Lesson Monday

Introduction to Programming (/introduction-to-programming) / JavaScript and Web Browsers (/introduction-to-programming/javascript-and-web-browsers) / Web APIs

Text Cheat sheet

Since being introduced to the `window` and `document` objects, as well as the DOM and events, we've stepped away from JavaScript to learn about structures and tools that web browsers make available to web developers. Now is a good time to learn where we can find documentation references on Mozilla Developer Network (MDN) Documentation for these tools.

In this lesson, we'll learn about these topics:

- What an Application Programming Interface (API) is.
- What a *Web* API is.
- All about Web API specifications and interfaces.
- Where to learn more about the DOM, the `window` object, and the `document` object on MDN.

Why We Learn about MDN Documentation

You don't need to worry about referencing MDN documentation now, however it is important to know how MDN documentation is structured so that you can get better at navigating it and understanding how the technology we work with is structured.

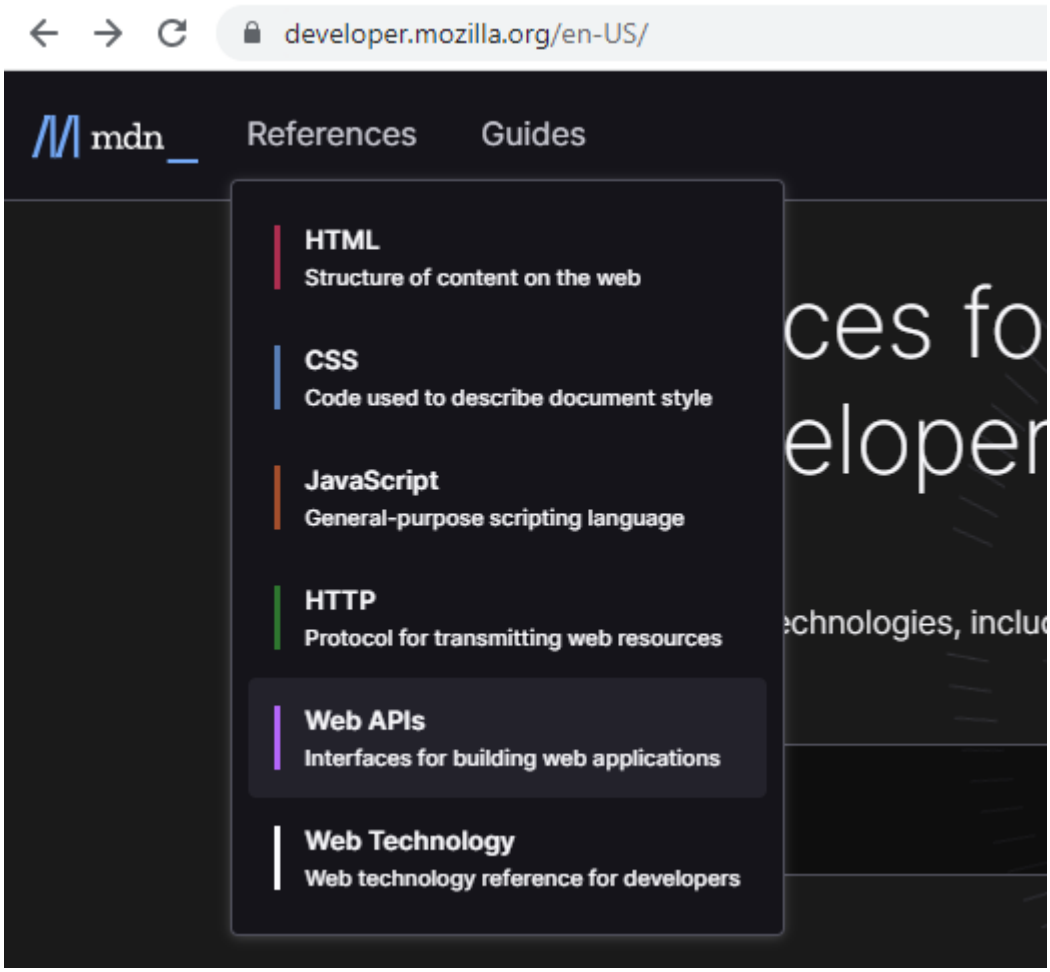
Learning how to drive and maintain a car is a good analogy for learning to write code. Once we know the basics, we can start driving without knowing at all how a car's engine works. This is just like being able to access `window` properties or create a string and store it in a variable without fully understanding how browser tools and JavaScript are structured. In fact, it's important to start driving before exploring what's going on under the hood of the car.

However, becoming a resourceful developer also involves looking under the hood of the car to understand how each piece of technology works and how it relates to other tech. Exploring MDN documentation enables us to do just that — look "under the hood" of JavaScript and the browser tools we are learning about!

Take note that we'll periodically pause to revisit the topic of navigating MDN and review the inner structures of the browser tools and JS we work with. So if something does not make sense now, there will be an opportunity to revisit it later. **Understanding how browsers are structured is not required to be successful on this section's independent project**, but it will improve your resourcefulness as a developer and is key to your long term growth. In other words, for the upcoming independent project, we want to see how you drive the car, not what you know about what's going on under the hood.

It's normal to feel overwhelmed with a bunch of new information. Talk about what doesn't make sense with your pair and instructor. Remember that learning takes place over time with a lot of repetition and practice. A solid understanding of what's going on "under the hood" also requires repetition and practice, and you'll get that by working through multiple course sections, not just this one.

The DOM, `window` and `document` Objects, and Events Are Web APIs



The DOM, the `window` object and `document` objects are a part of the structures and tools that web browsers make available to web developers. We call this collection of structures and tools **Web APIs**. Let's break down what this means:

- **API** is the acronym for **application programming interface**. An API is a set of code that lets us interact with more complex software in a simplified and pre-defined way. We'll expand on this definition when we discuss interfaces below.
- **Web APIs** are a collection of many APIs that are specifically made for accessing web technologies, like the browser window or the HTML document displayed in our browser window. These APIs are written in JavaScript, and we can use JavaScript

to access them, but they are not a part of the JavaScript language.
- Web APIs are made up of **specifications** that describe in detail what each individual API is and does, and **interfaces**, which are different types of objects that make up the actual functionality of Web APIs.

An Example Interface

We use interfaces all of the time. Consider a door knob — the knob is the interface that we use to open a door. The door knob interface handles abstracting the complex internal hardware of the door latch into a simple and easy to use interface. When we use the door knob, we're triggering the door hardware to do its thing to open the door latch. We don't need to concern ourselves with how the door latch works, we just need to know how to use the simplified interface that is the door knob.

Similarly with our web browsers, we don't need to know or understand the source code for our browser window, we just need to learn how to use the `window` object. The `window` object is the interface that we can learn to access and use to program our websites to work with the browser's window.

API specifically means that the interface we're working with is for programming applications, and it does the job of letting our program (that we create) talk to another program (for example, the web browser).

A **Web API** is even more specific: the API we're working with is specifically for working with the web browser. Later in the program, we'll work also with **3rd party APIs** (created by software companies to access the functionality of their products) and even create our own custom API.

Specifications & Interfaces

Let's look at a helpful example within Web APIs to understand the difference between a specification and an interface:

- The DOM is a Web API specification. Follow this link to see the specification page for the DOM (https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model).
- An interface belonging to the DOM is the `document` object.

A **specification** describes what an API's purpose is, how it is structured, what it can do, and what other technology it interacts with. An **interface** is an object of a specific type that contains a specific set of functionality. We interact with a Web API through its interface(s).

Every API is made up of one or more interfaces. For example, so far we have learned about the `document` interface for the DOM, but the DOM is actually made up of many, many interfaces, and each interface in the DOM describes a discrete bit of functionality. Optionally, check out all of the interfaces that belong to the DOM by following this link (https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model#dom_interfaces). Don't worry — we won't be working with all of these interfaces in the program. The idea here is to give you an idea of what's out there, and the tools to explore more on your own.

Also, it's common for Web APIs to have **entry points**. For the DOM, the entry point is the `document` object. For the browser window, the entry point is the `window` object. We can access other data (including other interfaces) through these entry points. Like we saw in the last lesson, we can access the `location` object through the `window` interface:

```
> window.location
```

Note that `location` is also an interface! How do we know? Anytime you are working with an interface (like `window` or `document`) and the data type of a property's value is an object (like `window.location`), that object is most likely an interface. To verify you'll need to find the name of the object in the Web APIs homepage, where all interfaces are listed (https://developer.mozilla.org/en-US/docs/Web/API#interfaces). If the name of the object is in that list, then it's an interface.

Take note! While all Web API interfaces are objects, not all objects are interfaces:

- An object is a data type, often defined as a collection of properties.
- An interface is a special kind of object that exposes (makes available for use) the functionality of an API.

Language Choice: Should I call `window` and `document` "interfaces" or "objects"?

As we just learned, `document` and `window` are interfaces within the collection of Web APIs that make browser functionality available to developers to use in their websites. These interfaces, like all Web API interfaces, are objects and we can use JavaScript to access them and manipulate them.

In this lesson and future lessons, we'll continue to call `window` and `document` by their data type, "object", because we'll be working on accessing object properties and setting the value of those properties. In other words, it's much less important to understand APIs, specifications, and interfaces conceptually, than it is to comfortably access object properties.

Going forward you can choose to call `window` and `document` as "interfaces" or "objects". Both are correct, only "interface" is more specific, and reminds us that we're working with a Web API.

Anecdotally, it's very common online to see developers referring to `window` and `document` as objects, because, well, that's what they are!

Web APIs Are Not JavaScript

Keep in mind that Web APIs "sit on top of" JavaScript. That means that we use JavaScript to access and manipulate Web APIs, but they are not a part of the core JavaScript language.

On MDN

So, on MDN the DOM, the `window` and `document` objects, and events are a part of **Web APIs**, the structures and tools that web browsers provide to developers to write code that interacts with browser structures.

- **`window` (https://developer.mozilla.org/en-US/docs/Web/API/Window)**
- **`document` (https://developer.mozilla.org/en-US/docs/Web/API/Document)**
- **An Introduction to the DOM (https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction)**

There are many interfaces that make up events, but we won't be going over those now. Instead, we'll review that information and documentation once we begin learning about events.

There are many more Web APIs that we won't learn about in this program. You can take a look at them here:

- **Web APIs (https://developer.mozilla.org/en-US/docs/Web/API)**

Summary

Remember, all of this information is for context-building and you won't be tested on it. We'll revisit this information in coming lessons, and it will help us better understand how to use tools like the `window` object in the applications we build.

In this lesson we learned:

- The DOM, the `window` object, the `document` object, and events are a part of the structures and tools that web browsers make available to web developers. We call this collection of structures and tools **Web APIs**.
- **API** is the acronym for **application programming interface**. An API is a set of code that lets us interact with more complex software in a simplified and pre-defined way.
- **Web APIs** are a collection of many APIs that are specifically made for accessing web technologies, like our browser window or our HTML document displayed in our browser window. These APIs are written in JavaScript, and we can use JavaScript to access them, but they are not a part of the JavaScript language.
- Web APIs are made up of **specifications** that describe in detail what each individual API is and does, and **interfaces**, which are different types of objects (with properties and methods) that make up the actual functionality of Web APIs.
    - An example specification is the DOM, and an example interface is the `document` object.
    - While all Web API interfaces are objects, not all objects are interfaces. An object is a data type, often defined as a collection of properties. An interface is a special kind of object that exposes (makes available for use) the functionality of an API.

In regards to using the new technical terminology covered in this lesson, you can:

- Refer to the `window` and `document` interfaces as "objects" or "interfaces". Going forward in the lessons, we'll refer to them as objects.

Please be patient with yourself as you wrap your mind around all of this information and start to use correct technical terminology. You won't be able to remember everything or understand every new concept on the first try. Always do your best and know that you will understand all of this over the course of the program.

Previous (/introduction-to-programming/javascript-and-web-browsers/accessing-window-properties)
Next (/introduction-to-programming/javascript-and-web-browsers/practice-accessing-window-properties)

Lesson 30 of 75

Last updated more than 3 months ago.

disable dark mode

(http://www.epicodus.com)

© 2023 Epicodus (http://www.epicodus.com/), Inc.