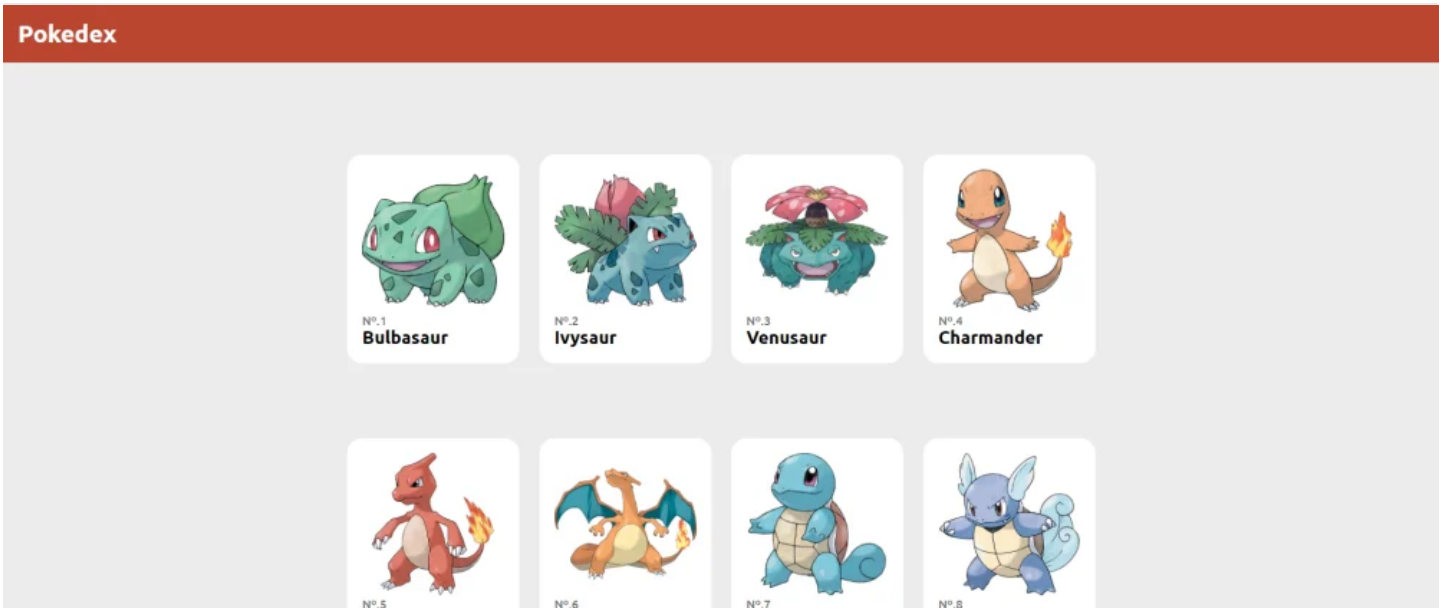


VikingCodeBlog
Posted on 5 abr

4 Maneras de llamar a una API Rest con JavaScript

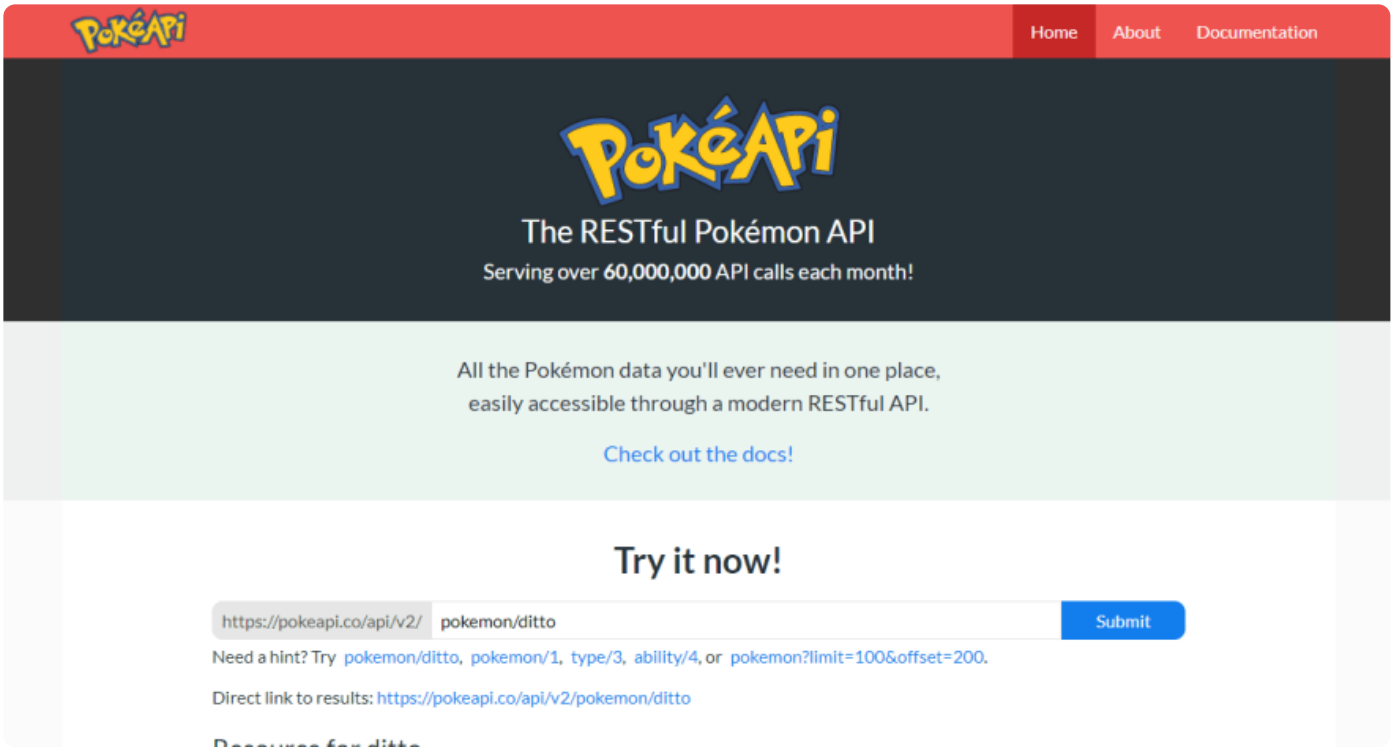
#javascript #webdev #angular #react

¿Qué es una API Rest?

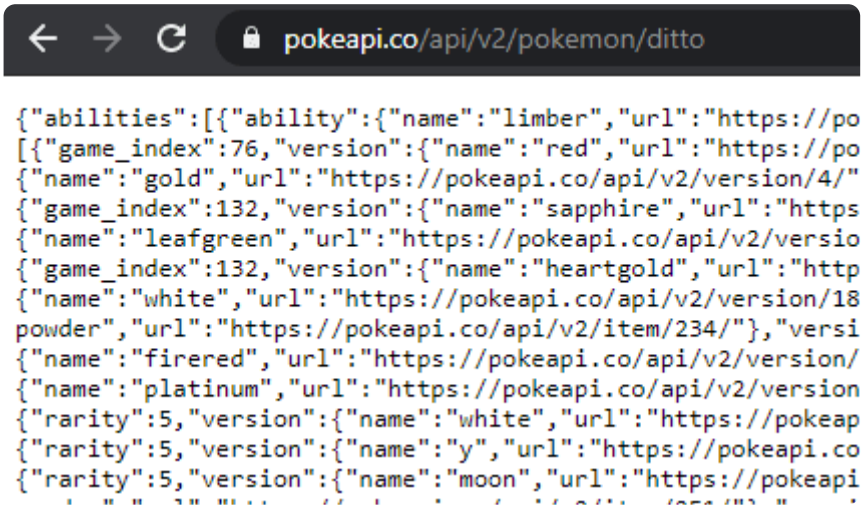
Antes de ver 4 Maneras de llamar a una API Rest con JavaScript, es posible que no conozcas que es esto de las API Rest, así que para ello, te comparto una definición que da BBVAAPIMarket, en esencia, una API Rest, es una interfaz que nos permite comunicarnos con otra aplicación que espera que hagamos una petición, puede ser para obtener datos, añadir datos, borrar...

Un Ejemplo

PokéApi es una API de prueba con temática nintendera, nos proporciona varias rutas como esta «https://pokeapi.co/api/v2/pokemon/ditto» la cual nos proporciona los datos del pokemon Ditto.



Puedes probar esta ruta simplemente desde tu navegador. Con ella obtenemos un JSON con todos los datos del pokemon.



4 Maneras de llamar a una API Rest con JavaScript

Ahora sí, vamos a ver 4 Maneras de llamar a una API Rest con JavaScript, todas ellas son válidas, iré opinando sobre ellas mientras te las muestro.

1. XMLHttpRequest (AJAX)

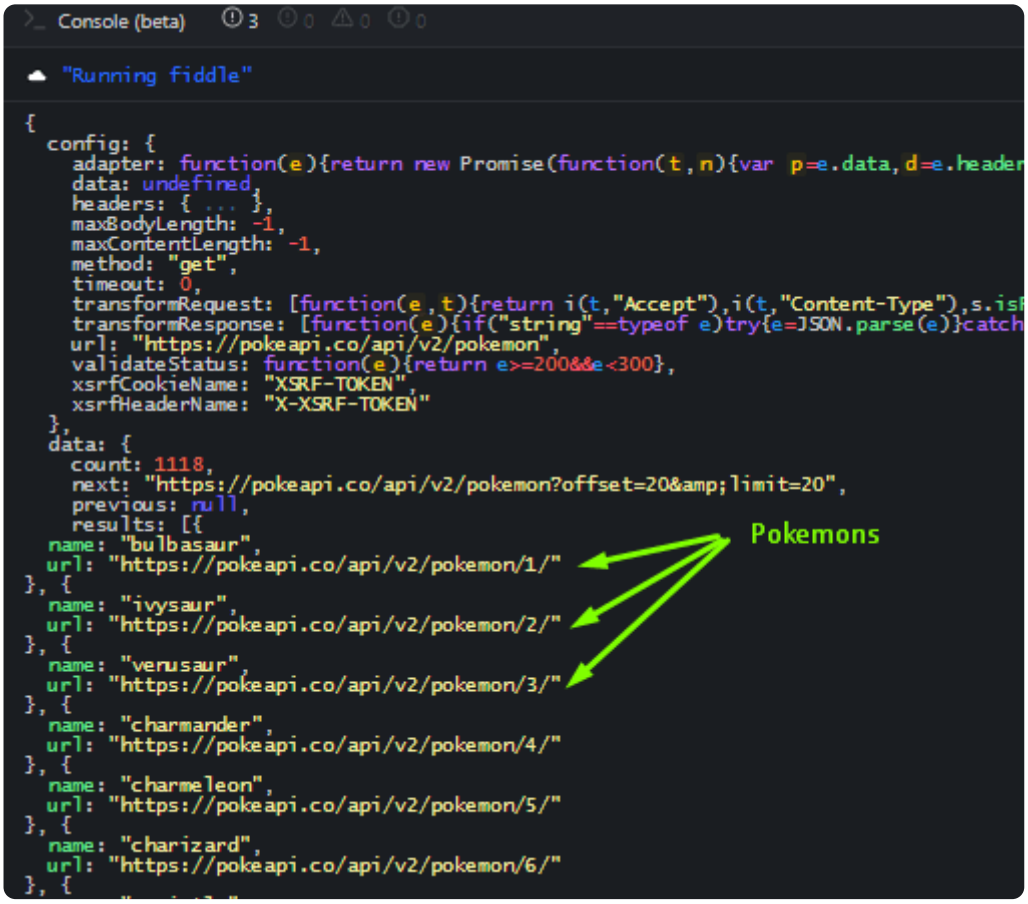
La primera forma de obtener datos que vamos a ver, va a ser mediante XMLHttpRequest, un objeto de JavaScript que fue diseñado por Microsoft y adoptado por Mozilla, Apple y Google.

Es un estándar de la W3C. Seguro que has oído hablar de la programación AJAX (Asynchronous JavaScript And XML), Esto era una forma de programar usando un conjunto de tecnologías que te permitían crear páginas más dinámicas, al hacer peticiones al backend para obtener datos nuevos sin tener que recargar la página al completo.

AJAX suena increíble ya que hoy en día todas las páginas webs hacen peticiones al backend sin necesidad de recargar la página actual pero estamos hablando de un término creado en 2005.

```
// Creamos un nuevo XMLHttpRequest
var xhttp = new XMLHttpRequest();

// Esta es la función que se ejecutará al finalizar la llamada
xhttp.onreadystatechange = function() {
  // Si nada da error
  if (this.readyState == 4 && this.status == 200) {
    // La respuesta, aunque sea JSON, viene en formato texto, por lo que tendremos que
    console.log(JSON.parse(this.responseText));
  }
};

// Endpoint de la API y método que se va a usar para llamar
xhttp.open("GET", "https://pokeapi.co/api/v2/pokemon", true);
xhttp.setRequestHeader("Content-type", "application/json");
// Si quisieramos mandar parámetros a nuestra API, podríamos hacerlo desde el método s
xhttp.send(null);
```

De esta manera obtenemos el siguiente resultado:



2. Fetch

Fetch es una nueva implementación de JavaScript, nos permite hacer lo mismo que **XMLHttpRequest** pero de manera más sencilla, el único defecto que le puedo ver es que al ser tan nueva, no funciona en internet Explorer, pero bueno, esto es un problema soportable para casi cualquier proyecto moderno.

```
fetch('https://pokeapi.co/api/v2/pokemon')
      .then(response => response.json())
      .then(json => console.log(json));
```

Con solo estas líneas estaríamos obteniendo el mismo resultado que en el código anterior con XMLHttpRequest, no es necesario importar ni instanciar fetch, simplemente se le llama😮.

3. Axios

Hasta ahora, tanto **XMLHttpRequest** como **Fetch** eran nativas de JavaScript, pero Axios es una librería externa que tendremos que importar en nuestro proyecto antes de usarla.

```
<head>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
</head>
```

Una vez importada, podrás usarla en tu proyecto.

```
axios.get('https://pokeapi.co/api/v2/pokemon')
  .then(function (response) {
    // función que se ejecutará al recibir una respuesta
    console.log(response);
  })
  .catch(function (error) {
    // función para capturar el error
    console.log(error);
  })
  .then(function () {
    // función que siempre se ejecuta
  });
```

¿Por qué se usa tanto Axios? Parece que Fetch es perfecto ¿para que iba a querer importar otra librería a mi proyecto?

El primer motivo es sencillo, Fetch es nuevo y no puede implementarse en proyectos que sigan usando tecnologías viejas, algunas veces está limitado, mientras que Axios tiene muy buena compatibilidad.

Pero hay más motivos, por ejemplo, Axios te permite añadir un timeout a la llamada para que se cierre cuando lleva un rato intentando obtener datos sin éxito.

Otro motivo muy importante es que Axios parsea automáticamente las respuestas JSON.

```
// axios
axios.get('https://pokeapi.co/api/v2/pokemon')
  .then(response => {
    console.log(response.data); // response.data ya es un JSON
  }, error => {
    console.log(error);
  });

// fetch()
fetch('https://pokeapi.co/api/v2/pokemon')
  .then(response => response.json())    // a fetch le llega una respuesta en string qu
  .then(data => {
    console.log(data)
  })
  .catch(error => console.error(error));
```

Axios tiene más funcionalidades como los interceptores, que te permiten interceptar las llamadas y reaccionar a ellas, esto se usa por ejemplo cuando nuestro backend tiene un sistema de seguridad que necesita que las llamadas lleven un token, podemos meterle el token a la llamada desde un interceptor para no tener que picarlo en código cada vez que lo vayamos a usar.

```
axios.interceptors.request.use(config => {
  // Aquí podríamos hacer algo con la llamada antes de enviarla
  console.log('Se ha enviado algo');
  return config;
});

// llamada común
axios.get('https://pokeapi.co/api/v2/pokemon')
  .then(response => {
    console.log(response.data);
  });
```

Creo que me he detenido demasiado tiempo en Axios, pero me interesaba hacer un pequeño análisis de por que es la librería que más he visto en los proyectos por los que he pasado.

4. jQuery.Ajax()

Obviamente si queremos hacer este tipo de llamadas, tendremos que importar la librería de jQuery a nuestro proyecto.

jQuery es una librería que muchos dan por muerta, pero creo que todavía le queda mucho recorrido, no hay más que ver las encuestas de StackOverflow o StateOfJs, sigue siendo una de las librerías/frameworks más usados, por lo que creo que es necesario conocer esta manera de llamar a una API Rest con jquery.

```
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
</head>
```

```
$.ajax({
  type: 'GET',
  url: 'https://pokeapi.co/api/v2/pokemon',
  dataType: 'json',
  success: function(data) {
    console.log(data)
```

```
  }
});
```

Realmente es un método bastante sencillo de utilizar.

Poniendo la teoría en práctica

Vamos a hacer uso de la API de PokéApi para obtener datos y pintar nuestra propia Pokedex.

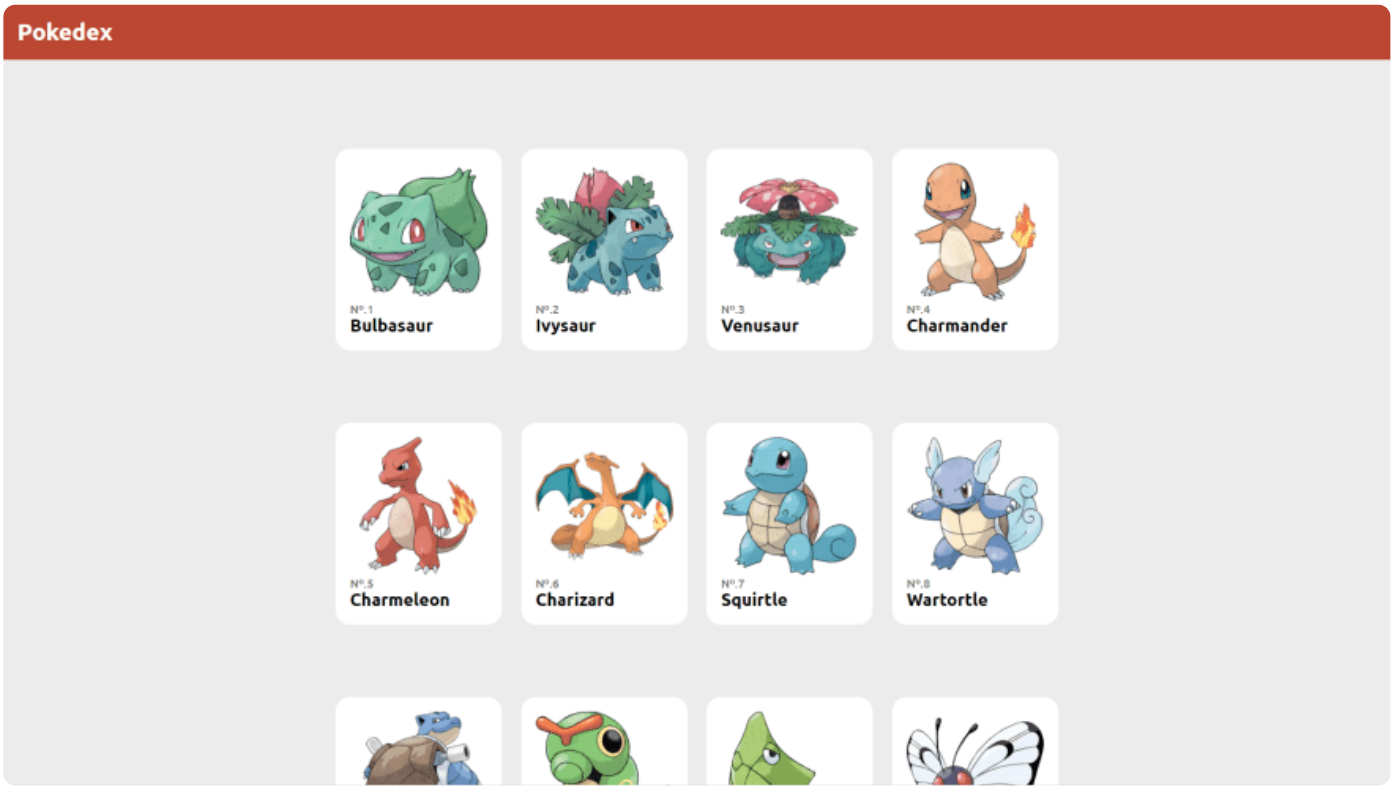
```
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Ubuntu:ital,wght@0,300;0,400;
</head>
<body>
    <div class="head">
      <h1>Pokedex</h1>
    </div>
    <div id="container"></div>
</body>
</html>
```

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }

  body{
    background-color: #ececec;
    font-family: 'Ubuntu', sans-serif;
  }

  .head{
    background-color: rgb(187, 70, 49);
    padding: 20px;
    position: fixed;
    top: 0;
    width: 100%;
    color: white;
  }
```

```
  #container{
    width: min(100%, 1000px);
    margin: 100px auto;
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
  }

  .card{
    width: 230px;
    margin-top: 100px;
    background-color: white;
    padding: 20px;
    border-radius: 20px;
  }

  .card img {
    width: 100%;
  }

  .card span{
    color: #6e6e6e;
    font-weight: 500;
  }


// Obtenemos los datos de todos los pokemon
fetch('https://pokeapi.co/api/v2/pokemon')
        .then(response => response.json())
        .then(json => {
            printPokemons(json.results);
        });

// Pinta todos los pokemos insertando un HTML dentro del #container
function printPokemons(pokemons) {
  const container = document.getElementById('container')
  pokemons.forEach(pokemon => {
    container.innerHTML = `
    ${container.innerHTML}
    <div class="card">
    <img src="https://pokeres.bastionbot.org/images/pokemon/${getPokemonId(pokemon.url
    <span>Nº.${getPokemonId(pokemon.url)}</span>
    <h2>${pokemon.name.charAt(0).toUpperCase() + pokemon.name.slice(1)}</h2>
    </card>
  `;
  });
}
```

```
// En esta ruta de la API no nos viene el id de cada pokemon, pero si que nos viene
// una URL, para poder obtener todos los datos de ese pokemon, la cual contiene su ID
// así que le extraigo el ID a la URL
function getPokemonId(url) {
  return url.replace('https://pokeapi.co/api/v2/pokemon/', '').replace('/','')
}
```

El resultado es el siguiente.



Más maneras de llamar a una API Rest desde JavaScript.

Me parece interesante destacar otras maneras de llamar a una API Rest usando JavaScript.

HttpClient

En el caso del framework de desarrollo Angular, se nos proporciona una librería para hacer llamadas a este tipo de servicios.

La librería HttpClient de Angular es muy potente, interceptores, testing, integrada con TypeScript... La verdad que dudo de que alguien use otra librería en proyectos de Angular.

SuperAgent

SuperAgent es una librería muy ligera, bastante similar a Axios.

Request

No conozco mucho Request, pero he visto muchas librerías de empresas que se basan en ella para implementar alguna funcionalidad propia, no se si será por algo en especial o simplemente es casualidad.

Conclusiones

Uso fetch siempre que voy a desarrollar un ejemplo para el blog, pero realmente si quisiera crear un proyecto entero con JavaScript, Axios sería mi primera opción ya que tiene varias funcionalidades muy sencillas y potentes.

jQuery.Ajax, no creo que la use ya que no trabajo en proyectos que contengan esa librería, pero nunca se sabe.

Y XMLHttpRequest, ahora que Microsoft ha matado internet Explorer, creo que ya no será necesario y pondré a fetch por delante.

Discussion (2)

maxdevjs • Apr 8

wow.. por suerte empecé a estudiar español 😃 Me encantó este tutorial

pokeapi.co/api/v2/pokemon/ditto» (pokeapi.co/api/v2/pokemon/ditto%C2%BB) is a broken link, should be pokeapi.co/api/v2/pokemon/ditto

Diego • Sep 24

hermoso <3

Code of Conduct • Report abuse

VikingCodeBlog

Desarrollador web, escribo un blog y conquisto países en mis ratos libres

LOCATION
Valencia

JOINED

5 abr 2021

Trending on DEV Community 🔥

5 things I struggled with when learning React with a Vue background

#vue #react #javascript #webdev

Javascript snippets you need to know right now 🔥 - #3

#javascript #webdev #beginners #programming

🤯 Did you know there are F13-F24 keys? 🤯

#discuss #todayilearned #watercooler