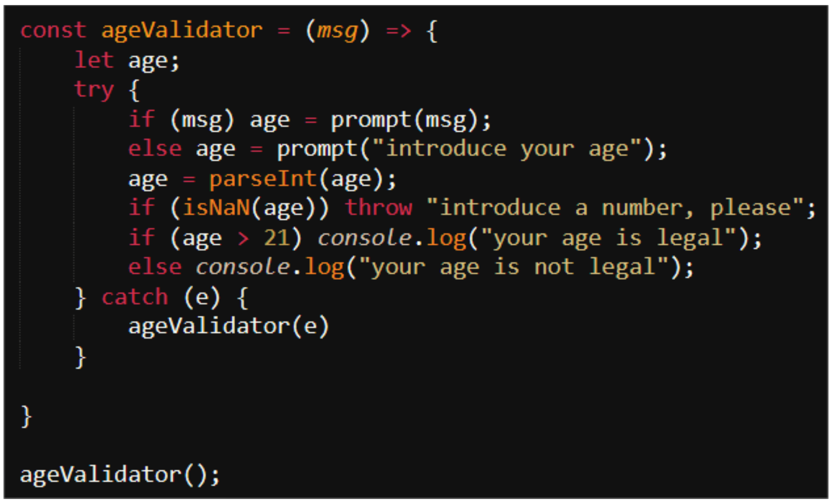
Student: Jorge Alvarez Grana (#249179033)
Class: WDD330
Term: Fall 2021
Subject: W07 Readings

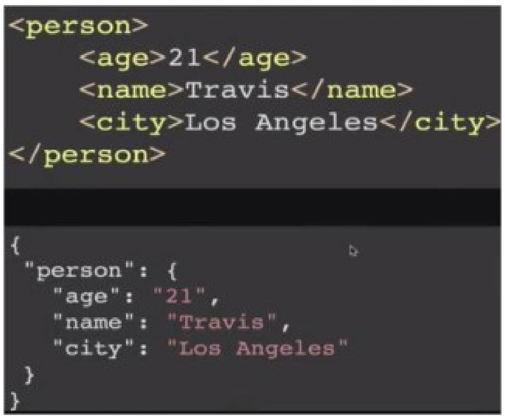
Ch. 11: Further Functions

Here it was very interesting to read about recursive functions, that is, a function that calls itself for iterative processes. For example, having to check age and a person introducing a number we could use the following code. In this situation if type any string on the prompt, it requires me to type a number. If I type 22, it tells me that I am on legal age. So it is an infinite loop so to speak until I do what I am required to do as a user. On the other hand, it is not an infinite loop of the type that crashes the web browser like when having set a *for-loop* with wrong conditions.



```
const ageValidator = (msg) => {
    let age;
    try {
        if (msg) age = prompt(msg);
        else age = prompt("introduce your age");
        age = parseInt(age);
        if (isNaN(age)) throw "introduce a number, please";
        if (age > 21) console.log("your age is legal");
        else console.log("your age is not legal");
    } catch (e) {
        ageValidator(e)
    }

}

ageValidator();
```

Ch. 13: AJAX

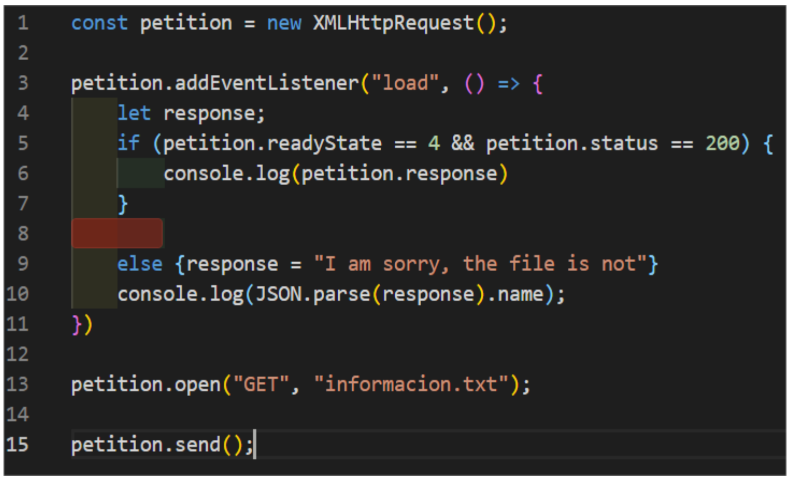


```
<person>
    <age>21</age>
    <name>Travis</name>
    <city>Los Angeles</city>
</person>
```

```
{
  "person": {
    "age": "21",
    "name": "Travis",
    "city": "Los Angeles"
  }
}
```

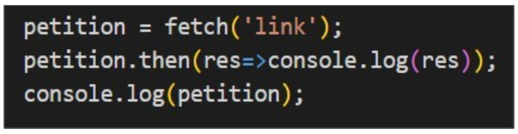
AJAX, JSON or XML are essentially data formats. When talking about websites for example, HTML is a way of formatting data but if we want just the pure information we need data formats so APIs can communicate and share information among computers. That's where XML or JSON comes into play. XML stands for *Extended Markup Language* and looks very similar to HTML, because encodes information between tags also, although in XML those tags are meaningless to the browser. However, nowadays JSON is more popular because it uses JavaScript notation (they are like JS objects) and it is easier for developers to make it parse with APIs coded using the most popular frameworks such as Node, Angular or React. In example we have the same information using XML (top) and JSON (bottom).

AJAX is asynchronous JavaScript. A synchronous operation is when hitting enter and having to wait for the website to upload & update. With AJAX we send request in parallel and we get also the answer in parallel mode. Before it was included in the web browsers but not for security reasons, it is no longer. An example of a code would be as follows, using GET because we request to retrieve a resource. The status 200 and state 4 verifies that there is connection in order to have access to a file named "informacion.txt" where there is a property called name and we get its value. Of course, because we use *JSON.parse* we can have access to an object and not just a string, because we receive (or send using *JSON.stringify*) the information as text and in order to use it we have to transform it into JSON. A website that helped me a bit with this matter was http://reqres.in



```
const petition = new XMLHttpRequest();

petition.addEventListener("load", () => {
    let response;
    if (petition.readyState == 4 && petition.status == 200) {
        console.log(petition.response)
    }

    else {response = "I am sorry, the file is not"}
    console.log(JSON.parse(response).name);
})

petition.open("GET", "informacion.txt");

petition.send();
```

All thus far was AJAX but now Fetch is the more modern alternative to be used, replacing *XMLHttpRequest*. Fetch returns encapsulated promises and we have to use some method to convert that information into a valid data. Fetch has GET method by default and the promise is asynchronous, so we don't have to do anything. When it is ready, we will get the result. So pretty much all that was written before now it could be as following, using methods as *json()*, *text()*, *blob()* to retrieve the information.[1]



```
petition = fetch('link');
petition.then(res=>console.log(res));
console.log(petition);
```

1Note: here link is a given link just that I don't use a real URL