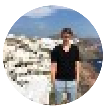
Browser and Web APIs



Antoine Leclercq Follow
May 11, 2017 · 3 min read

What are Web APIs?

The term Web APIs is a very generic term used in a lot of different instances. For this blog post I want to talk about Web APIs in the context of the browser. Basically, Web APIs are the APIs made available to us, front-end developers, by the browser. They are also referred to as BOM (Browser Object Model) APIs. For example, the DOM API is a BOM API, more specifically, it is a subset of the BOM APIs. Another example would be the `Event` interface or the `Element` interface, which both are part of the DOM API and consequently are also part of the BOM APIs.

So, a Web API, in the context of the browser, simply is an API, provided by the browser and that we can communicate with using JavaScript in order to solve our front-end problems. Even though these APIs are accessible with JavaScript, their implementation is in the language that the browser uses, for example, for Google Chrome it is C++.

Browser and Interfaces

A good thing to understand is the way the browser provides interfaces to work with. These APIs are available thanks to the interfaces implemented in the browser. Each browser implements interfaces the way it wants to, although, as the years have passed browsers came to mostly implement the same interfaces which makes it easier to have cross-browser compatibility. I won't get into details as to how each browser implements its interfaces, this isn't the subject of this blog post, the point is to have a good mental model of what the browser provides you with in terms of APIs.

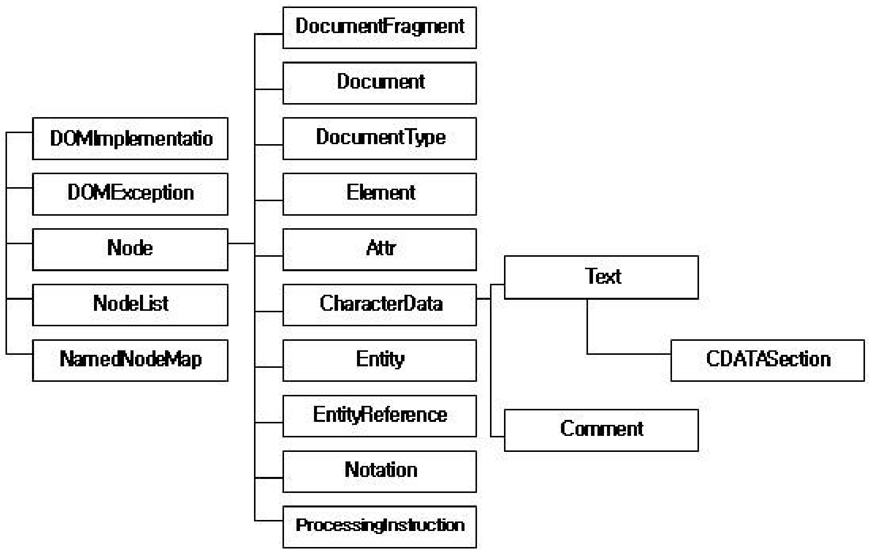
You can find the interfaces implemented in the browser referenced in the Mozilla Docs. These interfaces instantiate objects that are then used by the browser or in your scripts. For example, let's analyze the following JavaScript code in terms of Web APIs:

```
var aElements = document.querySelectorAll('a');

for (var i = 0; i < aElements.length; i++) {
  aElements[i].addEventListener('click', function (event) {
    // code
  });
}
```

- the `Document` interface—defined by the browser—defines the `querySelectorAll` method which makes it accessible to `Document` objects, such as `window.document`
- `querySelectorAll` returns a `NodeList` (browser interface) object, which is an Array-like object
- the `NodeList` browser interface provides us with the property `length` and the method `item`
- some JavaScript magic allows us to use the operator `[]` on the `NodeList` object, but what's really happening is that the method `item` is being called with the argument `i`
- each element in `aElements` is an `HTMLAnchorElement` object and the `HTMLAnchorElement` browser interface inherits from `HTMLElement` , which inherits from `Element` interface, which inherits from `EventTarget` interface , which defines the `addEventListener` method used on each element in our for loop
- the `addEventListener` registers an event listener —a callback function, the second argument in our example—on each element which will be triggered by an event of type `click` during the targeting or bubbling phase of the event

Having a strong mental model of how things work in the browser allows you to understand how to look at the documentation of each interfaces and how some interfaces relate to each other through inheritance. Here is a diagram showing the inheritance tree for the main interfaces composing the DOM API:



DOM Interfaces Inheritance Tree

As you can see, except for `NodeList` and `NamedNodeMap` every interface that we frequently use is inheriting from the `Node` interface. Even the `Document` interface inherits from the `Node` interface. This means that a `Document` object, which holds the DOM tree for a web page, will have properties and methods provided by the `Node` interface, such as `childNodes` or `nodeType` . Effectively, a `Document` object is also a node.

I hope this clarifies and strengthen your understanding of the browser and its Web APIs. If you have an understanding conflicting with my explanations, please leave me some feedback!

Happy Coding!