

# TS React Workspace

Antonio

Novembre 2023

## Sommario

Ciao team members! Ho pensato di creare questo documento per proporvi l'utilizzo di tecnologie di linting e formatting, ciò ci permetterà innanzitutto un controllo degli errori (che risulta inesistente in JavaScript) e delle linee guida stilistiche, oltre che un auto-formattazione per far sì che il codice sia consistente. Naturalmente prendete tutto ciò come una proposta, non ho nessuna autorità per imporvi di utilizzare tali tecnologie, anzi, l'obiettivo è semplificarci la vita; sono ben accetti suggerimenti. Nel caso si presentasse qualsivoglia problema contattatemi pure, cercherò di aiutarvi meglio che posso. Buona lettura e buona installazione!

## Indice

<b>1</b>	<b>Installazione di nvm ed npm</b>	<b>2</b>
<b>2</b>	<b>Creazione progetto di prova</b>	<b>2</b>
<b>3</b>	<b>Installazione e configurazione di Visual Studio Code</b>	<b>2</b>
<b>4</b>	<b>Configurazione del linter e del formatter</b>	<b>4</b>

# 1 Installazione di nvm ed npm

Questa sezione comprende la configurazione essenziale per l'implementazione di qualsiasi sito web basato su JavaScript. Iniziamo con l'installazione di npm (Node Package Manager) che ci permetterà di installare anche NodeJS e i vari pacchetti necessari per lavorare con React Native, TypeScript ed ESLint (il linter che andrò a proporvi).

Il primo passo consiste nell'installazione di nvm (Node Version Manager) che risulta necessario per la gestione delle installazioni di Node ed npm. Guide all'installazione:

- Windows: <https://github.com/coreybutler/nvm-windows>
- Linux/MacOS <https://github.com/nvm-sh/nvm>

Una volta installato con successo nvm potete procedere all'installazione di npm seguendo la documentazione ufficiale: <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

Arrivati a questo punto abbiamo gli strumenti necessari per tutte le attività, possiamo passare alla prossima sezione.

# 2 Creazione progetto di prova

In questa sezione andremo a creare un progetto di prova utilizzando un template per creare un progetto React Native che usa TypeScript, ciò ci permetterà di verificare che tutto stia procedendo nel modo giusto. Dopo aver creato la cartella necessaria al progetto aprite il vostro terminale. Tutto ciò che ci serve fare è creare il progetto: all'interno della vostra cartella digitate il seguente comando:

```
npx create-expo-app --template
```

Vi sarà chiesto di selezionare un template e inserire un nome, selezionate *Blank (TypeScript)* e il nome che preferite. Terminata questa operazione andiamo a scaricare dei pacchetti tramite npm, tali pacchetti sono necessari al corretto funzionamento dell'app:

```
npm install --save-dev eslint prettier eslint-plugin-react
eslint-plugin-import eslint-plugin-jsx-a11y @typescript-eslint/eslint-plugin
eslint-config-prettier eslint-plugin-react-hooks eslint-plugin-react-native
```

Abbiamo finalmente il progetto, non ci resta che impostare il linter e il formatter, ma prima di questo andremo a configurare Visual Studio Code, l'ambiente di lavoro proposto per tale scopo.

# 3 Installazione e configurazione di Visual Studio Code

Procediamo banalmente all'installazione dell'editor: <https://code.visualstudio.com/>

Fatta questa operazione possiamo passare all'installazione delle estensioni necessarie. Tramite il menù sulla sinistra trovate il pulsante *Extensions* che vi aprirà la schermata del Marketplace dove potete cercare e installare estensioni. Cercate e installate le seguenti estensioni:

- *esbenp.prettier-vscode* (l'integrazione del nostro formatter)
- *dbaeumer.vscode-eslint* (l'integrazione del nostro linter)
- *christian-kohler.npm-intellisense*
- *ritwickdey.LiveServer*
- *ecmel.vscode-html-css*
- *xabikos.JavaScriptSnippets*

- *formulahendry.auto-rename-tag*

Dopo aver finalmente installato tutte le estensioni, all'interno dell'editor digitate la combinazione di tasti Ctrl + Shift + P e cercate *Open User Settings (JSON)*, trovata l'impostazione premete invio. Questo file JSON non è altro che il file di configurazione dell'editor, da qui andremo a controllare delle impostazioni base, non dovrete far altro che incollare le varie impostazioni che vi indicherò.

Impostiamo la grandezza del font e andiamo a escludere file irrilevanti dal File Tree (il vostro menù di navigazione file):

```
{
  "editor.fontSize": 18,
  "files.exclude": {
    "**/__pycache__/*": true,
    "**/.ini*": true,
    "**/*.o": true,
    "**/.vscode/*": true
  },
}
```

Non ci resta che aggiungere le impostazioni relative al formatter:

```
{
  "[typescript]": {
    "editor.defaultFormatter": "esbenp.prettier-vscode",
    "editor.formatOnSave": true,
    "editor.tabSize": 2,
    "editor.indentSize": "tabSize"
  },
  "[typescriptreact]": {
    "editor.defaultFormatter": "esbenp.prettier-vscode",
    "editor.formatOnSave": true,
    "editor.tabSize": 2,
    "editor.indentSize": "tabSize"
  },
  "[javascript]": {
    "editor.defaultFormatter": "esbenp.prettier-vscode",
    "editor.formatOnSave": true,
    "editor.tabSize": 2,
    "editor.indentSize": "tabSize"
  },
  "[css]": {
    "editor.defaultFormatter": "esbenp.prettier-vscode",
    "editor.formatOnSave": true,
    "editor.tabSize": 2,
    "editor.indentSize": "tabSize"
  },
  "[jsonc]": {
    "editor.defaultFormatter": "esbenp.prettier-vscode",
    "editor.formatOnSave": true,
    "editor.tabSize": 2,
    "editor.indentSize": "tabSize"
  },
}
```

## 4 Configurazione del linter e del formatter

Se siete arrivati fin qui complimenti! Ci resta un'ultima cosa da fare, configurare Prettier ed ESLint, queste operazioni sono molto semplici e si riducono a impostare degli attributi JSON, similmente a come fatto con la Sezione 3.

Partiamo con Prettier: nella root del progetto create un file denominato *.prettierrc*. Creato il file apritelo e inserite le seguenti righe:

```
{
  "singleQuote": true,
  "printWidth": 80
}
```

Concludiamo con ESLint: nella root del progetto create un file denominato *.eslintrc.json*. Creato il file (se il file è già presente andate a sovrascriverne il contenuto) apritelo e inserite le seguenti righe:

```
{
  "env": {
    "browser": true,
    "es2021": true,
    "node": true
  },
  "extends": [
    "plugin:react/recommended",
    "plugin:react-hooks/recommended",
    "eslint:recommended",
    "plugin:@typescript-eslint/recommended"
  ],
  "overrides": [],
  "parser": "@typescript-eslint/parser",
  "parserOptions": {
    "ecmaVersion": "latest",
    "sourceType": "module",
    "ecmaFeatures": {
      "jsx": true
    }
  },
  "plugins": ["react", "react-native", "react-hooks", "@typescript-eslint"],
  "rules": {
    "indent": ["error", 2],
    "linebreak-style": ["error", "unix"],
    "quotes": ["error", "single"],
    "semi": ["error", "always"],
    "max-len": ["warn", { "code": 80 }],
    "@typescript-eslint/explicit-function-return-type": [
      "error",
      {
        "allowExpressions": true
      }
    ],
    "react-hooks/rules-of-hooks": "error",
    "react-hooks/exhaustive-deps": "warn"
  },
  "settings": {
    "react": {
```

```

    "createClass": "createReactClass",
    "pragma": "React",
    "fragment": "Fragment",
    "version": "detect",
    "flowVersion": "0.53"
  },
  "propWrapperFunctions": [
    "forbidExtraProps",
    { "property": "freeze", "object": "Object" },
    { "property": "myFavoriteWrapper" },
    { "property": "forbidExtraProps", "exact": true }
  ],
  "componentWrapperFunctions": [
    "observer",
    { "property": "styled" },
    { "property": "observer", "object": "Mobx" },
    { "property": "observer", "object": "<pragma>" }
  ],
  "formComponents": [
    "CustomForm",
    { "name": "Form", "formAttribute": "endpoint" }
  ],
  "linkComponents": [
    "Hyperlink",
    { "name": "Link", "linkAttribute": "to" }
  ]
}
}

```

Tale operazione dovrebbe generare degli errori all'interno del file */App.tsx*, questo è un buon segno! Ciò significa che il linter sta funzionando correttamente e dovrebbe segnalarci un errore relativo ad un "missed return type", per risolvere modificate la riga dell'*export default function App()* aggiungendo il tipo:

```
export default function App(): React.JSX.Element {
```

Provate anche a cambiare l'indentazione ed effettuare un salvataggio del file: se il formatter sta funzionando correttamente l'indentazione sarà automaticamente aggiustata. Installiamo degli ultimi pacchetti necessari ad avviare l'app sul web:

```
npx expo install react-native-web@~0.19.6 react-dom@18.2.0 @expo/webpack-config@~19.0.0
```

E lanciamo l'applicazione!

```
npm run web
```

Il risultato sarà una pagina bianca con scritto *Open up App.tsx to start working on your app!*.

Fatto ciò abbiamo terminato l'intera configurazione (che belloooo). Spero che tutto ciò vi sarà d'aiuto e che potremmo beneficiarne tutti noi, grazie della lettura e buona giornata ^^