AJAX y NodeJS

Programación Web 2

Document Object Model (DOM)

La idea básica es asociar a cada elemento HTML un objeto JavaScript, de este modo podemos extraer información de la página para nuestros programas y también modificar partes del documento HTML sin tener que recargarlo.

- `document.getElementById("demo").innerHTML = "Hello World!";`
  - https://www.w3schools.com/jsref/met_document_getelementbyid.asp
- `document.querySelector("p").innerHTML = "Hello World!";`
  - https://www.w3schools.com/jsref/met_element_queryselector.asp

Revise la documentación de ambos métodos y recuerde cómo extraer información de un formulario

Eventos

https://www.w3schools.com/js/js_htmldom_events.asp

Cuando se usa JavaScript en una página web varias “acciones” del usuario o del propio navegador pueden desencadenar eventos.

- onload, ocurre cuando se carga una página
  - `<body onload="checkCookies()">`
- onchange, ocurre cuando por ejemplo se cambia un campo en un formulario
  - `<input type="text" id="fname" onchange="upperCase()">`
- Onclick, ocurre cuando se hace click
  - `<button onclick="displayDate()">Try it</button>`

Revise la página de w3schools

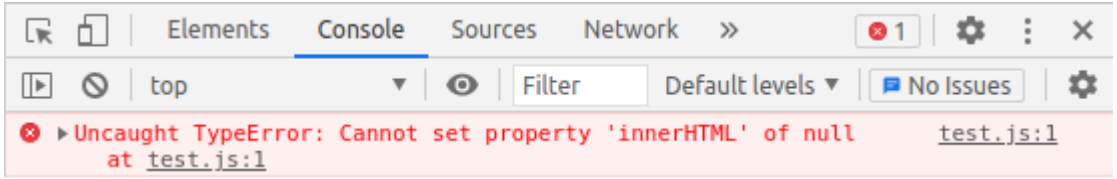
JavaScript ejecutado antes de la carga de la página

Es posible que un código de JavaScript se ejecute antes de que se termine de cargar una página

```html
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Ejemplos de JavaScript</title>
    <meta charset="UTF-8">
    <script src='test.js'></script>
  </head>

  <body>
    <h1>Probando JavaScript</h1>
    <div id='resultadoSuma'></div>
  </body>
</html>
```

```js
document.getElementById("resultadoSuma").innerHTML = 1 + 2;
```



Formularios y Eventos

Redefiniendo la propiedad onsubmit de un formulario, es posible capturar el envió de un formulario, sin embargo **la página NO se recargará**.

```html
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Ejemplos de JavaScript</title>
    <meta charset="UTF-8">
    <script src='test.js'></script>
  </head>

  <body>
    <h1>Probando JavaScript</h1>
    <form id="form1">
      <input type='text' name='userName' id='userName'
        placeholder='Ingrese su nombre' autofocus>
      <input id="enviar" type='submit'>
    </form>
    <div id='resultado'></div>
  </body>
</html>
```

```javascript
document.addEventListener('DOMContentLoaded', function() {

  const name = document.querySelector("#userName");
  const button = document.querySelector("#enviar");
  button.disabled = true;

  name.onkeyup = () => {
    if(name.value.length > 0){
      button.disabled = false;
    } else {
      button.disabled = true;
    }
  };

  document.querySelector('#form1').onsubmit = () => {
    document.getElementById("resultado").innerHTML = "Hola " + name.value;

    //No recargar la página
    return false;
  };
});
```

APIs WEB

https://apipheny.io/free-api/

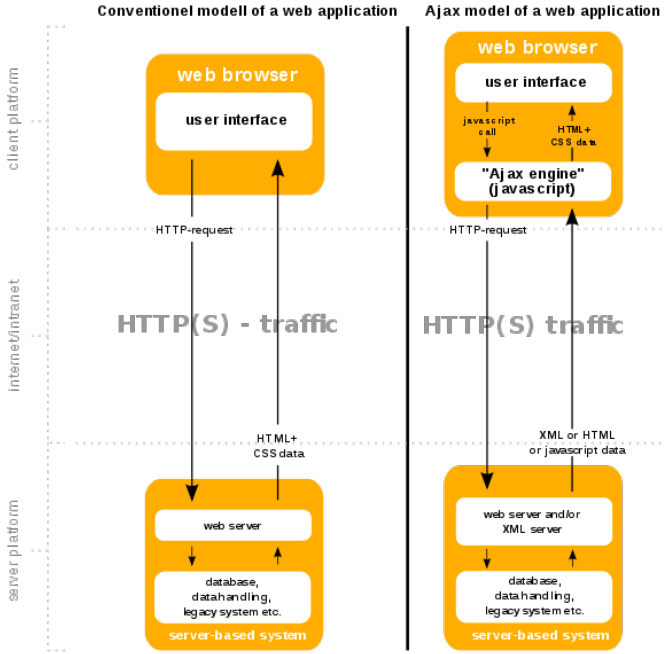
- Un API permite usar las funcionalidades de una biblioteca, tal como lo hace la API de Java
- Un API Web permite usar las funcionalidades de un programa web
- Las preguntas se pueden hacer por el URL con el método GET, o usando objetos JavaScript con el método POST.
- El formato Objetos de JavaScript de JavaScript se conoce como JSON


```
{
  "ip": "8.8.8.8",
  "anycast": true,
  "city": "Mountain View",
  "region": "California",
  "country": "US",
  "loc": "37.4056,-122.0775",
  "postal": "94043",
  "timezone": "America/Los_Angeles",
  "readme": "https://ipinfo.io/missingauth"
}
```


https://www.w3schools.com/js/js_objects.asp

AJAX (Asynchronous JavaScript And XML)

- Ajax combina un objeto asíncrono, JavaScript y XML ó JSON para permitir la comunicación con un servidor web.
- El comportamiento asíncrono se refiere que la comunicación con el servidor web se realizará en hilo de ejecución separado, esto permitirá que el código JavaScript ejecute varias acciones “al mismo tiempo”.
- De este modo se podrán modificar distintas partes del documento HTML de manera independiente.



https://www.w3schools.com/js/js_ajax_intro.asp

fetch: un objeto asíncrono basado en promises

Una promise es algo que sucederá en algún momento en el futuro y nosotros sólo la programamos para cuando llegue ese momento.

```html
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Cambio de Monedas</title>
    <meta charset="UTF-8">
    <script src='test.js'></script>
  </head>

  <body>
    <h1>Ingrese un número IP</h1>
    <form id="form1">
      <input type='text' name='ipAdress' id='ipAdress'
        placeholder='Ingrese un número IP' autofocus>
      <input id="enviar" type='submit'>
    </form>
    <div id='resultado'></div>
  </body>
</html>
```

```javascript
document.addEventListener('DOMContentLoaded', function() {

  const ip = document.querySelector("#ipAdress");
  const button = document.querySelector("#enviar");
  button.disabled = true;

  ipAdress.onkeyup = () => {
    if(ipAdress.value.length > 0){
      button.disabled = false;
    } else {
      button.disabled = true;
    }
  };

  document.querySelector('#form1').onsubmit = () => {
    const url = "https://ipinfo.io/" + ip.value + "/geo";
    fetch(url).then(
      response => response.json()
    ).then(
      data => {
        console.log(data);
      }
    );

    //No recargar la página
    return false;
  };
});
```

El evento DOMContentLoaded

addEventListener, asocia un “evento” de un objeto con una acción (función).

DOMContentLoaded es un evento que se lanza cuando una página se terminó de cargar.

Combinando esto, podemos escribir un código que sólo se ejecutará cuando la página esté cargada y así evitar el problema anterior.

```
document.addEventListener('DOMContentLoaded', function() {
  document.getElementById("resultadoSuma").innerHTML = 1 + 2;
});
```

https://www.w3schools.com/jsref/met_document_addeventlistener.asp

https://developer.mozilla.org/en-US/docs/Web/API/Window/DOMContentLoaded_event

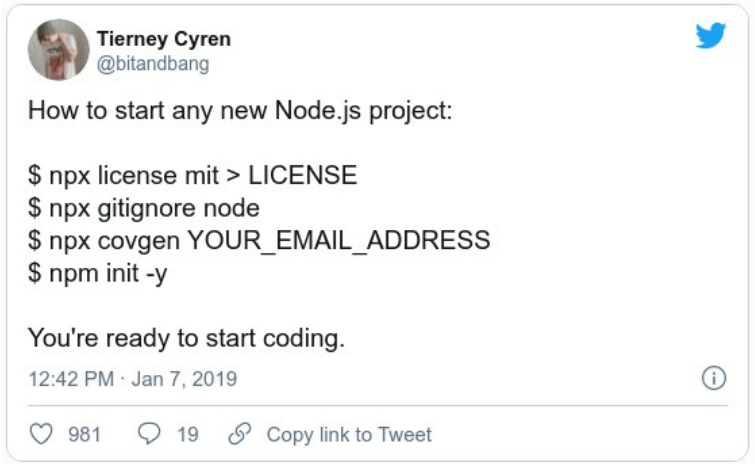
NodeJS

- Con NodeJS se podrá contar con un servidor WEB que entiende JavaScript.
- La instalación se debe hacer:
  - `$ sudo apt install nodejs`
  - O usando su página oficial https://nodejs.org/en/download/
- Una vez instalado se puede verificar su funcionamiento:
  - `$ node -v`
- Ahora se podrá ejecutar casi cualquier código de JavaScript tal como si fuera un programa en Perl, pero con algunas diferencias:
  - No habrá objetos DOM
  - console.log enviará mensaje a la salida estándar
  - En NodeJS se tendrá acceso al sistema de archivos del computadora
    - Este acceso debe ser no bloqueante!
    - Aquí se pueden aplicar también los promise
  - Los módulos podrían tener una sintaxis diferente

Revise el libro Begining Node.js, Express & MongoDB Development

Iniciar un proyecto en NodeJS

- NodeJS cuenta con un administrador de paquetes npm
  - `$ sudo apt install npm`



https://philna.sh/blog/2019/01/10/how-to-start-a-node-js-project/

https://www.contributor-covenant.org/es/version/2/0/code_of_conduct/code_of_conduct.md

Código de Conducta

- Ejemplos de comportamiento que contribuyen a crear un ambiente positivo para nuestra comunidad:
  - Demostrar empatía y amabilidad ante otras personas
  - Respeto a diferentes opiniones, puntos de vista y experiencias
  - Dar y aceptar adecuadamente retroalimentación constructiva
  - Aceptar la responsabilidad y disculparse ante quienes se vean afectados por nuestros errores, aprendiendo de la experiencia
  - Centrarse en lo que sea mejor no sólo para nosotros como individuos, sino para la comunidad en general
- Ejemplos de comportamiento inaceptable:
  - El uso de lenguaje o imágenes sexualizadas, y aproximaciones o atenciones sexuales de cualquier tipo
  - Comentarios despectivos (_trolling_), insultantes o derogatorios, y ataques personales o políticos
  - El acoso en público o privado
  - Publicar información privada de otras personas, tales como direcciones físicas o de correo electrónico, sin su permiso explícito
  - Otras conductas que puedan ser razonablemente consideradas como inapropiadas en un entorno profesional

Servidor Web básico en NodeJS

- En NodeJS se usa `require` para agregar módulos.
- El objeto `request`, contiene el URL que invocó el cliente
- El objeto `response`, contendrá la respuesta que recibirá el cliente.
- `console.log` imprime en la salida estandar

```
const http = require('http');
const server = http.createServer((request, response) => {
  console.log(request.url);
  response.end("Hola mundo");
});
server.listen(3000);
console.log("Escuchando en: http://localhost:3000")
```

```
apaz@ubuntu20:~/workspace/pweb2/nodejs$ nodejs markDownSever.js
Escuchando en: http://localhost:3000
```

Hola mundo

$ node index.js

Escuchando en http://localhost:3000

El framework Express

- Para hacer productivo a NodeJS es necesario usar frameworks
- Los frameworks nos permitirán reusar código y también hacer nuestros códigos más simples
- El framework express se puede instalar desde nuestro directorio de trabajo con npm
  - $ npm install express
- Esto instalará este framework sólo en nuestro directorio de trabajo y el archivo package.json, guardará las dependencias adecuadas.
- Es importante que los paquetes sean instalados localmente, para permitir que distinto proyectos puedan tener distintas versiones según lo necesiten

Servidor Web básico en NodeJS usando Express

- El código del servidor es muy similar al anterior, pero esta vez será más sencillo:
  - Responder a peticiones get o post
  - Responder a distintos URLs
  - Servir distintos archivos

```
const express = require('express');
const app = express();

app.listen(3000, () => {
  console.log("Escuchando en: http://localhost:3000")

});

app.get('/', (request, response) => {
  console.log(request.url);
  response.end("Hola mundo");
});
```

Servidor WEB en NodeJS con GET

- NodeJS nos permite escribir nuestros programas en el lado del Servidor usando JavaScript
- Estos scripts se pueden ejecutar en distintos SO por lo que la manera de especificar la ruta de un determinado Archivo puede ser distinta
- Por eso usaremos el módulo `path`
- Note que para cada URL distinta tendrá que crear un método `get` adicional

```
const path = require('path');
const express = require('express');
const app = express();

app.listen(3000, () => {
  console.log("Escuchando en: http://localhost:3000")

});

app.get('/', (request, response) => {
  response.sendFile(path.resolve(__dirname, 'index.html'));
});
```

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>NodeJS Test</title>
    <meta charset="UTF-8">
  </head>

  <body>
    <h1>Hola Mundo</h1>
  </body>
</html>
```

JavaScript en el cliente y el servidor

NodeJS reconoce a los archivos js, css, imágenes, etc. como archivos estáticos, para servirlos hay que crear un directorio y configurarlo en nuestro programa servidor.

```
const path = require('path');
const express = require('express');
const app = express();
app.use(express.static('pub'));

app.listen(3000, () => {
  console.log("Escuchando en: http://localhost:3000")

});

app.get('/', (request, response) => {
  response.sendFile(path.resolve(__dirname, 'index.html'));
});
```

markDownServer.js

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>NodeJS Test</title>
    <meta charset="UTF-8">
    <script src='test.js'></script>
  </head>

  <body>
    <h1>Hola Mundo</h1>
  </body>
</html>
```

index.html

```
console.log('Hola Mundo!');
```

pub/test.js

```
├── CODE_OF_CONDUCT.md
├── index.html
├── LICENSE
├── markDownServer.js
├── node_modules
├── package.json
├── package-lock.json
└── pub
```