2º Desarrollo de Aplicaciones Multiplataforma

Optativa

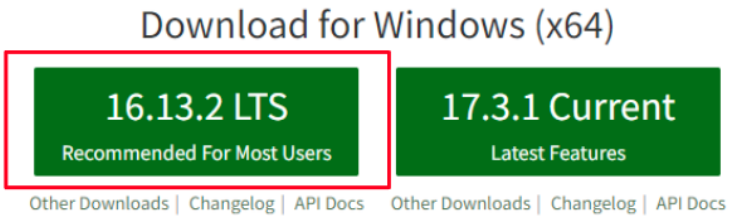
Proyecto de desarrollo de
aplicaciones multiplataforma

Jose Antonio Álvarez Romero

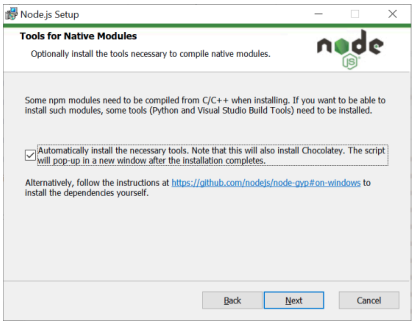
Manual de uso e instalación de NODE, NPM, HTTP y PATH.

Node y npm.

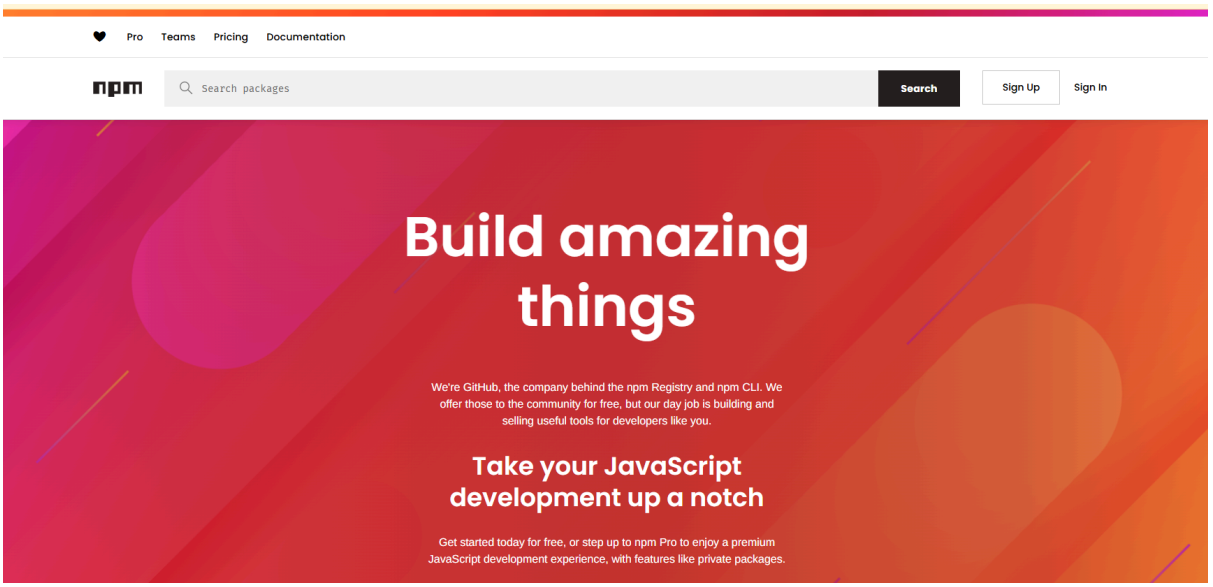
Node se descarga en la página nodejs.org
Se instala ejecutando el .exe y siguiendo los pasos. Al instalarlo se instalara conjuntamente **NPM.**



En consola, con el código *npm init -y* inicializamos el proyecto y con *npm install* y el nombre de un paquete instalaremos dicho paquete.



Para saber que paquetes existen y como se llama lo más sencillo es recurrir a la página oficial de npm donde están todos los paquetes con su información.
https://www.npmjs.com/




```
PM> node .\server.js
```

Para ejecutar un archivo con node es necesario, en consola, usar el comando node “nombre de archivo”

Http y path.

Para usar http y path en nuestro proyecto es necesario usar las líneas de código dentro del archivo javascript *require('http')* y *require('PATH')*

```
var http = require('http').
    path = require('path'),
```

Para el uso de http es necesario darle un nombre al host y definir un puerto al que acceder, y con la función *createServer* podemos levantar un servidor. Además de lo anterior, también podemos definir atributos como el header o una respuesta del servidor, en este caso "Hola mundo".

```
const { createServer } = require('node:http');

const hostname = '127.0.0.1';
const port = 3000;

const server = createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Al ejecutar levanta un servidor http en el puerto especificado con el nombre 127.0.0.1.

```
PS C:\Users\alumno\Desktop\JS_optativa\02ModulosNPM> node .\server.js
Server running at http://127.0.0.1:3000/
```

También se le puede pasar como parámetro un archivo, ya sea de texto o html.

```
function webServer(req, res){
    res.writeHead(200, {'Content-Type':'text/html'}).end('<h1>Hola Node.js<
}
```

Y con PATH podemos a un servidor, en función de la ruta, devolver un archivo html u otro si estos se encuentran en los definidos.

```
path = require('path'), //Requerimos el módulo path
urls = [  //Declaramos un array con 3 objetos
    { //Este primer objeto no contiene nada en la ruta
        ruta : '',
        output : '<h2>Home</h2>'
    },
    { //El segundo contendrá la ruta hacia acerca
        ruta : 'acerca',
        output : '<h2>Acerca</h2>'
    },
    {//El tercero contendrá la ruta hacia contacto
        ruta : 'contacto',
        output : '<h2>Contacto</h2>'
    }
]
```

De modo que la función webserver comparará la ruta que pasamos al servidor con todas las rutas posibles y devolverá la respuestas correspondiente.

```
    ]
function webServer(req, res)
{
    var message = '<h1>Hola Node.js</h1>',
        pathURL = path.basename(req.url)
        //El método basename eliminará todo de la ruta
        //excepto el nombre del fichero final (index.html, por ejemplo)
        //El objeto req devolverá la ruta completa de la petición
console.log("Ruta completa: "+req.url)
console.log("Ruta corta: "+pathURL)
//Recorremos todas las rutas del array                    pa que busque s
    urls.forEach(function (pos){
        if(pos.ruta == pathURL)
        {
            res.writeHead(200, {'Content-Type':'text/html'})
            res.end(message + pos.output)
        }
    })
    //En caso de que la URL no exista (SI NO OBTENEMOS RESPUESTA)
    if(!res.finished)
    {
        res.writeHead(404, {'Content-Type':'text/html'})
        res.end('<h1>Error 404: Not Found</h1><br><h2>Eres un melon, la rut
    }

}
```

Dentro de las funcionalidades de http también podemos usar una en la cual el servidor nos enseña un archivo html(GET), con un formulario por ejemplo, y devolverle esa información(POST), ya sea en texto plano o con otros formatos como Json.

```
var http = require('http').createServer(webServer),
    form = require('fs').readFileSync('form.html'),
    querystring = require('querystring'),
    util = require('util'),
    dataString = ''; //aqui se concatena el resultado

function webServer(req, res)
{
    if(req.method == 'GET')         //si la peticion es un get devuelve el
    {
        res.writeHead(200, {'Content-Type' : 'text/html'})
        res.end(form)
    }
    if(req.method == 'POST')
    {
        req
            .on('data', function (data){ //Mientras haya datos, ejecutaremo
                dataString += data //Que concatenará el dato en la variable
            })
            .on('end', function (){
                var dataObject = querystring.parse(dataString),
                dataJson = util.inspect(dataObject),
                templateString = `
                Los datos que enviaste por POST como string son : ${dataStr
                Los datos que enviaste por POST como Json son: ${dataJson}
                `
                console.log(templateString)
                res.end(templateString)
```

Próximos pasos.

Trabajas con las siguientes herramientas:

- GET
- POST
- PUT
- DELETE

Lo que viene a ser el CRUD desde el servidor.

Conclusiones.

Con esto podemos comprobar, no solo el potencial de estas herramientas, que nos permiten levantar servidores con cualquier tipo de archivo y encima devolver y procesar información

sino además la facilidad del proceso, que hace que en pocos minutos podamos tener todo funcional y sin errores.