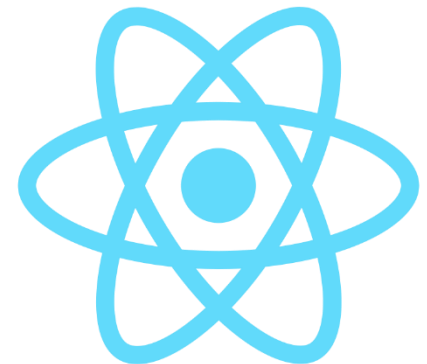


React

Riccardo Cattaneo



Introduzione

React è una libreria javascript, sviluppata e mantenuta da facebook, per creare interfacce grafiche. Lo scopo principale di Rect è quello di creare dei «componenti».

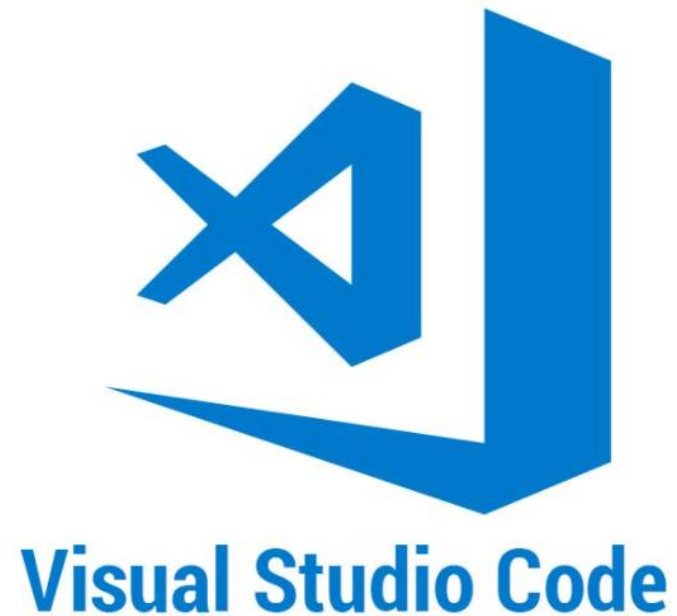
Ma che cos'è un componente ? Un componente è un elemento di interfaccia che può essere riutilizzato all'interno del nostro codice.

Prima di iniziare facciamo un veloce riepilogo sugli argomenti che dobbiamo conoscere :

- HTML
- CSS
- JAVASCRIPT (ES6)
- BOOTSTRAP

Ambiente di Sviluppo

- Node.js
- Chrome
- Visual Studio Code



Node.js

Node.js è una runtime di JavaScript Open source multiplatforma orientato agli eventi per l'esecuzione di codice JavaScript, costruita sul motore JavaScript V8 di Google Chrome.

In origine JavaScript veniva utilizzato principalmente lato client. In questo scenario gli script JavaScript, generalmente incorporati all'interno dell'HTML di una pagina web, vengono interpretati da un motore di esecuzione incorporato direttamente all'interno di un Browser. Node.js consente invece di utilizzare JavaScript anche per scrivere codice da eseguire lato server, ad esempio per la produzione del contenuto delle pagine web dinamiche prima che la pagina venga inviata al Browser dell'utente.

Node.js come detto è un runtime di javascript, ovvero un programma che, grazie al «motore» V8 di Google Chrome, viene eseguito codice JavaScript lato server.

Quando parliamo di «motore» JavaScript parliamo di quel componente sviluppato inizialmente solo per il browser, il cui compito è quello di prendere il nostro codice sorgente JavaScript e trasformarlo in codice macchina :

```
function add() {  
  return 2 + 2;  
}
```



V8 Engine



011011101

Ogni browser ha il suo «motore» JavaScript specifico. Ad esempio FireFox ha «SpiderMonkey», mentre chrome ha «V8». Inizialmente JavaScript è nato come linguaggio lato client e quindi eseguito dal «motore» residente all'interno del browser.

BROWSER



Gestione Eventi



Gestione DOM

...per rendere le pagine
web dinamiche

JavaScript

Accesso Filesystem



Accesso Database



Nel 2009 l'idea «rivoluzionaria» del progettista di Node.js Ryan Dahl è stata quella di prendere il motore V8 ed inserirlo all'interno di un nuovo ambiente in cui eseguire codice JavaScript. Questo ambiente prende il nome di Node.js che non opera lato client ma opera lato server.



Abbiamo quindi 1 motore JavaScript V8 e due runtime diversi, uno lato client ed uno lato server. Entrambi questi runtime espandono le funzionalità di V8 in base alla propria esigenza.

Ad esempio l'oggetto window e l'oggetto document presente nel runtime browser non è presente nel runtime node.js in quanto non abbiamo lato server una finestra del browser da gestire, ma ha un altro oggetto necessario al server che vedremo più avanti, cioè l'oggetto **global**.

Installazione Node.js

Per installare localmente Nodejs bisogna andare sul sito ufficiale www.nodejs.org e scaricare il pacchetto per il nostro sistema operativo.

Node.js® è un runtime JavaScript costruito sul motore JavaScript V8 di Chrome.

Join us at OpenJS World, a free virtual event on June 2-3, 2021

Download per Windows (x64)

14.17.0 LTS

Consigliata

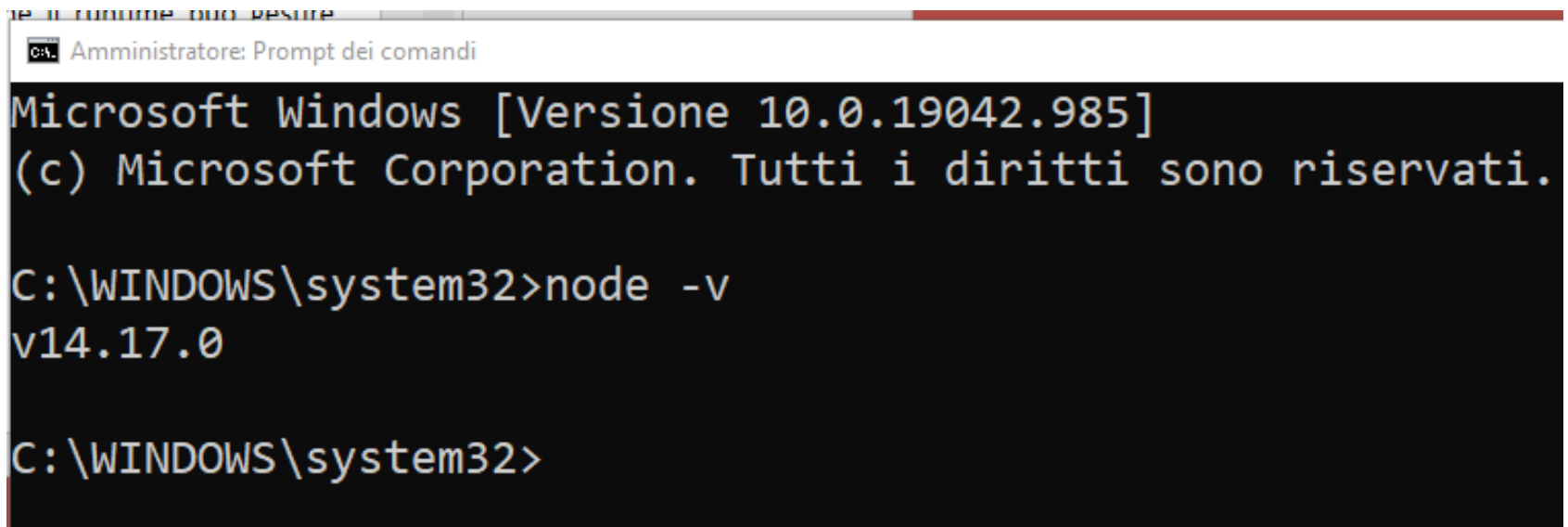
16.2.0 Corrente

Ultime funzionalità

[Altri download](#) | [Changelog](#) | [Documentazione API](#)

[Altri download](#) | [Changelog](#) | [Documentazione API](#)

Nodejs va utilizzato da riga di comando. Verifichiamo che l'installazione è andata a buon fine aprendo il terminale (dos) e digitare il comando `node -v`



```
Amministratore: Prompt dei comandi
Microsoft Windows [Versione 10.0.19042.985]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\WINDOWS\system32>node -v
v14.17.0

C:\WINDOWS\system32>
```

Il nostro primo script Node

Per prima cosa apriamo il nostro terminale e digitiamo semplicemente il comando `node`. In questo modo entriamo in una modalità interattiva per poter fare dei semplici test.

Proviamo a scrivere la seguente riga di codice :

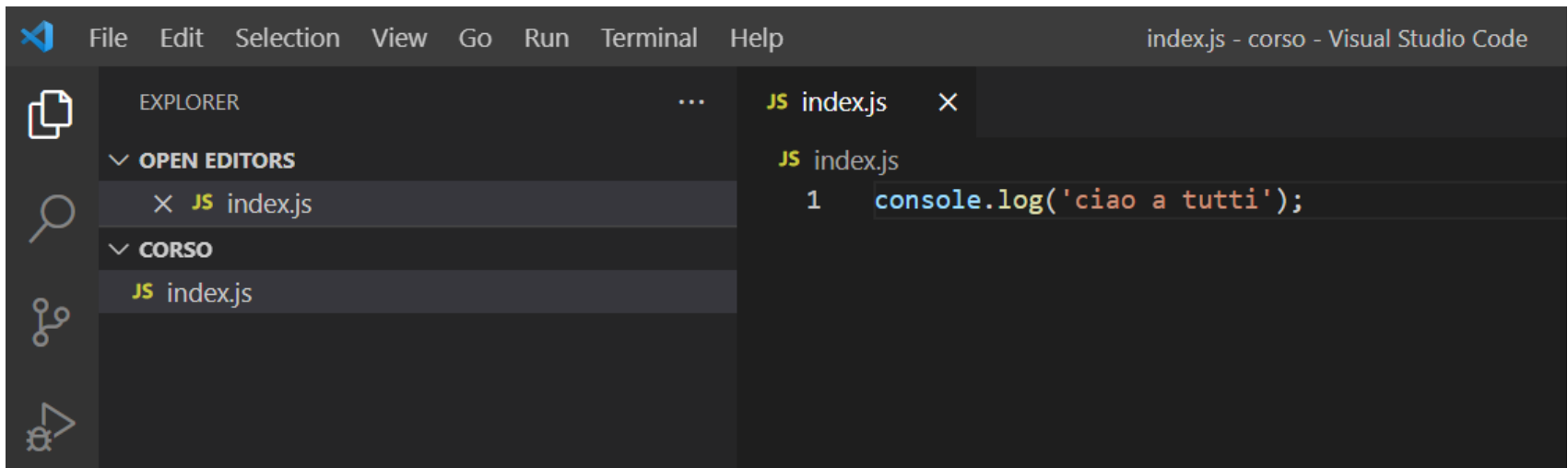
```
'node.js'.toUpperCase()
```

e poi diamo invio. Possiamo vedere il risultato a video. Per uscire dalla modalità node premiamo i tasti CTRL + D

```
C:\WINDOWS\system32>node
Welcome to Node.js v14.17.0.
Type ".help" for more information.
> 'node.js'.toUpperCase()
'NODE.JS'
>

C:\WINDOWS\system32>
```

Per fare un esempio meno banale andiamo a creare una cartella sul desktop e la chiamiamo «corso» e sempre tramite la riga di comando ci andremo a posizionare all'interno di essa e creeremo un file chiamato **index.js** (usando un editor di testo come ad esempio Visual Studio Code)



```
C:\Users\Rick\Desktop>cd corso
```

```
C:\Users\Rick\Desktop\corso>node index.js  
ciao a tutti
```

```
C:\Users\Rick\Desktop\corso>
```

Il registro NPM

NPM rappresenta un repository di librerie scritte apposta per poter essere utilizzate con Node.js.

Per installare un pacchetto basta un semplice:

npm install nome-pacchetto

Per vedere un esempio concreto consideriamo l'installazione del modulo **socket.io**. Questo modulo permette di gestire le connessioni via socket a basso livello. Il modulo ci servirà nel corso delle prossime lezioni, perciò non preoccupiamoci di capire in dettaglio a cosa serve: al momento lo useremo solo come esempio di procedura di installazione di un nuovo componente.

npm install socket.io

Comandi Terminale

- `mkdir`: creare una nuova cartella
- `rmdir` : eliminare una cartella
- `dir (ls mac)` : lista contenuto cartella
- `cd` : cambia cartella
- `cd..` : spostarsi indietro di una cartella
- `cls (clear mac)` : pulire la console
- freccia su e freccia giù per ripetere i comandi

Come creare una React App

Per creare la prima app è sufficiente digitare il comando **npx** che non è altro che un esecutore di pacchetti seguito dal pacchetto **create-react-app** e poi il nome dell'applicazione. Per fare un primissimo esempio creiamo sul desktop una cartella **ciaomondo** e tramite terminale digitiamo :

npx create-react-app ciaomondo

Success! Created ciao at C:\Users\Rick\Desktop\ciao
Inside that directory, you can run several commands:

`npm start`

Starts the development server.

`npm run build`

Bundles the app into static files for production.

`npm test`

Starts the test runner.

`npm run eject`

Removes this tool and copies build dependencies, configuration files and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

`cd ciao`

`npm start`

Happy hacking!

C:\Users\Rick\Desktop>

Appena lanciamo il comando partirà la configurazione della nostra nuova app, ci vuole qualche secondo. Una volta terminata l'installazione di tutte le librerie necessarie possiamo lanciare il nostro server locale digitando da riga di comando

npm start

```
Compiled successfully!
```

```
You can now view ciao in the browser.
```

```
Local:          http://localhost:3000
```

```
On Your Network: http://172.18.144.1:3000
```

```
Note that the development build is not optimized.  
To create a production build, use npm run build.
```

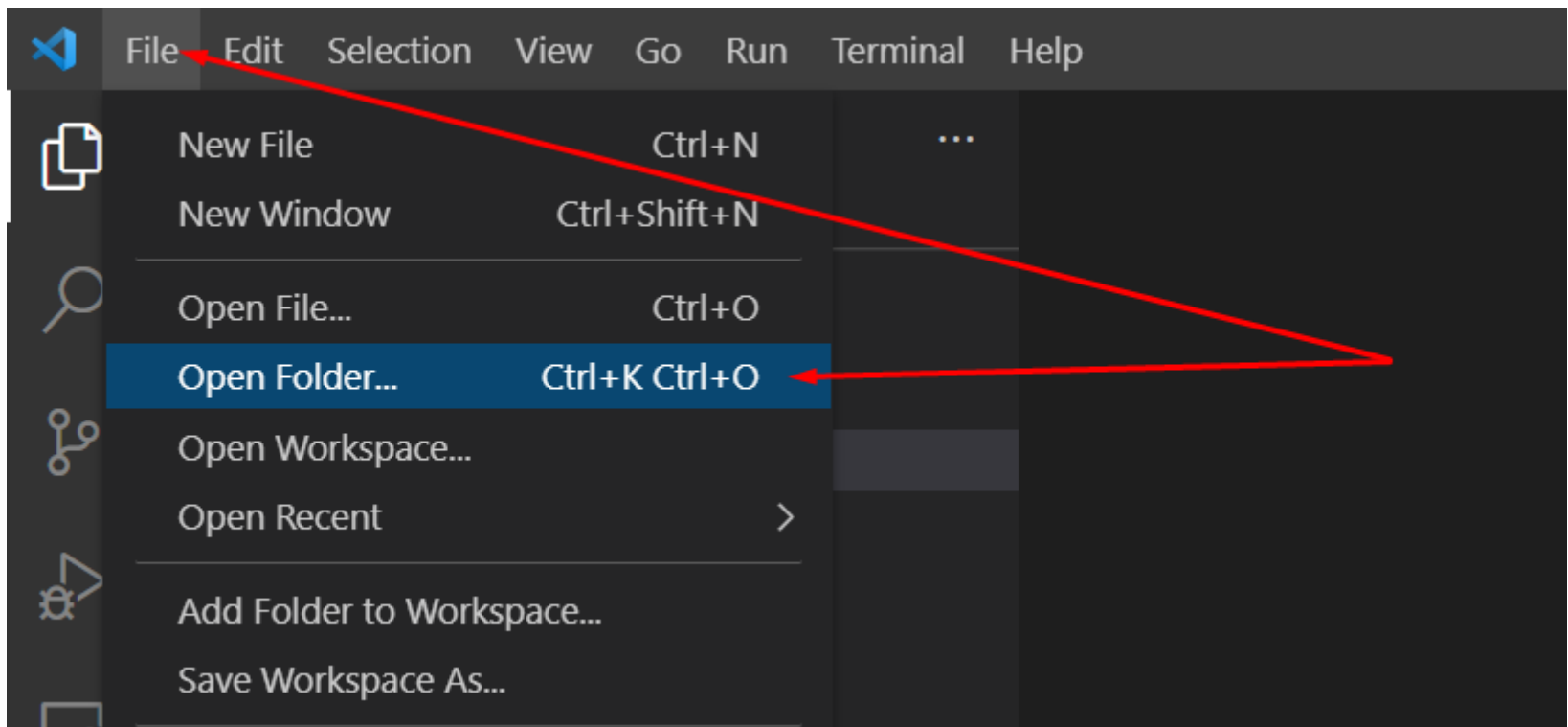
Automaticamente dovrebbe aprirsi il browser all'indirizzo `http://localhost:3000` che è la porta di default.

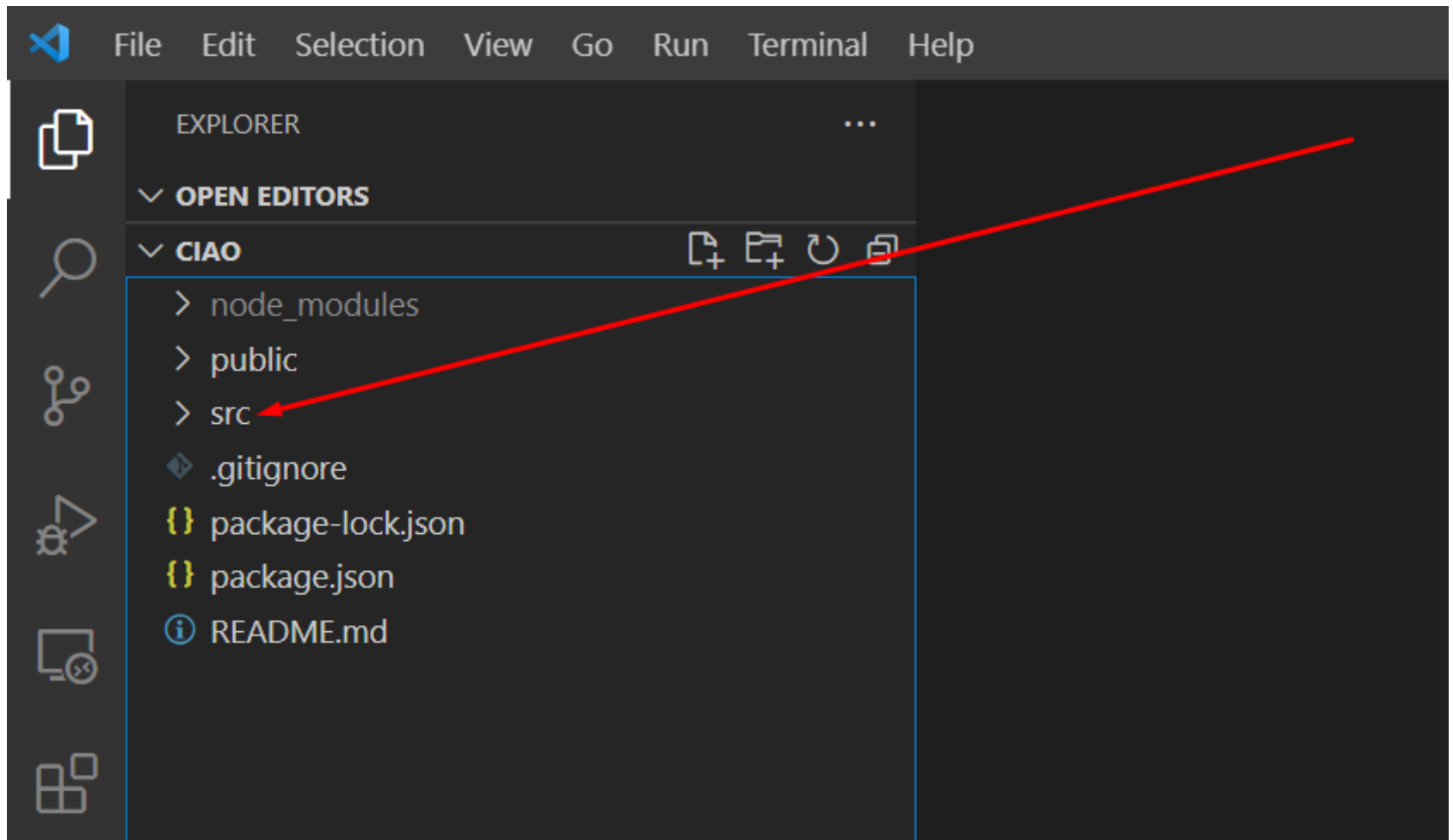


Edit `src/App.js` and save to reload.

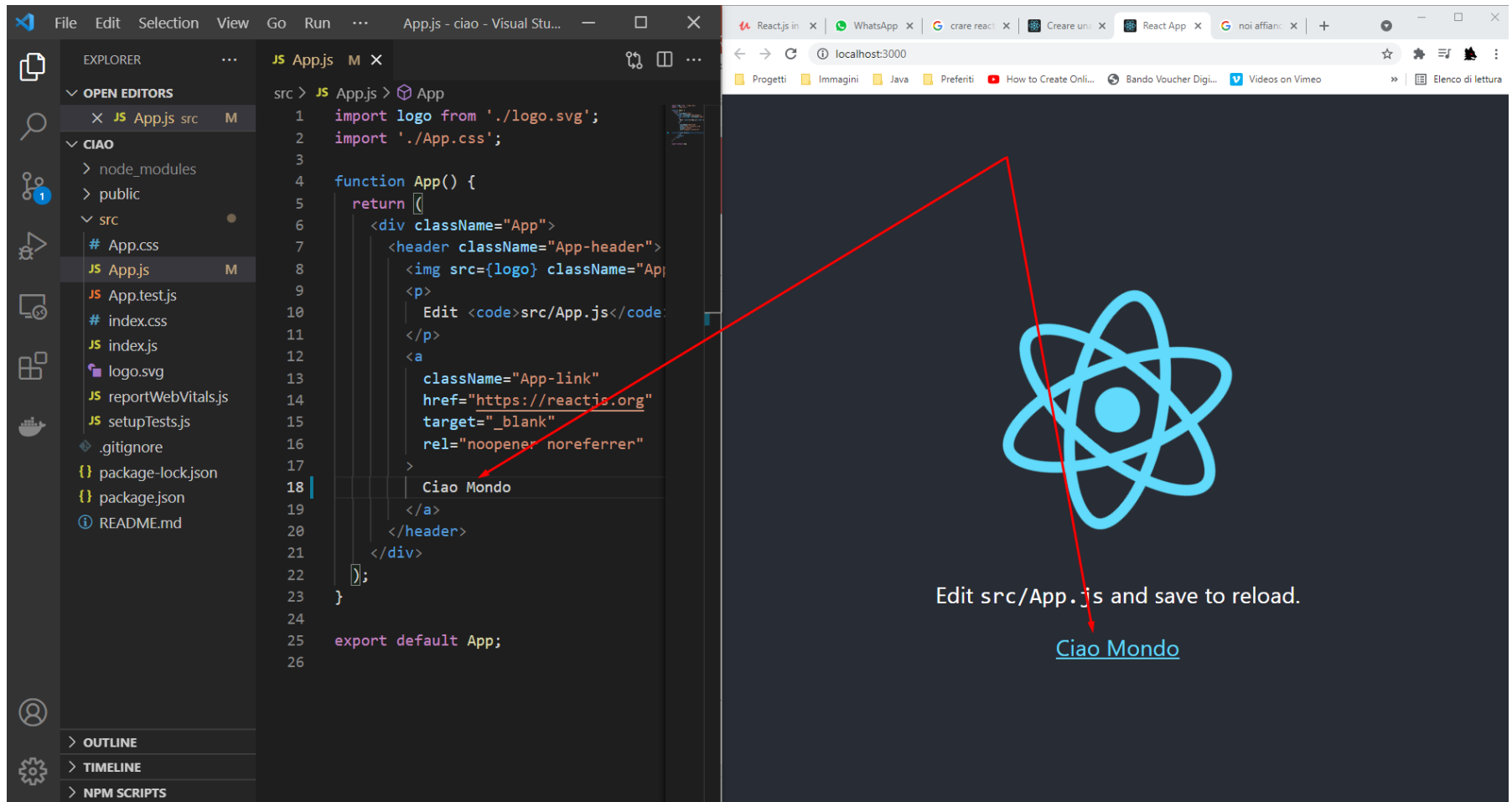
[Learn React](#)

Andiamo ora ad aprire la nostra applicazione attraverso Visual Studio Code :

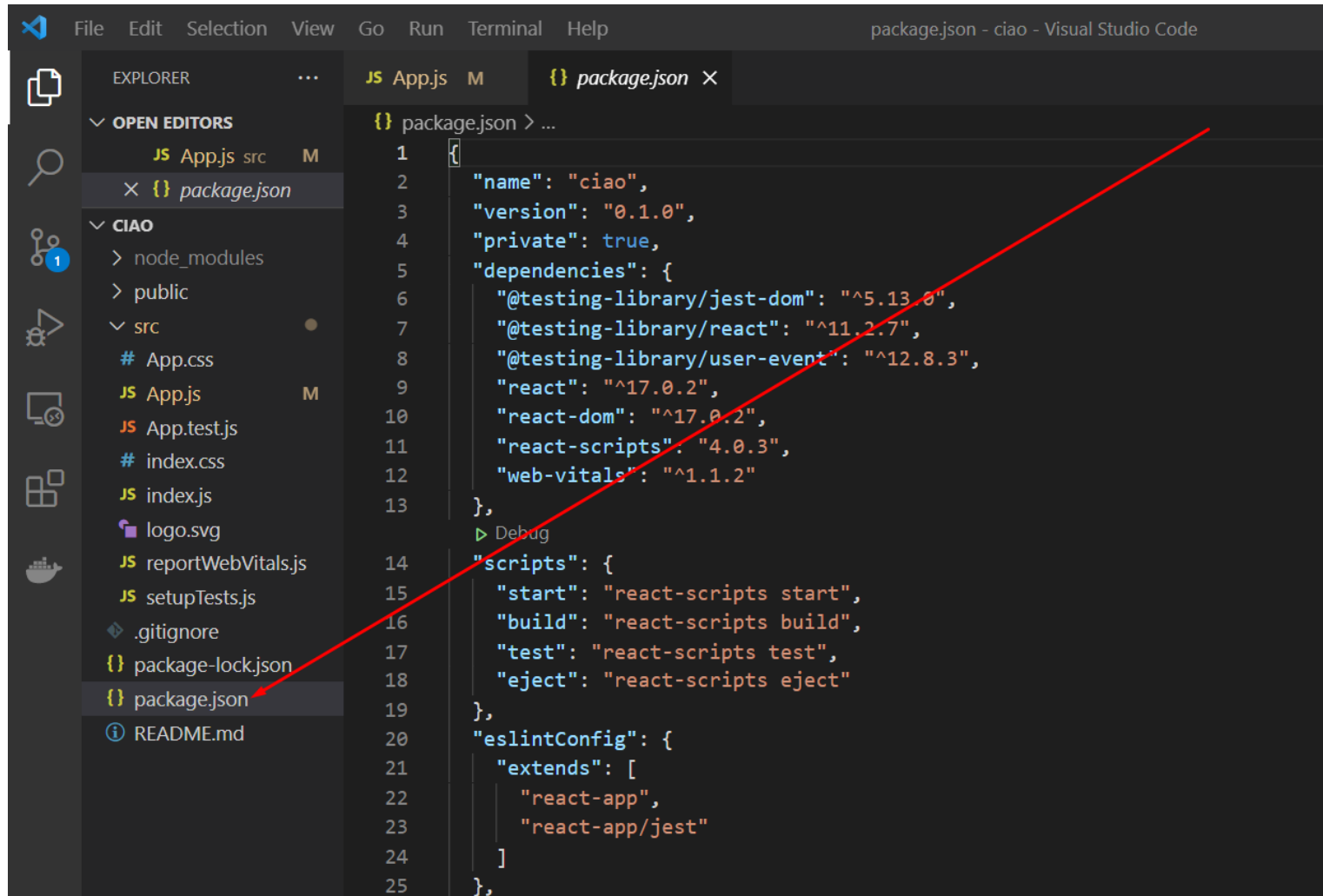




Per fare una prima prova apriamo il file App.js che troviamo all'interno della nostra cartella src e proviamo a modificare il testo all'interno del codice html, ad esempio sostituiamo la scritta «Learn React» con «Ciao Mondo», salviamo e vediamo cosa succede (per vederlo affianchiamo le due finestre browser e visual studio code) :



Package.json



The screenshot shows the Visual Studio Code interface with the Explorer on the left and the Editor on the right. The Explorer shows the project structure with the following files and folders:

- EXPLORED
- OPEN EDITORS
 - JS App.js src M
 - {} package.json
- CIAO
 - > node_modules
 - > public
 - > src
 - # App.css
 - JS App.js M
 - JS App.test.js
 - # index.css
 - JS index.js
 - logo.svg
 - JS reportWebVitals.js
 - JS setupTests.js
 - .gitignore
 - {} package-lock.json
 - {} package.json
 - README.md

The Editor shows the content of the package.json file:

```
1 {  
2   "name": "ciao",  
3   "version": "0.1.0",  
4   "private": true,  
5   "dependencies": {  
6     "@testing-library/jest-dom": "^5.13.0",  
7     "@testing-library/react": "^11.2.7",  
8     "@testing-library/user-event": "^12.8.3",  
9     "react": "^17.0.2",  
10    "react-dom": "^17.0.2",  
11    "react-scripts": "4.0.3",  
12    "web-vitals": "^1.1.2"  
13  },  
14  "scripts": {  
15    "start": "react-scripts start",  
16    "build": "react-scripts build",  
17    "test": "react-scripts test",  
18    "eject": "react-scripts eject"  
19  },  
20  "eslintConfig": {  
21    "extends": [  
22      "react-app",  
23      "react-app/jest"  
24    ]  
25  },  
26 }
```

Questo è il primo file del nostro progetto che dobbiamo conoscere (**package.json**) dove troviamo la nostra configurazione e tutte le librerie che sono state installate automaticamente quando abbiamo creato il progetto.

Fisicamente tutte le librerie scaricate possiamo vederle all'interno della cartella **node_modules**

index.html

All'interno della cartella `public` troviamo il file `index.html` che è il file che effettivamente viene caricato quando facciamo partire la nostra applicazione.

Tutto quello che scriveremo all'interno del nostro progetto, quindi nella cartella `src`, verrà visualizzato all'interno del `div` con `id root`.

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6     <meta name="viewport" content="width=device-width, initial-scale=1" />
7     <meta name="theme-color" content="#000000" />
8     <meta
9       name="description"
10      content="Web site created using create-react-app"
11    />
12    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
13    <!--
14      manifest.json provides metadata used when your web app is installed on a
15      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
16    -->
17    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
18    <!--
19      Notice the use of %PUBLIC_URL% in the tags above.
20      It will be replaced with the URL of the `public` folder during the build.
21      Only files inside the `public` folder can be referenced from the HTML.
22
23      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
24      work correctly both with client-side routing and a non-root public URL.
25      Learn how to configure a non-root public URL by running `npm run build`.
26    -->
27    <title>React App</title>
28  </head>
29  <body>
30    <noscript>You need to enable JavaScript to run this app.</noscript>
31    <div id="root"></div>
```

Esercizio

Creare un progetto react e creare una pagina di presentazione personalizzata sostituendo l'html presente di default nella pagina App.js