JavaScript APIs

Estimated Time: 15 mins

Introduction

In this section, you will explore JavaScript APIs and the REST architectural style.

Objectives

After completing this reading section you will be able to:

1. Define an API and a JavaScript API
2. Describe REST architecture
3. Explain CRUD operations
4. Identify popular JavaScript APIs

What are JavaScript APIs?

To understand what a JavaScript API is, it is important to first know what an API is. An API, or Application Programming Interface, is a way for two applications to communicate with each other. It delivers your request to another device, such as a database, and returns the response back to you.

Imagine you are sitting in a restaurant and have selected your order. The menu outlines a list of food items, and the corresponding meals are prepared in the kitchen. Your waiter is the link between you and the kitchen, who communicates your order to the kitchen and returns the food back to you. This communication is similar to how APIs work. An API is analogous to a waiter, as it communicates a request from one device to another, and returns the response back to the first device.

The menu in the example is the API documentation. In a restaurant, if you order a food item which does not exist on the menu, the waiter will inform you that it is an invalid choice, and will be incapable of delivering the food item. Similarly, each API has documentation that outlines the requests you are allowed to make, and the type of response you should expect to receive. If you try to make an invalid request, you will come across an error.

JavaScript APIs are Application Programming Interfaces that use JavaScript scripting to dynamically access and modify content.

REST Architecture

Most JavaScript APIs follow the Representational State Transfer (REST) architectural style. These are referred to as RESTful APIs, and follow the CRUD paradigm. CRUD stands for Create, Read, Update, and Delete, and model the four basic functionalities needed when communicating between services and with a database. In a REST environment, these CRUD operations are often aliased as follows:

- Create → POST
- Read → GET
- Update → PUT
- Delete → DELETE

As an example, imagine an API which communicates with a banking service to process online payments. It is possible to use all four CRUD operations for this API. An example of each is provided below.

Method	**URL**	**Description**
POST	api/customer	Create a new banking customer
GET	api/customers/{id}	Retrieve the information of a customer
PUT	api/customers/{id}	Update information for a specific customer
DELETE	api/customers/{id}	Delete a banking customer

The URLs in each example are important, as they determine the specific item/customer that is being accessed.

In the POST request, you can see that a new customer is created with the API. Depending on the specifications of the API, this may include options to provide data for this customer, such as a name or credit card details. It may also automatically generate new information upon creation, such as an id.

The GET request allows you to retrieve all the information associated with a customer. The API assumes a unique id for each customer, that is used in the URL to specify which customer's information you are searching. A response for this request can come in many different formats, such as JSON or XML, depending on the API.

In the PUT request, you are able to update the information for a specific customer. This will overwrite the current data with new data. Similar to the GET request, you can specify a specific customer using the id. You can provide new data to be updated in different ways that are specific to the API. Some APIs allow you to include a "body" in which you can specify a load of data to be sent with the request. For example, you can attach the following body to the PUT request in order to update a customer's first and last name:

```
{
    "first_name": "Thomas",
    "last_name": "Watson"
}
```

In the DELETE request given in the example, you can remove a customer entirely by once again providing the id.

Document Object Model (DOM) API

The Document Object Model (DOM) API is one of the most basic JavaScript APIs available. It connects web pages to scripts by representing the structure of a document (e.g. an HTML web page) in memory, making it accessible for modification as required.

The DOM API is covered in more detail in the videos, and will not be reviewed here.

XMLHttpRequest

A popular JavaScript API is XMLHttpRequest (XHR), which allows you to retrieve data without refreshing the entire page. This is important when you to want to update only a part of a page without disrupting what a user is currently doing on the page.

XMLHttpRequest is used heavily in Asynchronous JavaScript And XML (AJAX) programming. Full documentation on its usage can be found on (this page) https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

Advanced APIs

There are more advanced JavaScript APIs available, each with a different use and specification. Many of these APIs can be found on the official Mozilla Developer website or you can search the internet for the required API.

Author(s)

Michelle Saltoun