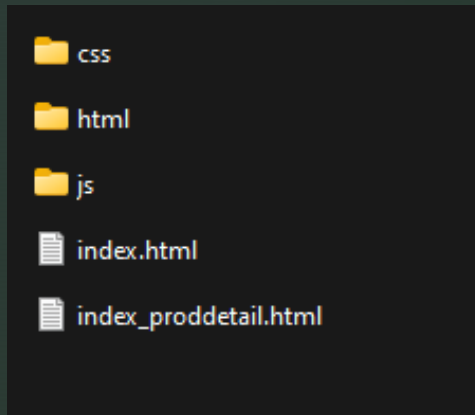




Diego Samudio Makaia

# Proyecto final modulo de fundamentos

# Indicar estructura de carpetas y archivos



> Escritorio > Trabajos de programacion > SincronizadoGitHub > PFINAL_E-COMMERCE > js			
Nombre	Fecha de ...	Tipo	Tamaño
funcion de botones	24/03/202...	Text Docu...	0 KB
mostrarpructor	24/03/202...	Text Docu...	0 KB
ProductosID	24/03/202...	Text Docu...	0 KB

> Escritorio > Trabajos de programacion > SincronizadoGitHub > PFINAL_E-COMMERCE > html			
Nombre	Fecha de modificación	Tipo	Tamaño
htmls necesario en la marcha	24/03/2023 3:18 a. m.	Carpeta de archivos	

Nombre	Fecha de modificación	Tipo	Tamaño
style.css	24/03/2023 3:18 a. m.	Text Document	0 KB

- Indicar si usuario recursos externos / librerias o frameworks .

No tengo claro como es el uso de los frameworks, no entendi



## ► Como implementaria - Consumir una lista de productos desde un JSON-serve

1. Obtener el JSON con HTTP GET
2. Analizar el JSON con `JSON.parse()`
3. Recorrer en el JS el la lista con `forEach()`

En resumen se utiliza `fetch()` para obtener el archivo JSON desde el servidor y `JSON.parse()` para analizarlo y convertirlo en un objeto JavaScript. Luego, se utiliza `forEach()` para recorrer la lista de productos y mostrarlos en la consola o en la página HTML.

En la página principal, Filtrar por categoría de productos desde la página principal:

- Para poder filtrar por categoría agregaría un botón con la función de Get by Class para usar el ID en algo mas especifico]





Debe permitir la opción de  
agregar a favoritos.

- Agregaría un botón de corazón en el card para que así se agregue a favoritos y que el botón solo aparezca cuando el mouse pasa encima de la imagen y no del card (Hover img y Button)



■ Usuario debe tener la oportunidad de eliminar el producto de favoritos.


- agregando en el top bar el carrito para así poder darle la X (button) de borrar lo que este en esa fila del array. Mientras que los demás permanecieran hasta que se borren



## Agregar productos al carrito

- En el card agregaría el botón (+) y que un Java script lo escuche para que escuche el evento y lo muestre en lo que se desplegara del header de carrito





El botón "Process To Checkout" debe abrir un formulario, donde permita ingresar: nombre, dirección y teléfono.

- Le agregaría un (a con ref) a la imagen del icono para que me lleve a un index mas limpio donde se recolecten los datos para finalizar la compra

## Al presionar el botón "Buy Now" debe guardar la compra en un array de compras dentro del Json-Server

- El botón buy now debe tener la función de agarrar todos los IDs únicos del carrito o crear uno con `ID + Math.random()`
- Obtener el JSON para agregar la compra al array de compras con HTTP GET
- Analizar el JSON con `JSON.parse()`
- Después con el método `push` esto se agrega al array de compras

En una página adicional de administrador,  
permitir al usuario:

- a. Ver productos actuales,  
b. Modificar un producto,  
c. borrar un producto, y  
d. crear un nuevo producto  
e. usando JSON-server ver lista de compras de usuarios

# Definiciones

# a. Funciones (arrow functions)

=

```
7 //Ahora
8 let saludo = nombre => `Hola ${nombre}`;
9 console.log( saludo('Jonathan') ); //Imprime Hola Jonathan
```

# b. Peticiones HTTPs (GET, POST, PUT, PATCH, delete) =

1 | GET /index.php

```
1 <html>
2 <body>
3 <form action="newsletter.php" method="post">
4 Name: <input type="text" name="name"><br>
5 E-mail: <input type="text" name="email"><br>
6 <input type="submit">
7 </form>
8 </body>
9 </html>
```



# c. Condicionales

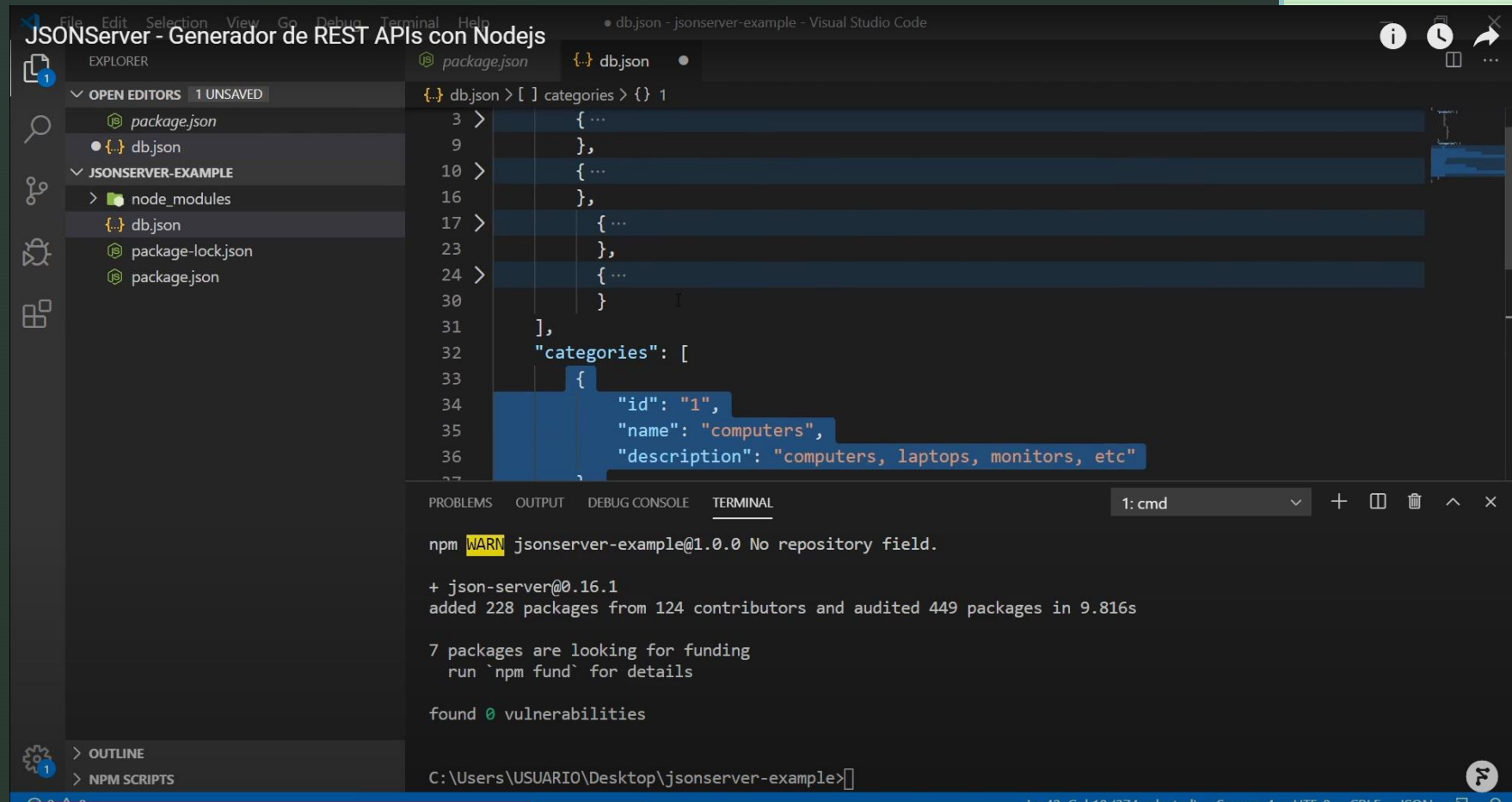
=

```
1  //Condicional que los operadores == || != recordad igual a o distinto a
2  var nombre="Juan"
3  var nombre2="Pedro"
4  if(nombre=='Juan' || nombre2!='Pedro'){
5      console.log("Eres Juan y no Pedro")
6  }else{
7      console.log("No eres Juan y puede que Pedro")
8  }
9  //Condicional que los operadores == && == recordad igual a Y distinto a
10 if(nombre=='Juan' && nombre2=='Pedro'){
11     console.log("Eres Juan y Pedro")
12 }else{
13     console.log("No estais los dos juntos")
14 }
```





# d. JSON-server =



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows the project structure for 'JSONServer - Generador de REST APIs con Nodejs'. The main editor displays the 'db.json' file, which contains a JSON array of categories. The 'categories' array has one object with the following properties: 'id': '1', 'name': 'computers', and 'description': 'computers, laptops, monitors, etc'. The terminal window at the bottom shows the output of running 'npm install json-server' in the project directory. The output includes a warning about the repository field, the installation of 'json-server@0.16.1', and information about funding and vulnerabilities.

```
JSONServer - Generador de REST APIs con Nodejs
EXPLORER
  OPEN EDITORS 1 UNSAVED
    package.json
    db.json
  JSONSERVER-EXAMPLE
    node_modules
      db.json
      package-lock.json
      package.json

package.json
db.json
db.json > [ ] categories > {} 1
3 > { ...
9 },
10 > { ...
16 },
17 > { ...
23 },
24 > { ...
30 }
31 ],
32 "categories": [
33   {
34     "id": "1",
35     "name": "computers",
36     "description": "computers, laptops, monitors, etc"
37   }
38 ],
39 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: cmd
npm WARN jsonserver-example@1.0.0 No repository field.
+ json-server@0.16.1
added 228 packages from 124 contributors and audited 449 packages in 9.816s

7 packages are looking for funding
  run `npm fund` for details

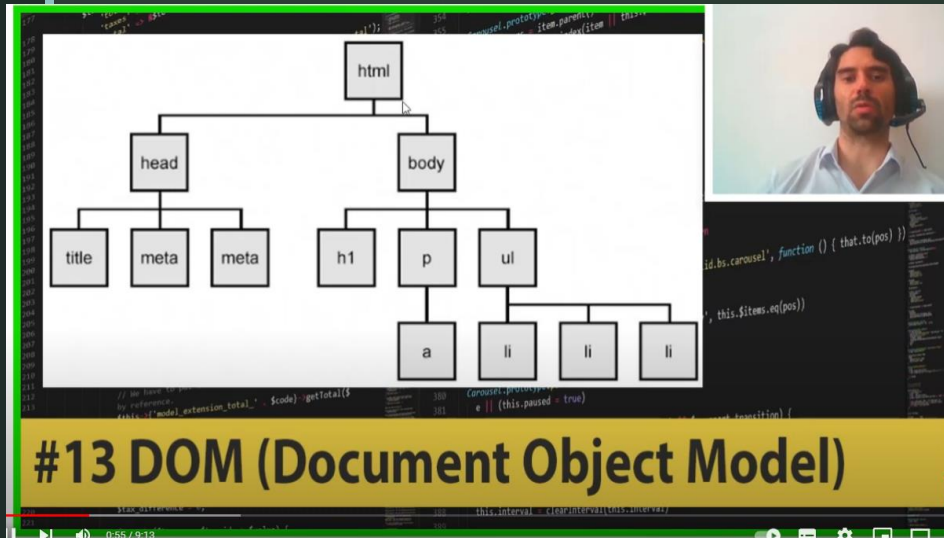
found 0 vulnerabilities

C:\Users\USUARIO\Desktop\jsonserver-example>
```

# e. Eventos =

Forma	Ejemplo	Artículo en profundidad
Mediante <b>atributos HTML</b>	<code>&lt;button onClick="..."&gt; &lt;/button&gt;</code>	<a href="#">Eventos JS desde atributos HTML</a>
Mediante <b>propiedades Javascript</b>	<code>.onclick = function() { ... }</code>	<a href="#">Eventos JS desde propiedades Javascript</a>
Mediante <b>addEventListener()</b>	<code>.addEventListener("click", ...)</code>	<a href="#">Eventos JS desde listeners</a>

# f. Métodos del DOM HTML =



The screenshot shows a video lecture interface. On the left, a DOM tree diagram is displayed. The root node is 'html', which branches into 'head' and 'body'. 'head' branches into 'title', 'meta', and 'meta'. 'body' branches into 'h1', 'p', and 'ul'. The 'p' node branches into 'a', and the 'ul' node branches into three 'li' nodes. On the right, a small video window shows a man with a beard and headphones. Below the diagram, a yellow banner contains the text '#13 DOM (Document Object Model)'. The background of the video shows code snippets.

```
graph TD
    html[html] --> head[head]
    html --> body[body]
    head --> title[title]
    head --> meta1[meta]
    head --> meta2[meta]
    body --> h1[h1]
    body --> p[p]
    body --> ul[ul]
    p --> a[a]
    ul --> li1[li]
    ul --> li2[li]
    ul --> li3[li]
```

#13 DOM (Document Object Model)

- `document.getElementById(id)`
- `document.getElementsByClassName(nombreDeClase)`
- `document.getElementsByTagName(nombreDeEtiqueta)`
- `document.querySelector(selectorCss)`
- `document.querySelectorAll(selectorCss)`

# g. Async/Await o .then().catch =

```
const getDatos = () => {  
  return new Promise((resolve, reject) => {  
    setTimeout(() => {  
      resolve(datos);  
    }, 1500);  
  });  
}  
  
// getDatos()  
//   .then((datos) => console.log(datos));  
  
async function fetchingData () {  
  const datosFetched = await getDatos();  
  console.log(datosFetched);  
}
```



# h. localStorage =

```
<body class="light">
  <div class="toggle-switch">
    <p>Dark</p>
    <span>
      <label class="switch">
        <input id="theme-switch" type="checkbox">
        <span class="slider round"></span>
      </label>
    </span>
    <p>Light</p>
  </div>
  <script src="./index.js"></script>
</body>
```

```
const themeSwitcher = document.getElementById("theme-switch");
themeSwitcher.checked = false;
function clickHandler() {
  if (this.checked) {
    document.body.classList.remove("light");
    document.body.classList.add("dark");
    localStorage.setItem("theme", "dark");
  } else {
    document.body.classList.add("light");
    document.body.classList.remove("dark");
    localStorage.setItem("theme", "light");
  }
}
themeSwitcher.addEventListener("click", clickHandler);
```

```
window.onload = checkTheme();

function checkTheme() {
  const localStorageTheme = localStorage.getItem("theme");

  if (localStorageTheme !== null && localStorageTheme === "dark") {
    // set the theme of body
    document.body.className = localStorageTheme;

    // adjust the slider position
    const themeSwitch = document.getElementById("theme-switch");
    themeSwitch.checked = true;
  }
}
```