Web Application Development

Web APIs, JSON, FETCH, AXIOS

Web APIs

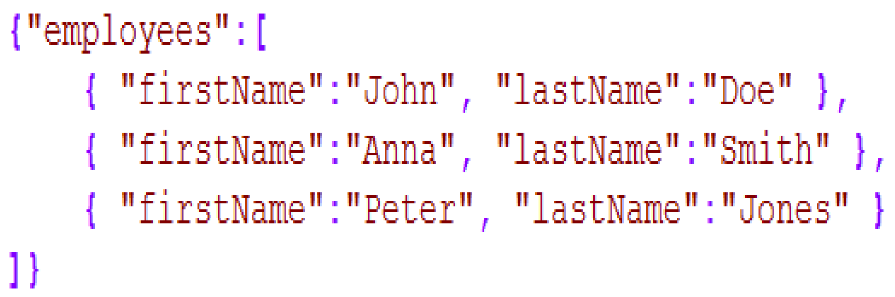
- A Web API (Application Programming Interface) is a set of rules and protocols that allows one application to communicate with another over the web.
- You can consume Web API Services using any front-end technology like JavaScript, Jquery, Angular or React.
- We will study Jquery to consume APIs, and Laravel to create our own API.

Web API

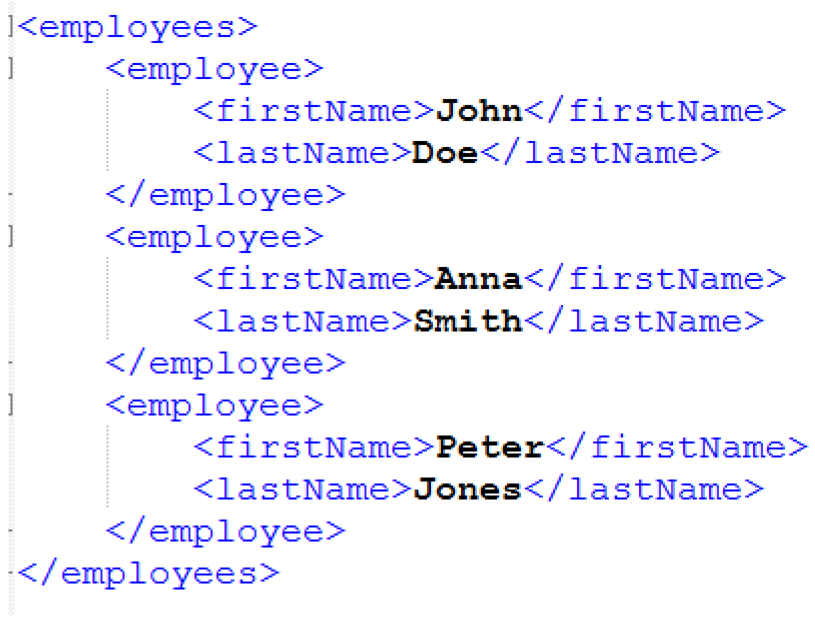
- Web APIs are created using server side lanaguages/technogies, and front-end consumes it.
- WebAPis are just a set of link that provides data in JSON formate.
  - For example, https://jsonplaceholder.typicode.com/users
  - The response message contains a JSON object.
  - Some APIs may return data in XML format.

JSON vs XML

- Comparing JSON with XML
  - The simple difference of JSON and XML are as following



```json
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```

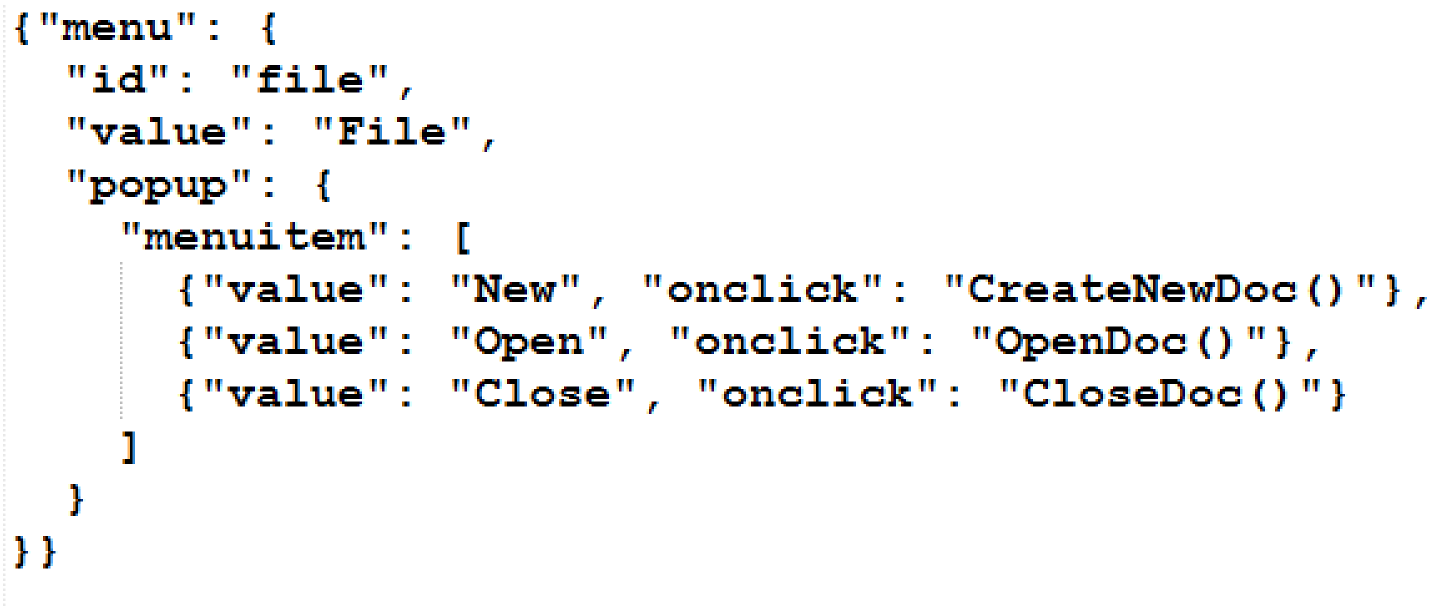


```xml
<employees>
    <employee>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName>
        <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName>
        <lastName>Jones</lastName>
    </employee>
</employees>
```

JSON **XML**

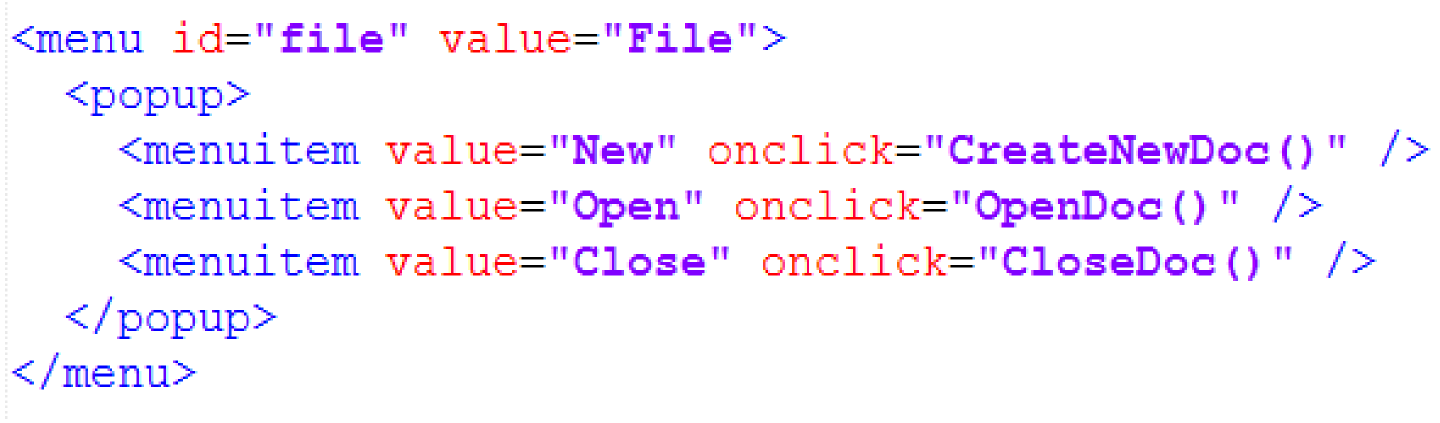
JSON VS XML

- JSON



```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

- XML



```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

JSON

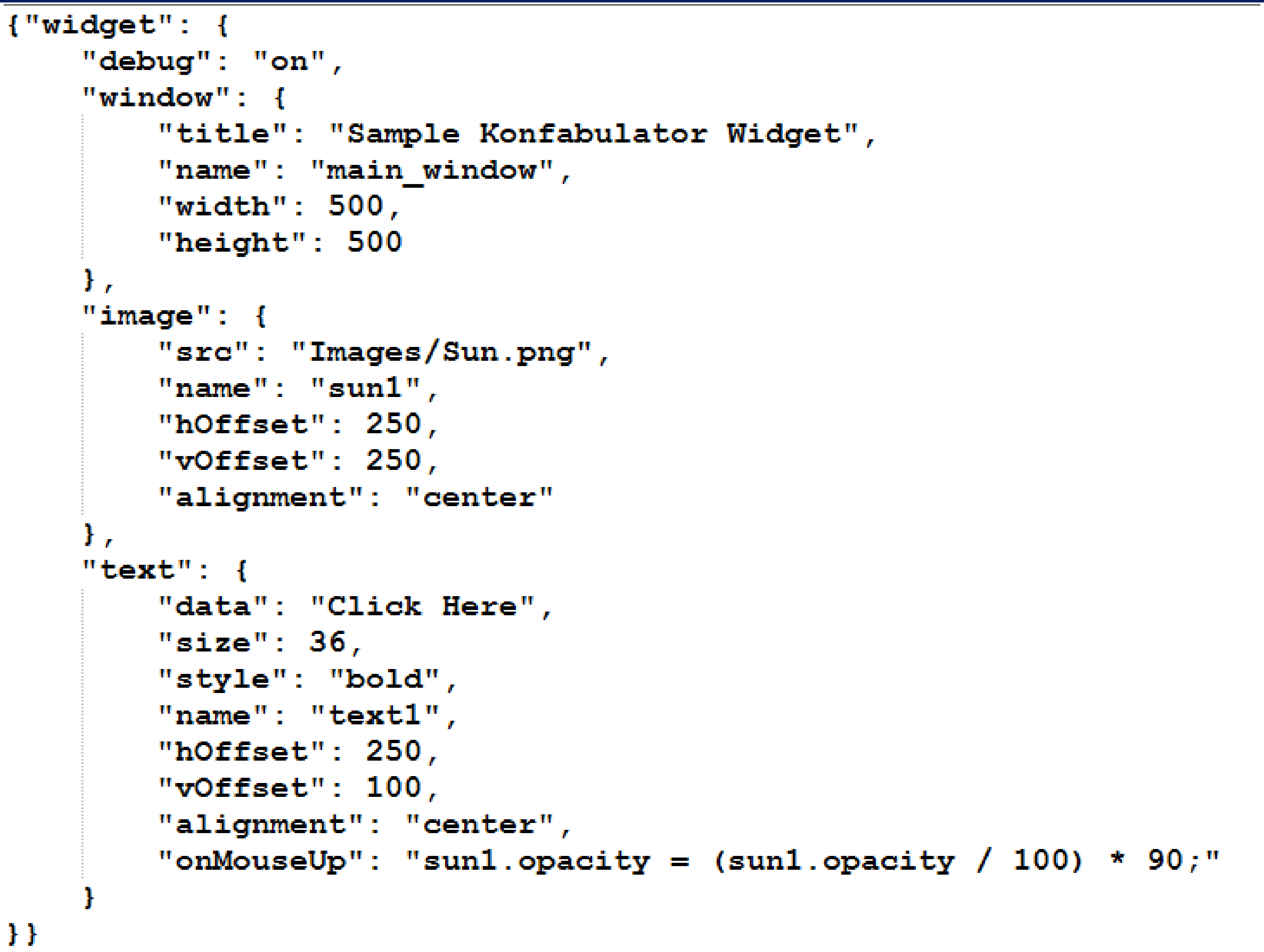
- JSON
  - **JSON** (JavaScript Object Notation) is a lightweight data-interchange format.
  - It is easy for humans to read and write.
  - It is easy for machines to parse and generate.
  - JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.

JSON

- JSON object Syntax
  - { "name":"John", "age":30, "car":null }
  - JSON objects are surrounded by curly braces {}.
  - JSON objects are written in key/value pairs.
  - Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
  - Keys and values are separated by a colon.
  - Each key/value pair is separated by a comma.

JSON VS XML

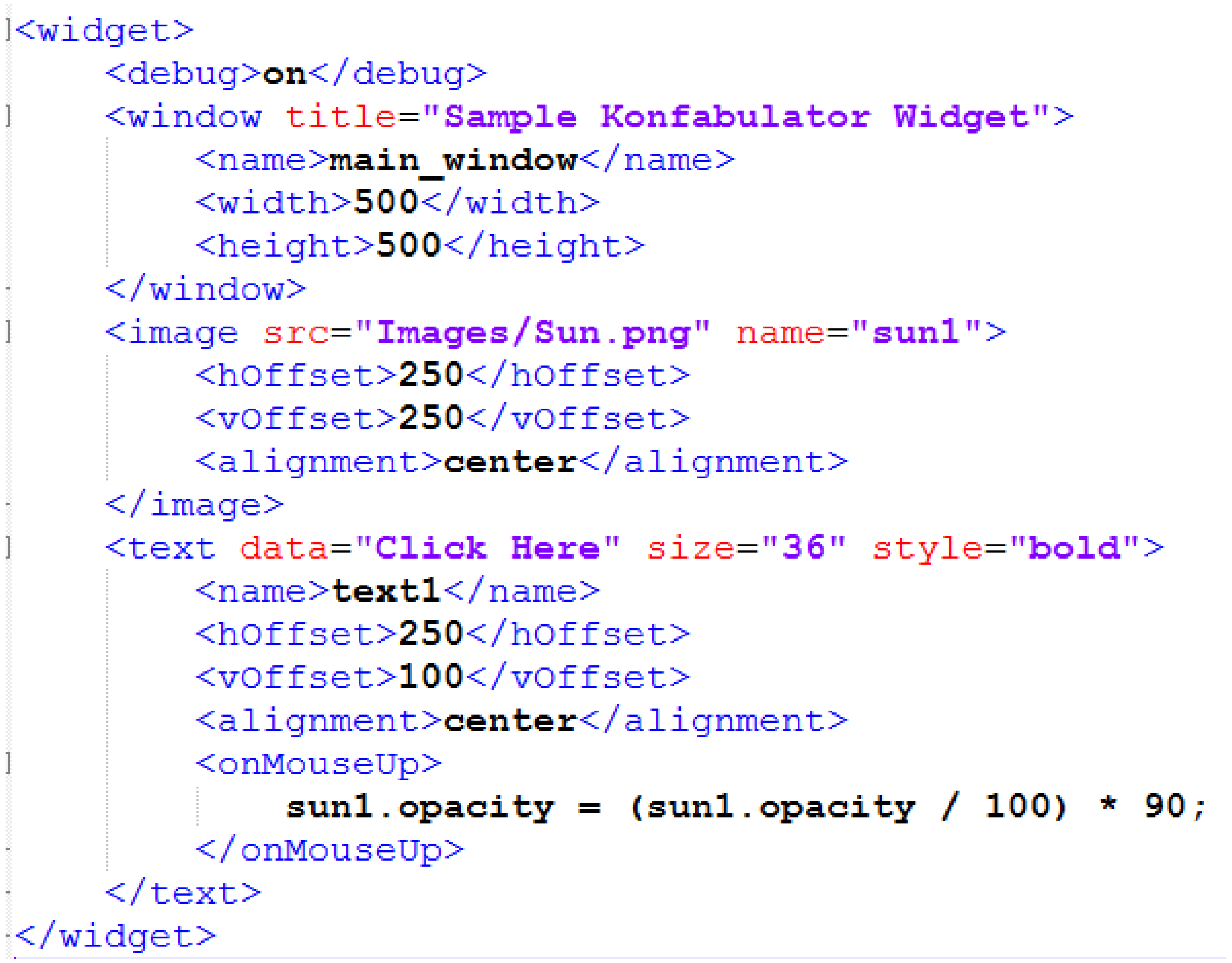
- JSON



```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "name": "sun1",
        "hOffset": 250,
        "vOffset": 250,
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "hOffset": 250,
        "vOffset": 100,
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

JSON VS XML

- XML



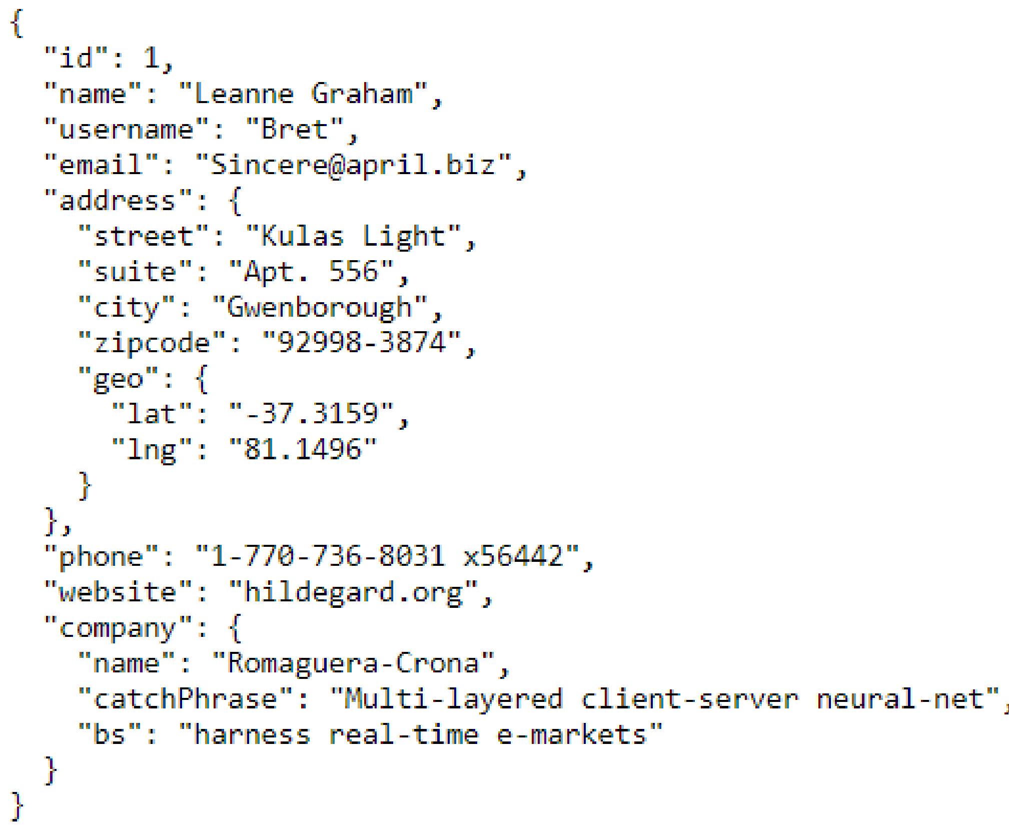
```xml
<widget>
    <debug>on</debug>
    <window title="Sample Konfabulator Widget">
        <name>main_window</name>
        <width>500</width>
        <height>500</height>
    </window>
    <image src="Images/Sun.png" name="sun1">
        <hOffset>250</hOffset>
        <vOffset>250</vOffset>
        <alignment>center</alignment>
    </image>
    <text data="Click Here" size="36" style="bold">
        <name>text1</name>
        <hOffset>250</hOffset>
        <vOffset>100</vOffset>
        <alignment>center</alignment>
        <onMouseUp>
            sun1.opacity = (sun1.opacity / 100) * 90;
        </onMouseUp>
    </text>
</widget>
```

JSON

- Comparing JSON with XML
  - Both are self descriptive
  - Both are hierarchical
  - Both can be parsed by programming languages
  - JSON is shorter and therefore quicker
  - JSON doesn’t uses tags as XML

WebAPI Response

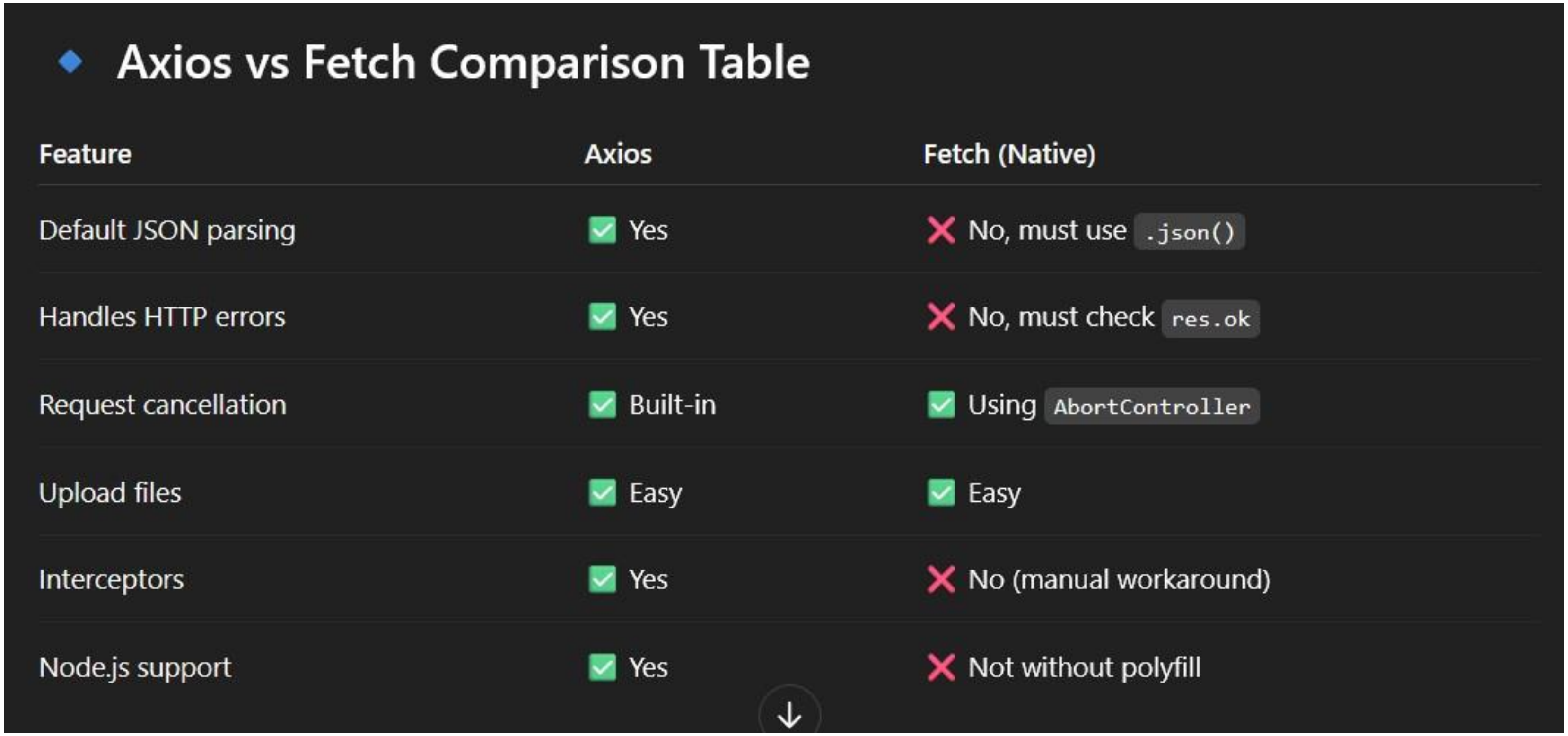
- https://jsonplaceholder.typicode.com/users/1
- The response message contains a json object as described in following



```
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
}
```

How to Call an WebAPIs

- Use Fetch
- Use AXIOS
  - npm install axios

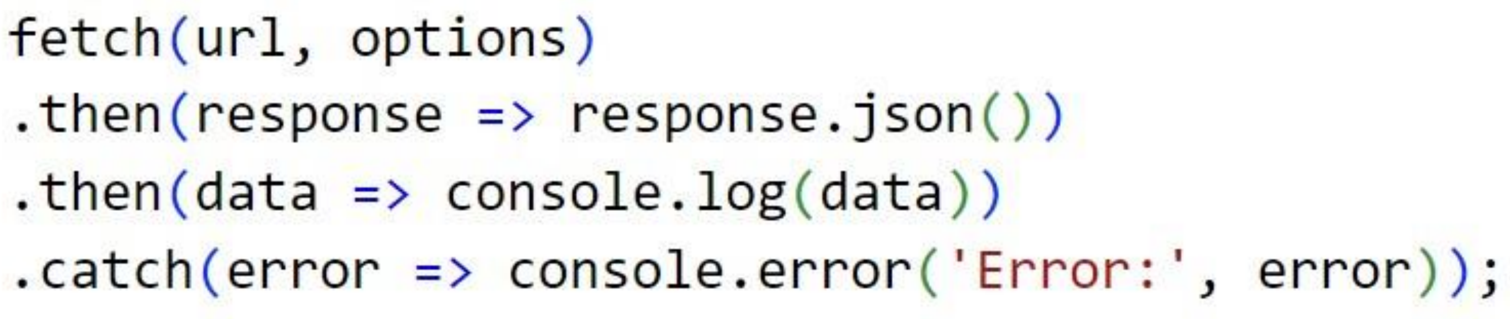


◆ **Axios vs Fetch Comparison Table**

Feature	Axios	Fetch (Native)
Default JSON parsing	✅ Yes	❌ No, must use `.json()`
Handles HTTP errors	✅ Yes	❌ No, must check `res.ok`
Request cancellation	✅ Built-in	✅ Using `AbortController`
Upload files	✅ Easy	✅ Easy
Interceptors	✅ Yes	❌ No (manual workaround)
Node.js support	✅ Yes	❌ Not without polyfill

Fetch API

Syntax



```javascript
fetch(url, options)
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```

Fetch Data using AXIOS

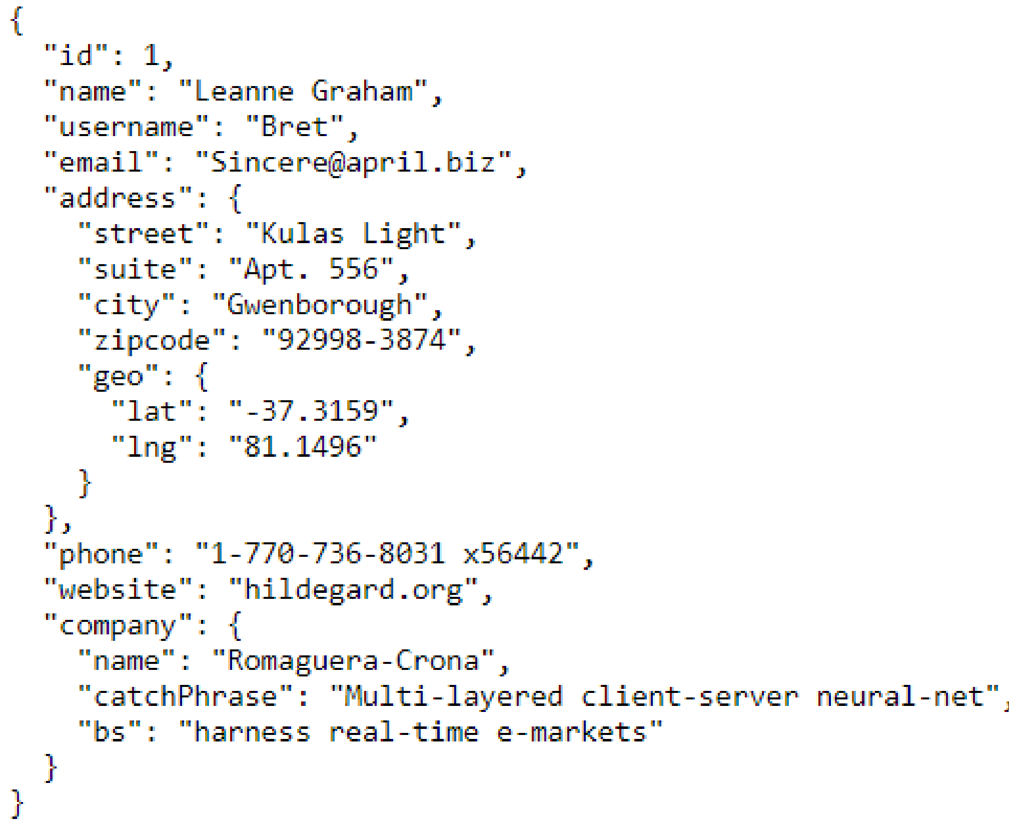
```
axios.get(‘Web-api link')
 .then(response => {
   console.log(response.data); })
 .catch(error => {
   console.error(error);
 });
```

AXIOS GET Request

```
axios.get(‘Web-api link')
 .then(response => {
   console.log(response.data); })
 .catch(error => {
   console.error(error);
 });
```

FETCH AXIOS response

- https://jsonplaceholder.typicode.com/users/1
- The response message contains a json object as described in following

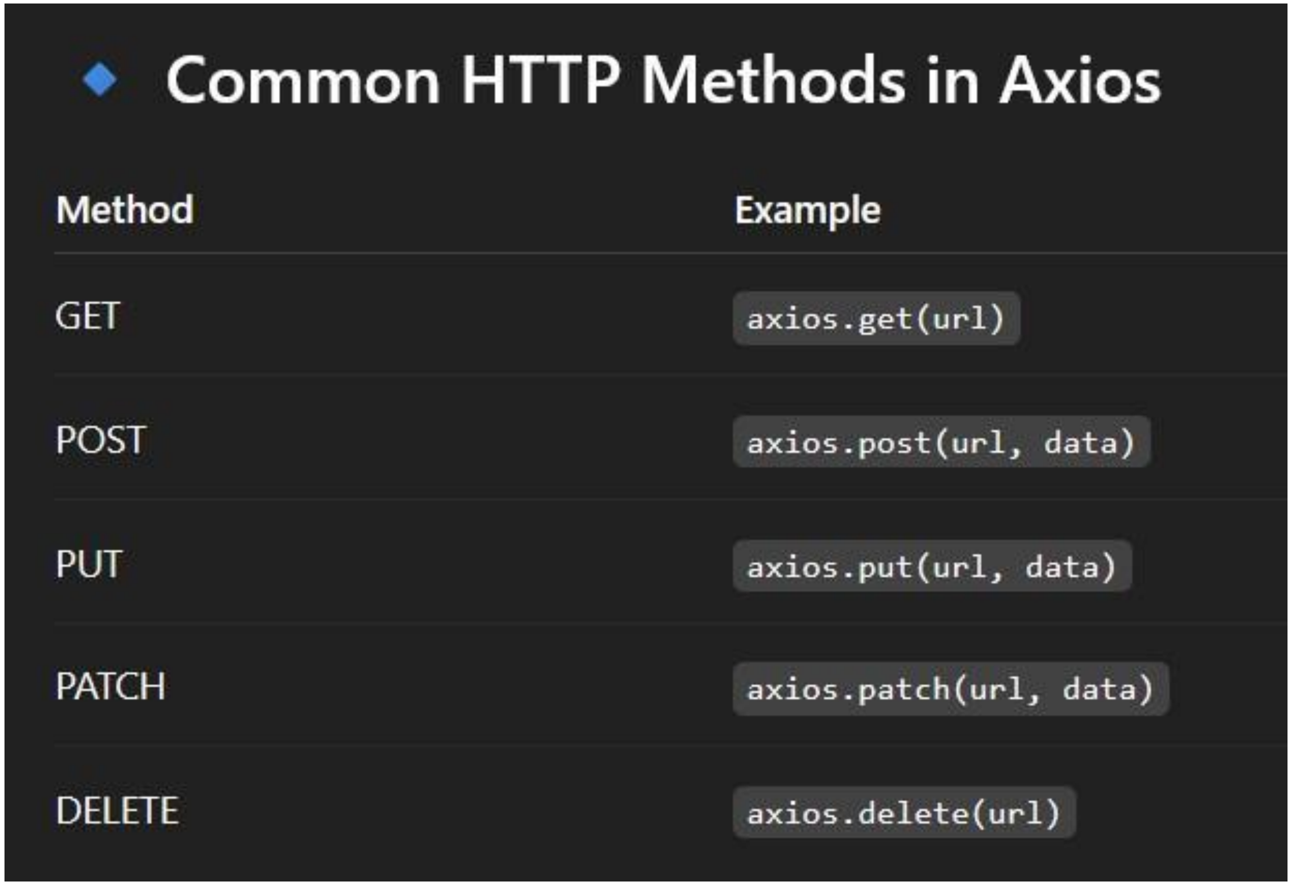


```
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
}
```

AXIOS POST Request

```
axios.post('https://jsonplaceholder.typicode.com/posts', {
  title: 'New Post',
  body: 'Post content',
  userId: 1
})
.then(response => {
  console.log(response.data);
})
.catch(error => {
  console.error(error);
});
```

AXIOS POST Request



◆ Common HTTP Methods in Axios

Method	Example
GET	`axios.get(url)`
POST	`axios.post(url, data)`
PUT	`axios.put(url, data)`
PATCH	`axios.patch(url, data)`
DELETE	`axios.delete(url)`

Async and Await (modern javascript)

In JavaScript, async and await are used to handle asynchronous operations more cleanly and clearly than using traditional Promises or callbacks.

- **Async:** When you declare a function with async, it automatically returns a **Promise**, even if you return a non-promise value. This lets you use await inside that function.
- **Await:** The await keyword can only be used inside an async function. It pauses the execution of the function until the awaited Promise is resolved or rejected

Async and Await Example

```
async function getUser() {
  try {
    let response = await fetch("/user");
    let user = await response.json();
    console.log(user);
  } catch (error) {
    console.error("Error:", error);
  }
}
```

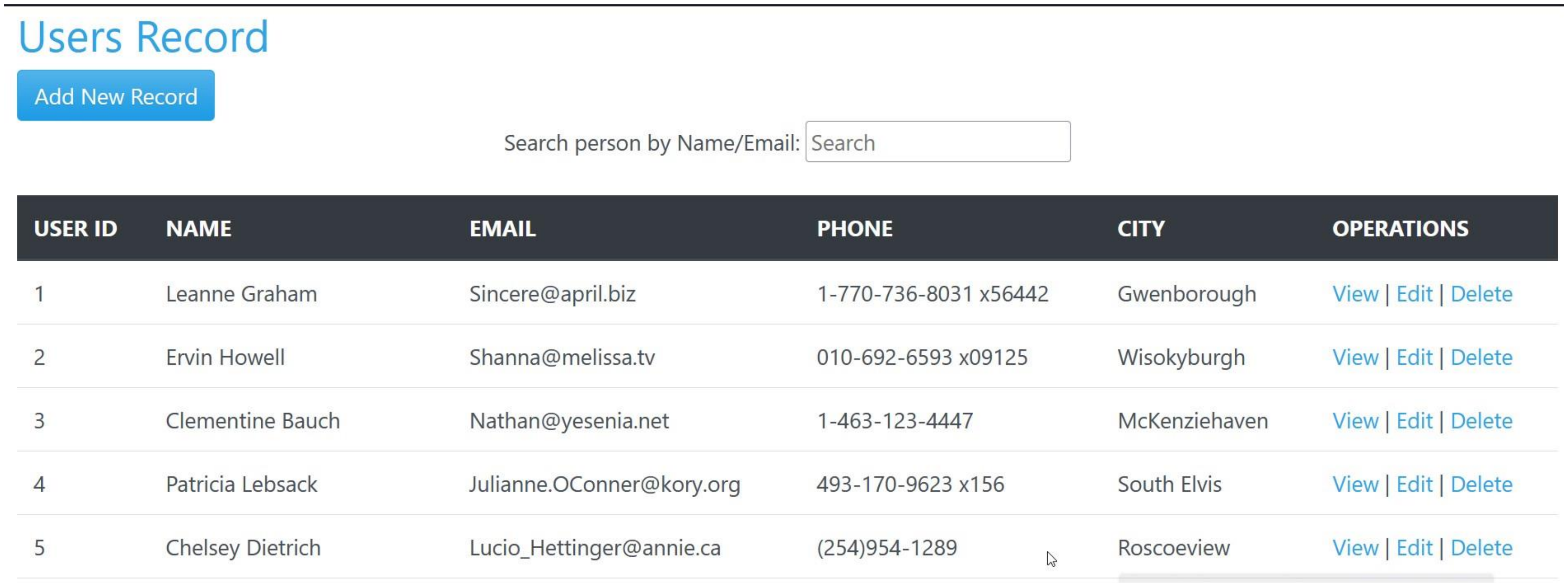
Task

- Once you have web APIs in hand, now it is possible to consume them from any type of application irrespective of their technology.
- We can use React to send some new data to the server. And get response from server in JSON format, Process or display the data using DOM.

Task

- Create a dashboard where you can execute
  - Add
  - Update
  - Get
  - Get all
  - Delete
- Operation on
  - http://exampleapi.somee.com/api/person

Dashboard



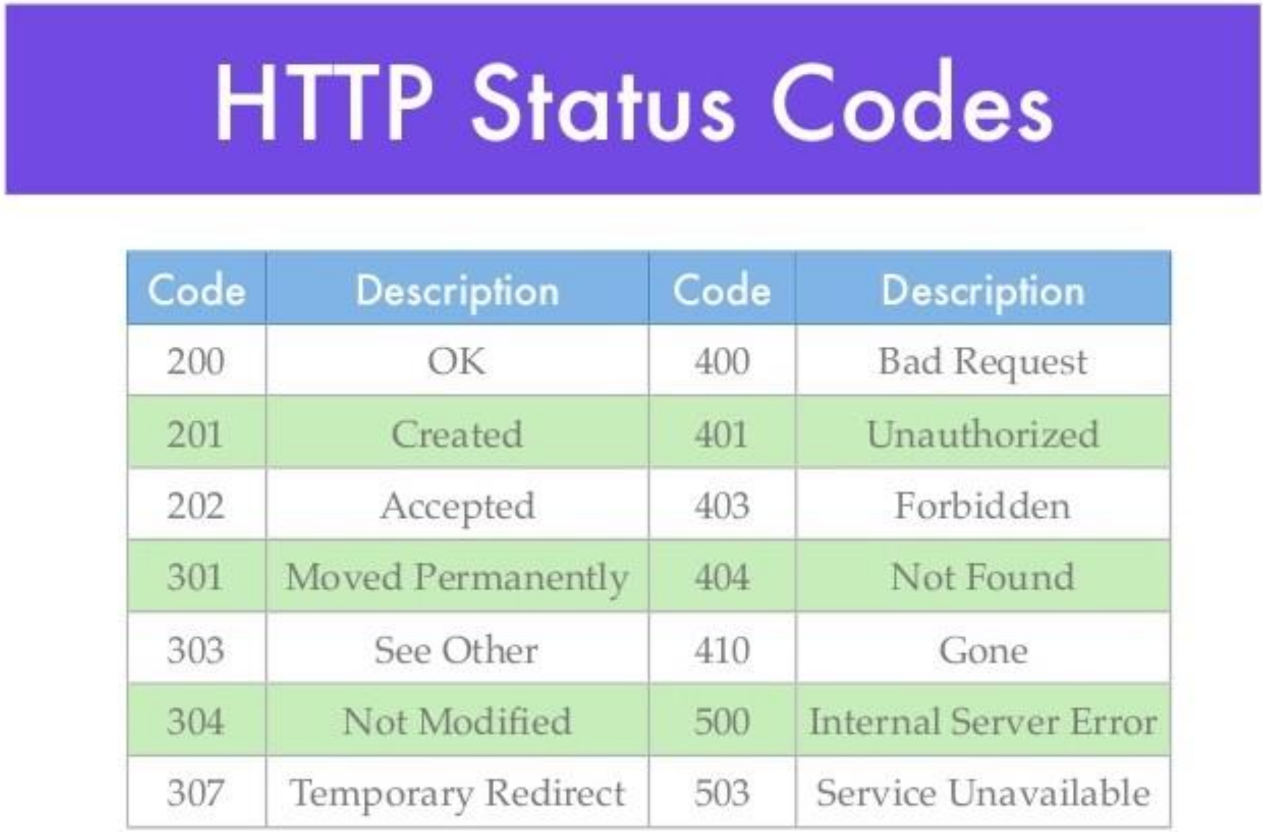
Users Record

Search person by Name/Email: Search

USER ID	NAME	EMAIL	PHONE	CITY	OPERATIONS
1	Leanne Graham	Sincere@april.biz	1-770-736-8031 x56442	Gwenborough	View | Edit | Delete
2	Ervin Howell	Shanna@melissa.tv	010-692-6593 x09125	Wisokyburgh	View | Edit | Delete
3	Clementine Bauch	Nathan@yesenia.net	1-463-123-4447	McKenziehaven	View | Edit | Delete
4	Patricia Lebsack	Julianne.OConner@kory.org	493-170-9623 x156	South Elvis	View | Edit | Delete
5	Chelsey Dietrich	Lucio_Hettinger@annie.ca	(254)954-1289	Roscoeview	View | Edit | Delete

Reference contents

- Some of HTTP status code



HTTP Status Codes

Code	Description	Code	Description
200	OK	400	Bad Request
201	Created	401	Unauthorized
202	Accepted	403	Forbidden
301	Moved Permanently	404	Not Found
303	See Other	410	Gone
304	Not Modified	500	Internal Server Error
307	Temporary Redirect	503	Service Unavailable

  - https://developer.mozilla.org/en-US/docs/Web/HTTP/Status