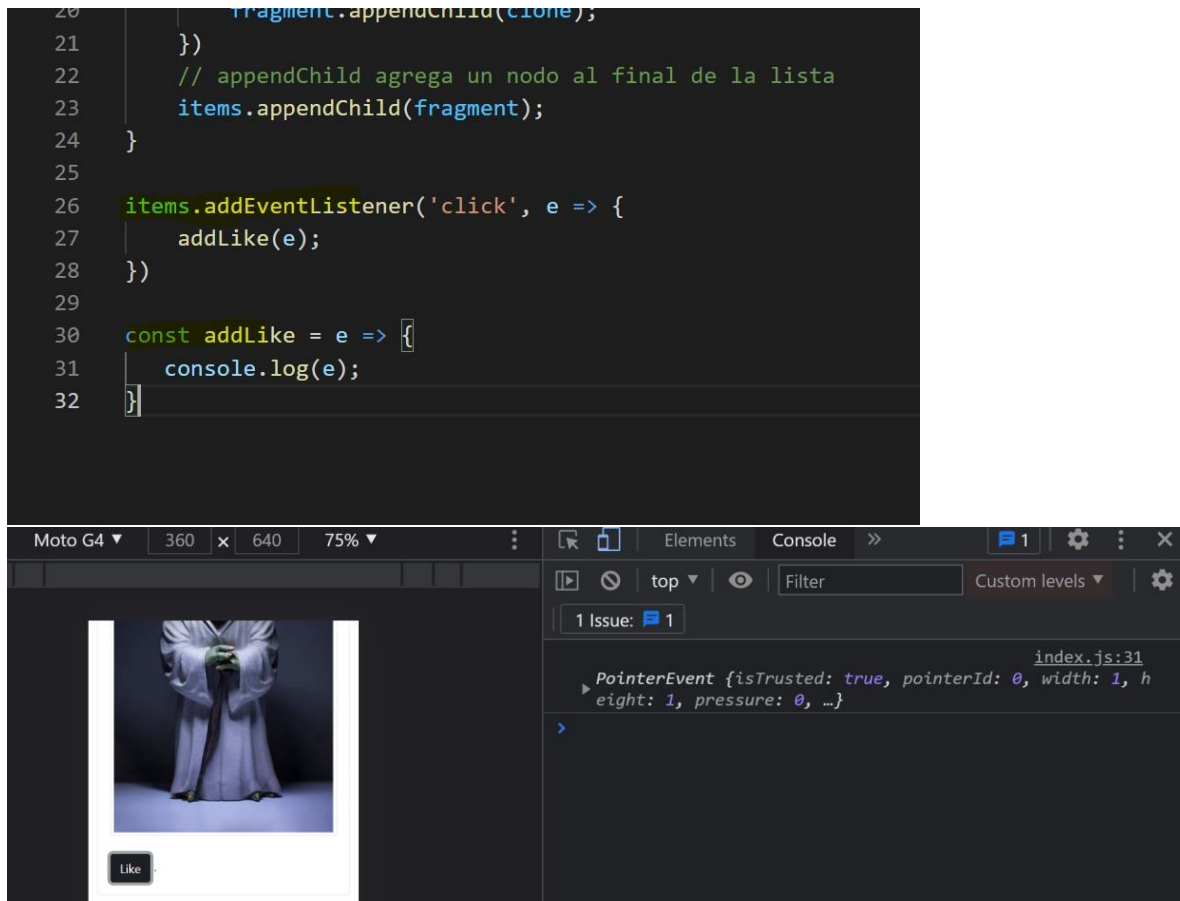
Se debe visualizar de la siguiente manera



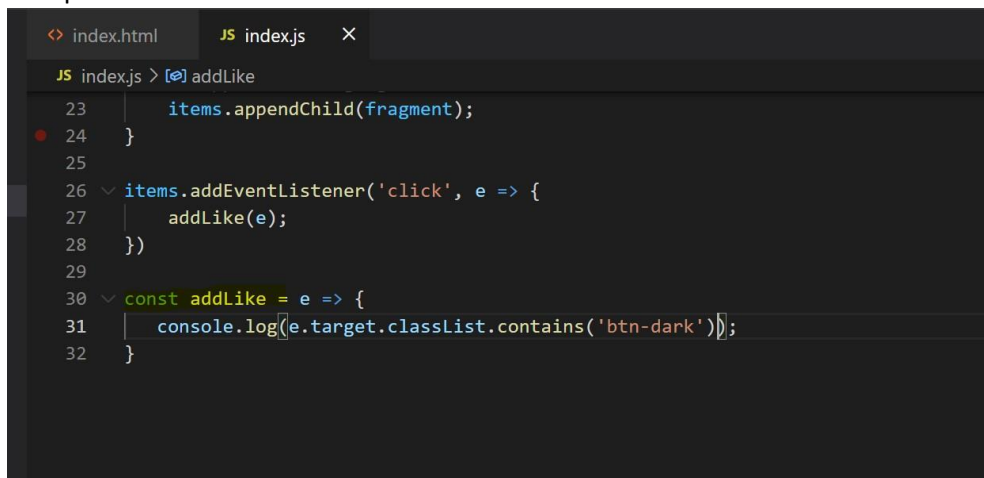
En la función loadData vemos que al botón le asignamos el id de la data, esto es con el fin de saber a qué id le vamos a asignar el like y se modifique solo en uno y no en todas las tarjetas

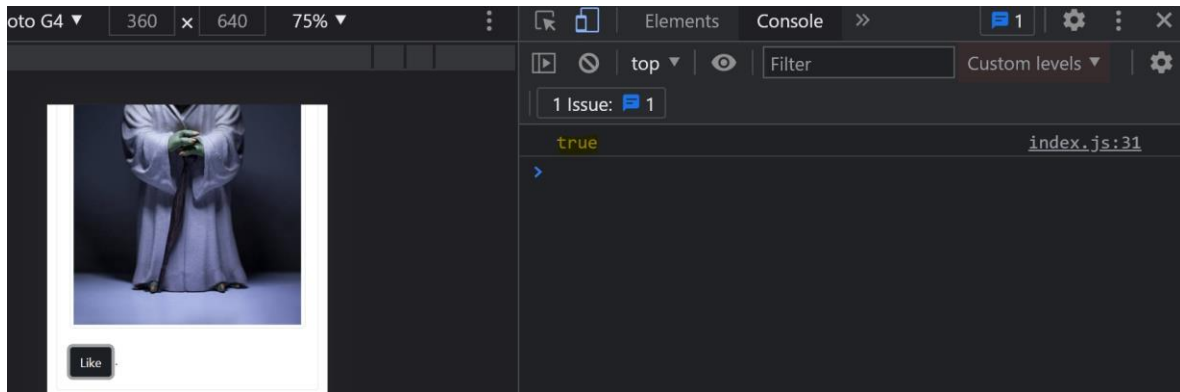
Vamos a llamar al evento clic del botón like y crearemos una función que se llame addlike la cual capturará el evento





Debemos de validar que seleccionemos el botón y no otra parte de la tarjeta y realizaremos con `contains`. El `e.target.classList.contains` nos devuelve un `true` o un `false`. Si hacemos clic en el botón `Like` que se llama `btn-dark` no devolvería un `true`





Si nos devuelve true, llamaremos una función que llamaremos setLike y le enviaremos toda la tarjeta completa del objeto que seleccionó y esto lo haremos con parentElements

```

APP      JS index.js > [e] addLike
ml
23     items.appendChild(fragment);
24   }
25
26   items.addEventListener('click', e => {
27     addLike(e);
28   })
29
30   const addLike = e => {
31     //que contenga btn dark y devuelve true
32     if (e.target.classList.contains('btn-dark')) {
33       //captura todos los elementos de la target
34       setLike(e.target.parentElement);
35     }
36   }

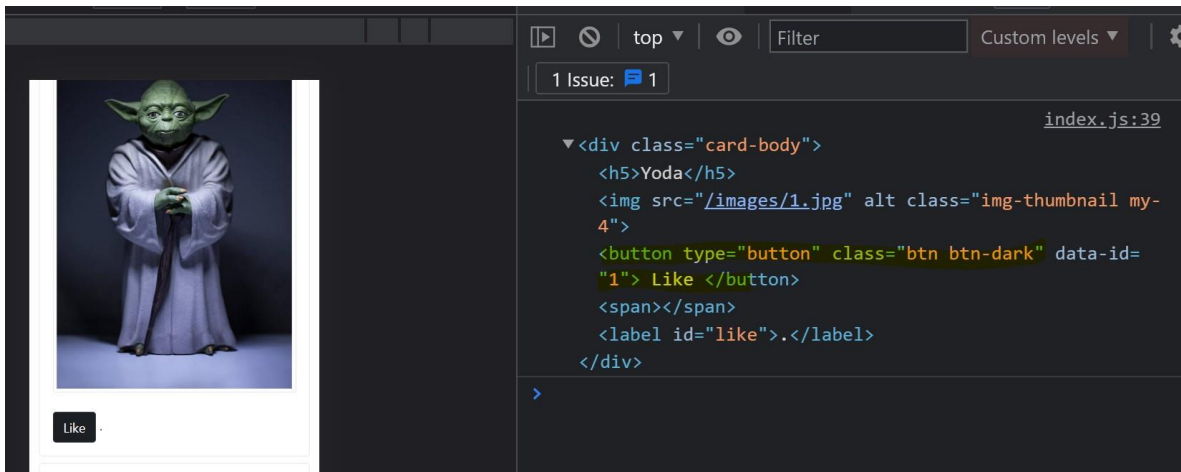
```

SetLike recibirá el objeto y luego de ese objeto capturaremos el id correspondiente

```

33     //captura todos los elementos de la target
34     setLike(e.target.parentElement);
35   }
36 }
37
38 const setLike = objeto =>{
39   console.log(objeto);
40 }

```



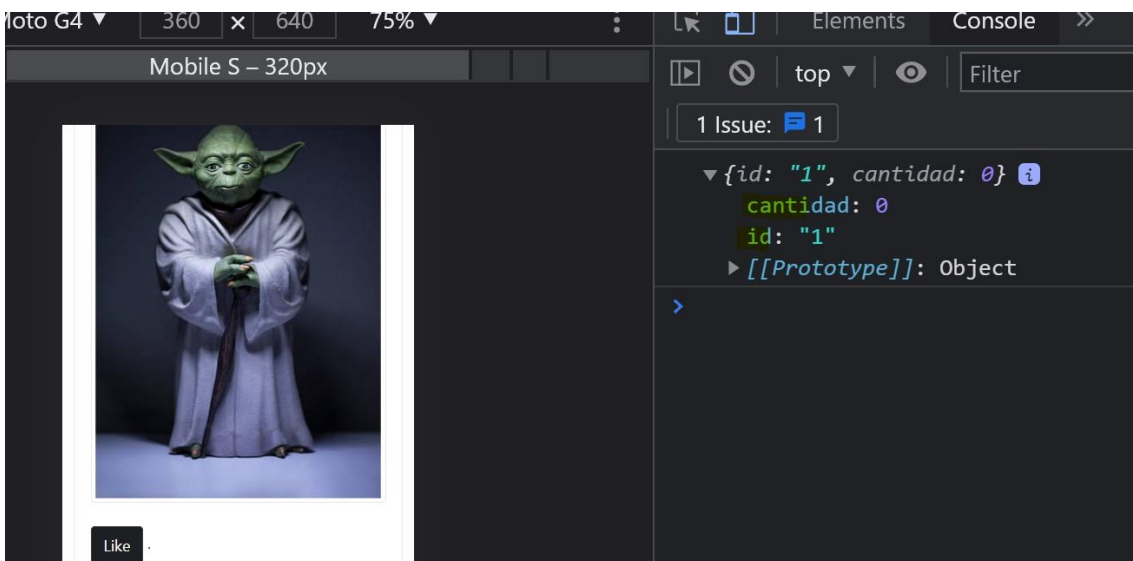
Capturaremos el botón btn dark, su data-id

Esto lo adicionaremos dentro de un objeto y crearemos una propiedad para almacenar la cantidad

```

36   }
37
38   <v> const setLike = objeto =>{
39   <v>     const boton = {
40         id: objeto.querySelector('.btn-dark').dataset.id,
41         cantidad: 0
42     }
43
44     console.log(boton);
45 }

```



Creamos un arreglo llamado like y que inicie vacío

```

JS index.js > [⌘] setLike
1  import { data } from './data/
2
3  const items = document.getEle
4  const templateCard = document
5  const fragment = document.cre
6  let like = [];
7
8  document.addEventListener('DC
9  |   loadData(data);
10 }
11

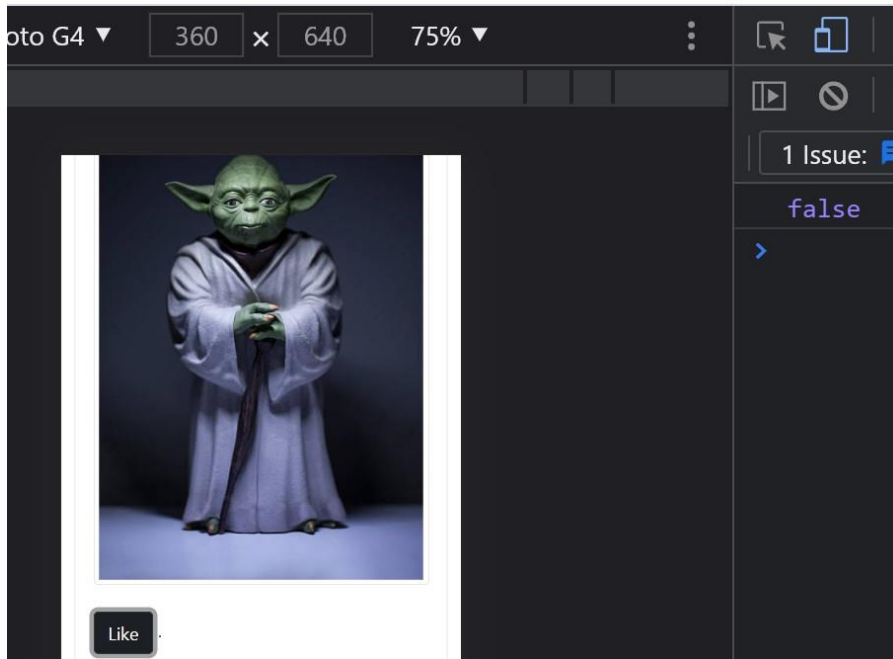
```

hasOwnProperty es un booleano que nos permite validar si un objeto tiene la propiedad que le estamos especificando.

```

... index.html JS index.js X
JS index.js > ...
33   (e.target.classList.contains('btn-dark')) {
34       //captura todos los elementos de la target
35       setLike(e.target.parentElement);
36   }
37
38   const setLike = objeto =>{
39       const boton = {
40           id: objeto.querySelector('.btn-dark').dataset.id,
41           cantidad: 0
42       }
43
44
45       console.log(like.hasOwnProperty(boton.id));
46   }

```

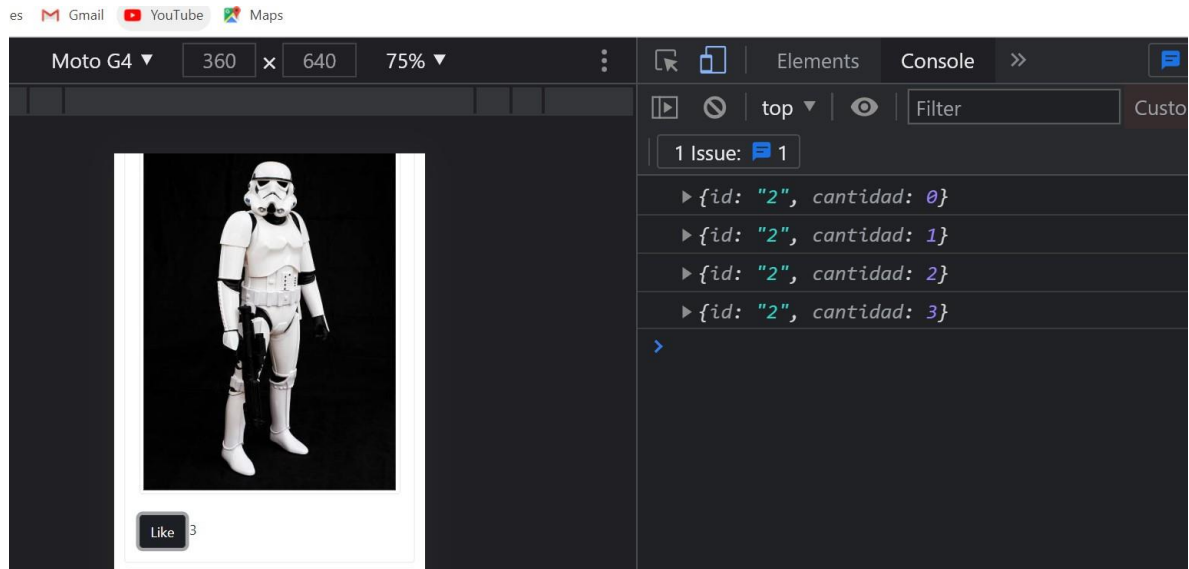


En este caso nos devuelve un false y como la tiene se la vamos a establecer. Adicionamos la cantidad y la mostramos en el label

```
40     id: objeto.querySelector('.btn-dark').dataset.id,  
41     cantidad: 0  
42   }  
43  
44   if (like.hasOwnProperty(boton.id)) {  
45     boton.cantidad = like[boton.id].cantidad + 1;  
46     objeto.querySelector('#like').textContent = boton.cantidad;  
47   }  
48
```

Llevamos el spread del botón a la propiedad que especificamos en el array like

```
45     boton.cantidad = like[boton.id].cantidad + 1;  
46     objeto.querySelector('#like').textContent = boton.cantidad;  
47   }  
48  
49   like[boton.id] = {...boton};  
50  
51   console.log(like[boton.id]);  
52 }
```



## Reto

Realizar el procedimiento correspondiente para hacer el dislike. La actividad la deben realizar de forma grupal Formulario entrega