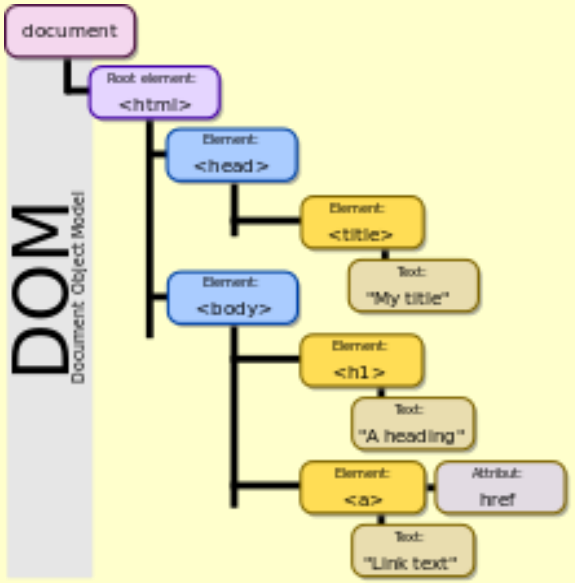
Ref Card: **Intermediate** JavaScript [ES6]

DOM [Document Object Model]



The DOM is essentially a programming interface that browsers use to render source HTML 'sections' as objects that contain all the parts of a web page. The DOM can be manipulated dynamically by JavaScript via the process of Event Listening/Handling. A simple event could be a button click. JS can be programmed to listen/handle the click by modifying the DOM, e.g. Hiding a div & displaying a new div.
REFERENCE 1 & REFERENCE 2 + TUTORIAL VIDEO

Functional Programming [Asynchronous]

Maps & Sets

Ajax [Client-Side] (Becoming Deprecated to Fetch API)

```
performAjax(requestNum, sendToNode, callback) {
    let bustCache = '?' + new Date().getTime();
    const XHR = new XMLHttpRequest(); //THIS is Ajax!!
    XHR.open('POST', document.url + bustCache, true);
    XHR.setRequestHeader('X-Requested-with', requestNum);
    XHR.send(sendToNode);
    XHR.onload = () => {
        if (XHR.readyState == 4 && XHR.status == 200 && callback) {
            return callback(XHR.responseText);
        } else {
            return `ERROR`;
        }
    };
}
```

Simple client method to pass data to Node.js server & handle server response. Use **JSON.stringify()** to send & **JSON.parse()** to receive.

Ajax [Server-Side] (Becoming Deprecated to Fetch API)

```
if (request.method === 'POST' && request.headers['x-requested-with'] === 'XMLHttpRequest0') {
    const FORMIDABLE = require('formidable');
    let formData = {};
    new FORMIDABLE.IncomingForm().parse(request).on('field', (field, name) => {
        formData[field] = name;
        }).on('error', (err) => {
             next(err);
        }).on('end', () => {
            DATA_HANDLER.addData(formData); //points to external class that writes data to DB
            formData = JSON.stringify(formData);
            response.writeHead(200, {'content-type': 'application/json'});
            response.end(formData);
        });
    }
}
```

Simple Node.js routine to receive data from DOM & return results. Use **JSON.stringify()**

Ref Card: **Intermediate** JavaScript [ES6]

Asynchronous File I/O [Server-Side]

```
const IO = require('fs');  // Library for file I/O
handleUserData(data, callback) {
    data = JSON.parse(data);
    const FILE_PATH = 'data/users.csv';
    IO.readFile(FILE_PATH, 'utf8', (err, file) => {
         let user = {};
         const COLUMNS = 4;
         let tempArray, finalData = [];
         tempArray = file.split(/\r?\n/); //remove newlines
         for (let i = 0; i < tempArray.length; i++) {
              finalData[i] = tempArray[i].split(/,/).slice(0, COLUMNS);
         }
         for (let i = 0; i < finalData.length; i++) {
              if (data === finalData[i][0]) {
                   user = JSON.stringify({
                        'email': finalData[i][0],
                        'position': finalData[i][1],
                        'lastName': finalData[i][2],
                        'firstName': finalData[i][3]
                   });
                   break;
              } else {
                  user = 'false';
              }
         }
         callback(user);
    });
}
```

DOM Event Listening/Handling [Client-Side]

```
document.getElementById('continue').addEventListener('click', () => {
    this.performAjax('XMLHttpRequest0',
      JSON.stringify(document.getElementById('getEmail').value), (response) => {
        if (response === 'false') {
            alert('You must provide your proper email address to continue.');
        } else {
            this.user = JSON.parse(response);
            document.getElementById('login').style.display = 'none';
            document.getElementById('log').style.display = 'block';
            document.getElementById('name').innerHTML = `${this.user.firstName
                                                          ${this.user.lastName}`;
        }
});
```

Simple method that demonstrates **addEvenListener()** technique for listening for DOM events & **anonymous arrow function callback** for handling event. Events list HERE.
Important DOM stuff:
http request ~ client -> server. Use GET to receive from server, use POST to transmit to server.
Node.js http response ~ server -> client. Use **writeHead()**, **write()**, & **end()** to return data to client.