

REACT PARA PRINCIPIANTES

TEMA 2: SETUP

Material obtenido de Open Webinars

TEMAS DEL CURSO

REACT PARA

PRINCIPIANTES



1. **FUNDAMENTOS:** qué saber antes de iniciarse con React
2. **SETUP:** crear un proyecto React desde cero.
3. **RENDERIZADO:** cómo aprovechar las capacidades de renderizado de React
4. **PROPS & STATE:** comunicación de componentes

ÍNDICE DEL TEMA 2

INTRODUCCIÓN A REACT. 2 - SETUP



Dependencies



Bundling



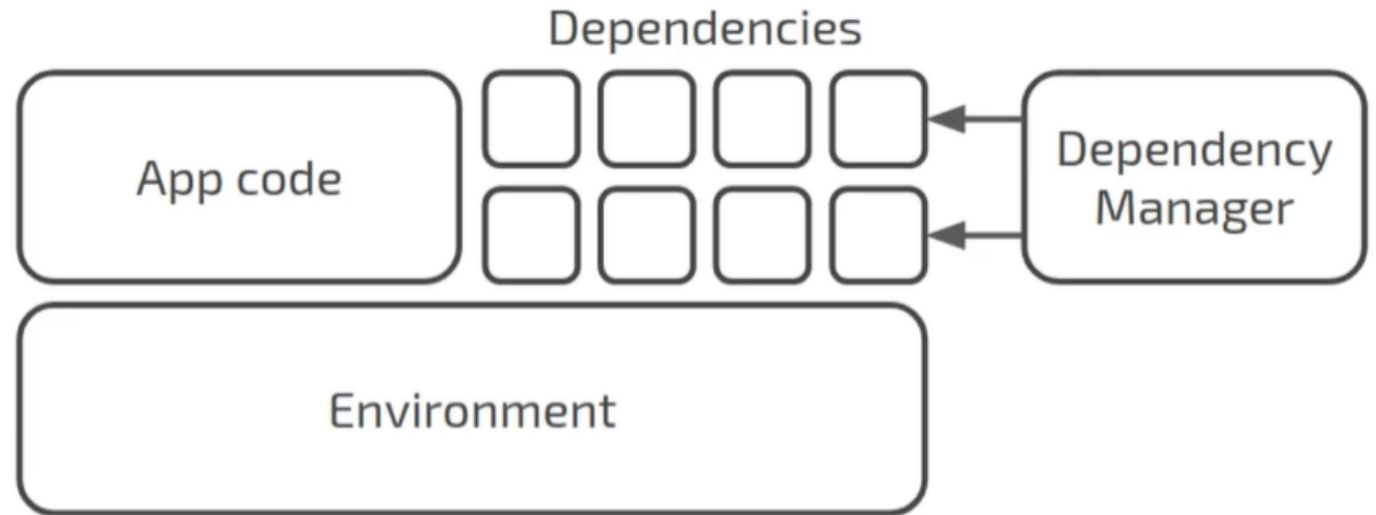
Code style



Dev Tools

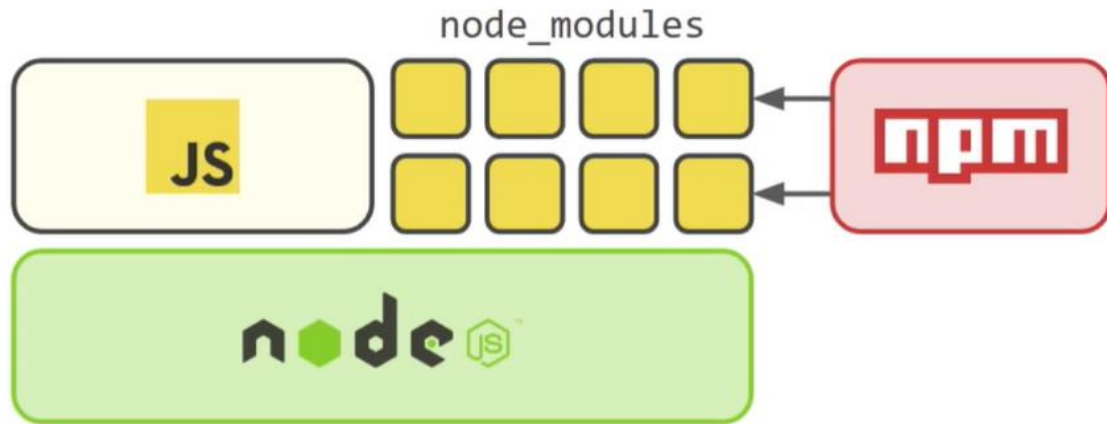
DEPENDENCIAS

- Estructura básica de una app común de cualquier tipo (web, cliente, servidor...)

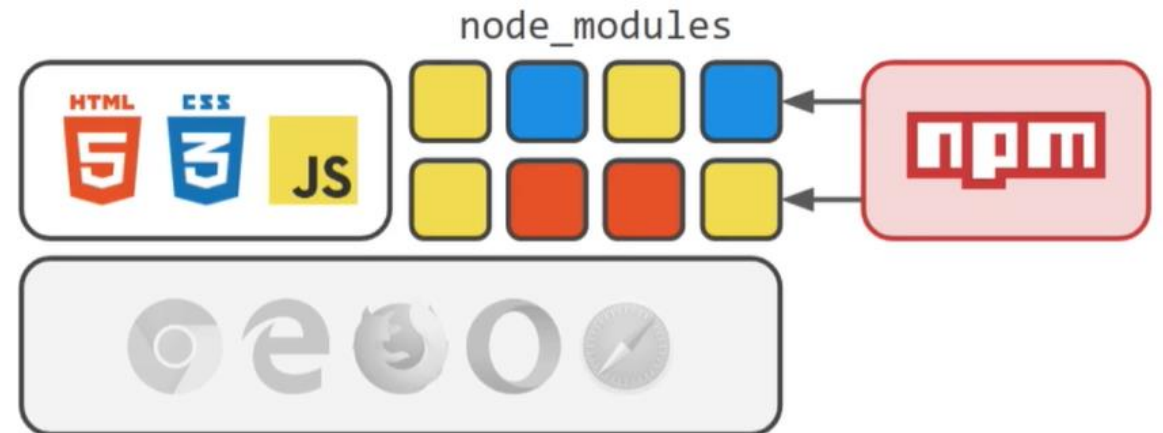


- ❑ **Entorno:** la base sobre la que se ejecuta nuestro código
- ❑ **App code:** nuestro código (nuestra lógica de negocio). Va acompañado de dependencias
- ❑ **Dependencias:** librerías o paquetes (generalmente de terceros) que aportan funcionalidad a nuestro módulo. Reutilización.
- ❑ **Gestor de dependencias:** versionado, interdependencias, etc.

EJEMPLOS TÍPICOS



Ejemplo de
Estructura típica en servidor con javascript



Estructura típica de dependencias
actual en React



INICIAMOS NUESTRO PROYECTO

- Desde Visual Code nos clonamos el repositorio con github

<https://github.com/vanesaespin/whishlist-react.git>

PASOS PARA LA CREACIÓN DEL PROYECTO

1. Install NodeJS

```
https://nodejs.org
```

Si no lo tenemos..
node -v nos dice la versión

2. Check NPM version

```
npm -v
```

Chequeamos que tenemos instalado node.js y npm

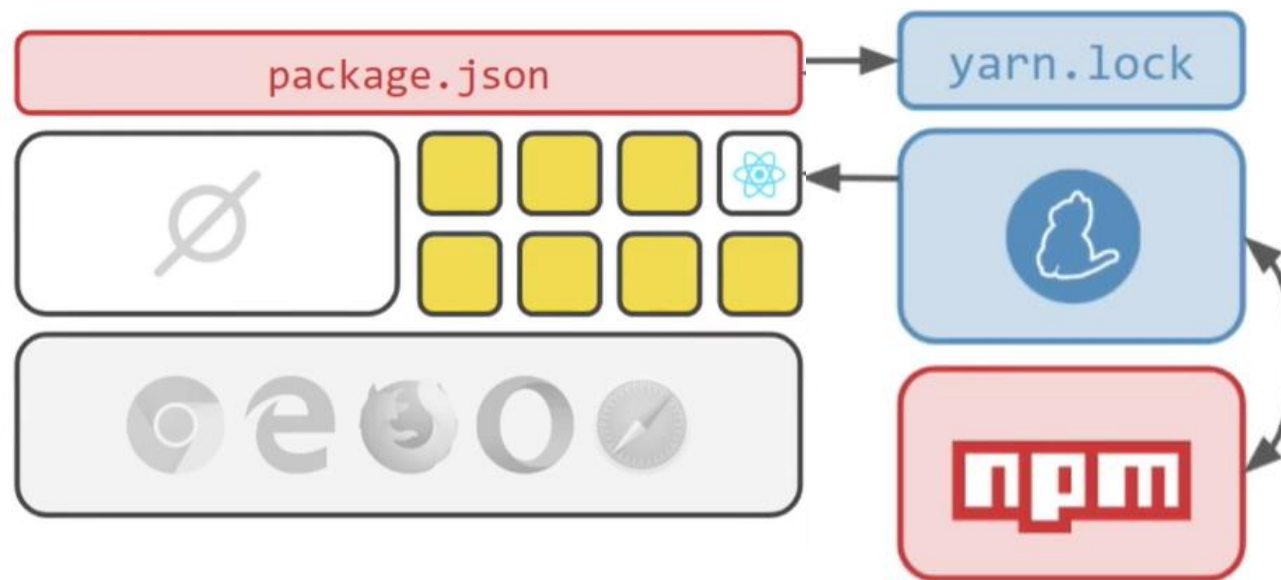
4. Start a project in the folder

```
npm init
```

INSTALAMOS REACT EN NUESTRO PROYECTO

```
npm install --save react
```

Ya tenemos esto



En la actualidad se usa una capa sobre npm llamada **yarn** que facilita su uso

REACT-DOM

- Instalamos también react-dom

```
npm install --save react-dom
```

- En nuestro package.json habrán aparecido ya las dos dependencias instaladas

```
"author": "vanesa espin",  
"license": "MIT",  
"dependencies": {  
  "react": "^18.2.0",  
  "react-dom": "^18.2.0"  
}
```

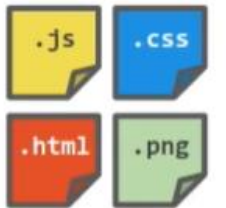
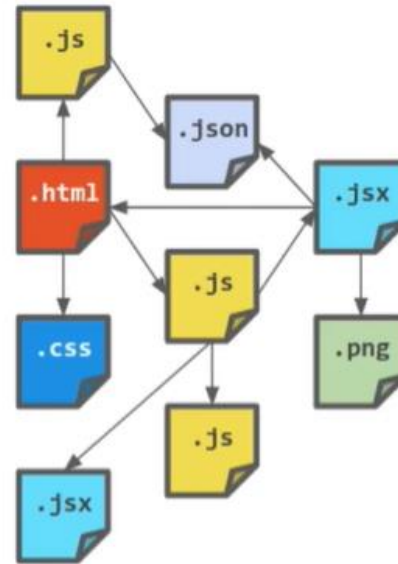
JSON FINAL

```
{  Alvaro Yuste Torregrosa, 4 years ago • Create package
  "name": "react-course-wishlist",
  "version": "1.0.0",
  "description": "An exercise for an introduction to React",
  "main": "index.js",
  ▶ Debug
  "scripts": {
    "start": "parcel ./src/index.html --open"
  },
  "keywords": [
    "parcel",
    "react"
  ],
  "author": "Alvaro Yuste (alvaroyuste.com)",
  "license": "MIT",
  "dependencies": {
    "react": "^16.8.6",
    "react-dom": "^16.8.6"
  },
  "devDependencies": {
    "parcel-bundler": "^1.12.3"
  }
}
```

EMPAQUETADO DE LA APP



Bundling



Developer friendly



Builder



Browser friendly

INSTALAMOS EL BUILDER O EMPAQUETADOR

ELEGIMOS PARCEL.

DE MOMENTO NO USAREMOS YARN, SOLO NPM

Versión con yarn: `yarn add --dev parcel-bundler`

1. Install Parcel in the project

```
npm install --save-dev parcel-bundler
```

Podemos comprobar las dependencias instaladas en nuestro proyecto con

```
npm list
```

```
├─ parcel-bundler@1.12.5
├─ react-dom@18.2.0
└─ react@18.2.0
```

```
"author": "vanessa espin",
"license": "MIT",
"dependencies": {
  "react": "^18.2.0",
  "react-dom": "^18.2.0"
},
"devDependencies": {
  "parcel-bundler": "^1.12.5"
}
```

Mirando carpeta node-modules

```
> parcel-bundler
```

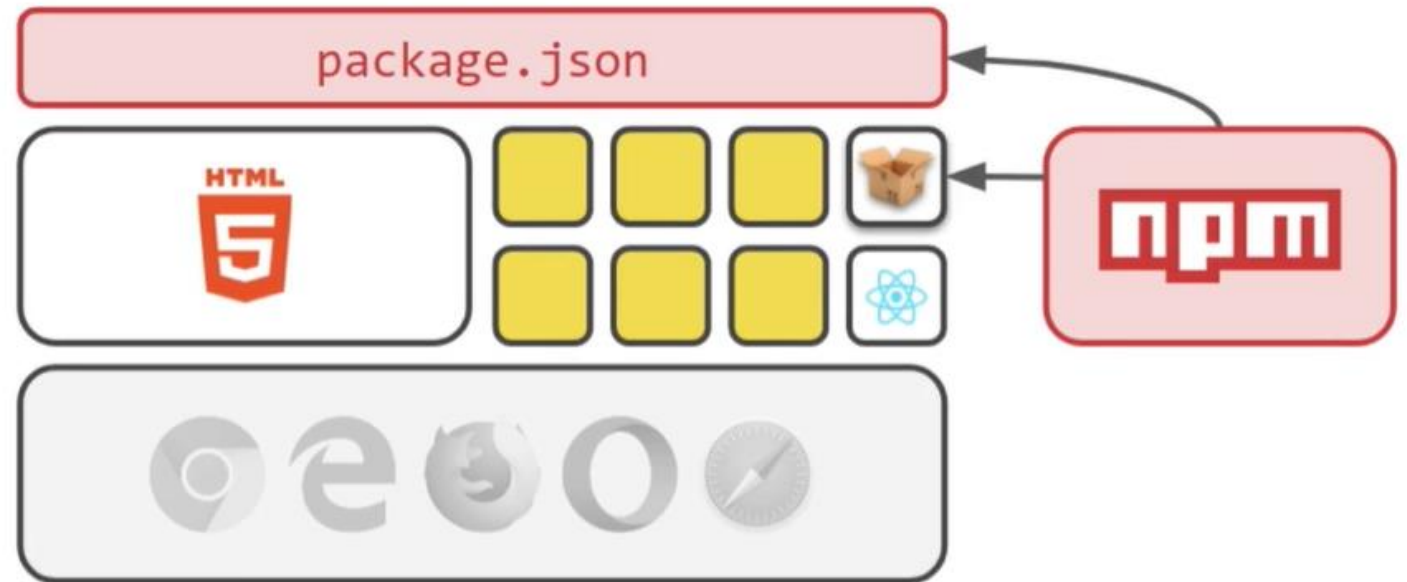
SI NO ESTÁ CREADO EL INDEX.HTML..

- Lo creamos dentro de src

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Wishlist</title>
  </head>
  <body>
    My wishlist
  </body>
</html>
```

HERRAMIENTAS DE EMPAQUETADO

- Ya tenemos esto..



START POINT

- Dentro de “scripts” definimos nuestro script de punto de acceso en “**start**” apuntando a nuestro index.html
- Arrancamos el proyecto con npm **start** o yarn **start**

package.json

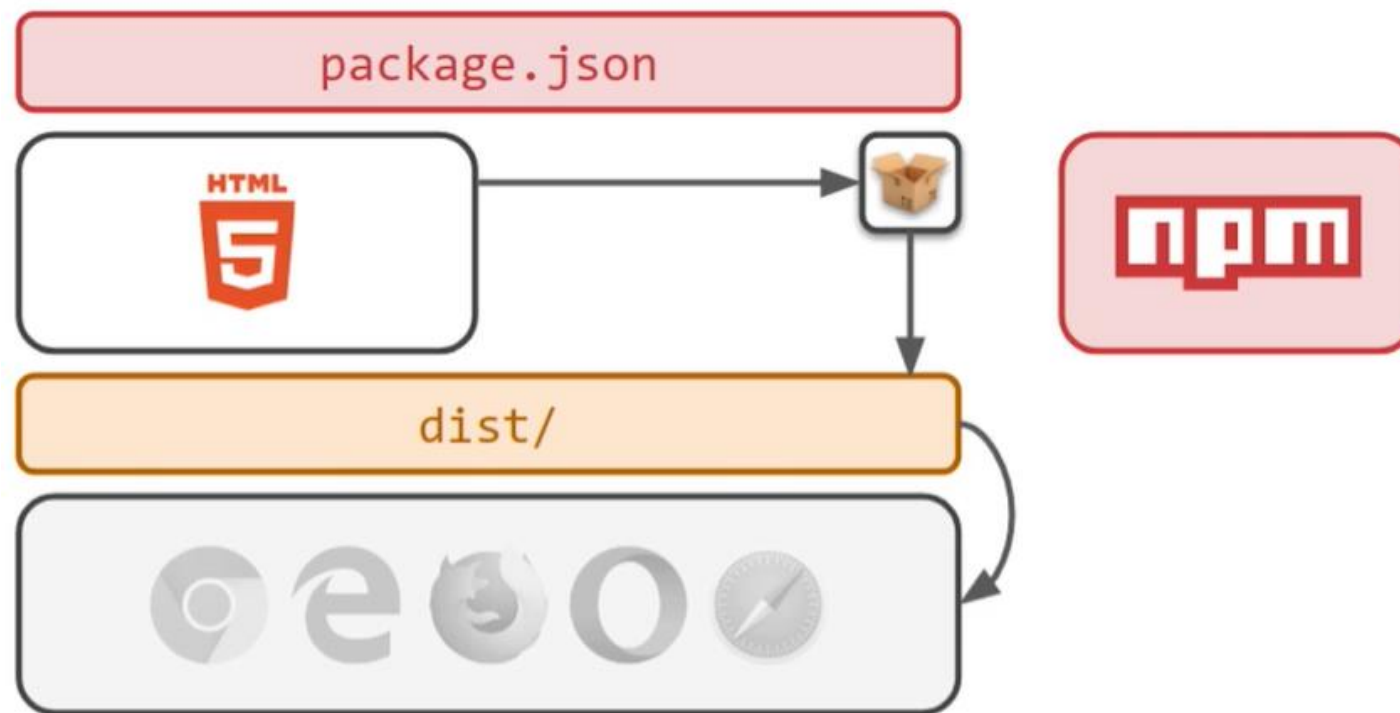
```
{  
  "name": "whishlist-react",  
  "version": "1.0.0",  
  "description": "ejemplo de app con react",  
  "main": "index.js",  
   Debug  
  "scripts": {  
    "start": "parcel src/index.html --open"  
  },  
}
```

```
PS V:\1-CURSOS\2023-REACT openwebinars\react-course-wishlist> npm start  
Debugger attached.  
  
> react-course-wishlist@1.0.0 start  
> parcel ./src/index.html --open  
  
Debugger attached.  
Server running at http://localhost:1234  
√ Built in 154ms.  
Debugger attached.  
Debugger attached.
```

Se abre index.html
en el navegador

HEMOS HECHO NUESTRO PRIMER EMPAQUETADO

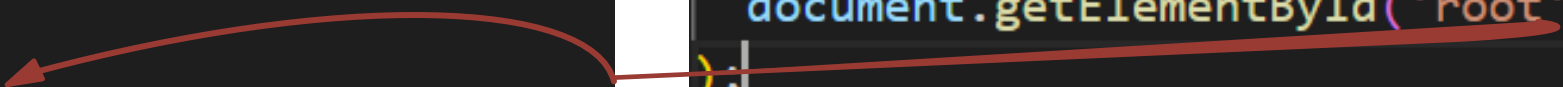
- Se ha creado nueva carpeta dist que es la que usa el empaquetador para mostrar al navegador



```
▼ dist
  <> index.html
```


index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Wishlist</title>
  </head>
  <body>
    <div id="root"> </div>
    <script src="index.jsx"> </script>
  </body>
</html>
```



index.jsx

```
import React from 'react';
import ReactDOM from 'react-dom/client';

const root = ReactDOM.createRoot(
  document.getElementById('root')
);
root.render(
  <div> Mi wishlist de REACT</div>,
);
```

EJEMPLO DE EMPAQUETADO DE CÓDIGO REACT

- Creamos el archivo index.jsx (javascript para React) y lo enlazamos desde nuestro index.html haciendo los cambios que se indican
- Si *volvemos a* la página web tenemos lo mismo (el proyecto aun se está ejecutando con *parcel*).
- Probar a modificar el div del render y ver comportamiento

ESTILO DEL CÓDIGO

- Estructura de carpetas eficaz
- Linting: herramientas que revisan nuestro código y comprueba que no tiene problemas
- Formato del código, puede automatizarse



Code style



Folders Structure



Linting



Format

project-folder/

src/

dist/

package.json

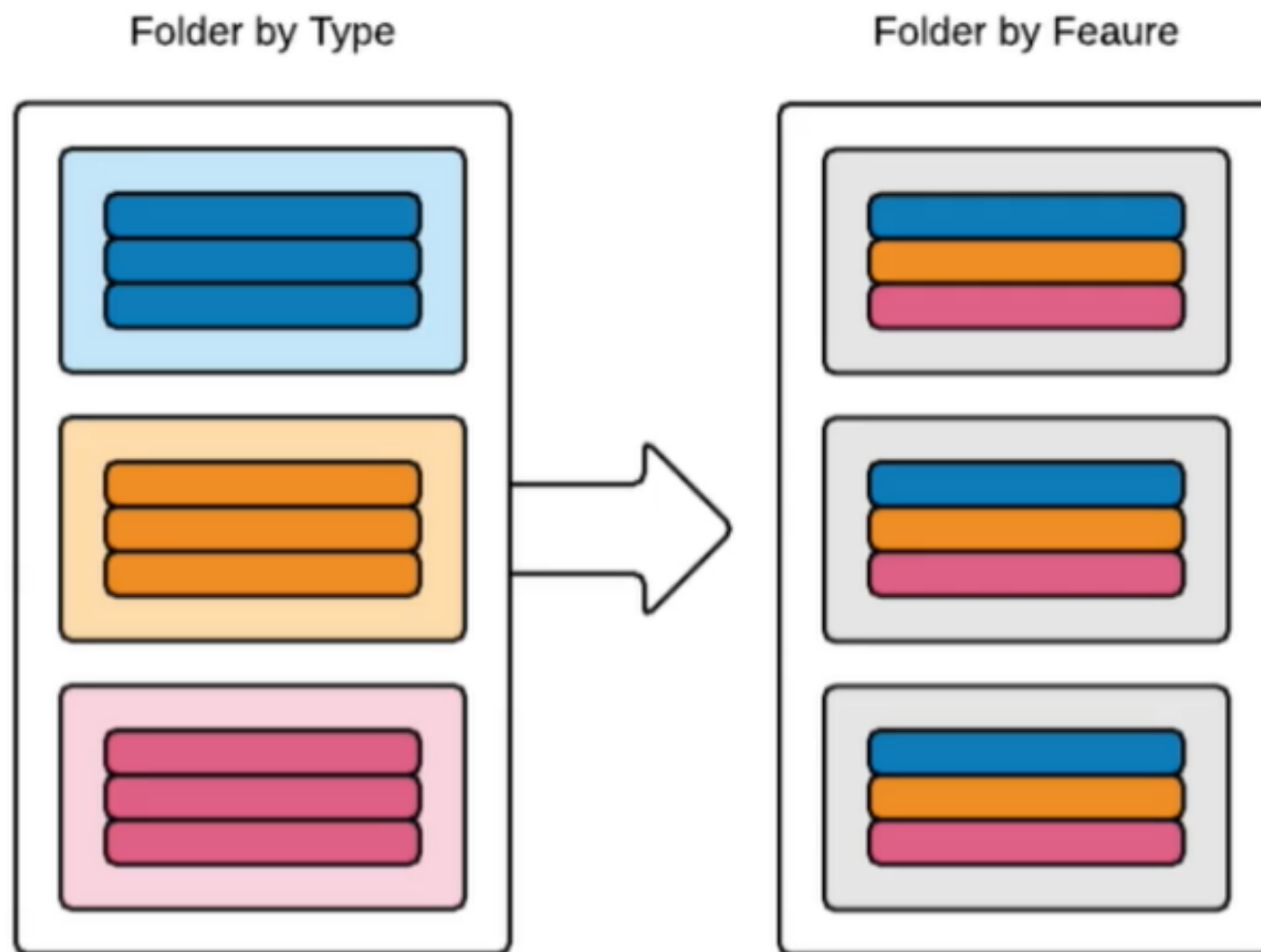
.all-config-files

ESTRUCTURA DE CARPETAS TIPICA

- package.json es la entrada a nuestro proyecto
- src tendrá nuestro código
- dist el código del empaquetado (lo que ve el navegador. No entramos

NUESTRO SRC. OPCIONES DE ESTRUCTURA

- Por tipo:
agrupamos según
el tipo de fichero
(lógica, estructura,
estilo)
- Por
funcionalidades (o
componentes)



Folder-by-Type

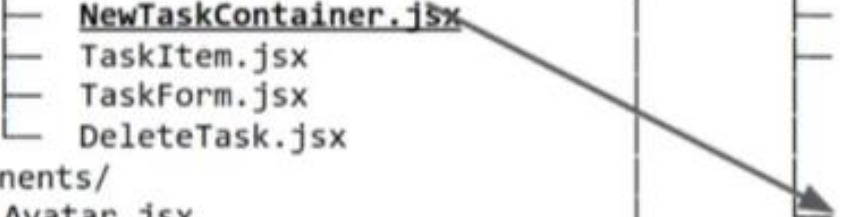
```
src/
├── index.jsx
├── App.jsx
├── components/
│   ├── TaskItem.jsx
│   ├── TaskForm.jsx
│   ├── DeleteTask.jsx
│   ├── ProfileForm.jsx
│   ├── DeleteProfile.jsx
│   ├── FriendItem.jsx
│   └── Avatar.jsx
├── pages/
│   ├── TasksPage.jsx
│   └── ProfilePage.jsx
├── containers/
│   ├── TaskListContainer.jsx
│   ├── NewTaskContainer.jsx
│   ├── FriendsContainer.jsx
│   └── ProfileContainer.jsx
└── services/
    ├── users.js
    └── tasks.js
```

Folder-by-Feature

```
src/
├── index.jsx
├── App/
│   ├── index.jsx
│   ├── ProfilePage/
│   │   ├── index.jsx
│   │   ├── FriendsContainer.jsx
│   │   ├── ProfileContainer.jsx
│   │   ├── ProfileForm.jsx
│   │   ├── DeleteProfile.jsx
│   │   └── FriendItem.jsx
│   └── TasksPage/
│       ├── index.jsx
│       └── TaskListContainer.jsx
│           ├── NewTaskContainer.jsx
│           ├── TaskItem.jsx
│           ├── TaskForm.jsx
│           └── DeleteTask.jsx
├── components/
│   └── Avatar.jsx
└── services/
    ├── users.js
    └── tasks.js
```

Extreme Fold.-by-Feat.

```
src/
├── index.jsx
├── App/
│   ├── index.jsx
│   ├── ProfilePage/
│   │   ├── index.jsx
│   │   ├── FriendsContainer/
│   │   │   ├── index.jsx
│   │   │   └── FriendItem.jsx
│   │   └── ProfileContainer/
│   │       ├── index.jsx
│   │       └── ProfileForm.jsx
│   └── DeleteProfile.jsx
│       └── TasksPage/
│           ├── index.jsx
│           ├── TaskListContainer/
│           │   ├── index.jsx
│           │   ├── DeleteTask.jsx
│           │   ├── TaskItem.jsx
│           │   └── NewTaskContainer/
│           │       ├── index.jsx
│           │       └── TaskForm.jsx
├── components/...
└── services/...
```



CUAL ELEGIMOS?

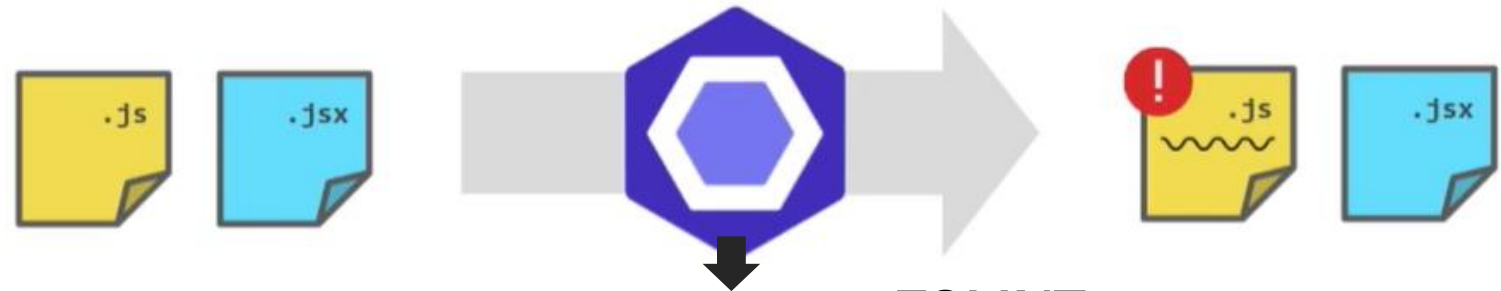
DEPENDE DE LA COMPLEJIDAD

 Code style - Folders structure



LINTING

REVISA EL CÓDIGO Y NOS AVISA
DE POSIBLES ERRORES



Ya que estamos en JavaScript usaremos **ESLint** <https://eslint.org/>

1. Install ESLint globally

```
npm install -g eslint
```

No hace falta que sea global,
podemos quitar el `-g` y poner
`--save-dev`

2. Init ESLint in the project

```
eslint --init
```

```
npm init @eslint/config
```

3. Add a lint script in package.json

```
{  
  "scripts": {  
    ...  
    "lint": "eslint --fix src/**/*.js,src/**/*.jsx"  
  },  
}
```

ESLINT – EXTENSIÓN AIRBNB

4. Adjust `.eslint.json` with this optional recommendation

Can be placed as a `eslintConfig` field in the `package.json`

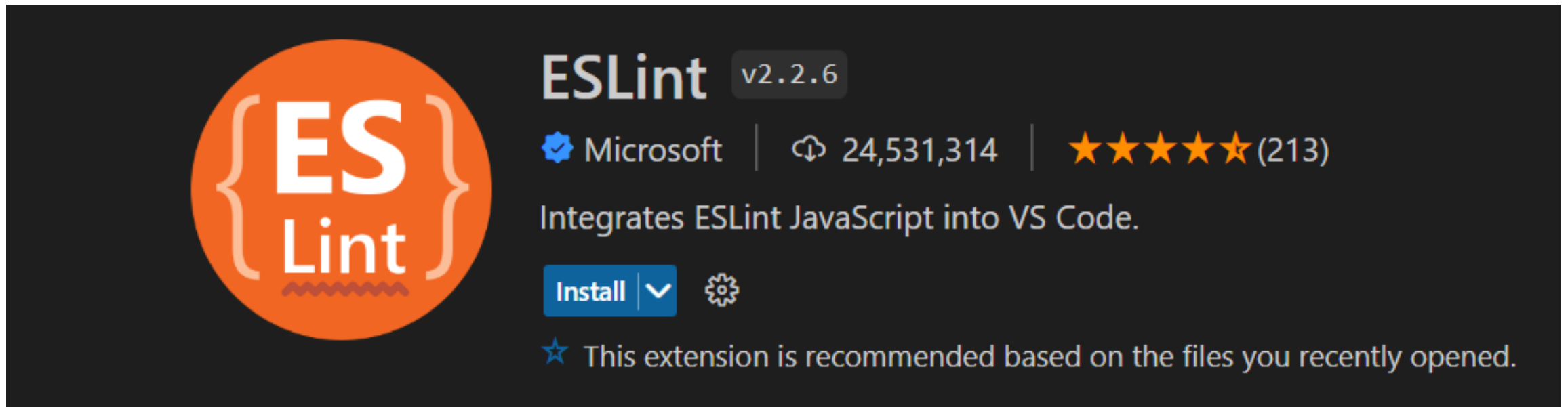
```
npm install --save-dev eslint-config-airbnb
```

5. Run linter

```
{  
  "extends": "airbnb",  
  "env": {  
    "browser": true  
  },  
  "rules": {  
    "linebreak-style": "off"  
  }  
}
```

```
npm run lint
```


TAMBIÉN PODEMOS INSTALAR LA EXTENSIÓN



The screenshot displays the ESLint extension page in the Visual Studio Code marketplace. On the left is the ESLint logo, which consists of an orange circle containing the text 'ES' in large white letters and 'Lint' in smaller white letters below it, with a wavy line underneath. To the right of the logo, the text 'ESLint' is shown in a large font, followed by a version tag 'v2.2.6'. Below this, there is a blue checkmark icon followed by the publisher name 'Microsoft', a download icon followed by the number '24,531,314', and five yellow stars followed by '(213)'. A description below reads 'Integrates ESLint JavaScript into VS Code.' There is a blue 'Install' button with a dropdown arrow and a gear icon for settings. At the bottom, a blue star icon is followed by the text 'This extension is recommended based on the files you recently opened.'

ESLint v2.2.6

Microsoft | 24,531,314 | ★★★★★ (213)

Integrates ESLint JavaScript into VS Code.

[Install](#) ⌵ ⚙️

★ This extension is recommended based on the files you recently opened.

ETAPAS DEL LINTING

- En etapa de desarrollo (en el editor)
- Etapa de integración: antes de hacer el commit (Husky)
- En etapa de despliegue



Editor
(e.g. VSCode)



Commit
(e.g. Husky)



Pipeline
(e.g. Jenkins)

Instalamos las dependencias
lint-staged y husky

LINTING EN ETAPA DE INTEGRACIÓN

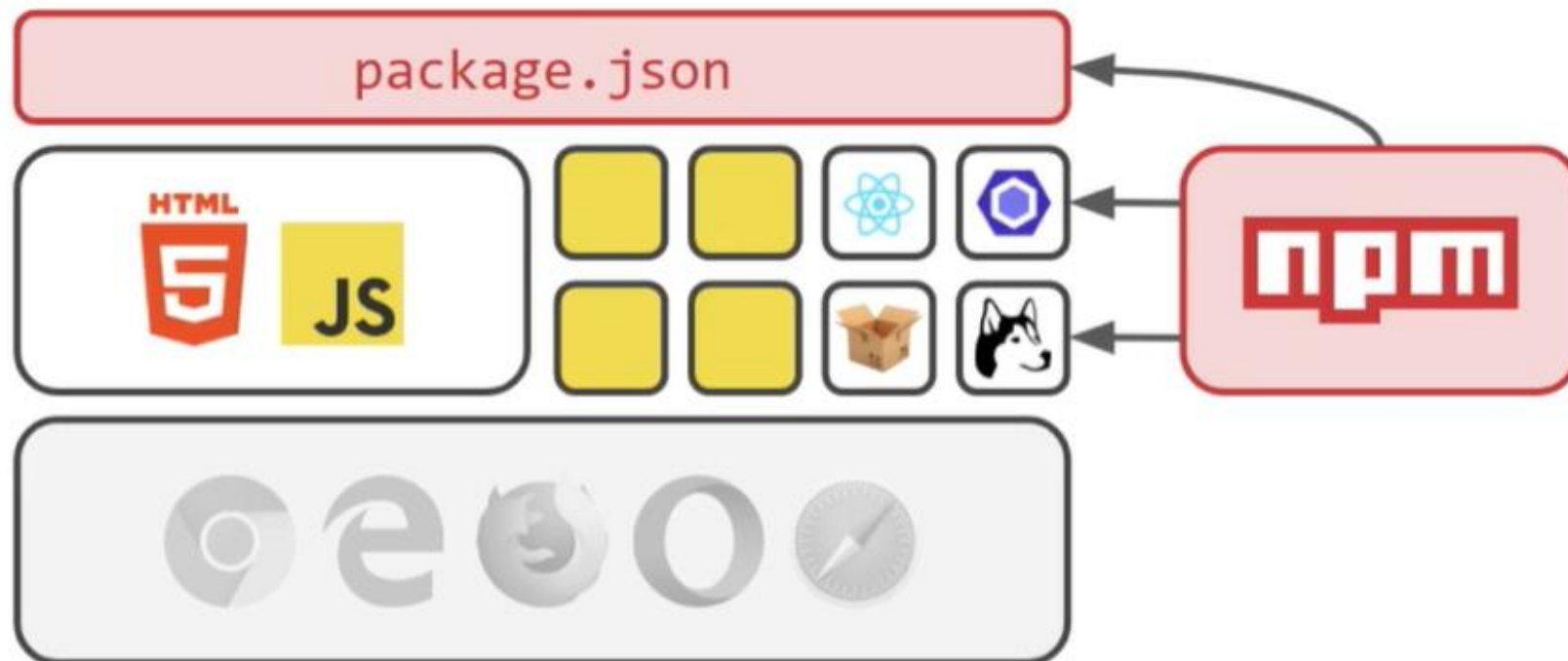
1. Install `lint-staged` and `husky`

```
npm install --dev lint-staged husky
```

2. Configure the pre-commit hook in `package.json`

```
{  
  ...  
  "husky": {  
    "hooks": {  
      "pre-commit": "lint-staged"  
    }  
  },  
  "lint-staged": {  
    "src/**/*.js,jsx": [ "npm run lint" ]  
  }  
}
```

YA TENEMOS ESTO..



FORMATO DEL CÓDIGO

- Vamos a ver las herramientas que nos garantizan la uniformidad del formato de nuestro código.
- Se ocupan de saltos de página, sangrado, tabulación, etc



Prettier más enfocado al ámbito general
Editor config se comunica con nuestro editor para automatizar estos estilizados

INSTALAMOS PRETTIER

1. Install Prettier

```
npm install --dev prettier
```

2. Add the script and the preferred rules in package.json

```
{  
  ...  
  "scripts": {  
    ...  
    "format": "prettier --write \"*.{js,jsx,json,css}\"",  
  },  
  "prettier": {  
    "trailingComma": "all",  
    "singleQuote": true  
  },  
}
```

Si añadimos, por ejemplo:
"semi": true, indicamos que
se escriban puntos y coma
al final de las líneas y ya me
olvido de hacerlo yo

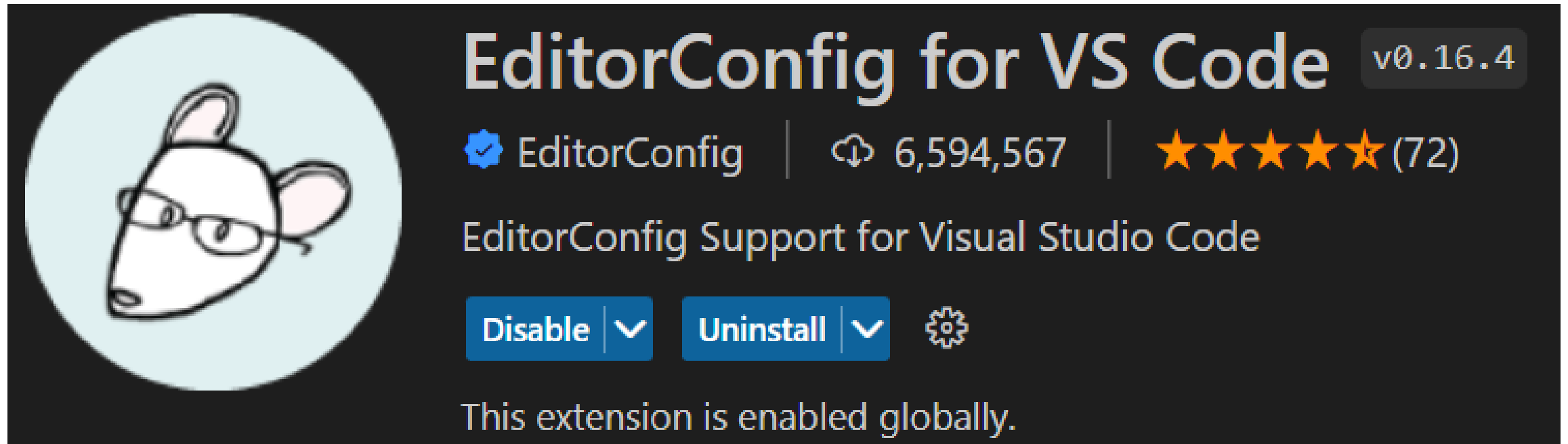
LINCAMOS PRETTIER CON NUESTRA CONFIGURACIÓN DE HUSKY MEDIANTE LINT-STAGED

3. Add the prettier script to lint staged fired by husky

```
{
  ...
  "lint-staged": {
    "src/**/*.js,jsx": [
      "npm run lint"
    ],
    "src/**/*.js,jsx,json,css,scss": [
      "prettier --write",
      "git add"
    ]
  }
}
```

Así lanzo el prettier con la opción de - -write para que lo añada también al commit

EDITOR CONFIG LO PUEDO INSTALAR COMO EXTENSIÓN



The image shows the Visual Studio Code extension interface for 'EditorConfig'. On the left is a circular profile picture of a white mouse wearing glasses. To the right, the extension name 'EditorConfig for VS Code' is displayed in a large, light blue font, with the version 'v0.16.4' in a smaller font to its right. Below the name, there is a blue checkmark icon followed by 'EditorConfig', a cloud icon with the download count '6,594,567', and five yellow stars with the rating '(72)'. The description 'EditorConfig Support for Visual Studio Code' is written in a smaller font. Below this, there are two blue buttons: 'Disable' and 'Uninstall', each with a dropdown arrow. To the right of these buttons is a gear icon. At the bottom, a status message reads 'This extension is enabled globally.'

EditorConfig for VS Code v0.16.4

✓ EditorConfig | 6,594,567 | ★★★★★ (72)

EditorConfig Support for Visual Studio Code

Disable | Uninstall | ⚙️

This extension is enabled globally.

REACT DEV TOOLS

- Nos centramos en Chrome
- Chrome web store:



React Developer Tools

Offered by: Facebook

★★★★★ 1,095

Developer Tools

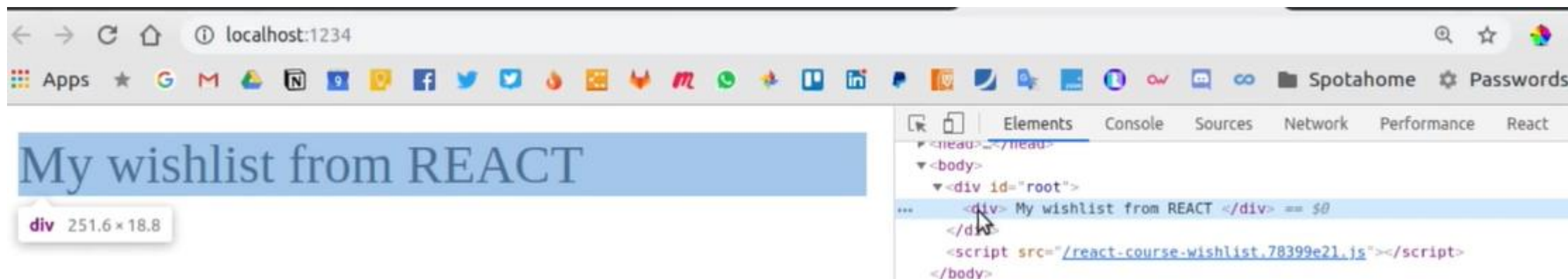
👤 1,489,422 users



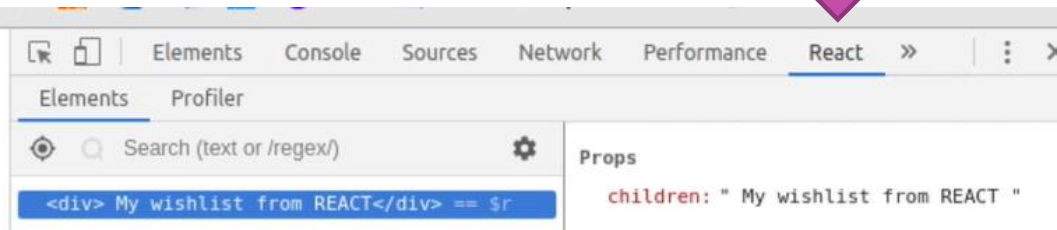
Extensions



YA PODEMOS VER EL DOM



My wishlist from REACT





FIN DEL TEMA 2

A CONTINUACIÓN: TEMA 3 RENDERIZADO