Trabajando con Node.js

- PASO PREVIO: Comprobar que node está instalado. Ejecutar por consola de comandos:

```
$ node --version
```

- A modo de consulta, utilizar la API https://nodejs.org/api/index.html

1. Probar la consola de node.js

   Escribiendo "node" en la consola de comandos, tendremos acceso a la consola interna de node. Nos permite hacer ejecución de comandos.
   Ejemplo:

```
$ node
> var v1 = 1;
> var v2 = 2;
> console.log(v+v2)


> for(var i = 0;i<3;i++){
... console.log(i);
... };
```

2. Crear tu primer servidor con Node.js

   Crear un nuevo archivo Node.js en el directorio raíz de tu proyecto llamado "ejemplo_servidor.js" y añadir el siguiente código:

```
// usar módulo HTTP
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.end('MI primer servidor en Node');
}).listen(8080);
```

   A través del navegador, acceder a la siguiente URL. Nuestro servidor está escuchando en el puerto 8080.

```
http://localhost:8080
```

   2.1. Verificar que el servidor está funcionando. Mostrar mensaje por consola. [Para el alumno]

3. Utilizando el módulo HTTP.

   3.1. En la respuesta del servidor, devolver la URL de consulta que se ha realizado.
   Ejemplo URL petición: https://localhost:8080/ejemploconsulta
   Debería escribirnos en el navegador: /ejemploconsulta
   Nota: para acceder a la URL de petición, usar en el código del servidor **req.url,** que devuelve el la URL de petición

[Para el alumno]

3.2. Pasar parámetros en la petición HTTP y formatear la respuesta para que muestre los parámetros de la petición en el navegador.
Ejemplo URL petición:
http://localhost:8080/?param1=hola&param2=mundo&param3=1
Ejemplo de respuesta del servidor y cómo se vería en el navegador. “Hola mundo 1”
Nota:

- Es necesario hacer uso del módulo URL

```
var url = require('url');
```

- Es necesario parsear (url.parse().query) la URL para poder acceder a los parámetros.

4. Sistema de archivos

4.1. Crear un archivo ejemplo_archivo.html de ejemplo y guardar el archivo en el proyecto.

```
<html>

      <body>
      <h1>Sistema de ficheros</h1>
      <p>Probando ficheros Node</p>
      </body>
</html>
```

Node.js permite que hagamos operaciones de apertura/escritura de archivos. Para ello hay que utilizar el modelo correspondiente:

```
var fs = require('fs');
```

4.2. Con el siguiente código de ejemplo en el servidor se podría hacer lectura de archivo HTML y enviarlos como respuesta de nuestro servidor node JS

```
      var http = require('http');
      // Modulo file sistem
      var file = require('fs');

      http.createServer(function (req, res) {
        file.readFile('html/index.html', function(err, data) {
          res.writeHead(200, {'Content-Type': 'text/html'});
          res.write(data);
          res.end();
        });
}).listen(8080);
```

5. Manejo de Streams. Son objetos que se utilizan como canal de lectura, escritura o Duplex(lectura y escritura) de datos. Vamos a hacer uso del fichero HTML creado anteriormente.
   5.1. Vamos a utilizar un Stream de lectura de datos. Crear el archivo stream_datos.js

```
// Hacer uso del modelo fs
var fs = require("fs");
var datos_guardados= '';

// Crear Stream de lectura
var readerStream = fs.createReadStream('index.html');

// Manejar eventos del stream de lectura --> data, end, error
readerStream.on('data', function(caracter) {
   datos_guardados += caracter;
});
// Evento para cuando termine de leer fichero
readerStream.on('end',function(){
   console.log(datos_guardados);
});
// Evento en caso de error en la lectura
readerStream.on('error', function(err){
   console.log(err.stack);
});
console.log("Fin de lectura de datos");
```

Comprobar el resultado ejecutando en la consola el archivo

- **node stream_datos.js**

6. Eventos en Node.js. Los objetos utilizados en Node.js pueden disparar eventos. En nuestro caso, vamos a abrir nuestro archivo HTML creado anteriormente y comprobar cómo manejar el evento de apertura de archivo.

   6.1. Hacer uso del archivo HTML creado para manejo de ficheros.

```
<html>
      <body>
      <h1>Sistema de ficheros</h1>
      <p>Probando ficheros Node</p>
      </body>
</html>
```

6.2. Crear el archivo eventos.js e introducir el siguiente código:

```
var fs = require('fs');
//Stream de lectura de datos
var readStream = fs.createReadStream('./ejemplo_archivo.html);
//Cuando se abra el archivo, mostrar por consola un mensaje
readStream.on('open', function () {
      console.log('El fichero se ha abierto');
});
```

6.3. Ejecutar en la consola eventos.js

- **Node eventos.js**

6.4. Debe mostrarte por consola: **'El fichero se ha abierto'**

7. Manejo de Eventos usando el Módulo de Eventos. Tiene la ventaja de que te permite crear, disparar y escuchar tus propios eventos.
   7.1. Crear un fichero mi_evento.js y escribir el siguiente código.

```
// Módulo para manejar eventos
var events = require('events');
// Objeto emisor de eventos
var eventEmitter = new events.EventEmitter();

// Función manejadora del evento creado
var alarma_horno = function () {
  console.log('Pizza lista para comer');
}

//Asignar mi manejador de eventos
eventEmitter.on('pizza_lista', alarma_horno);

//Fire the 'scream' event:
eventEmitter.emit('alarma_lista');
```

7.2. Ejecutar en la consola mi_evento.js

- **Node mi_evento.js**

7.3. Debe mostrarte por consola: **'Pizza lista para comer'.**