college CD

Data processing technologies (TTD)

Class 07

APIs

Standing for *Application Programming Interface*, an API consists in a set of rules and specifications that applications can follow in order to communicate with each other. They govern how applications can talk to each other, and how data gets shared over the Internet.

APIs should be considered by developers as tools made available by different companies that can save a lot of time, solve various problems and to create complex functionality more easily.

Client-side APIs

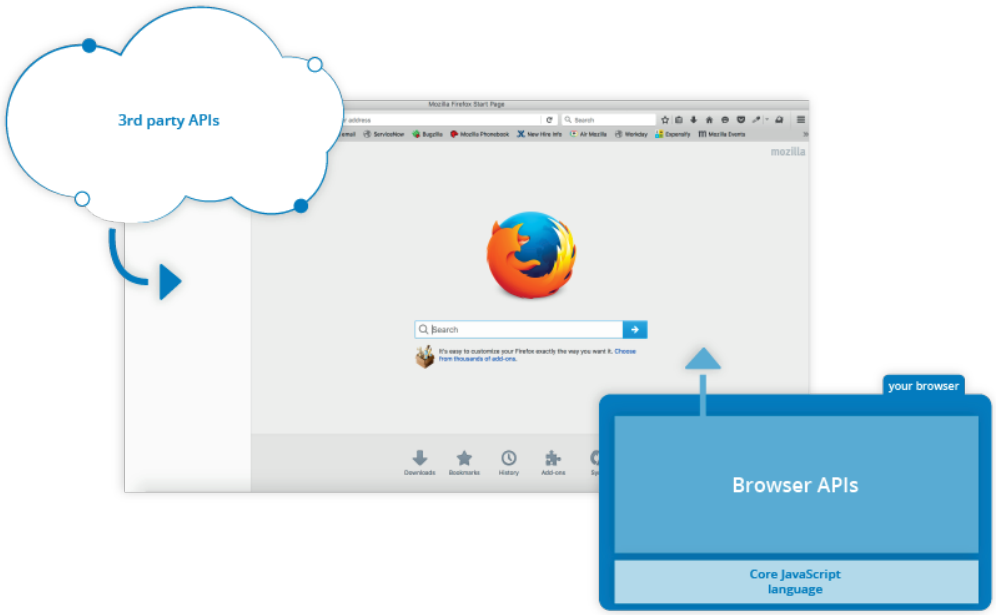
Many APIs are available to be used in JavaScript. They provide extra superpowers to use in the JavaScript code generally falling into two categories :

Browser APIs

Browser APIs are built into the web browser and make it possible to show data from the browser as well as from distant servers. For instance, the *Web Audio API* makes it possible to manipulate audio in the browser. In fact, applications programmed in complex lower-level languages are used in the background to do the real job, but the API acts somehow like a plugin.

Third-party APIs

Third-party APIs are not built into the browser, and it is generally necessary to retrieve their code and data from an external source on the web. For instance, the Twitter API allows you to do different things such as displaying your latest tweets on a web document. It provides functionalities capable of making queries to the Twitter service and return specific information.



APIs possibilities

There are hundreds of APIs available online doing all sorts of things. Some are better and more useful than others.

Common browser APIs

These are the most common categories of browser APIs allowing you to:

APIs for manipulating documents loaded into the browser.
The best example is the DOM API, making it possible to manipulate HTML and CSS.

APIs that fetch data

Some APIS fetch data from the server to update small sections of a webpage automatically. For instance, stock listings continuously updated without having to reload a page are provided using such APIs. They use XMLHttpRequest or AJAX requests and the Fetch API.

Drawing and graphics manipulation APIS

These APIS are now mostly supported in browsers (Canvas and WebGL, for instance). They make it possible to update the pixels information within a <canvas> element making it possible to create 2D and 3D views and to apply different effects and behaviours.

Audio and Video APIs

These APIs allow you to do things such as creating custom UI controls for playing audio and video, displaying captions and subtitles, grabbing video from your web camera to be manipulated via a canvas or displayed on someone else's browser in a web conference, etc.

Device APIs

Device APIs are made for manipulating and retrieving data from modern device hardware which is very useful for web apps. For instance, an API may be telling the user that an update is available for a specific app sending a notification.

Client-side storage APIs

These make it possible to store data on the client-side, which is useful to save the state of an app between page loads, and perhaps even work when the device is offline.

Public APIs

Also known as *Open APIs*, public APIs are application programming interfaces made available to developers. Using the APIs, the developer may give access different services to users. Public APIs include what is called *Big Data* which consist of very large amount of data made publicly available. There are a very large number of public APIs to choose from and there are different ways to connect to them.

Data from an endpoint

An endpoint is a URL where data is made available. When you decide to use an endpoint to retrieve data, first test it in a browser to make sure data is available before starting any coding. The result are very often made available in JSON format, so it is quite simple to extract using an AJAX request.

Find and read the documentation

Whenever you want to use an API, one of the very first step to take is to find and read it's documentation (most of APIs are documented. This is how you will find out about the API's features, the endpoint addresses, and how to use it.

Parsing content in your browser

Reading from a raw data block can be quite awful. In order to have your JSON data look good and well structured on screen, you can use JSONview extension for Chrome or Firefox (https://jsonview.com/).

Loading JSON data into a web document

As you already know, importing data from another server requires an AJAX request. Although, using jQuery, it is possible to use the *getJSON()* function.

Loading JSON data and displaying text content to a web document

In your web document containing the jQuery CDN and between <script></script> tags or using an external JavaScript file, make you request.

```
<script type="text/javascript">
$.getJSON("http://api.mycoolapi.com", function(data) {
        ...
});
</script>
```

Explanation :

In the example above, *getJSON()* function is used to make an *AJAX request* allowing to import the content and to store it in the variable *data*. Using this last variable's name as prefix, it becomes easy to later access the available content.

Your first API

Geolocalization

Different APIs offer developers to locate the user geographically using their IP adresses. Most of these API are paid services, but some offer free plan with restrictions.

Test the endpoint

First of all, test the endpoint to make sure data is available (https://ipapi.co/json/).

If you use JSONview extension for Chrome or Firefox, the data will be much more easy to read, as you can see in the example beside.

This one API is very easy to use since there is only one object (root) containing different name/value pairs.

As you can see, the API detects the current IP address and quite a few informations about the user that can be used in different ways.


```
{
    ip: "135.19.214.110",
    city: "Longueuil",
    region: "Quebec",
    region_code: "QC",
    country: "CA",
    country_code: "CA",
    country_code_iso3: "CAN",
    country_capital: "Ottawa",
    country_tld: ".ca",
    country_name: "Canada",
    continent_code: "NA",
    in_eu: false,
    postal: "J4K",
    latitude: 45.5185,
    longitude: -73.5043,
    timezone: "America/Toronto",
    utc_offset: "-0500",
    country_calling_code: "+1",
    currency: "CAD",
    currency_name: "Dollar",
    languages: "en-CA,fr-CA,iu",
    country_area: 9984670.0,
    country_population: 33679000.0,
    asn: "AS5769",
    org: "Videotron Telecom Ltee"
}
```


Make your request and display data

Now that the endpoint has been tested and that we know about the data structure, we need to use *getJSON()* function to make an AJAX request. As usual with jason, the data will be retrieve usign the *root.name* format.

```
<h2> </h2>

<script type="text/javascript">
$.getJSON("https://ipapi.co/json/", function(data) {

        $("h2").html("You are now in the city of: " + data.city);
});
</script>
```

Explanation

In the example above, HTTP request is made to the endpoint, and the data is displayed using concatenation.

Redirecting users based on country

Using the same API as for the last lesson, it becomes very easy to use the data in different conditional structures. For instance, to show specific contents or to redirect users to various pages. Here, we will use the country name to redirect users to a CANADA or a USA page depending of the data supplied by the API.

```
<script type="text/javascript">
$.getJSON("https://ipapi.co/json/", function(data) {                    // AJAX request

        let country = data.country;                                     // User's country name

        if (country == "CA"){                                           // For Canada
                location.replace("canada.html");
        } else if (country == "US") {
                location.replace("usa.html");                           // For USA
        }
        else {
                alert("You are not from CANADA nor USA")                // For all other countries
        }
});
</script>
```

Explanation :
In the example above, an AJAX request is first mad and the result (the JSON data) is stored in the variable *country*. Conditional structures than evaluate if the variable is equal to CA or USA to redirect the user to the corresponding pages. Otherwise, an alert message is showed.