

Índice.

1. Integrando JavaScript y HTML en navegadores.
2. Ejecutar javascript con node.
3. Proyectos de node.

El objetivo del módulo DI es realizar interfaces gráficas de usuario para entornos escritorio. Este se puede hacer con lenguajes de programación como java o c# pero también se puede hacer usando tecnologías web para generar aplicaciones de escritorio. En este último caso usaremos node, que permite ejecutar javascript directamente sin usar navegador. Este tema es una introducción a node y los proyectos node.

1. Integrando JavaScript y HTML en navegadores.

Para integrar el código JavaScript en nuestro HTML, necesitamos usar la etiqueta `<script>`. El sitio recomendado para poner la etiqueta es justo antes de cerrar la etiqueta `</html>`, para que algunos navegadores puedan cargar y construir el DOM antes de procesar el código JavaScript, de otro modo, el navegador bloqueará el renderizado de la página hasta que se haya procesado todo el código JS.

Dentro de la etiqueta `<script>`:

```
<!DOCTYPE>
<html>

<head>
  <title>Ejemplo JS</title>
</head>

<body>
  <p>Hola Mundo!</p>
  <script>
```

```
    console.log("Hola Mundo!");  
  </script>  
</body>  
  
</html>
```

En un archivo separado:

Archivo: ejemplo1.html

```
<!DOCTYPE>  
<html>  
  
<head>  
  <title>Ejemplo JS</title>  
</head>  
  
<body>  
  <p>Hola Mundo!</p>  
  <script src="ejemplo1.js"></script>  
</body>  
  
</html>
```

Será necesrio crear el archivo ejemplo1.js:

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

2. Ejecutar javascript con node.

Node.js es un entorno de ejecución en el lado del servidor construido utilizando el motor Javascript de Google Chrome, llamado V8. Lo que se hace es utilizar de manera externa el mismo motor de ejecución que emplea Google Chrome para compilar y ejecutar Javascript en el navegador: el motor V8. Dicho motor se encarga de compilar y ejecutar el código Javascript, transformándolo en código más rápido (código máquina).

También se encarga de colocar los elementos necesarios en memoria, eliminar de ella los elementos no utilizados (garbage collection), etc.

V8 está escrito en C++, es open-source y de alto rendimiento. Se emplea en el navegador Google Chrome y

"variantes" como Chromium (adaptación de Chrome a sistemas Linux), además de en Node.js y otras aplicaciones. Podemos ejecutarlo en sistemas Windows (XP o posteriores), Mac OS X (10.5 o posteriores) y Linux (con procesadores IA-32, x64, ARM o MIPS).

Además, al estar escrito en C++ y ser de código abierto, podemos extender las opciones del propio Javascript. Como hemos comentado, inicialmente Javascript era un lenguaje concebido para su ejecución en un navegador. No podíamos leer un fichero de texto local, por ejemplo. Sin embargo, con Node.js se ha añadido una capa de funcionalidad extra a la base proporcionada por V8, de modo que ya es posible realizar estas tareas, gracias a que con C++ sí podemos acceder a los ficheros locales, o conectar a una base de datos. En concreto desde node disponemos:

- Mecanismos para acceder al sistema de ficheros, lo que es particularmente útil para leer ficheros de texto, o subir imágenes al servidor, por poner dos ejemplos.
- Mecanismos para conectar con bases de datos.
- Comunicarnos a través de Internet (el estándar ECMAScript no dispone de estos elementos para Javascript).
- Aceptar peticiones de clientes y enviar respuestas a dichas peticiones.

Instalaciones necesarias para este curso

Node puede descargarse desde la web:
<https://nodejs.org/es/>

Es muy sencillo de instalar.

Visual Studio Code

Es un editor de código multilenguaje muy usado con node:

<https://code.visualstudio.com/download>

Durante la instalación de code nos pregunta si deseamos incluir en el menú contextual la opción de abrir ficheros y carpetas con code, le diremos que sí, pues nos será muy útil.

Hola mundo en Node

Podemos ejecutar archivos javascript sin necesidad de un navegador, en el ejemplo que hemos creado punto 1, para ejecutar este fichero, desde la línea de comando, nos situamos dentro del directorio que contiene ejemplo1.js y escribimos: node ejemplo1.js

3. Proyectos de node.

NPM

npm es el sistema de gestión de paquetes por defecto para Node.js

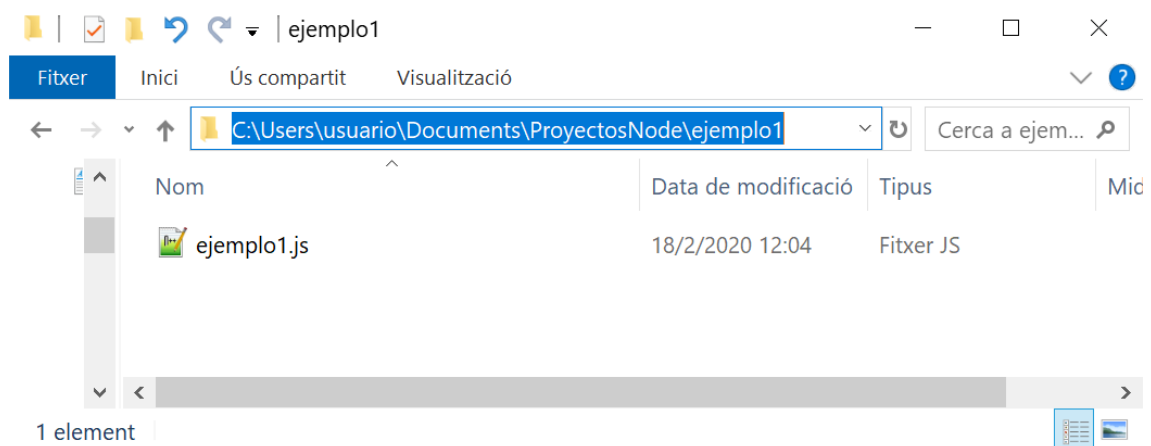
Proyectos en Node

Lo habitual cuando se trabaja con node es tener una carpeta que contendrá el proyecto. Para iniciar un proyecto usaremos npm:

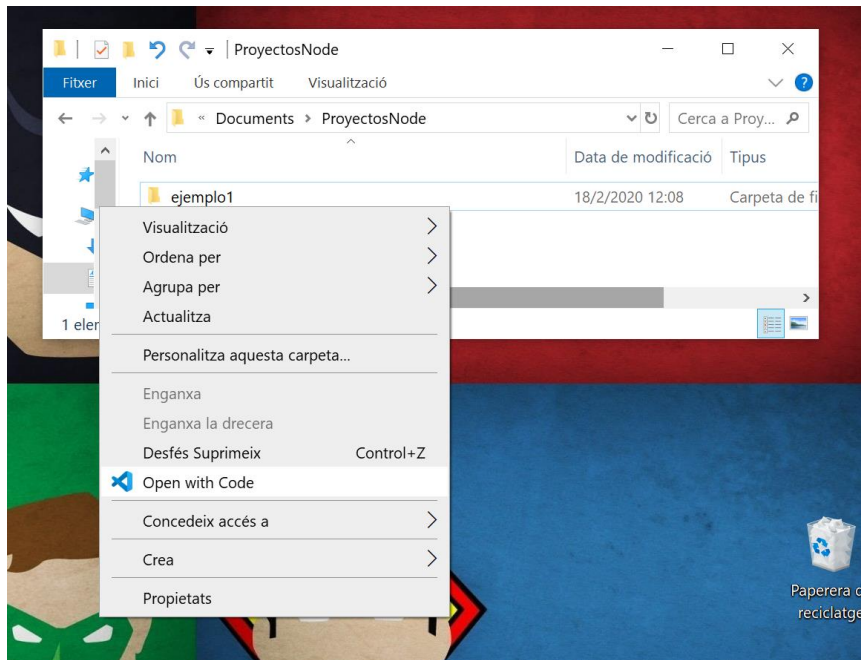
1. Crea una carpeta en la ubicación que desees, dale un nombre sin espacios en blanco.
2. Después accede a la carpeta desde tu consola o terminal y ejecuta el siguiente comando: `npm init`. El comando anterior genera un archivo `package.json` personalizado para cada proyecto node.

Ejemplo de un proyecto con node

Previamente tenemos que realizar la instalación de node y visual studio code. Después vamos a crear un proyecto con node usando el anterior ejemplo `ejemplo1.js`. Primero creamos un directorio llamado `ejemplo1` y copiamos el archivo `ejemplo1.js`:

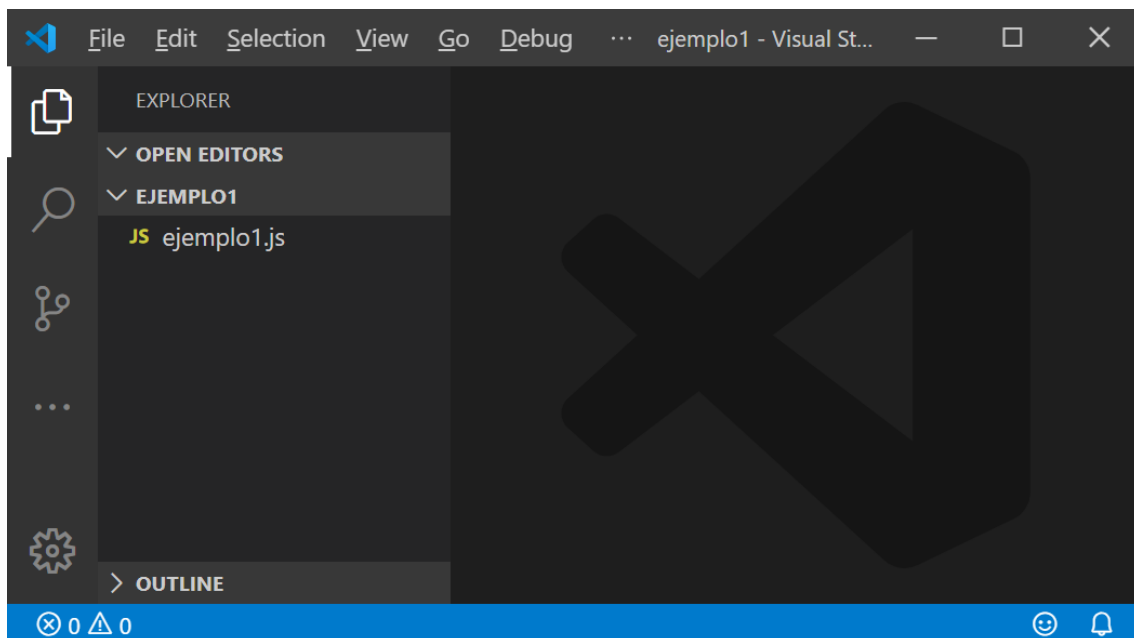


Abre la carpeta con visual studio code haciendo clic con el botón derecho sobre la carpeta `ejemplo1` y seleccionado la opción de abrir con code:

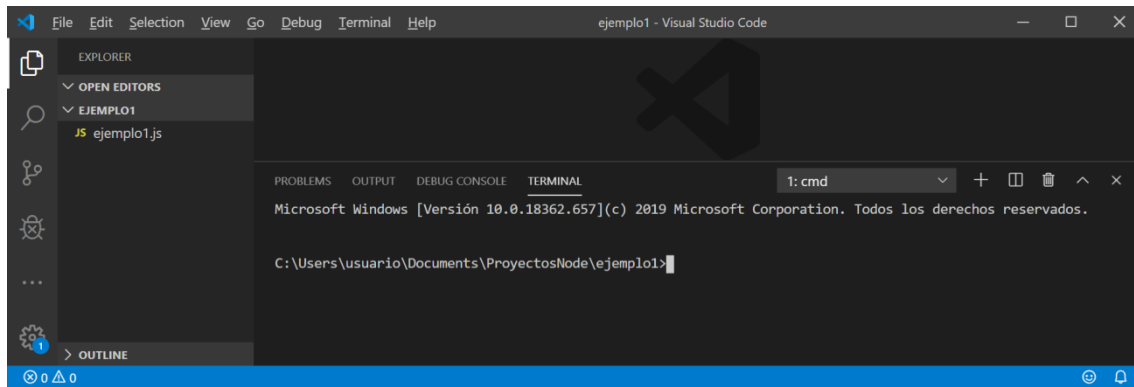


Es importante abrir la carpeta adecuada, no una superior, esto puede llevar a errores.

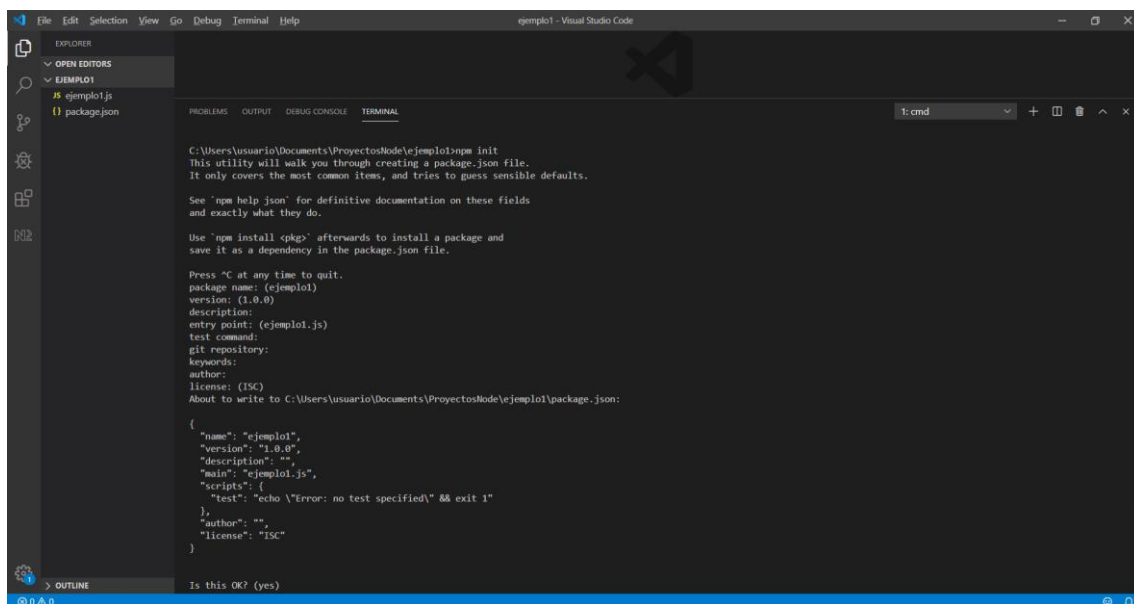
Te abrirá visual studio code con el contenido de esa carpeta, para saber que estás en la carpeta adecuada tiene que poner su nombre, en este caso ejemplo1, en primer nivel y no como una subcarpeta:



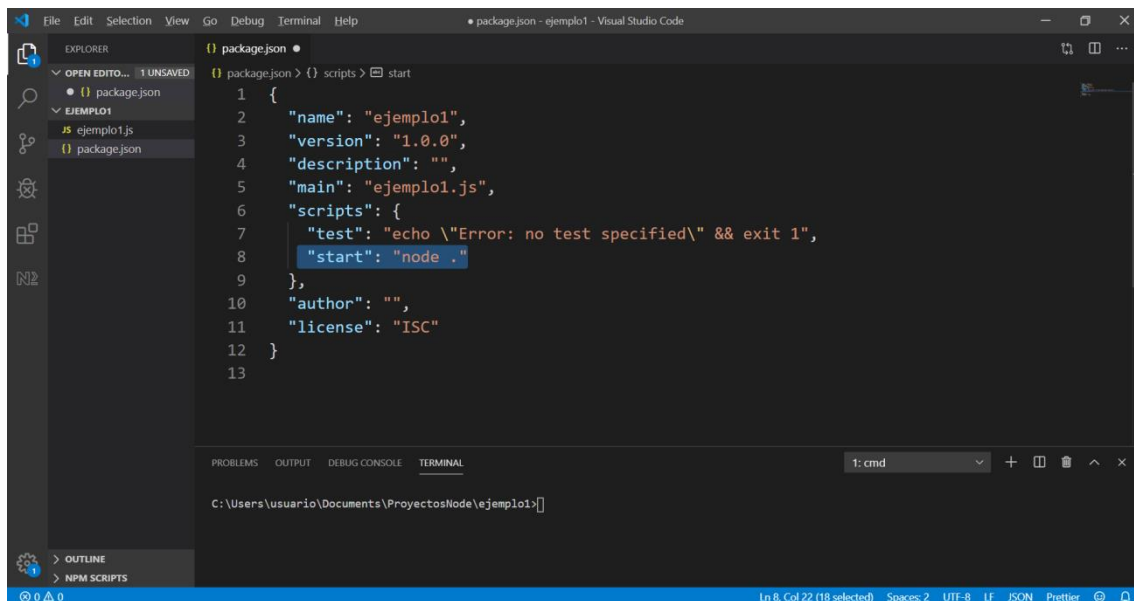
Para ejecutar npm podemos hacerlo desde la línea de comandos habitual o usando la de code. Para desplegarla haz “CTRL + ñ”:



Con el comando `npm init` crearemos el proyecto, lo que supone la generación un fichero `package.json` que defina nuestro proyecto. Al ejecutar `npm init` desde dentro de nuestra carpeta de proyecto (en este caso `ejemplo1`) podemos dejar todas las opciones por defecto dándole a enter:



Vamos a añadir una línea a `package.json` para ejecutar el proyecto usando `npm`:

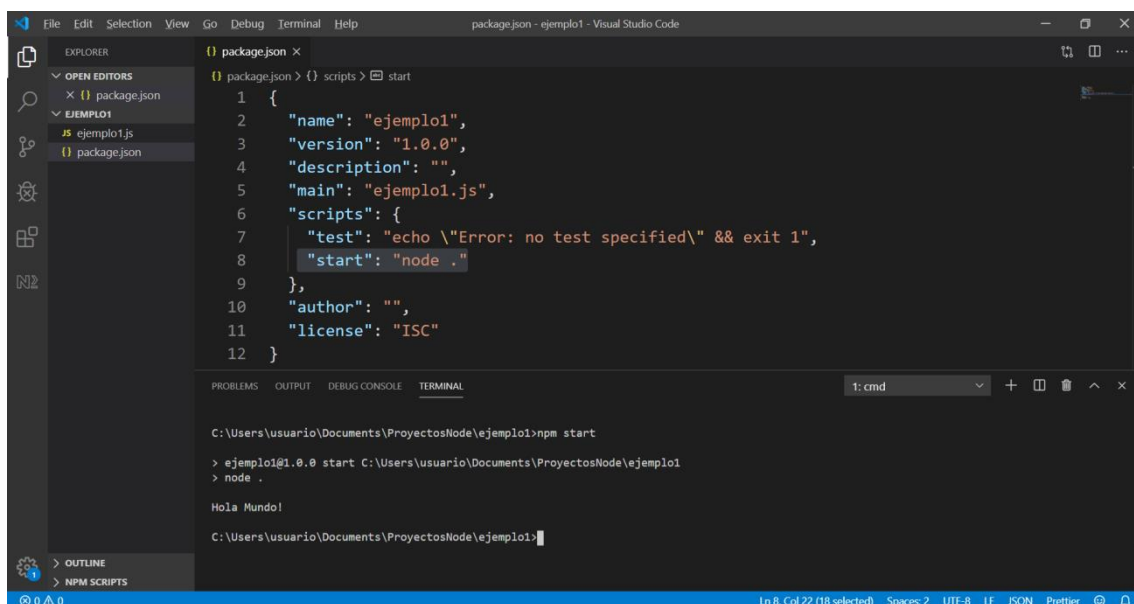


The screenshot shows the Visual Studio Code interface with the `package.json` file open. The file content is as follows:

```
1 {
2   "name": "ejemplo1",
3   "version": "1.0.0",
4   "description": "",
5   "main": "ejemplo1.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "start": "node ."
9   },
10  "author": "",
11  "license": "ISC"
12 }
13
```

The `"start": "node ."` line is highlighted in blue. The terminal at the bottom shows the command prompt: `C:\Users\usuario\Documents\ProyectosNode\ejemplo1>`.

Guardamos el contenido de `package.json`, ahora podemos ejecutar el proyecto usando `npm start`:



The screenshot shows the Visual Studio Code interface with the `package.json` file open. The terminal at the bottom shows the command `npm start` being executed, resulting in the output:

```
C:\Users\usuario\Documents\ProyectosNode\ejemplo1>npm start
> ejemplo1@1.0.0 start C:\Users\usuario\Documents\ProyectosNode\ejemplo1
> node .
Hola Mundo!
C:\Users\usuario\Documents\ProyectosNode\ejemplo1>
```

Lo que hace `npm start`, es ejecutar el comando que hemos puesto en la línea `start` del `package.json`, usando como fichero javascript el especificado en la entrada `main` de `package.json`.