Programación con JavaScript II

Sesión sincrónica 6

API

Bienvenida y actividad de bienestar

Duración: 10 minutos.

Nombre de la práctica: Plantea tus objetivos como metas de aproximación y replantea tus metas de evitación.
Descripción de la práctica: Con base en lo que plantea Grenville (2012), en la práctica podrás definir diferentes tipos de metas y encontrar la mejor manera de conseguirlas.
Palabras clave: Objetivos, metas y planes.

Instrucciones para el aprendedor:
La autora Bridget Grenville-Cleave (2012) comenta que en el establecimiento de metas es importante distinguir los tipos de metas que hay. Ella menciona dos:

1. Metas de aproximación (*approach*): son las metas con resultados positivos (deseables, placenteros, benéficos o que nos gustaría tener) y hacia las cuales trabajamos.

2. Metas de evitación (*avoidance*): son las metas con resultados negativos (indeseables, dolorosos, dañinos o que nos disgustan) y en las cuales trabajamos para evitarlas.

Ejemplo

Meta de aproximación:

- Ser más eficiente.
- Ser amigable y extrovertido en las reuniones.
- Asumir el rol de líder en el trabajo.

Meta de evitación:

- Dejar de aplazar.
- Dejar de ser tan tímido en las reuniones.
- No pasar desapercibido en el trabajo.

Bienvenida y actividad de bienestar

Las investigaciones que se han realizado respecto a estos tipos de metas muestran que perseguir metas de evitación resulta en un detrimento del bienestar. Por otro lado, estos descubrimientos sugieren que el establecer metas de aproximación o replantear las metas de evitación es benéfico.

Reflexiona:

- ¿Qué tipo de metas te has planteado tú?
- ¿Hay algunas metas que puedas replantear en una forma más positiva?
- ¿Cuándo las tendrás listas?

Fuente: Grenville, B. (2012). GOAL-SETTING SECRETS. Recuperado de http://positivepsychologynews.com/news/bridget-grenville-cleave/2012013120696

Actividad guiada
Parte 1

Duración: 75 minutos.

En este tema pondrás en práctica algunas de las nuevas características y tipos de datos aprendidos durante la lección. Para alcanzar el objetivo, realiza los siguientes ejercicios.

Ejercicio 1

1. Accede a la siguiente página en donde podrás usar la API de prueba: https://jsonplaceholder.typicode.com/
2. Accede a la siguiente página en donde podrás encontrar el recurso de comentarios: https://jsonplaceholder.typicode.com/comments
3. Describe la estructura de información JSON que contiene.
4. Analiza el siguiente código y redacta comentarios, colocándolos a lo largo del texto, que describan las acciones a ejecutar.
5. Determina y describe el resultado que mostrará la consola al ejecutarse con en Visual Code + Live Server.

Actividad guiada
Parte 1

```
<!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">
    <title>Document</title>
</head>
<body>
    <div id="app"></div>
    <script>
        const xhr = new XMLHttpRequest();

        function onRequestHandler(){
            if (this.readyState==4 && this.status==200) {
            console.log(this.response);
            }
        }
        xhr.addEventListener("load", onRequestHandler);
        xhr.open("GET","https://jsonplaceholder.typicode.com/comments");
        xhr.send();
    </script>
</body>
```

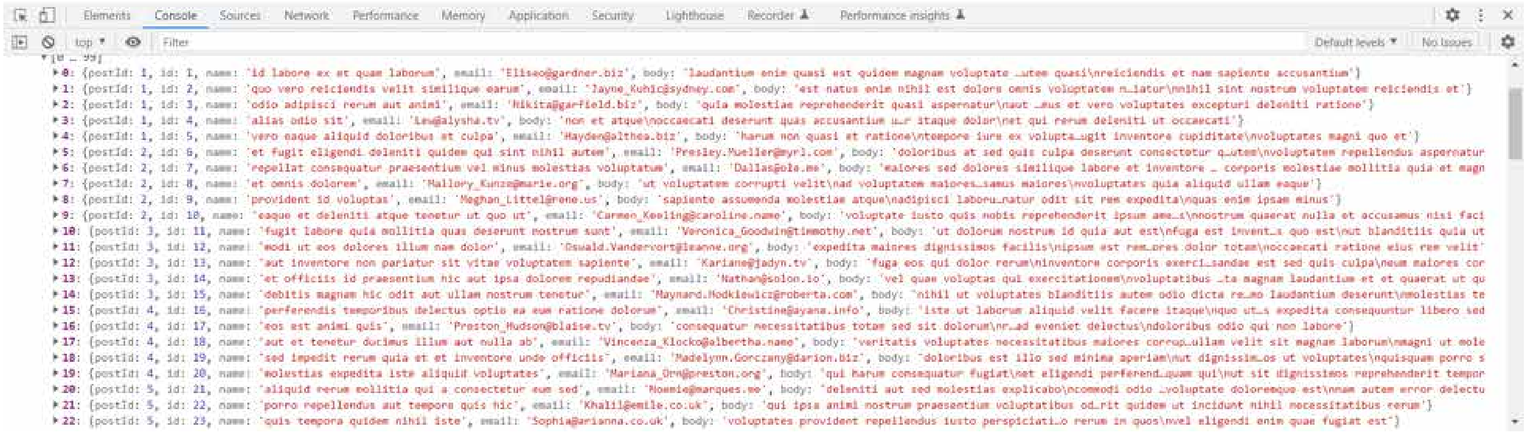
Nota: Para ejecutar el código anterior, puedes hacer clic derecho en alguna línea del código y seleccionar la opción “Open with Live Server”.

Actividad guiada
Parte 1

6. Utiliza el método JSON.PARSE() para tomar los comentarios y desplegarlos en consola, de tal manera que se muestren los datos de la siguiente forma:

```
const data = JSON.parse(this.response);
console.log(data);
```

La siguiente imagen muestra los valores JSON en la consola del navegador. Para verlo así, solo debes presionar F12.



Actividad guiada Parte 1

7. Finalmente, usando una tabla, muestra el id, el nombre y el correo electrónico en una página HTML.

Id	Name	Email	Comment
1	id labore ex et quam laborum	Eliseo@gardner.biz	laudantium enim quasi est quidem magnam voluptate ipsam eos tempora quo necessitatibus dolor quam autem quasi reiciendis et nam sapiente accusantium
2	quo vero reiciendis velit similique earum	Jayne_Kuhic@sydney.com	est natus enim nihil est dolore omnis voluptatem numquam et omnis occaecati quod ullam at voluptatem error expedita pariatur nihil sint nostrum voluptatem reiciendis et
3	odio adipisci rerum aut animi	Nikita@garfield.biz	quia molestiae reprehenderit quasi aspernatur aut expedita occaecati aliquam eveniet laudantium omnis quibusdam delectus saepe quia accusamus maiores nam est cum et ducimus et vero voluptates excepturi deleniti ratione
4	alias odio sit	Lew@alysha.tv	non et atque occaecati deserunt quas accusantium unde odit nobis qui voluptatem quia voluptas consequuntur itaque dolor et qui rerum deleniti ut occaecati
5	vero eaque aliquid doloribus et culpa	Hayden@althea.biz	harum non quasi et ratione tempore iure ex voluptates in ratione harum architecto fugit inventore cupiditate voluptates magni quo et
6	et fugit eligendi deleniti quidem qui sint nihil autem	Presley.Mueller@myrl.com	doloribus at sed quis culpa deserunt consectetur qui praesentium accusamus fugiat dicta voluptatem rerum ut voluptate autem voluptatem repellendus aspernatur dolorem in
7	repellat consequatur praesentium vel minus molestias voluptatum	Dallas@ole.me	maiores sed dolores similique labore et inventore et quasi temporibus esse sunt id et eos voluptatem aliquam aliquid ratione corporis molestiae mollitia quia et magnam dolor
8	et omnis dolorem	Mallory_Kunze@marie.org	ut voluptatem corrupti velit ad voluptatem maiores et nisi velit vero accusamus maiores voluptates quia aliquid ullam eaque
9	provident id voluptas	Meghan_Littel@rene.us	sapiente assumenda molestiae atque adipisci laborum distinctio aperiam et ab ut omnis et occaecati aspernatur odit sit rem expedita quas enim ipsam minus
10	eaque et deleniti atque tenetur ut quo ut	Carmen_Keeling@caroline.name	voluptate iusto quis nobis reprehenderit ipsum amet nulla quia quas dolores velit et non aut quia necessitatibus nostrum quaerat nulla et accusamus nisi facilis
11	fugit labore quia mollitia quas	Veronica_Goodwin@timmothy.net	ut dolorum nostrum id quia aut est fuga est inventore vel eligendi explicabo quis consectetur

Actividad guiada Parte 1

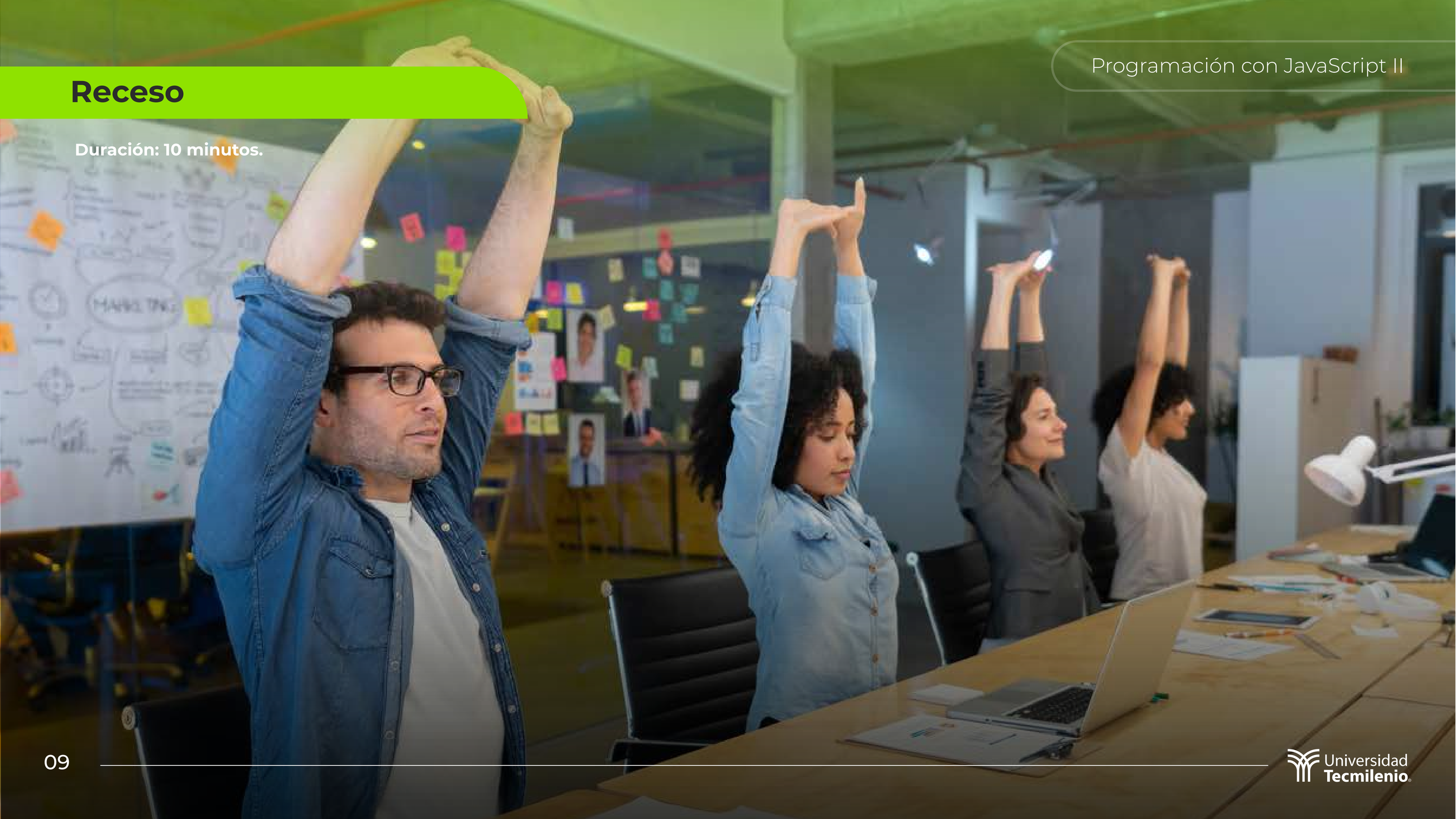
Consideraciones:

Instructor: Para realizar esta actividad se requerirá de un servidor web y una base de datos que permita hacer pruebas de lectura y escritura.
Se recomienda usar la siguiente liga para acceder a un recurso que permite usar una API libre para pruebas y prototipado:
https://jsonplaceholder.typicode.com/
De manera local, se recomienda usar Live Server como extensión a Visual Code.

Participante: Al realizar esta actividad, recuerda instalar el Live Server como extensión en Visual Studio Code.
El formato de la tabla usa archivo CSS, pero la presentación estética no es tan importante como su funcionamiento.
Puedes hacer más prácticas con otros archivos JSON desde la página: https://jsonplaceholder.typicode.com/

Receso

Duración: 10 minutos.



Actividad guiada Parte 2

Duración: 75 minutos.

Ejercicio 2

1. Accede a la siguiente página en donde podrás usar la API de prueba: https://jsonplaceholder.typicode.com/
2. Accede a la siguiente página en donde podrás encontrar el recurso de publicaciones en un foro llamado POSTS. Observa que al usar la liga accederás a la página que mostrará el contenido de los datos en formato JSON https://jsonplaceholder.typicode.com/posts
3. Describe la estructura de información JSON que contiene y la cantidad de registros.
4. Crea un archivo HTML en Visual Code llamado MetodoPut.html e incluye un formulario que recopile información en donde se use el método PUT en una API Fetch.

Actividad guiada
Parte 2

```
<!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">
    <title>Document</title>
</head>
<body>
    <h1 id="title">METODO PUT</h1>
    <form class="form">
      <p><label>Id: <input type="text" name="id"></label></p>
      <p><label>Name: <input type="text" name="name"></label></p>
      <p><label>Email: <input type="text" name="email"></label></p>
      <p><label>Comment: <input type="text" name="comment"></label></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
  </body>
</html>
```

Actividad guiada
Parte 2

5. Crea un archivo de JavaScript llamado scripts.js que contendrá las líneas de código necesarias para establecer peticiones Fetch a una API. Opcionalmente, puedes incluir las etiquetas de <script> en el archivo html entre las siguientes etiquetas <body></body> y colocar el código JavaScript en un archivo alterno llamado “script.js”.

```
<script src="scripts.js"></script>
```

6. Utiliza la propiedad querySelector para recuperar la información que el usuario ponga en el formulario:

```
const formE1 = document.querySelector('.form');
```

7. Utiliza la función addEventListener para transferir la información del formulario a la Fetch API al presionar el botón “Enviar”. Para evitar que la página se recargue y que se agilice el tiempo de respuesta, utiliza la función event.preventDefault();.

```
formE1.addEventListener('submit', event =>{
    event.preventDefault();
});
```

8. Crea un par de constantes, una para capturar toda la información de la forma y la otra para usarla como parámetro para el método stringify. Para ello puedes usar new FormData y pasarle como parámetro el objeto de la forma usada en el paso 6.

```
const formData = new FormData(formE1);
const data = Object.fromEntries(formData);
```

Actividad guiada
Parte 2

9. Abre una secuencia fetch que apunte hacia la API Fetch con el método PUT https://jsonplaceholder.typicode.com/posts/1.

Nota: La API de prueba que se está usando solo puede agregar un valor a la vez, es por eso que la dirección URL termina en 1.

```
fetch('https://jsonplaceholder.typicode.com/posts/1', {
        method: 'PUT',
        headers: {
            'Content-type': 'application/json; charset=UTF-8',
            },
        body: JSON.stringify(data)
    })
```

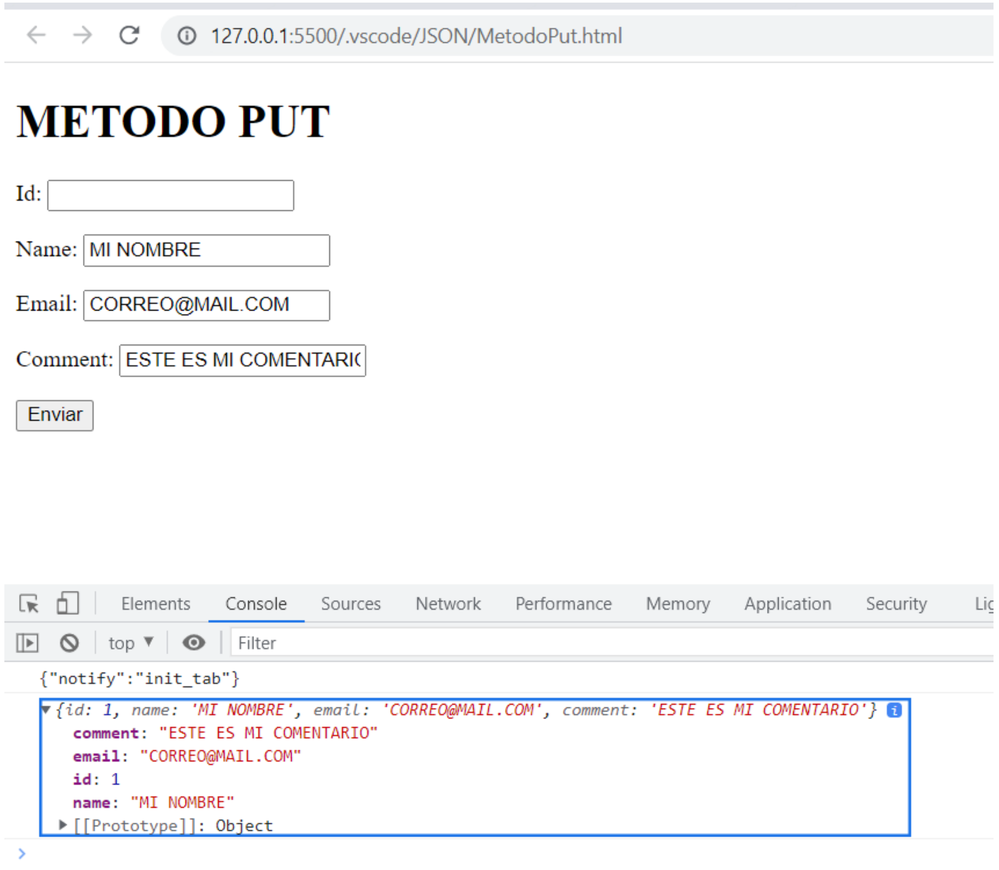
10. Incluye una transformación del objeto “data” declarado en el paso 8 para convertirlo en una estructura JSON utilizando el método stringify().

```
body: JSON.stringify(data)
```

11. Termina el código enviando la respuesta JSON. De forma opcional, puedes enviar a la consola los datos y cualquier error que se genere.

```
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.log(error));
```

Actividad guiada
Parte 2



Actividad guiada Parte 2

Consideraciones:

Instructor: Para realizar esta actividad se requerirá de un servidor web y una base de datos que permita hacer pruebas de lectura y escritura.

Se recomienda usar https://jsonplaceholder.typicode.com/, que es un recurso que permite usar una API de acceso libre para pruebas y prototipado.

De manera local, se recomienda usar Live Server como extensión a Visual Code.

El punto 11 puede ser omitido en caso de que consideres que la actividad requiere de más tiempo.

A continuación, se incluyen las dos secciones del código que pueden servir como respuesta a la actividad:

CÓDIGO DEL ARCHIVO MetodoPut.html

Actividad guiada
Parte 2

```
<!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">
    <title>Document</title>
</head>
<body>
    <h1 id="title">METODO PUT</h1>
    <form class="form">
      <p><label>Id: <input type="text" name="id"></label></p>
      <p><label>Name: <input type="text" name="name"></label></p>
      <p><label>Email: <input type="text" name="email"></label></p>
      <p><label>Comment: <input type="text" name="comment"></label></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
      <script src="scripts.js"></script>
  </body>
</html>
```

Actividad guiada Parte 2

CÓDIGO DEL ARCHIVO scripts.js

```
const formE1 = document.querySelector('.form');

formE1.addEventListener('submit', event =>{
    event.preventDefault();

    const formData = new FormData(formE1);
    const data = Object.fromEntries(formData);

    fetch('https://jsonplaceholder.typicode.com/posts/1', {
        method: 'PUT',
        headers: {
            'Content-type': 'application/json; charset=UTF-8',
            },
        body: JSON.stringify(data)
    })
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.log(error));
});
```

Actividad guiada
Parte 2

Consideraciones:

Aprendedor: Para realizar esta actividad es necesario usar un servidor web que acepte peticiones GET y PUT. En caso de no contar con este servidor, puedes usar el propuesto en la actividad.

Recuerda que también puedes usar Live Server de manera local desde Visual Studio Code.

Actividad guiada
Parte 2

```
<!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">
    <title>Document</title>
</head>
<body>
    <h1 id="title">METODO PUT</h1>
    <form class="form">
      <p><label>Id: <input type="text" name="id"></label></p>
      <p><label>Name: <input type="text" name="name"></label></p>
      <p><label>Email: <input type="text" name="email"></label></p>
      <p><label>Comment: <input type="text" name="comment"></label></p>
      <p><input type="submit" value="Enviar"></p>
    </form>
      <script src="scripts.js"></script>
  </body>
</html>
```

Cierre

Duración: 10 minutos.

Ahora has aprendido a usar uno de los pilares fundamentales de JavaScript: la asincronía. Esto te ayudará a evitar que tus aplicaciones se queden en espera de algún resultado y que tu usuario se desespere y termine por no querer utilizar tu aplicación.