EXERCISES QUE TRABAJAREMOS EN LA CUE

- EXERCISE 1: SINTAXIS DE FETCH API
- EXERCISE 2: CONSUMIENDO API CON FETCH. METODO GET
- EXERCISE 3: CONSUMIENDO API CON FETCH. METODO POST
- EXERCISE 4: ELIMINAR Y MODIFICAR CON FETCH

EXERCISE 1: SINTAXIS DE FETCH API

Hace tiempo, se utilizó **XMLHttpRequest** para realizar solicitudes a las **APIs**. No incluía promesas y no permitía un código **JavaScript** limpio.

Luego, se comenzó a utilizar **Ajax** con **JQuery**, permitiendo una sintaxis más limpia.

Actualmente, **JavaScript** tiene su propia forma integrada para realizar las solicitudes a las **APIs**, nos referimos a la **API Fetch**, un nuevo estándar para realizar solicitudes de servidor con promesas.

Para trabajar con **Fetch** comenzaremos invocando al método **fetch()**, que acepta como parámetro la **URL** de la **API** que queremos consumir:

```
fetch(url)
```

Luego del método **fetch()** incluiremos el método de promesa **then()**.

```
fetch(url)
.then(function() {

})
```

Si la promesa devuelta se resuelve correctamente, se ejecutarán las instrucciones que se encuentren dentro del método **then()**, en cambio, si ocurre un error al invocar la **API**, podemos utilizar el método **catch()** para gestionarlo.

La sintaxis completa básica para el uso de **Fetch** es la siguiente:

```
fetch(url)
.then(function() {

})
.catch(function() {

});
```

Además de esta sintaxis que hemos conocido, es importante mencionar que, si bien el método **fetch()** solo nos solicita la **URL** para poder funcionar, podemos agregar otros parámetros de manera opcional, por ejemplo, si recibe parámetros extras acerca del tipo de petición que vamos a realizar, si queremos enviar datos, bajo que codificación se encuentra (por ejemplo, **JSON** o **XML**), etc.

La sintaxis sería la siguiente:

```
Fetch('url ', {
    method: ('GET/POST/PUT/DELETE ',        //optional
    headers: {                              //optional
        Content-Type ': application/json '  //optional
        ...
    },
    body: formData                          //optional
})
```

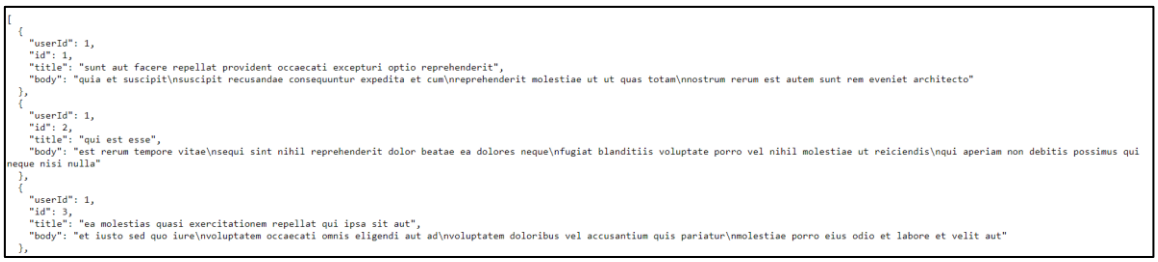
EXERCISE 2: CONSUMIENDO API CON FETCH. METODO GET

Vamos a consumir la **API** publica https://jsonplaceholder.typicode.com/ que es una **API** falsa y gratuita para pruebas. Esta **API** admite todos los métodos **HTTP** y para conocer las rutas debemos dirigirnos al enlace y ahí revisar el apartado **Routes** donde aparecen detallado cada ruta y su método **HTTP**.

Lo primero que haremos será traer datos utilizando el método **GET** y los desplegaremos en una tabla dinámica.

Para comenzar, debemos revisar la **API** para ver como nos devuelve la información: nos dirigiremos a la ruta **POST** para traer todos los posts.

En el navegador se desplegará la información y podremos ver un **JSON** con los atributos “userId”, “id”, “title” y “body”.



Ahora que conocemos la estructura de nuestro **JSON**, crearemos una carpeta (cuyo nombre es irrelevante) para comenzar con nuestra ejemplificación. La ubicaremos en el *workspace* de nuestro editor de texto y crearemos en ella el archivo **index.html**.

En este archivo vamos a agregar la estructura básica para un **HTML** y el enlace a **Bootstrap**. Utilizaremos **Bootstrap** para traer una tabla.

Vamos a dejar solamente la cabecera y una primera fila de datos.

En la etiqueta **<tbody>** vamos a colocar un id de nombre “contenido”. Finalmente, vamos a colocar a la tabla la clase “container” y crearemos un archivo **JavaScript** el cual enlazaremos con nuestro **index.html**.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/boots
trap.min.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
    <title>Document</title>
</head>
<body>

    <table class="table container">
        <thead>
          <tr>
            <th scope="col">userID</th>
            <th scope="col">id</th>
            <th scope="col">title</th>
            <th scope="col">body</th>
          </tr>
        </thead>
        <tbody id="contenido">
          <tr>
            <th scope="row">1</th>
            <td>Mark</td>
            <td>Otto</td>
            <td>@mdo</td>
          </tr>
        </tbody>
      </table>
    <script src="assets/js/logica.js"></script>
</body>
</html>
```

Si nos dirigimos al navegador a ver que tenemos realizado hasta el momento, nos encontraremos con la tabla que hemos creado previamente. Es importante destacar que para conceptos de la ejemplificación hemos colocado en la cabecera los atributos que vamos a intentar consumir.

userID	id	title	body
1	Mark	Otto	@mdo

Ahora, comenzaremos a trabajar en el archivo **JavaScript**.

Comenzaremos la ejemplificación consumiendo nuestra **API** e imprimiéndola por consola, para eso, invocaremos al método **fetch()** y como parámetro pasaremos la ruta a la que nos queremos dirigir.

```
fetch('https://jsonplaceholder.typicode.com/posts')
```

Luego, invocaremos **then()**, que contendrá una función con un parámetro llamada *response* que tomará el valor del objeto devuelto en la función **fetch()** (que obtendríamos de la **URL** que intentamos consumir). Utilizaremos el objeto **json()** para convertir nuestro *response* a **JSON**.

```
.then(response => response.json())
```

Finalmente agregaremos otra función **then()** con una función que contendrá un parámetro llamado “datos”, que se encargará de imprimir en nuestra consola, los datos que consumamos.

```
.then(datos => {
        console.log(datos);
    })
```

Ahora nos podemos dirigir nuevamente a nuestro navegador, actualizar nuestra página web y observaremos que nuestra tabla sigue igual. Esto se debe a que estamos mandando a imprimir los datos en la consola.

Ingresaremos al inspector de elementos, a la consola y, si todo se realizó de manera correcta, deberíamos tener ahí los datos consumidos:

¡Muy bien! Hemos consumido la **API** con **Fetch** pero ¿cómo podemos hacer que toda esta información se pinte en nuestra tabla?

Para hacerlo, lo primero será capturar nuestro id "contenido" de la etiqueta **<tbody>** que es donde intentaremos pintar los datos. Para eso, utilizaremos **querySelector()**.

```
var contenido = document.querySelector("#contenido");
```

Luego, para mantener orden en nuestro código, vamos a crear una función para pintar nuestra tabla. De esta forma, no escribiremos todo el código dentro del segundo **then()** y será mucho más legible.

Crearemos una función llamada "tabla" que recibirá como parámetro los datos capturados. Dentro de esta función, vamos a sobrescribir los datos que se encuentran en nuestro id "contenido". Para eso, escribiremos:

```
contenido.innerHTML = ""
```

Al pasar un **String** vacío estamos sobrescribiendo los datos estáticos que teníamos pintados, dejando la tabla en blanco.

Continuaremos recorriendo todos los datos que obtuvimos con un ciclo **for**, esto, porque pudimos observar que tenemos en total 100 datos. Vamos a hacerlo con un ciclo **for of**, que ejecuta un bloque de código para cada elemento de un objeto iterable. La sintaxis es:

```
for (variable_temporal of dato_iterable) {
  instrucciones
}
```

De esta forma, nuestro bloque **for** quedaría de la siguiente manera:

```
for (let temp of datos) {
}
```

Siendo “temp” nuestra variable temporal y “datos” el valor que recibiremos como parámetro y que vamos a iterar.

Para finalizar, vamos a pintar los datos en nuestra tabla, utilizando nuevamente **contenido.innerHTML**, pero esta vez vamos a indicarle que en cada iteración pinte datos en el cuerpo de la tabla.

Para pintar la tabla traeremos las columnas que se encuentran dentro del **<tbody>** y, usando las comillas invertidas de **JavaScript** (``), que nos permiten agregar código **HTML** dentro del **JavaScript** vamos a indicar cada dato que queremos pintar en cada iteración, quedando de la siguiente manera:

```
contenido.innerHTML += `
            <tr>
               <th scope="row">${temp.userId}</ th>
               <td>${temp.id}</td>
               <td>${temp.title}</td>
               <td>${temp.body}</td>
            </tr>
            `
```

Si actualizamos el navegador ahora, podremos observar como nuestra tabla ha comenzado a pintar los datos que estamos capturando.

userID	id	title	body
1	1	sunt aut facere repellat provident occaecati excepturi optio reprehenderit	quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto
1	2	qui est esse	est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla
1	3	ea molestias quasi exercitationem repellat qui ipsa sit aut	et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut
1	4	eum et est occaecati	ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit
1	5	nesciunt quas odio	repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque
1	6	dolorem eum magni eos aperiam quia	ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciatis et ea nemo ab reprehenderit accusantium quas voluptate dolores velit et doloremque molestiae
1	7	magnam facilis autem	dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas
1	8	dolorem dolore est ipsam	dignissimos aperiam dolorem qui eum facilis quibusdam animi sint suscipit qui sint possimus cum quaerat magni maiores excepturi ipsam ut commodi dolor voluptatum modi aut vitae
1	9	nesciunt iure omnis dolorem tempora et accusantium	consectetur animi nesciunt iure dolore enim quia ad veniam autem ut quam aut nobis et est aut quod aut provident voluptas autem voluptas
1	10	optio molestias id quia eum	quo et expedita modi cum officia vel magni doloribus qui repudiandae vero nisi sit quos veniam quod sed accusamus veritatis error
2	11	et ea vero quia laudantium autem	delectus reiciendis molestiae occaecati non minima eveniet qui voluptatibus accusamus in eum beatae sit vel qui neque voluptates ut commodi qui incidunt ut animi commodi
2	12	in quibusdam tempore odit est dolorem	itaque id aut magnam praesentium quia et ea odit et ea voluptas et sapiente quia nihil amet occaecati quia id voluptatem incidunt ea est distinctio odio
2	13	dolorum ut in voluptas mollitia et saepe quo animi	aut dicta possimus sint mollitia voluptas commodi quo doloremque iste corrupti reiciendis voluptatem eius rerum sit cumque quod eligendi laborum minima perferendis recusandae assumenda consectetur porro architecto ipsum ipsam

Pero, nuestra **API** de prueba nos devuelve 100 datos, por lo que vamos a indicar que solo nos imprima 10. Posterior al cierre de las llaves especiales vamos a indicar que, cuando el id sea igual a 10 detenga la ejecución.

El código completo de esta función y su llamada en la función **then()** quedará así:

```
var contenido = document.querySelector("#contenido");

fetch('https://jsonplaceholder.typicode.com/posts')
    .then(response => response.json())
    .then(datos => {
        tabla(datos)
    });
function tabla(datos) {
    contenido.innerHTML = ""
    for (let temp of datos) {
        contenido.innerHTML += `
            <tr>
```

```
                    <th scope="row">${temp.userId}</ th>
                    <td>${temp.id}</td>
                    <td>${temp.title}</td>
                    <td>${temp.body}</td>
                </tr>

                ` //acá cierran las comillas invertidas
            if (temp.id == 10) {
                break
            }
        }
}
```

Si actualizamos nuestro navegador, podremos encontrar nuestra tabla con los 10 datos indicados:

userID	id	title	body
1	1	sunt aut facere repellat provident occaecati excepturi optio reprehenderit	quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto
1	2	qui est esse	est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla
1	3	ea molestias quasi exercitationem repellat qui ipsa sit aut	et iusto sed quo iure voluptatem occaecati omnis eligendi aut ad voluptatem doloribus vel accusantium quis pariatur molestiae porro eius odio et labore et velit aut
1	4	eum et est occaecati	ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit
1	5	nesciunt quas odio	repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque
1	6	dolorem eum magni eos aperiam quia	ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciatis et ea nemo ab reprehenderit accusantium quas voluptate dolores velit et doloremque molestiae
1	7	magnam facilis autem	dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas
1	8	dolorem dolore est ipsam	dignissimos aperiam dolorem qui eum facilis quibusdam animi sint suscipit qui sint possimus cum quaerat magni maiores excepturi ipsam ut commodi dolor voluptatum modi aut vitae
1	9	nesciunt iure omnis dolorem tempora et accusantium	consectetur animi nesciunt iure dolore enim quia ad veniam autem ut quam aut nobis et est aut quod aut provident voluptas autem voluptas
1	10	optio molestias id quia eum	quo et expedita modi cum officia vel magni doloribus qui repudiandae vero nisi sit quos veniam quod sed accusamus veritatis error

EXERCISE 3: CONSUMIENDO API CON FETCH. METODO POST

Continuando con nuestro ejemplo, vamos a hacer un **POST** utilizando la **API Fetch**, esto quiere decir que enviaremos datos a la **API** que estamos consultando.

Si observamos la **API** que consultamos, podremos encontrar que nos informa como enviar los datos (al igual que consumirlos, actualizarlos o eliminarlos), haciendo clic en el apartado **guide**, que podemos encontrar en las **Routes**.

Nosotros comenzaremos incorporando un formulario en nuestro **index**, ya que deseamos tomar los datos que un usuario nos ingrese. Iremos a **Bootstrap** y seleccionaremos el formulario que sea de nuestra preferencia. En él vamos a dejar tres `<input>` ya que el id no será necesario solicitarlo, puesto que, la **API** se encarga sola de ese parámetro (eso lo observamos en el apartado **guide**).

```
<form id="formulario" class="container" style="border-color: black;
border-style: solid; width: 50%;">
        <div class="mb-3">
          <label for="userId" class="form-label">ID DE
USUARIO</label>
          <input type="text" class="form-control" id="userID"
placeholder="userId"  aria-describedby="emailHelp">
        </div>
        <div class="mb-3">
          <label for="title" class="form-label">TITULO</label>
          <input type="text" class="form-control" id="title"
placeholder="title">
        </div>
        <div class="mb-3">
          <label for="body" class="form-label">Cuerpo</label>
          <input type="text" class="form-control" id="body"
placeholder="body">
        </div>
        <input type="button" class="btn btn-primary" value="enviar"
id="boton">

      </form>
```

Colocaremos al formulario el id “formulario”, a cada `<input>` un id que lo represente y al botón el id “botón”.

En este ejemplo se agregó un estilo simple solo para estética, no es necesario si no deseamos hacerlo.

En nuestro archivo **JavaScript** vamos a comenzar capturando el valor del botón a través de su id.

```
var formulario = document.getElementById("boton");
```

A formulario vamos a asignarle un evento **Listener**. Este es un escuchador que capturará la interacción del usuario, en otras palabras, capturará la acción que el usuario realicé sobre el botón, para este caso, hacer clic, es por esto que indicaremos “click” y una función de flecha.

```
formulario.addEventListener("click", () =>{
})
```

Luego de eso, vamos a tomar los datos que provienen desde el formulario y asignarlos a una variable. En el caso del **userId** debemos tener la precaución de *parsearlo*, ya que los datos capturados del **DOM** vienen en formato **String** y este dato es de tipo numérico en la **API.**

No estamos realizando validación de los datos en este ejemplo, pero lo correcto en un ambiente real es siempre validar el tipo de dato que estamos recibiendo.

```
var idUsuario = parseInt(document.getElementById("userID").value);
var titulo = document.getElementById("title").value;
var cuerpo = document.getElementById("body").value;
```

Ahora que hemos capturado esta información, vamos a crear una constante de nombre newPost que será un objeto y en este objeto asignaremos cada valor a cada parámetro de la **API** de la siguiente forma:

```
const newPost = {
        title: titulo,
        body: cuerpo,
        userId: idUsuario
    }
```

Hasta este momento, aun no hemos realizado el **POST** a la **API**. Vamos a comenzar llamando al método **fetch()** con la **URL** necesaria para realizar un **POST** (para este caso, la dirección es igual que para realizar un **GET**). Antes de llamar al método **then()** vamos a pasar un segundo parámetro, que en este caso será necesario (al realizar un **GET** podemos omitirlo).

Vamos a indicar el método, que será **POST** y pasaremos el *body* del mensaje que será nuestro objeto **newPost**, previamente convertido a un objeto **JSON**.

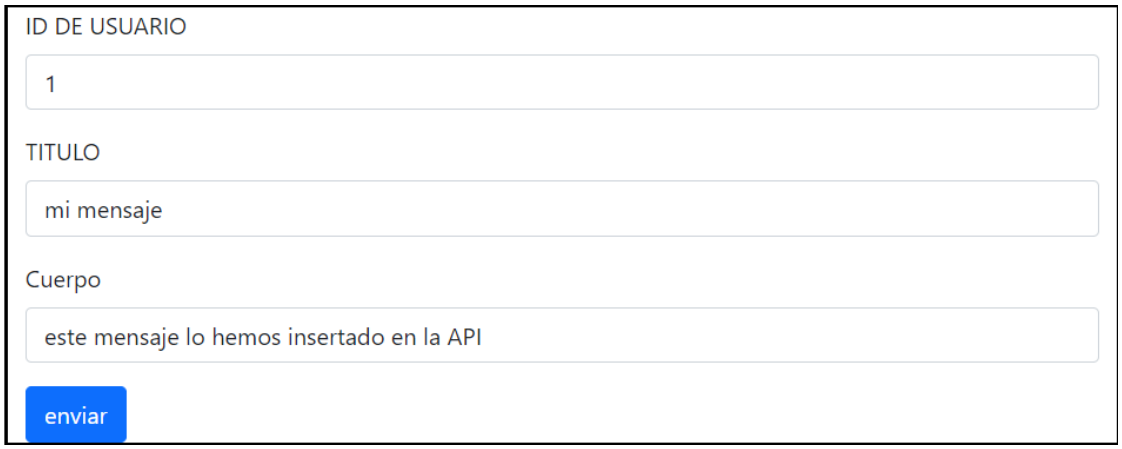
Finalmente pasaremos una cabecera indicando que el contenido es de tipo **JSON**.

```
fetch('https://jsonplaceholder.typicode.com/posts', {
        method: 'POST',
        body: JSON.stringify(newPost),
        headers: {
            'Content-type': 'application/json; charset=UTF-8',
        }
    })
```

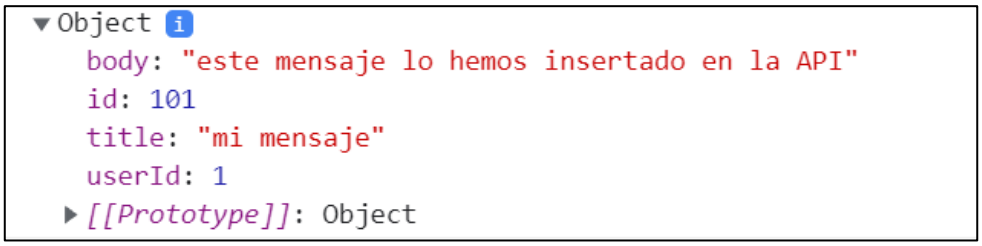
Luego llamaremos al método **then()** con una variable que capturará el objeto **JSON** y el cual imprimiremos en la consola.

```
.then(response => response.json())
.then(json => console.log(json))
```

Ahora que hemos realizado todos estos pasos, vamos a dirigirnos al navegador y enviaremos un dato desde el formulario, recordando que el **userId** debe ser numérico.



Apretaremos el botón enviar y nos dirigiremos a la consola.



Podemos ahora observar que estamos imprimiendo el objeto que insertamos y que nos ha entregado el id 101. Este id es por defecto el que entrega la **API** y, como es una **API** publica de prueba, no podremos ver este objeto insertado realmente en la lista de objetos, ya que esa acción la omite. Este resultado nos demuestra que realizamos bien la inserción.

Podemos observar que nuestro formulario no se limpia después de enviar los datos (es decir, quedan estancados los datos que ingresamos), para eso crearemos una función para limpiar el formulario que llamaremos posterior a la captura de cada dato para que se limpie en nuestra página.

El código completo, incluyendo la función de limpieza, quedará de la siguiente forma:

```
var formulario = document.getElementById("boton");

formulario.addEventListener("click", () => {
    var idUsuario =
parseInt(document.getElementById("userID").value);
    var titulo = document.getElementById("title").value;
    var cuerpo = document.getElementById("body").value;
    limpiar()
    const newPost = {
        title: titulo,
        body: cuerpo,
        userId: idUsuario
    }
    fetch('https://jsonplaceholder.typicode.com/posts', {
        method: 'POST',
        body: JSON.stringify(newPost),
        headers: {
            'Content-type': 'application/json; charset=UTF-8',
        }
    })
        .then(response => response.json())
        .then(json => console.log(json))

})

function limpiar() {
    document.getElementById("formulario").reset();
}
```

EXERCISE 4: ELIMINAR Y MODIFICAR CON FETCH

Para eliminar y modificar un dato con **fetch()** mantenemos la misma lógica de las acciones mostradas previamente.

En esta **API** no podemos eliminar ni modificar realmente, por lo que la ejemplificación será simple:

La sintaxis para eliminar un dato llevará el método **then()** y como parámetro la ruta para eliminar un dato, en este caso, es la misma ruta utilizada previamente, más el id que queremos eliminar. Además de eso, es necesario pasar como segundo parámetro el método *delete*.

```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
  method: 'DELETE',
});
```

La variable 1 es la que debe ir modificándose para indicar el id que se eliminará. Vamos a concatenar una variable con el valor deseado.

Veamos el ejemplo usando una variable con un valor:

```
var valor = 2;

fetch('https://jsonplaceholder.typicode.com/posts/' + valor, {
  method: 'DELETE',
});
```

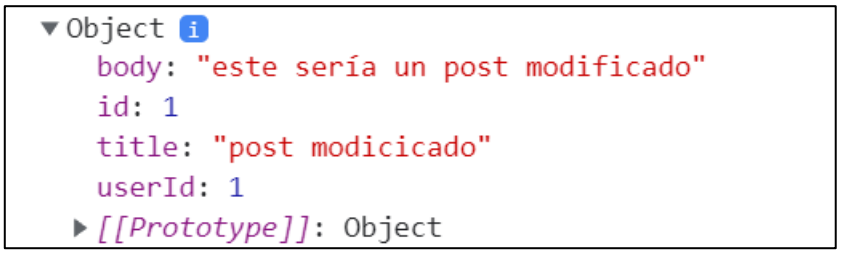
Lamentablemente no podremos ver eliminado el post con id 2, pero ya sabemos cómo concatenar el valor que queremos eliminar.

Para el caso del método **PUT**, este se realiza de manera muy similar al método **POST**, pero, en este caso, requerimos indicar el id que queremos modificar (la **URL** también cambia).

Vamos a ejemplificarlo de manera estática, pero la lógica sería la misma que en el **POST**, capturando los datos desde el **DOM**.

```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
  method: 'PUT',
  body: JSON.stringify({
    id: 1,
    title: 'post modificado',
    body: 'este sería un post modificado',
    userId: 1,
  }),
  headers: {
    'Content-type': 'application/json; charset=UTF-8',
  },
})
  .then((response) => response.json())
  .then((json) => console.log(json));
```

En la consola obtendremos impreso el objeto que acabamos de modificar, pero por desgracia no será realmente modificado en la **API**.



¡Felicitaciones! Ya aprendimos a trabajar con **Fetch** y pintar nuestros datos en nuestra página web.