Programación con JavaScript II
Sesión sincrónica 6



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



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.



```
<!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".



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.

```
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder A Performance insights A
 FO: (postId: 1, id: 1, namm: 'id labore ex et quam laborum', smail: 'Eliseo@gardner.biz', body: 'laudantium enim quasi est quidem magnam voluptate ...utem quasi\nreiclendis et nam sapiente accusantium')
        1: (postid: 1, id: 2, name: 'quo vero reiciendis velit similique earum', email: 'layne Kuhic@sydney.com', body: 'est natus enim mihil est dolors cemis voluptatem m.latur\mnihil sint mostrum voluptatem reiciendis et')
        * 2: [postId: 1, id: 3, name: 'odio adipisci rerum aut animi', small: 'Nikita@garfield.biz', body: 'quia molestiae reprehenderit quasi aspernatur\naut _sus et vero voluptates excepturi deleniti ratione')
        *3: {postld: 1, id: 4, name: "alias onlo sit", email: 'lengalysha.tv', body: 'non et atque'noccaecati deserunt quas accusantium uur itaque dolor/net qui rerum deloniti ut occaecati'}
        P4: (postId: 1, Id: 5, namm: 'vero eaque aliquid doloribus et culpa', small: 'Haydengalthea.biz', body: 'harum non quasi et ratione\ntempore inre ex volupta_ugit inventore cupiditate\nvoluptates magni quo et')
        F5: (postId: 2, id: 6, name: (et fugit eligendi deleniti quidem qui sint mihil autem', small: 'Presley. Mueller@myrl.com', body: 'doloribus at sed quis culpa deserunt consectetur quitem\nvoluptatem repellendus aspernatur
        > 6: {postId: 2, id: 7, name: 'repellat consequator praesentium vel minus molestias voluptatum', small: 'Dallasgole.me', body: 'majores sed dolores similique labore et inventore _ corporis molestias mollitia quia et magn
        *7: {postId: 2, id: 8, name: 'et cenis dolorem', email: 'Hallory Kunzegmarie.org', budy: 'et voluptatem majores assus majores as
        >8: (postId: 2, Id: 9, namm: 'provident id voluptas', email: 'Meghan_Littel@rene.us', body: 'sapiente assumenda molestiam atque\nadipisci labbru_ratur odit sit rem expedita\nquas enim ipsam minus')
        F9: (postId: 2, id: 18, name: "eaque et deleniti atque tecetur ut quo ut", email: "Carmeo Keeling@caroline.come", body: 'voluptate iusto quis nobis reprehenderit ipsum ame_s\nnostrum quaerat nulla et accusamus nisi faci
        *18: [postid: ], id: 11, name: 'fugit labore quis collitia quas deserunt mostrum sunt', email: 'Veronica_Goodwin@timmothy.met', body: 'ut dolorum mostrum id quis aut est\nfuga est invent.s quo est\nut blanditis quis ut
        *11: (postid: 3, id: 12, name: "modi ut cos dolores illum nam dolor", email: "Gauald.Vandercort@league.org", body: "expedita mainres dignissimus facilis\mipsum est rem.ores dolor totam\nocceecati ratione eius rem velit"
        12: [postId: 3, id: 13, name: "aut inventore non pariatur sit vitae voluptatem majores cor
        > 13: (postīd: 3, id: 14, namm: 'et officiis id praesentium hic aut ipsa dolorem repudiandae', small: 'Nathan@Solon.io', body: 'vel quas voluptas qui execcitationem\nvoluptatibus its magnam laudantium et et quaerat ut qu
        > 14: [postid: 3, id: 15, name: 'debitis magnam hic odit aut ullam nostrum tenetur', smail: 'Maymard. Modklewlczgroberta.com', body: 'nihil ut voluptates blanditis mutem odio dicta re.mo laudentium deserunt\nmolestias te
        *15: {postId: 4, Id: 15, name: 'perferendis temporibus delectus optio ma eum ratione dolorum', small: 'ChristineSayana.info', body: 'iste ut laborum aliquid velit facere itaque', nque ut a expedita consequentur libero sed
        16: [postld: 4, id: 17, name: 'eas est animi quis', small: 'Preston Hudson@blaise.tv', body: 'consequatur mecessitatibus totam sed sit dolorum\nr.ad eveniet delectus\ndoloribus odio qui non labore')
        > 17: (postid: 4, id: 18, name: "aut et tenetur dorimus lilum aut nulla ab", emuil: "Vincenza_Klocko@albertha.name", body: "veritatis voluptates necessitatibus maiores corrup.ullam velit sit magnam laborum\nmagni ut mole
        *18: [postid: 4, id: 19, name: 'sed impedit norum quia et et inventore unde officiis', email: 'Madelyon.Gorczany@darion.biz', body: 'doloribus est illo sed minima aperiam\mut dignissim.os ut voluptates\nquisquam porro s
        ▶ 19: {postId: 4, id: 20, name: 'molestias expedita iste aliquid voluptates', email: 'Mariana_Orn@preston.org', hody: 'qui harum consequator fugiat\net eligendi perferend.quam qui\nut sit dignissimos reprehenderit tempor
        ▶ 20: [postId: 5, id: 21, name: 'aliquid rerum molitia qui a consectatur mum sed', email: 'Nommin@marques.me', hody: 'deleniti aut med molestias explicabo\ncommodi odio _voluptate doloremente est\name autem error delectu
        P 21: [postid: 5, id: 22, name: 'porno repellendus aut tempora quis hic', email: 'Khalil@emile.co.uk'; body: 'qui ipsa animi nostrum praesentium voluptatibus od_rit quidem ut incidunt mihil mocessitatibus cerum')
        > 22: [postid: 5, id: 23, name: 'quis tempora quidem nihil iste', smail: 'Sophia@arisnna.co.uk', body: 'voluptates provident repellandus iusto perspiciati.o renum im quos\nwel eligendi enim quae fugiat est']
```



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

ld	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	



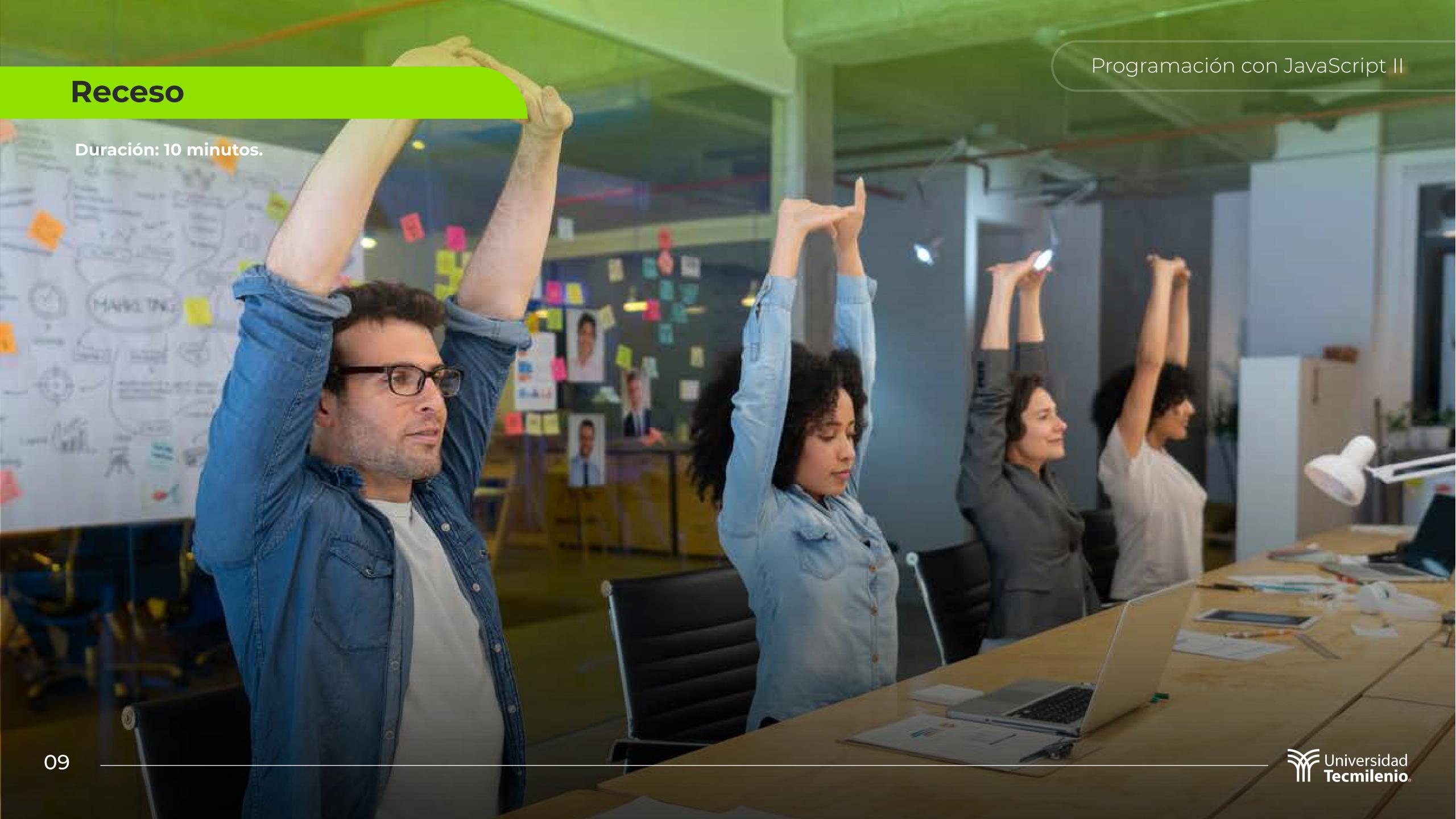
Consideraciones:

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/





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.





```
<!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">
   <|abel>Id: <input type="text" name="id"></label>
   <label>Name: <input type="text" name="name"></label>
   <label>Email: <input type="text" name="email"></label>
   <label>Comment: <input type="text" name="comment"></label>
   <input type="submit" value="Enviar">
  </form>
 </body>
</html>
```



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);
```



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.

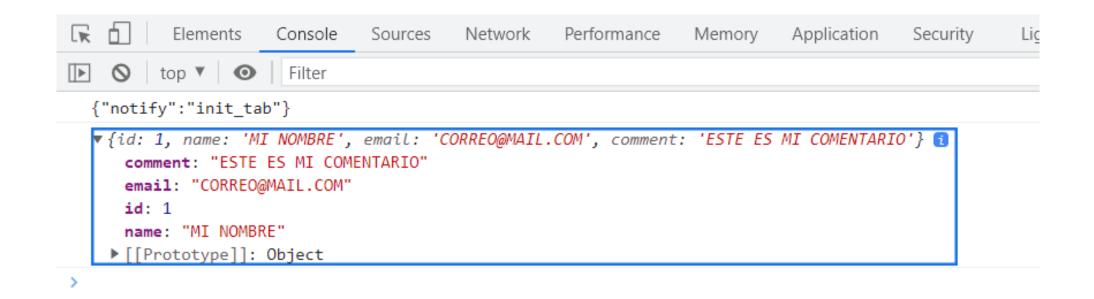
```
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.log(error));
```



← → G	127.0.0.1:5500/.vscode/JSON/MetodoPut.html	

METODO PUT

Id:
Name: MI NOMBRE
Email: CORREO@MAIL.COM
Comment: ESTE ES MI COMENTARIO
Enviar





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



```
<!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">
   <label>Id: <input type="text" name="id"></label>
   <label>Name: <input type="text" name="name"></label>
   <label>Email: <input type="text" name="email"></label>
   <label>Comment: <input type="text" name="comment"></label>
   <input type="submit" value="Enviar">
  </form>
   <script src="scripts.js"></script>
 </body>
</html>
```



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));
```



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.



```
<!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">
   <label>Id: <input type="text" name="id"></label>
   <label>Name: <input type="text" name="name"></label>
   <|abel>Email: <input type="text" name="email"></label>
   <label>Comment: <input type="text" name="comment"></label>
   <input type="submit" value="Enviar">
  </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.



La obra presentada es propiedad de ENSEÑANZA E INVESTIGACIÓN SUPERIOR A.C. (UNIVERSIDAD TECMILENIO), protegida por la Ley Federal de Derecho de Autor; la alteración o deformación de una obra, así como su reproducción, exhibición o ejecución pública sin el consentimiento de su autor y titular de los derechos correspondientes es constitutivo de un delito tipificado en la Ley Federal de Derechos de Autor, así como en las Leyes Internacionales de Derecho de Autor.

El uso de imágenes, fragmentos de videos, fragmentos de eventos culturales, programas y demás material que sea objeto de protección de los derechos de autor, es exclusivamente para fines educativos e informativos, y cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por UNIVERSIDAD TECMILENIO.

Queda prohibido copiar, reproducir, distribuir, publicar, transmitir, difundir, o en cualquier modo explotar cualquier parte de esta obra sin la autorización previa por escrito de UNIVERSIDAD TECMILENIO. Sin embargo, usted podrá bajar material a su computadora personal para uso exclusivamente personal o educacional y no comercial limitado a una copia por página. No se podrá remover o alterar de la copia ninguna leyenda de Derechos de Autor o la que manifieste la autoría del material.

