Repaso LMSGI 2º trimestre

Mientras que en el backend hemos empezado a usar NodeJS y Express, en el front hemos estado trabajando con HTML, CSS y JavaScript Vanilla.

Nos hemos centrado en conocer las diferentes tecnologías de las que disponemos nativamente en el navegador, en concreto `XMLHttpRequest` y `fetch` para hacer peticiones a un servidor, y `JS Vanilla` para manipular el DOM con los datos que recibimos.

XMLHttpRequest

Hasta ahora, para obtener datos dinámicos, hemos usado `XMLHttpRequest` para hacer peticiones a un servidor.

```
const xhr = new XMLHttpRequest();
xhr.open('GET', 'https://eazywarez.glitch.me/htmx/hola', true);
xhr.onreadystatechange = function() {
  if (xhr.readyState === 4 && xhr.status === 200) {
    const respuesta = JSON.parse(xhr.responseText);
    document.getElementById('hola_mundo').innerHTML = respuesta;
  }
}
xhr.send();
```

Fetch API

En lugar de `XMLHttpRequest`, podemos usar `fetch` para hacer peticiones a un servidor.

```
fetch('https://eazywarez.glitch.me/')
  .then(response => response.json())
  .then(data => {
    procesar(data)
  });
```

Mandar headers con fetch

Para interactuar con muchas APIs es necesario configurar un header, aqui podemos especificar cosas como el tipo de contenido que esperamos recibir, el que mandamos u otros datos como tokens de autenticación, API keys, etc.

```
fetch('https://eazywarez.glitch.me/', {
    headers: {
    'Content-Type': 'application/json',
    'Accept': 'application/json',
    'API-Key': '123456'
    }
})
```

Promesas

`fetch` devuelve una promesa, que es un objeto que representa un valor que puede estar disponible ahora, en el futuro o nunca.

```
const promesa = fetch('https://eazywarez.glitch.me/');
const respuesta = promesa.then(response => response.json());
const datos = respuesta.then(data => procesar(data));

function procesar(data) {
  const contenedor = document.getElementById('#contenedorEazy');
    contenedor.innerHTML = data;
}
```

Una promesa tiene dos métodos: `then` y `catch`. El método `then` se ejecuta si la promesa se cumple con `resolve`, y el método `catch` se ejecuta si la promesa no se cumple con `reject`.

```
const promesa = new Promise((resolve, reject) => {
  const valor = Math.random();
    if (valor > 0.5) {
        resolve('La promesa se ha cumplido');
    } else {
        reject('La promesa no se ha cumplido');
    }
});
promesa.then((valor) => {
  console.log(valor);
}).catch((error) => {
  console.log(error);
});
```

`<form>` y `event.preventDefault()`

Si queremos manejar el envio de datos de un formulario, podemos usar el evento `submit` y el método `preventDefault` para evitar que la página se recargue.

```
<form id="formulario">
  <input type="text" name="user" id="user">
  <input type="text" name="password" id="password">
  <button type="submit">Enviar</button>
</form>
```

Usando `event.preventDefault()`:

```
document.getElementById('formulario').addEventListener('submit', function(event) {
  event.preventDefault(); // Evita que la página se recargue
  //capturamos el input que lanzó el evento
  const whoTriggered = event.target;
  const user = document.getElementById('user').value;
  const password = document.getElementById('password').value;
  fetch('/login', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Accept': 'application/json',
      'API-Key': '123456'
    },
    body: JSON.stringify({user, password})
  });
});
```

HTMX

HTMX es una librería que nos permite hacer peticiones a un servidor y actualizar el contenido de la página sin recargarla.

```
<p hx-trigger="load" hx-get="https://eazywarez.glitch.me"></p>
```

Este código hace una petición a `https://eazywarez.glitch.me` cuando la página se carga y actualiza el contenido de la etiqueta `p` con la respuesta.

Este código hace una petición a `https://eazywarez.glitch.me` cuando la página se carga y actualiza el contenido de la etiqueta que tenga el id `contenido` con la respuesta que nos da el servidor.

```
<div hx-trigger="load" hx-get="https://eazywarez.glitch.me" hx-target="#contenido" hx-swap="innerHTML"></div>
<p id="contenido"></p>
```