JSON Syntax

JSON (JavaScript Object Notation), is a standard for representing structured data based on JavaScript's object syntax. It is commonly used to transmit data in web apps via HTTP. For example, the HTTP `fetch()` requests we have been using in this course have been returning Jello projects, users, and issues as JSON.

JSON supports the following primitive data types:

Strings, e.g. `"Hello, World!"` Numbers, e.g. `42` or `3.14` Booleans, e.g. `true` Null, e.g. `null`

And the following collection types:

Arrays, e.g. `[1, 2, 3]` Object literals, e.g. `{"key": "value"}`

Because we already understand what JavaScript objects look like, understanding JSON is easy! JSON is just a stringified JavaScript object. The following is valid JSON data:

```
{
    "movies": [
        {
            "id": 1,
            "genre": "Action",
            "title": "Iron Man",
            "director": "Jon Favreau"
        },
        {
            "id": 2,
            "genre": "Action",
            "title": "The Avengers",
            "director": "Joss Whedon"
        }
    ]
}
```

Parsing HTTP Responses as JSON

JavaScript provides us with some easy tools to help us work with JSON. After making an HTTP request with the fetch() API, we get a Response object. That response object offers us some methods that help us interact with the response. One such method is the `.json()` method. The `.json()` method takes the response stream returned by a fetch request and returns a promise that resolves into a JavaScript object parsed from the JSON body of the HTTP response!

```
const resp = await fetch(...)
const javascriptObjectResponse = await resp.json()
```

JSON Review

JSON is a stringified representation of a JavaScript object, which makes it perfect for saving to a file or sending in an HTTP request. Remember, an actual JavaScript object is something that exists only within your program's variables. If we want to send an object outside our program, for example, across the internet in an HTTP request, we need to convert it to JSON first.

JSON isn't just for JavaScript

Just because JSON is called JavaScript Object Notation doesn't mean it's only used by JavaScript code! JSON is a common standard that is recognized and supported by every major programming language. For example, even though Boot.dev's backend is written in Go, we still use JSON as the communication format between the front-end and backend.

Common use-cases

In HTTP request and response bodies As formats for text files. .json files are often used as configuration files. In NoSQL databases like MongoDB, ElasticSearch and Firestore

Sending JSON

`JSON` isn't just something we get from the server, we can also send JSON data!

In JavaScript, two of the main methods we have access to are `JSON.parse()`, and `JSON.stringify()`.

JSON.stringify()

`JSON.stringify()` is particularly useful for sending JSON.

As you may expect the JSON `stringify()` method does the opposite of parse. It takes a JavaScript object or value as input and converts it into a string. This is useful when we need to serialize the objects into strings to send them to our server or store them in a database.

Assignment Users need to be able to mark their projects as completed. However, there's a bug in the updateProjectById() function. It looks like the project's completed property isn't getting saved properly by the server.

Fix the code. The body property on the object we're passing into the fetch function should be a JSON string, not a JavaScript object.

Parsing JSON

The `JSON.parse()` method takes a JSON string as input and constructs the JavaScript value/object described by the string. This allows us to work with the JSON as an object!

```
const json = '{"title": "Avengers Endgame", "Rating":4.7, "inTheaters":false}';
const obj = JSON.parse(json)
console.log(obj.title)
// Avengers Endgame
```

XML

We can't talk about JSON without mentioning XML. XML, or "Extensible Markup Language" is a text-based format for representing structured information, like JSON - but different (and a bit more verbose).

XML syntax

XML is a markup language like HTML, but it's more generalized in that it does not use predefined tags. Just like how in a JSON object keys can be called anything, XML tags can also have any name.

XML representing a movie:

```xml
<root>
  <id>1</id>
  <genre>Action</genre>
  <title>Iron Man</title>
  <director>Jon Favreau</director>
</root>
```

The same data in JSON:

```json
{
  "id": "1",
  "genre": "Action",
  "title": "Iron Man",
  "director": "Jon Favreau"
}
```

Why use XML?

XML and JSON both accomplish similar tasks, so which should you use?

XML used to be used for the same things that today JSON is primarily used for. Configuration files, HTTP bodies, and other data-transfer can work with either JSON or XML. Personally, I believe that if JSON works, you should favor it over XML. JSON is:

- Lighter-weight
- Easier to read
- Has better support in most programming languages

There are cases where XML might still be the better, or maybe even necessary, but that tends to be the exception rather than the rule.