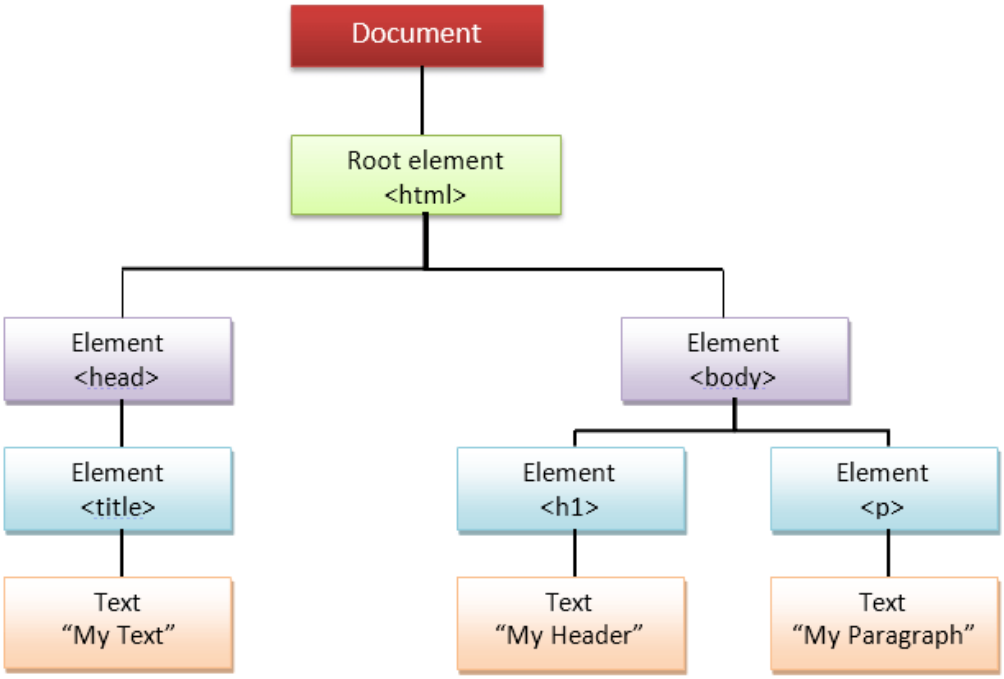
JavaScript DOM Programming

A front-end web application consists of HTML, CSS, and JavaScript code sent from a web server to a user's browser. After the HTML and CSS is displayed, the JavaScript program continues to run, managing the user's experience including all the application inputs and outputs.

▶ JavaScript DOM (Document Object Model)

Web based puzzles, games, and apps work because users can interact with elements in the browser. The browser stores a list of all page elements in a tree data structure called the DOM (document object model). JavaScript code can be written to target specific elements. The code can listen for user inputs like mouse clicks and can change both content and styles in the page.



https://www.guru99.com/how-to-use-dom-and-events-in-javascript.html

▶ JQuery

The code to manipulate the DOM is complicated and for years varied from browser to browser. *JQuery* is a pre-written JavaScript library of code that any programmer can use for easier cross-platform JavaScript development. JQuery is free to use. Google and other large content companies host the library.

Using JQuery on your page requires the following line in your HTML head element:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

▶ JavaScript vs. JQuery

Reasons to use **JavaScript**

- Pages use less bandwidth because no extra code library is downloaded
- Some JavaScript APIs like Canvas don't have JQuery wrappers
- You'll have a better understanding of client programming

Reasons to use **JQuery**

- Code is faster and easier to write
- Cross-platform library code runs the same on all browsers
- Your framework or template might already be using it

▶ JavaScript DOM Selectors

This script uses the DOM's getElementByID() function on the HTML page (referred to as the document object) to select the element with the "content" ID. When selected, the HTML content inside the content div (referred to as the innerHTML property) is replaced with "Hello World!" text.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <title>Document Object Model</title>
</head>

<body>
    <p id="content"></p>

    <script>
        document.getElementById("content").innerHTML = "Hello World!";
    </script>

</body>
</html>
```

▶ JQuery DOM Selectors

The **dollar sign ($)** character is a shortcut for accessing the JQuery object. Selectors inside the parenthesis are written using the same syntax CSS uses to select elements, classes and ids. The JQuery html() function replaces the HTML content. Note that the head element contains a link to the JQuery script. This is required for all JQuery projects.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <title>Document Object Model</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
</head>

<body>
    <p id="content"></p>

    <script>
        $('#content').html("Hello World!");
    </script>

</body>
</html>
```

▶ JavaScript DOM Methods

GET an element's content:

document.getElementById(id).innerHTML;

```
document.getElementById("message").innerHTML;
```

SET an element's content:

document.getElementById(id).innerHTML = value;

```
document.getElementById("message").innerHTML = "Hello World";
```

GET a form field's content:

document.getElementById(id).value;

```
document.getElementById("firstname").value;
```

SET a form field's content:

document.getElementById(id).value = value;

```
document.getElementById("firstname").value = "David";
```

GET an attribute's value:

document.getElementById(id).getAttribute();

```
document.getElementById("photo").getAttribute("src");
```

SET an attribute's value:

document.getElementById(id).setAttribute(attribute, value);

```
document.getElementById("photo").setAttribute("src", "newphoto.png");
```

Note: The style attribute should be modified using the className property demonstrated next.

Note: Also, you can access attributes directly like document.getElementById("element").src, but the JavaScript property names are not always the same as the HTML attribute names.
(https://www.w3.org/TR/REC-DOM-Level-1/idl-definitions.html)

GET an element's class for CSS styling:

document.getElementById(id).className;

```
document.getElementById("content").className;
```

SET an element's class for CSS styling:

document.getElementById(id).className = value;

```
document.getElementById("content").className = "newClass";
```

Add an HTML element:

document.createElement(element);
document.appendChild(element);

```
var btn = document.createElement("button");
btn.innerHTML = "CLICK ME";
document.body.appendChild(btn);
```

Note: you should store the newly created element in a variable and can attach it to any node.
(see https://www.w3schools.com/jsref/met_document_createelement.asp)

Remove an HTML element:

document.getElementById(parent).remove(document.getElementById(child));

```
var parent = document.getElementById("parent-element");
var child = document.getElementById("child-element");
parent.removeChild(child);
```

Working with Collections:

```
document.getElementByTagName(element) // Selects all elements with that element tag
document.getElementByClassName(class) // Selects all elements with that class
document.getElementByName(name)       // Selects all elements with that name value
```

```
var myNodeList = document.getElementsByClassName("class");
    for (var i = 0; i < myNodeList.length; i++) {
    myNodeList[i].style.backgroundColor = "red";
}
```

▶ JQuery DOM Methods

GET an element's content:

$(selector).html();

```
$("#content").html();
```

SET an element's content:

$(selector).html(value);

```
$("#content").html("Hello World!");
```

GET a form field's content:

$(selector).val();

```
$("#firstname").val();
```

SET a form field's content:

$(selector).val(value);

```
$("#firstname").val("David");
```

GET an attribute's value:

$(selector).attr(attribute);

```
$("#image").attr("src");
```

SET an attribute's value:

$(selector).attr(attribute);

```
$("#image").attr("class", "floatleft");
```

Add a Class for CSS Styling

$(selector).addClass(class-name);

```
$("#submit-button").addClass("visible");
```

Remove a Class:

$(selector).removeClass(class-name);

```
$("#submit-button").removeClass("visible");
```

Toggle a Class (Alternate On/Off):

$(selector).toggleClass(property, value);

```
$("#submit-button").toggleClass("visible");
```

Add an HTML element:

$(selector).append(element);

```
$("body").append("<button>CLICK ME</button>");
```

Remove an HTML element:

$(selector).remove();

```
document.getElementById("content").remove();
```

Working with Collections:

You don't have to write loops if your JQuery selects more than one element, like when selecting elements or class names, as you do with JavaScript DOM programming. The reason is the JQuery functions above automatically loop through the selected elements and apply the command to all of them.

▶ JavaScript DOM Events

JavaScript DOM methods can be inside an event handler, which is a function called in response to an event like a user mouse click or key press message. **Note:** Make sure references to elements are made at the end of the HTML, just before the closing body tag, to be sure the element exists in the DOM before you reference it.

Writing an Event Handler

1. Write a DOM **selector**, like *document.getElementById("btn");*

2. Add a DOM **method**, like *addEventListener(event, handlerFunction);*
    a. Commonly used event values are listed on the next page.
    b. handlerFunction could be **anyonymous** but is better to pass name of function defined earlier.

3. Write a handler **function**, like *elementEventHandler(e) {}*
    a. Commonly used e (event object parameter) properties are listed on the next page.
    b. The *this* keyword gives you access to the selected element.

```
function btnClickHandler(e) {
    // logs the id attribute name of my_element
    console.log(this.id);

    // logs the id attribute name of my_element
    document.getElementById("content").innerHTML = "Hello World!";

    // logs the id attribute name of my_element
    if (e.shiftKey) {
        alert("The SHIFT key was pressed!");
    } else {
        alert("The SHIFT key was NOT pressed!");
    }
}
document.getElementById("btn").addEventListener("click", btnClickHandler);
document.getElementById("btn").removeEventListener("click", btnClickHandler);
```

Alternatively, you could assign a function to a global event handlers. The properties that exist are created by prefixing the event name with “on”. Only one function can be attached at a time, so it doesn’t really follow best programming practices. If you are writing a simple program and are sure you only need to write one handler, then a global event handler is an easy to read and write shortcut.

```
document.getElementById("btn").onclick = btnClickHandler
```

Events and Event Object Properties

Keyboard Events: *keydown, keyup*
Keyboard Event Object Properties: *key, shiftKey, ctrlKey, altKey*

```
document.getElementById("btn").onkeyup = function (e) {
    // keyboard events give access to properties like e.key
};
```

Mouse Events: *click, dblclick, mousedown, mouseup, mouseenter, mouseleave, mouseover, mouseout, wheel*
Mouse Event Object Properties: *pageX, pageY, movementX, movementY*

```
document.getElementById("btn").onclick = function (e) {
    // e.clientX and e.clientY report the mouse position when clicked
};
```

Touch Events: *touchstart, touchmove, touchend, touchcancel*
Touch Event Object Properties: *touches*

```
document.getElementById("btn").ontouchstart = function (e) {
    // e.touches.length reports the number of simultaneous touches
};
```

UI Events: *resize, scroll*
UI Event Object Properties: *view*

Form Events: *blur* (element loses focus), *change* (element content changed), *focus* (element gains focus), *submit*
Form Event Object Properties: *data*

More Events at – https://www.w3schools.com/jsref/dom_obj_event.asp
More Event Objects at – https://www.w3schools.com/jsref/obj_events.asp

Anonymous Functions

The following example creates an **anonymous function** for onclick:

```
document.getElementById("btn").onclick = function (e) {
    // ...
};
```

Anonymous functions can make the code look much more complicated, so best practice is to use them only when the code in the function is very short.

▶ JQuery DOM Events

The syntax for handling events is again easier in JQuery than in pure JavaScript.

For more JQuery events see: https://www.w3schools.com/jquery/jquery_events.asp

```
$(document).ready(function() {
  $("button").click(function() {
    $("#content").html = "Hello World!";
  });
});
```

▶ Timers

Display an alert box **once** after 3 seconds (3000 milliseconds):

```
setTimeout(function(){ alert("Hello"); }, 3000);
```

Display an alert box **repeatedly** after every 3 seconds (3000 milliseconds):

```
setInterval(function(){ alert("Hello"); }, 3000);
```

▶ Programming Challenge: CSS Image Sprites

1. Read about Image Sprites at https://www.w3schools.com/css/css_image_sprites.asp
2. Search for a “walk cycle spritesheet” image, ideally with a single row of sprites.
3. Figure out the width and height of a single sprite and add the following CSS
   (Note the width of each sprite should be the width of the sheet divided by the number of sprites)

```
#home {
  width: 46px;
  height: 44px;
  background: url(img_navsprites.gif) 0 0;
}
```

4. Write a JavaScript timer to flip the sprite to the next one at a every 100 milliseconds (frame rate = 10fps)
5. The timer function should reposition the background position. If the last sprite is shown, the background position should be reset to 0 0.

▶ Web Services

Someone who wants to see a weather forecast online could go to a website like weather.com. They would get a web page formatted for viewing in a browser.

Interactive media applications can also visit websites and request information. These programs will not get the usual HTML and CSS pages that people see. Instead, they get specially formatted data. Web services that send data to a program have an interface called an API (application programming interface).

Programs access APIs in the same way people visit websites – using an URL. The URL query string (the part after the question mark) allows you to pass variable values to the API, such as the city name.

If you sign up for the free OpenWeatherMap API, you will receive a message containing a sample API call:

```
api.openweathermap.org/data/2.5/weather?q=London,uk&APPID=...
```

If you paste that URL in a browser window, you will see the data returned:

```
{"coord":{"lon":-0.13,"lat":51.51},"weather":[{"id":500,
"main":"Rain","description":"light rain","icon":"10n"}],
"base":"stations","main":{"temp":280.82,"pressure":1018.65,"humidity":
100,"temp_min":280.82,"temp_max":280.82,"sea_level":1026.31,"grnd_leve
l":1018.65},"wind":{"speed":2.71,"deg":121.501},"rain":{"3h":0.46},"cl
ouds":{"all":92},"dt":1485827164,"sys":{"message":0.01,"country":"GB",
"sunrise":1485848401,"sunset":1485881318},"id":2643743,"name":"London"
,"cod":200}
```

You can change the city name and pass other parameters to the API, like units. To get temperatures in Farenheit, add the following to the query string: **units=imperial**

```
api.openweathermap.org/data/2.5/weather?q=Boston&units=imperial&APPID=...
```

▶ JSON

This is JSON data. JSON stands for JavaScript Object Notation. JSON is essential a **JavaScript Object** data with a special format for saving an arbitrary amount of information about an object. JSON originally was written for JavaScript but became so popular it is now a standard for sharing data across many types of applications and programming languages.

Paste the JSON data you received from the weather page into a JSON parser to see how the data is organized:

```
http://json.parser.online.fr/
```

JSON contains attribute-value pairs and arrays, and each of those can be nested.

Data values inside the JSON files passed between programs can be accessed using **Dot Notation**.

▶ JavaScript AJAX Request and Callback

XMLHttpRequest is a JavaScript class used to transfer data between a web page and an API running on a server. For more information visit https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

You need to call two functions from that class object.

- The open function creates a request for the external URL.
- The send function sends off the request to the other website.

Next, you write a listener to handle the onreadystatechange event, which is called a *callback* function.

When the state changes to "4", loading is completed. If the request was successful, the code will be 200. If the request failed, the code will be in the 400 or 500 range. Failure codes are 404 (not found) and 500 (server error).

If the AJAX call was successful, you can get the returned data using the property *xmlhttp.responseText*.

COPY AND PASTE THIS CODE INTO YOUR <BODY> ELEMENT. EDIT THE URL. SAVE AND TEST.

```
<div id="data"></div>

<script>
var xmlhttp = new XMLHttpRequest();
var url = "http://api.openweathermap.org/data/2.5/weather?q=Boston&units=imperi...";

xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {

                document.getElementById("data").innerHTML = xmlhttp.responseText;

        }
}

xmlhttp.open("GET", url, true);
xmlhttp.send();
</script>
```

▶ Traversing JSON

Our final task is to extract any values we need from the raw data and update the HTML.

Note that the data returned by the API is just a long string of text. It must be cast from the string data type to a JavaScript JSON object type. This code needs to be in your readystatechange handler:

```
var weatherData = JSON.parse(xmlhttp.responseText);
```

Next, access values in the JSON object using the dot notation. Each time you encounter a nested property (there will be an open curly brace in the JSON data), add another dot and the property name.

Here is how to get the current temperature from the Current Weather Data API response:

```
var currentTemp = weatherData.main.temp;
```

Accessing JSON arrays is more difficult. When you encounter a nested array, use brackets, and put the number of the item in the array you want to access inside the brackets. The first item is always 0. The second item is 1.

Here is how to get the icon. Note that only a code is returned, and you need to build the path to the icon which is located at http://openweathermap.org/img/w/XXX.png

```
var currentIcon = "<img src='http://openweathermap.org/img/w/" +
  weatherData.weather[0].icon + ".png'>";
```

A complete list of icons is located at https://openweathermap.org/weather-conditions

Next, use the JavaScript DOM properties to update the rest of the HTML.

```
document.getElementById("temperature").innerHTML = currentTemp;
document.getElementById("icon").innerHTML = currentIcon;
```

COPY AND PASTE THIS CODE INTO YOUR <BODY> ELEMENT. EDIT THE URL. SAVE AND TEST.

```
<div><span id="temperature"></span></div>
<div><span id="icon"></span></div>
<div id="data"></div>

<script>
var xmlhttp = new XMLHttpRequest();
var url = "http://api.openweathermap.org/data/2.5/weather?q=Boston&units=imperi...";

xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {

                document.getElementById("data").innerHTML = xmlhttp.responseText;

                var weatherData = JSON.parse(xmlhttp.responseText);

                var currentTemp = weatherData.main.temp;

                var currentIcon = "<img src='http://openweathermap.org/img/w/" +
                  weatherData.weather[0].icon + ".png'>";

                document.getElementById("icon").innerHTML = currentIcon;
                document.getElementById("temperature").innerHTML = currentTemp;
        }
}

xmlhttp.open("GET", url, true);
xmlhttp.send();
</script>
```

Assignment – Finish the JavaScript AJAX Request and Callback

1. Round the temperature and add the °F units in the HTML.

2. Try adding a new div and the JavaScript code for the **current condition** text.

3. Try replacing the default icons with a weather font like the one at: https://www.webappers.com/2013/06/03/multi-layered-weather-icons-from-forecast-font/

4. Call the 5 day / 3 hour forecast and try adding new div elements and a JavaScript loop for the forecast conditions over the next five days.

▶ JQuery AJAX Request and Callback

JQuery includes the ajax object, which can call XMLHttpRequest for you and spare you from dealing with the state changes and other messy details. The JQuery library will always be slower than native code, but it results in cleaner syntax, and makes sense if you are already using other JQuery objects or plugins.

Your data will be returned as the first parameter in the callback function.

Note: Don't forget to include the JQuery library in the HTML head element!

```
<div><span id="temperature"></span></div>
<div><span id="icon"></span></div>

<script>
$(function() {
var url = "http://api.openweathermap.org/data/2.5/weather?q=Boston&units=imperi...";

        window['wxCallback'] = function(weatherData) {

                var currentTemp = weatherData.main.temp;
                $('#temperature').html(currentTemp);

                var currentIcon = "<img src='http://openweathermap.org/img/w/" +
                        weatherData.weather[0].icon + ".png'>";
                $('#icon').html(currentIcon);
        };

        $.ajax({
                url: url,
                dataType: 'jsonp',
                cache: true,
                jsonpCallback: 'wxCallback'
        });
});
</script>
```