

TEMA 10:

Promesas y fetch

B. FETCH

1. FETCH

- El API **fetch** es un nuevo estándar para interactuar por HTTP, con un diseño moderno, basado en **promesas**, con mayor flexibilidad y capacidad de control a la hora de realizar llamadas al servidor.
- Este es un estándar publicado por **WHATWG** (Web Hypertext Application Technology Working Group), los responsables de algunos de los estándares más interesantes que se han publicado en los últimos tiempos.
- **Fetch** está soportado de forma nativa por Edge 14, Firefox 39, Chrome 42 y Opera 29

2. PETICIONES HTTP CON FETCH

- La forma de realizar una petición es muy sencilla, básicamente se trata de llamar a **fetch** y pasarle por parámetro la URL de la petición a realizar:

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

- Fetch, es una promesa, por tanto, lleva asociado un método **.then**, en este caso encadenaremos un **.then** con otro, ya que hasta que no se resuelva el primero no se ejecutará el segundo.

2. PETICIONES HTTP CON FETCH

```
fetch('https://reqres.in/api/users?page=2')  
  .then(function (respuesta) {  
    return respuesta.json();  
  })  
  .then(function (datosjson) {  
    console.log(datosjson);  
    console.log(datosjson.total_pages);  
  });
```

2. PETICIONES HTTP CON FETCH

```
▼ {page: 2, per_page: 6, total: 12, total_pages: 2, data: Array(6), ...} ⓘ  
  ► data: (6) [{...}, {...}, {...}, {...}, {...}, {...}]  
    page: 2  
    per_page: 6  
    ► support: {url: "https://reqres.in/#support-heading", text: "To keep ReqRes free, contribut...  
      total: 12  
      total_pages: 2  
    ► __proto__: Object
```

2. PETICIONES HTTP CON FETCH

- Utilizando las funciones **flecha** (en este caso en particular todo el mundo las usa... I'm sorry) se nos queda más elegante el código:

```
fetch('https://reqres.in/api/users?page=2')  
  .then(respuesta => respuesta.json())  
  .then(datosjson => {  
    console.log(datosjson);  
    console.log(datosjson.total_pages);  
  });
```

2. PETICIONES HTTP CON FETCH

- Podemos comparar ahora con jQuery para ver las similitudes y diferencias entre ambos métodos.

| jQuery | Fetch |
|---|--|
| <pre>\$.ajax({ type: "GET", url: "https://reqres.in/api/users?page=2", data: "data", dataType: "JSON", success: datosjson => { console.log(datosjson); console.log(datosjson.total_pages); } });</pre> | <pre>fetch('https://reqres.in/api/users?page=2') .then(respuesta => respuesta.json()) .then(datosjson => { console.log(datosjson); console.log(datosjson.total_pages); });</pre> |

2. PETICIONES HTTP CON FETCH

- Podemos incluso comparar ejemplos más complejos:

```
<div id="contenedor" class="contenedor">  
  <div><h3>ID</h3></div>  
  <div><h3>Nombre</h3></div>  
  <div><h3>Año</h3></div>  
  <div><h3>color</h3></div>  
  <div><h3>Pantone</h3></div>  
</div>
```

HTML

```
.contenedor {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr 1fr 1fr;  
}
```

CSS

2. PETICIONES HTTP CON FETCH

```
fetch('https://reqres.in/api/unknown')
.then (respuesta=>respuesta.json())
.then (json => {

    let salida=" <div class='contenedor'><div><h3>ID</h3></div><div><h3>Nombre</h3></div><div><h3>Año</h3></div><div><h3>color</h3></div><div><h3>Pantone</h3></div>";

    for (let element of json.data) {
        salida+="

" +element.id+"</div>";
        salida+="

" +element.name+"</div>";
        salida+="

" +element.year+"</div>";
        salida+="

" +element.color+"</div>";
        salida+="

" +element.pantone_value+"</div>";
    }
    salida+="


```

2. PETICIONES HTTP CON FETCH

```
$.ajax({
  type: 'GET',
  dataType: 'JSON',
  url: 'https://reqres.in/api/unknown',
  success: function (json) {
    let salida=" <div class='contenedor'><div><h3>ID</h3></div><div><h3>Nombre</h3></div><div><h3>Año</h3></div><div><h3>color</h3></div><div><h3>Pantone</h3></div>";

    for (let element of json.data) {
      salida+="<div>" + element.id + "</div>";
      salida+="<div>" + element.name + "</div>";
      salida+="<div>" + element.year + "</div>";
      salida+="<div>" + element.color + "</div>";
      salida+="<div>" + element.pantone_value + "</div>";
    }
    salida+="</div>";
    $("#contenedor").html(salida);
  },
});
```

2. PETICIONES HTTP CON FETCH


- Por supuesto, el resultado es el mismo en ambos:

| ID | Nombre | Año | color | Pantone |
|----|----------------|------|---------|---------|
| 1 | cerulean | 2000 | #98B2D1 | 15-4020 |
| 2 | fuchsia rose | 2001 | #C74375 | 17-2031 |
| 3 | true red | 2002 | #BF1932 | 19-1664 |
| 4 | aqua sky | 2003 | #7BC4C4 | 14-4811 |
| 5 | tigerlily | 2004 | #E2583E | 17-1456 |
| 6 | blue turquoise | 2005 | #53B0AE | 15-5217 |

2. ENVIAR DATOS CON FETCH

- Para enviar datos podemos hacerlo de 2 formas: mediante GET y POST.
- Para enviar datos mediante GET normalmente concatenaremos el dato a la cabecera de la url

2. ENVIAR DATOS CON FETCH



```
// 20210131235435
// https://reqres.in/api/users

{
  "page": 1,
  "per_page": 6,
  "total": 12,
  "total_pages": 2,
  "data": [
    {
      "id": 1,
      "email": "george.bluth@reqres.in",
      "first_name": "George",
      "last_name": "Bluth",
      "avatar": "https://reqres.in/img/faces/1-image.jpg"
    },
    {
      "id": 2,
      "email": "janet.weaver@reqres.in",
      "first_name": "Janet",
      "last_name": "Weaver",
      "avatar": "https://reqres.in/img/faces/2-image.jpg"
    },
    {
      "id": 3,
      "email": "emma.wong@reqres.in",
      "first_name": "Emma",
      "last_name": "Wong",
      "avatar": "https://reqres.in/img/faces/3-image.jpg"
    },
  ]
}
```

2. ENVIAR DATOS CON FETCH



A screenshot of a web browser window displaying the JSON response from the API endpoint `reqres.in/api/users/2`. The browser's address bar shows the URL. The response is a JSON object with a `data` field containing user information and a `support` field containing a URL and a message. The JSON is formatted with syntax highlighting and line numbers on the left.

```
1 // 20210131235615
2 // https://reqres.in/api/users/2
3
4 {
5   "data": {
6     "id": 2,
7     "email": "janet.weaver@reqres.in",
8     "first_name": "Janet",
9     "last_name": "Weaver",
10    "avatar": "https://reqres.in/img/faces/2-image.jpg"
11  },
12  "support": {
13    "url": "https://reqres.in/#support-heading",
14    "text": "To keep ReqRes free, contributions towards server costs are appreciated!"
15  }
16 }
```

2. ENVIAR DATOS CON FETCH

- Método POST, aquí la cosa se complica un poco con respecto a lo que hemos visto hasta ahora sobre fetch.

```
✓ let datos = {  
  name: 'Juan',  
  job: 'Master Race'  
}  
datosJSON=JSON.stringify(datos);
```

2. ENVIAR DATOS CON FETCH

```
fetch('https://reqres.in/api/users', {  
  method: 'POST',  
  body: datosJSON,  
  headers: {  
    'Content-Type': 'application/json'  
  },  
})  
  .then(function (response) {  
    return response.json();  
  })  
  .then(function (json) {  
    console.log(json);  
    console.log(json.id);  
    console.log(json.name)  
  });
```


2. ENVIAR DATOS CON FETCH

- Si lo comparamos a jQuery:

| jQuery | Fetch |
|---|--|
| <pre>let datos = { name: 'Juan', job: 'Master Race' }</pre> | <pre>✓ let datos = { name: 'Juan', job: 'Master Race' } datosJSON=JSON.stringify(datos);</pre> |

2. ENVIAR DATOS CON FETCH

jQuery



```
$.ajax({  
  type: "POST",  
  url: "https://reqres.in/api/users",  
  data: datos,  
  dataType: "JSON",  
  success: function (response) {  
    console.log(response);  
    console.log(response.id);  
    console.log(response.name)  
  }  
});
```

Fetch

```
fetch('https://reqres.in/api/users', {  
  method: 'POST',  
  body: datosJSON,  
  headers: {  
    'Content-Type': 'application/json'  
  },  
})  
  .then(function (response) {  
    return response.json();  
  })  
  .then(function (json) {  
    console.log(json);  
    console.log(json.id);  
    console.log(json.name)  
  })  
});
```

2. ENVIAR DATOS CON FETCH

- Al igual que antes, produce el mismo resultado en ambos casos.

| | | |
|---|--|--------------------------|
|  <code>{name: 'Juan', job: 'Master Race', id: '760', createdAt: '2022-01-31T22:30:16.516Z'}</code> | | <code>fetch.js:20</code> |
| 760 | | <code>fetch.js:21</code> |
| Juan | | <code>fetch.js:22</code> |
|  | | |

3. ASYNC / AWAIT

- En la versión de JavaScript ES2017 se introducen las palabras clave **async/await**.
- No son más que una forma de simplificar la sintaxis para gestionar las promesas de una forma más sencilla.
- Con async/await seguimos utilizando promesas, pero abandonamos el modelo de encadenamiento de **.then()** para utilizar uno en el que trabajamos de forma más tradicional.

3.1 ASYNC / AWAIT. La palabra clave async

- En primer lugar, tenemos la palabra clave **async**. Esta palabra clave se colocará previamente a **function**, para definirla así como una función asíncrona, el resto de la función no cambia:

```
async function funcion_asincrona() {  
  return 42;  
}
```

- En el caso de que utilicemos función flecha, se definiría colocando el **async** justo antes de los parámetros de la función:

```
const funcion_asincrona = async () => 42;
```

3.2 ASYNC / AWAIT. La palabra clave await

- Cualquier función definida con **async**, o lo que es lo mismo, cualquier **promesa** puede utilizarse junto a la palabra clave **await** para manejarla.
- Lo que hace **await** es esperar a que se resuelva la promesa, mientras permite continuar ejecutando otras tareas que puedan realizarse:

3.2 ASYNC / AWAIT. La palabra clave await

```
const funcion_asincrona = async () => 42;  
  
document.querySelector("#boton").addEventListener("click", async () => {  
    const resultado=await funcion_asincrona();  
    console.log(resultado);  
});
```

3.3 Transformamos fetch con async/await

| .then | Await/async |
|--|--|
| <pre data-bbox="96 496 1113 868">' fetch('https://reqres.in/api/users?page=2') .then(respuesta => respuesta.json()) .then(datosjson => { console.log(datosjson); console.log(datosjson.total_pages); });</pre> | <pre data-bbox="1154 511 2428 875">async function peticion() { let respuesta = await fetch('https://reqres.in/api/users?page=2'); let datosjson = await respuesta.json(); console.log(datosjson); console.log(datosjson.total_pages); } peticion();</pre> |

3.3 Transformamos fetch con async/await

- Por supuesto, el resultado no varía

```
▶ {page: 2, per_page: 6, total: 12, total_pages: 2, data: Array(6), ...}
```

```
2
```