Curso de NodeJS

Unidad Didáctica 03: Gestión de módulos

http://cursosdedesarrollo.com/

Ayuntamiento
de Vitoria-Gasteiz
Vitoria-Gasteizko
Udala

Índice de contenidos

http://cursosdedesarrollo.com/

Introducción

Un módulo es un conjunto de funcionalidades gestionadas desde npm

http://cursosdedesarrollo.com/

Introducción

Para iniciar un módulo basta con ejecutar

npm init

sobre el directorio que se quiere gestionar

http://cursosdedesarrollo.com/

Introducción

Desde ese momento se puede gestionar el contenido completo del módulo identificando todos sus elementos principales: nombre, versión, dependencias, …

http://cursosdedesarrollo.com/

Introducción

Toda esta información se gestiona dentro del fichero packages.json

http://cursosdedesarrollo.com/

packages.json

Dentro del fichero sólo hay dos elementos que son imprescindibles: nombre (name) y versión (version)

http://cursosdedesarrollo.com/

packages.json

```
{
  "name": "my-awesome-package",
  "version": "1.0.0"
}
```

http://cursosdedesarrollo.com/

packages.json

- name: el nombre del directorio y del módulo
- version: siempre x.y.z
- description: información del readme, o una cadena vacía ""
- main: fichero principal, por defecto: index.js
- scripts: por ejemplo los scripts de test
- keywords: palabras clave
- author: nombre <direccion@dominio.com>
- license: GPL
- bugs: información de fallos normalmente una url
- homepage: información del módulo, normalmente una url

http://cursosdedesarrollo.com/

packages.json


```
{
  "name": "my_package",
  "description": "",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": { "test": "echo \"Error: no test specified\" && exit 1"  },
  "repository": { "type": "git", "url": "https://github.com/ashleygwilliams/my_package.git"  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {  "url": "https://github.com/ashleygwilliams/my_package/issues"  },
  "homepage": "https://github.com/ashleygwilliams/my_package"
}
```


http://cursosdedesarrollo.com/

packages.json

Se pueden cambiar las opciones por defecto:

> npm set init.author.email "wombat@npmjs.com"

> npm set init.author.name "ag_dubs"

> npm set init.license "MIT"

packages.json

O bien crear un fichero de plantilla .npm-init.js:

```
module.exports = {
  customField: 'Custom Field',
  otherCustomField: 'This field is really cool'
}
```

http://cursosdedesarrollo.com/

packages.json

Que cada vez que se ejecute npm init:

```
{
  customField: 'Custom Field',
  otherCustomField: 'This field is really cool'
}
```

http://cursosdedesarrollo.com/

packages.json

Dentro del fichero se pueden manejar las dependencias del módulo con el atributo:

“dependencies”

http://cursosdedesarrollo.com/

packages.json

Dentro del fichero se pueden manejar también las dependencias de desarrollo o pruebas del módulo con el atributo:

“devDependencies”

http://cursosdedesarrollo.com/

packages.json

```
{
  "name": "my_package",
  "version": "1.0.0",
  "dependencies": {
    "my_dep": "^1.0.0"
  },
  "devDependencies" : {
    "my_test_framework": "^3.1.0"
  }
}
```

http://cursosdedesarrollo.com/

packages.json

Cada vez que utilizamos el --save en un npm install

modificamos el packages.json para reflejar esa dependencia

http://cursosdedesarrollo.com/

packages.json

Cada vez que utilizamos el --save-dev en un npm install

modificamos el packages.json para reflejar esa dependencia de desarrollo

Webpack

Es un empaquetador de módulos para aplicaciones Javascript

https://webpack.js.org/

http://cursosdedesarrollo.com/

Webpack

Conceptos importantes:

Entry

Output

Loaders

Plugins

http://cursosdedesarrollo.com/

Webpack

Entry gestiona un grafo con las dependencias del módulo para webpack

webpack.config.js

```
module.exports = {

entry: './path/to/my/entry/file.js'

};
```

http://cursosdedesarrollo.com/

Webpack

Output es capaz de compilar los assets del módulo e indicarle un sitio donde colocar ese bundle

http://cursosdedesarrollo.com/

Webpack

webpack.config.js

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

module.exports = {
    entry: './path/to/my/entry/file.js',
    output: {
      path: path.resolve(__dirname, 'dist'),
      filename: 'my-first-webpack.bundle.js'
    }
};
```

http://cursosdedesarrollo.com/

Webpack

Loaders son los responsables de la carga de assets para que el navegador no sea el responsable de su carga, sino sea Webpack quien sepa qué cargar

http://cursosdedesarrollo.com/

Webpack

La idea con los Loaders es que los assets sean dependencias del grafo que se transformen para su inclusión

http://cursosdedesarrollo.com/

Webpack

Pero Webpack sólo entenderá Javascript, no CSS, no JPG, etc…

http://cursosdedesarrollo.com/

Webpack

Se pueden gestionar los “módulos” de web pack en base a reglas (rules)

http://cursosdedesarrollo.com/

Webpack

Se pueden gestionar los “módulos” de web pack en base a reglas (rules)

http://cursosdedesarrollo.com/

Webpack

webpack.config.js

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

const config = {
  entry: './path/to/my/entry/file.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'my-first-webpack.bundle.js'
  },
  module: {
    rules: [ {test: /\.(js|jsx)$/, use: 'babel-loader'}    ]
  }
};

module.exports = config;
```

http://cursosdedesarrollo.com/

Webpack

Aunque los loaders están bien, la manera más habitual de ampliar las funcionalidades de Webpack es con Plugins

http://cursosdedesarrollo.com/

Webpack

Los plugins son mucho más personalizables y extensibles

https://webpack.js.org/plugins/

http://cursosdedesarrollo.com/

Webpack

webpack.config.js

```
const HtmlWebpackPlugin = require('html-webpack-plugin'); //installed via npm

const webpack = require('webpack'); //to access built-in plugins

const path = require(‘path');

const config = {

….

plugins: [

    new webpack.optimize.UglifyJsPlugin(),

    new HtmlWebpackPlugin({template: './src/index.html'})

  ]

};

module.exports = config;
```

http://cursosdedesarrollo.com/

Publicación de Paquetes

Para publicar paquetes es necesario disponer de un usuario en https://www.npmjs.com

Publicación de Paquetes

Creamos el usuario con

npm adduser

http://cursosdedesarrollo.com/

Publicación de Paquetes

Una vez introducido nombre de usuario, contraseña
e email ya podemos publicar módulos

http://cursosdedesarrollo.com/

Publicación de Paquetes

Para hacer login

npm login

para desbloquearse

npm logout

http://cursosdedesarrollo.com/

Publicación de Paquetes

Una vez creado el usuario y logueado ya podemos publicar

npm publish

http://cursosdedesarrollo.com/

.gitignore y .npmignore

Cuando publicamos se van a tener en cuenta estos dos ficheros para saber lo que se puede y no se puede subir

http://cursosdedesarrollo.com/

.gitignore y .npmignore

Ambos ficheros son compatibles y se tendrán en cuenta, básicamente los dos ficheros tienen la misma sintaxis

https://git-scm.com/docs/gitignore

.gitignore y .npmignore

Aquello que se defina en el .npmignore prevalecerá por encima del .gitignore

si no existe el .npmignore se usará el .gitignore si existe

http://cursosdedesarrollo.com/

Versionado

Cada vez que queramos actualizar el módulo será necesario que cambiemos la versión definida en el packages.json

http://cursosdedesarrollo.com/

Versionado

Y luego podremos volver a publicar el módulo:

npm publish

http://cursosdedesarrollo.com/

Módulos privados

En el sitio de npm se puede pagar para disponer de módulos privados

https://www.npmjs.com/pricing

Módulos privados

Esto permite no publicar a todos los usuarios los paquetes subidos

http://cursosdedesarrollo.com/

Módulos privados

Los módulos pueden ir asociados a un determinado scope o nombre de usuario

@scope/project-name

@username/project-name

http://cursosdedesarrollo.com/

Módulos privados

Cuando se inicializa un módulo se puede indicar el scope:

npm init --scope=<your_scope>

http://cursosdedesarrollo.com/

Módulos privados

O bien se puede cambiar el scope de un módulo ya creado

npm config set scope <your_scope>

Módulos privados

Los módulos publicados por defecto de manera privada si tienen un scope asociado, sino serán módulos públicos

Módulos privados

Un módulo privado puede cambiarse su visibilidad desde la página de npmJS

https://docs.npmjs.com/private-modules/intro

Módulos privados

Si los módulos son privados cómo podemos descargarlos?

http://cursosdedesarrollo.com/

Módulos privados

Si los módulos son privados cómo podemos descargarlos?

Usando un token de autenticación disponible en .npmrc

Módulos privados

//
registry.npmjs.org/:_authToken=00000000-0000-0000
-0000-000000000000

http://cursosdedesarrollo.com/

Módulos privados

Luego este token se puede exportar como una variable de sistema:

export NPM_TOKEN="00000000-0000-0000-0000-000000000000"

http://cursosdedesarrollo.com/

Módulos privados

//registry.npmjs.org/:_authToken=${NPM_TOKEN}

http://cursosdedesarrollo.com/

Dependencias Git

Si no queremos utilizar los repositorios públicos de módulos de NPM, también podemos utilizar dependencias Git

http://cursosdedesarrollo.com/

Dependencias Git

git://github.com/user/project.git#commit-ish

git+ssh://user@hostname:project.git#commit-ish

git+ssh://user@hostname/project.git#commit-ish

git+http://user@hostname/project/blah.git#commit-ish

git+https://user@hostname/project/blah.git#commit-ish

http://cursosdedesarrollo.com/

Dependencias Git

O incluso dependencias GitHub:

```
{
  "name": "foo",
  "version": "0.0.0",
  "dependencies": {
    "express": "visionmedia/express",
    "mocha": "visionmedia/mocha#4727d357ea"
  }
}
```

http://cursosdedesarrollo.com/

Conclusiones

Hemos visto cómo crear y publicar módulos node con NPM

http://cursosdedesarrollo.com/

Datos de Contacto

http://www.cursosdedesarrollo.com
david@cursosdedesarrollo.com

http://cursosdedesarrollo.com/

Licencia

David Vaquero Santiago

http://cursosdedesarrollo.com/