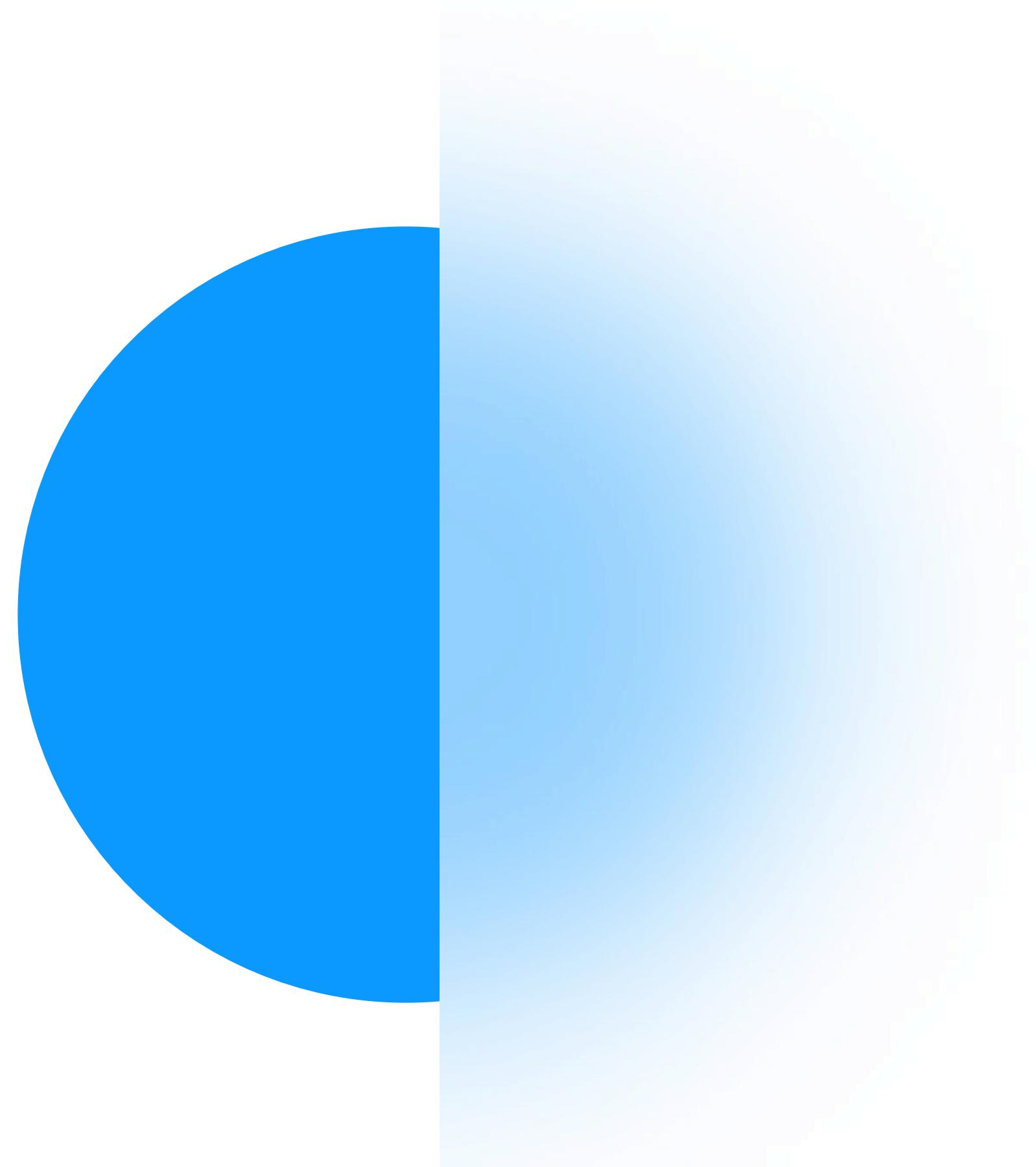


JS (JavaScript)

Les 3 API & Fetch data (pokeapi)



الموضوع	الوقت (التوقيت السوري)	التاريخ
HTML: Basics & Elements & page anatomy	12:00	2025\3\1
HTML: Basics check	12:00	2025\3\8
CSS: Basics & Selectors	12:00	2025\3\15
Typography & Colors	12:00	2025\3\22
Positions & :has() :not()	12:00	2025\3\29
Flexbox	12:00	2025\4\5
Grid	19:30	2025\4\7
States & Media Querie	19:30	2025\4\9
Animations	19:30	2025\4\11
JS: Basics & Events & First function (html+css+js)	19:30	2025\4\14
JS: Conditional Statements & Arrays & Loops	19:30	2025\4\16
JS: API & Fetch data (pokeapi)	19:30	2025\4\18
Last check	19:30	2025\4\20

API

| Application Programming Interface) API هي مجموعة من القواعد والبروتوكولات التي تتيح لتطبيقات مختلفة التفاعل مع بعضها البعض. في الويب، تُستخدم الـ API لجلب أو إرسال البيانات من/ إلى الخادم.



API

- أهمية الـ API:

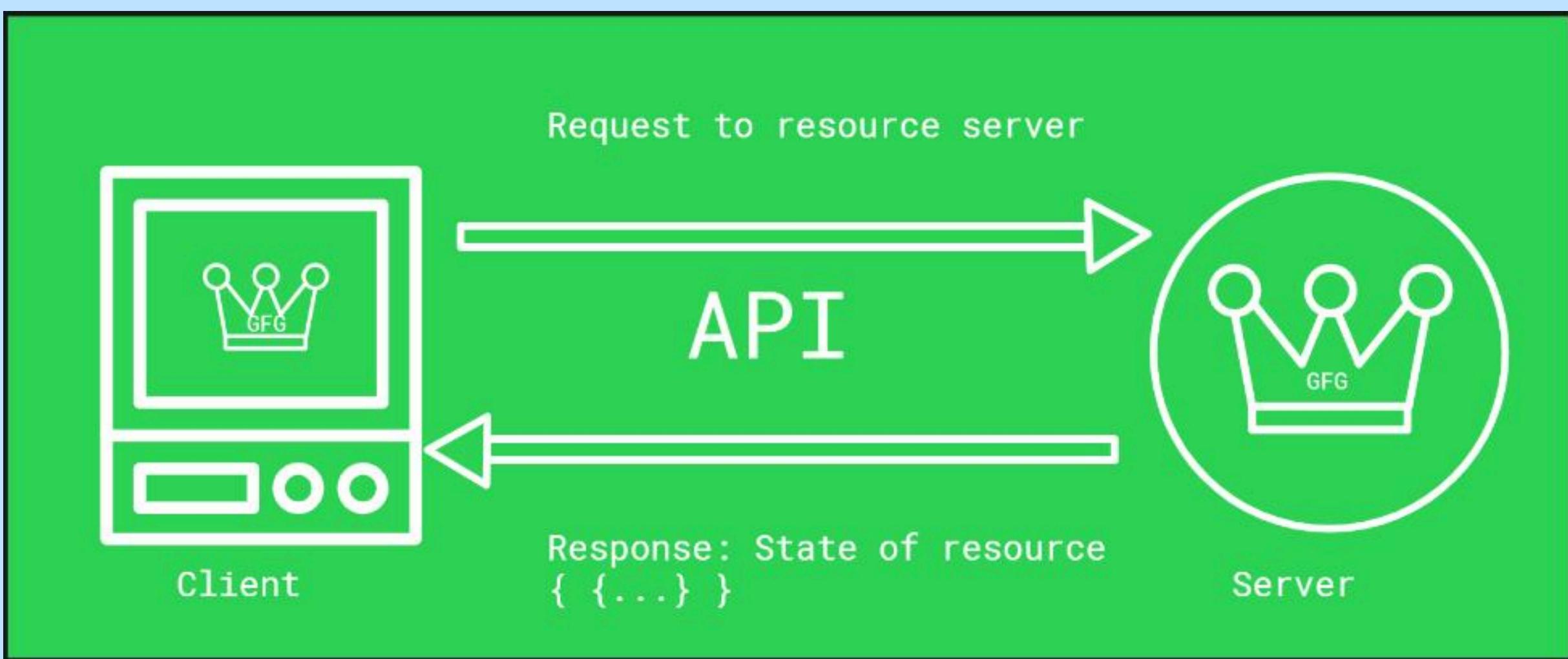
تُستخدم لجلب بيانات ديناميكية من الخادم، ويمكن دمجها مع تقنيات الويب الأخرى لبناء تطبيقات متقدمة.

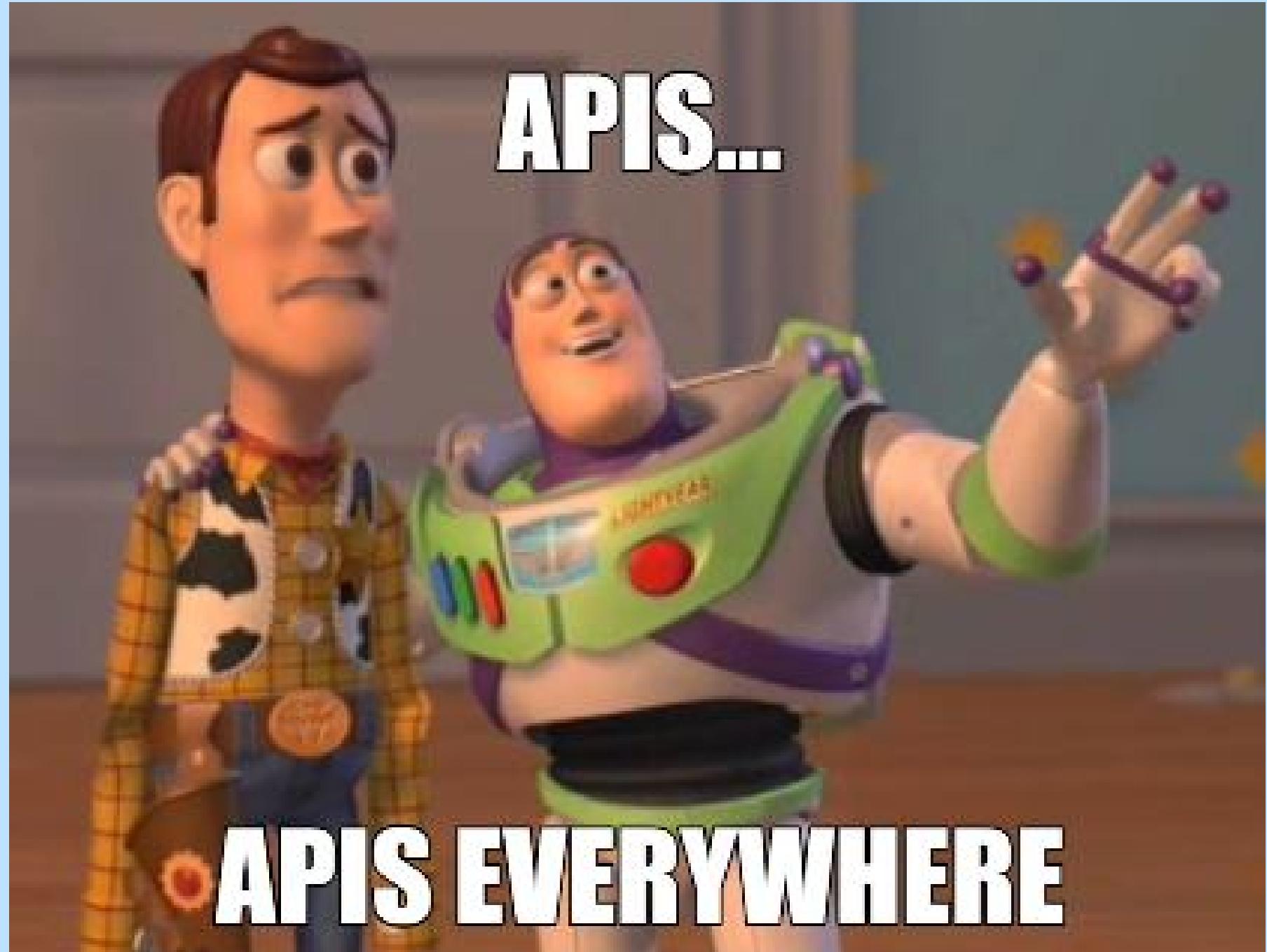
- التعامل مع الأخطاء:

من الضروري دائمًا التحقق من حالة الاستجابة والتعامل مع الأخطاء بشكل مناسب (مثل التحقق من `response.ok`).

- التعامل مع البيانات:

غالبًا ما تكون البيانات المرسلة من الـ API بصيغة JSON، لذا يتم استخدام `JSON.parse()` لتحويل الاستجابة إلى كائنات قابلة للاستخدام في JavaScript.





API

كيف يمكننا التعامل مع الـ API ؟

`fetch()` •

تُستخدم لجلب البيانات من عنوان URL (عادةً باستخدام بروتوكول HTTP).

• تعمل بطريقة Promise مما يسمح بالتعامل مع العمليات غير المتزامنة.

`async/await`

تساعد على كتابة كود غير متزامن بطريقة تبدو متزامنة، مما يحسن من قراءة الكود وصيانته.

• لماذا نستخدم Promises ؟

- للتعامل مع تأخير الشبكة.
- لتحسين قراءة الكود بدلاً من التعشيش في الـ callbacks.
- لتوحيد طريقة التعامل مع أي عملية غير متزامنة.
- (استدعاء راجع) callback
- هو ببساطة دالة يتم تمريرها كوسيلة لتنفيذها لاحقاً، عادةً بعد انتهاء مهمة معينة.

```
function getDataCallback(callback) {
  setTimeout(() => {
    callback('البيانات وصلت');
  }, 1000);
}

getDataCallback(function(result) {
  console.log(result); // البيانات وصلت
});
```

- العيب: إذا احتجت إلى سلسلة من العمليات، يصبح الكود متداخلاً ومعقداً

```
getDataCallback(function(result1) {
  getDataCallback(function(result2) {
    getDataCallback(function(result3) {
      // جحيم التعشيش callback hell
    });
  });
});
```

• ما هو ال Promise ؟

Promise هو كائن يمثل عملية غير متزامنة (asynchronous) قد تنجح أو تفشل في المستقبل. الفكرة أنك "تَعد" بأنك سترجع قيمة لاحقاً، وليس الآن.

- القصد من "إرجاع Promise": عندما تقول دالة إنها ترجع Promise، فهذا يعني:
 - العملية بداخلها تأخذ وقتاً (مثلاً تحميل بيانات من الإنترنت).
 - أنت لا تحصل على النتيجة فوراً، بل تحصل على وعد بأنك ستحصل عليها لاحقاً.
 - يمكنك استخدام await() أو then() للحصول على القيمة عندما تصبح جاهزة.

```
async function loadData() {
  const result = await getDataPromise();
  console.log(result);
}

loadData();
```

constant API url



```
1 let dataResults = [];
2
3 async function apiInfo() {
4   const url = "https://pokeapi.co/api/v2/pokemon/?limit=1"; // ?limit=3 is used to limit the data to 1 result
5   try {
6     const response = await fetch(url);
7     const data = await response.json();
8     console.log("data:", data); ----->
9
10    dataResults = data.results;
11    console.log("dataResults:", dataResults); ----->
12
13    dataResults.forEach(printData);
14  } catch (error) {
15    console.error(error.message);
16  }
17 }
18
19 function printData(item, index) {
20   console.log("printData:", "item:", item.name, "index:", index); ----->
21 }
22
23 apiInfo();
```

```
data: script.js:8
  {count: 1304, next: 'https://pokeapi.co/
  ▶ api/v2/pokemon/?offset=1&limit=1', previous: null, results: Array(1)}
dataResults: script.js:11
  ▼ [{}]
    ▶ 0: {name: 'bulbasaur', url: 'https://pc
      length: 1
    ▶ [[Prototype]]: Array(0)
printData: item: bulbasaur script.js:20
index: 0
```

Dynamic API Link

```
● ● ●  
1 let dataResults = [];  
2 let pokemonName = "pikachu";  
3  
4 async function apiInfo() {  
5   const url = `https://pokeapi.co/api/v2/pokemon/${pokemonName}`;  
6   try {  
7     const response = await fetch(url);  
8     const data = await response.json();  
9     dataResults = data;  
10    console.log("dataResults:", dataResults);  
11  } catch (error) {  
12    console.error(error.message);  
13  }  
14}  
15  
16 apiInfo();
```

```
dataResults: script.js:10  
  abilities: Array(2), base_experience: 1  
  ▼ 12, cries: {...}, forms: Array(1), game_in  
  dices: Array(20), ...} ⓘ  
    ► abilities: (2) [{...}, {...}]  
    base_experience: 112  
    ► cries: {latest: 'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/112.png'}  
    ► forms: [{...}]  
    ► game_indices: (20) [{...}, {...}, {...}, {...},  
      height: 4  
    ► held_items: (2) [{...}, {...}]  
    id: 25  
    is_default: true  
    location_area_encounters: "https://pokeapi.co/api/v2/location-area/112"  
    ► moves: (105) [{...}, {...}, {...}, {...}, {...},  
      name: "pikachu"  
      order: 35  
    ► past_abilities: []  
    ► past_types: []  
    ► species: {name: 'pikachu', url: 'https://pokeapi.co/api/v2/species/112'}  
    ► sprites: {back_default: 'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/back/112.png'}  
    ► stats: (6) [{...}, {...}, {...}, {...}, {...}, {...}]  
    ► types: [{...}]  
    weight: 60  
  ► [[Prototype]]: Object
```

Add elements to html with js [appendChild]



```
1 // call the list item from the html (where the pokemons names will be displayed)
2 const pokemonsList = document.querySelector(".pokemonsNames");
3
4 let dataResults = [];
5
6 // Fetch the data from the API
7 async function apiInfo() {
8   const url = "https://pokeapi.co/api/v2/pokemon/?limit=6";
9   try {
10     const response = await fetch(url);
11     const data = await response.json();
12     dataResults = data.results;
13     // send the data to the printData function one by one
14     dataResults.forEach(printData);
15   } catch (error) {
16     console.error(error.message);
17   }
18 }
19
20 function printData(item, index) {
21   // item is the current element in the array
22   const li = document.createElement("li"); // Create a <li> node
23   li.textContent = index + ") " + item.name; // Add text to the <li> node
24   pokemonsList.appendChild(li); // Append the <li> node to <ul>
25 }
26
27 apiInfo();
```

html



```
1 <body>
2   <ul class="pokemonsNames"></ul>
3
4   <script src="./script/script.js"></script>
5 </body>
```

In the browser

0) Bulbasaur

1) Ivysaur

2) Venusaur

3) Charmander

4) Charmeleon

5) Charizard

Add elements to html with js [.innerHTML]



```

1 // call the list item from the html (where the pokemons names will be displayed)
2 const pokemonsList = document.querySelector(".pokemonsNames");
3
4 let dataResults = [];
5
6 // Fetch the data from the API
7 async function apiInfo() {
8   const url = "https://pokeapi.co/api/v2/pokemon/?limit=6";
9   try {
10     const response = await fetch(url);
11     const data = await response.json();
12     dataResults = data.results;
13     // send the data to the printData function one by one
14     dataResults.forEach(printData);
15   } catch (error) {
16     console.error(error.message);
17   }
18 }
19
20 function printData(item, index) {
21   // item is the current element in the array
22   pokemonsList.innerHTML += `<li>${index}. ${item.name}</li>`;
23 }
24
25 apiInfo();
26

```

html



```

1 <body>
2   <ul class="pokemonsNames"></ul>
3
4   <script src="./script/script.js"></script>
5 </body>

```

In the browser

0. Bulbasaur**1. Ivysaur****2. Venusaur****3. Charmander****4. Charmeleon****5. Charizard**

JSON

بما يُستخدم JSON؟

1. تبادل البيانات بين الخادم والمتصفح:
 - عند استخدام `fetch()` أو AJAX في الموضع.
 - مثلاً: `fetch('/users')` يرجع بيانات JSON فيها معلومات المستخدمين.
2. حفظ الإعدادات أو البيانات مؤقتاً:
 - في ملفات `.json` أو في `.localStorage`.
3. واجهات برمجة التطبيقات (APIs):
 - أغلب APIs تُرجع أو تستقبل JSON.
4. برمجة الأنظمة وتطبيقات الهواتف:
 - يتم تبادل البيانات بين التطبيقات والخوادم عبر JSON أيضاً.

اختصار ل JavaScript Object Notation

وهو صيغة (تنسيق) خفيفة لتخزين وتبادل البيانات بين الخوادم والمتصفحات أو التطبيقات. يشبه شكل كائنات JavaScript، لكنه مجرد نص (Text) يمكن إرساله أو استلامه بسهولة.

```
{  
  "name": "Ali",  
  "age": 30,  
  "isStudent": false,  
  "skills": ["JavaScript", "HTML", "CSS"]  
}
```

The end of JS les 3