Umberto Emanuele

Introduzione React

Novembre 2023

Cos’è REACT?

Una **libreria** JavaScript **OpenSource**, sviluppata da **Facebook**, per creare interfacce utente (prima release 2013).

https://it.reactjs.org/

React rende la creazione di UI interattive facile e indolore. Progetta interfacce per ogni stato della tua applicazione.

Ad ogni cambio di stato React aggiornerà efficientemente solamente le parti della UI che dipendono da tali dati.

Tra i grandi nomi che utilizzano **React** in produzione ci sono **Facebook**, **Instagram, WhatsApp**, **Netflix**, **Airbnb**, **eBay**, **PayPal**, **New York Time** e molte altre.

- **React** ha una sintassi simile ad **HTML.**
- I componenti **React** sono scritti in **Javascript**.
- **React** ha un approccio **Dichiarativo** e non Imperativo, specifichiamo cosa vogliamo che venga fatto e non come.
- Architettura basata su **Componenti.**
- **Modulare**, i componenti sono riutilizzabili
- **Componenti** complessi possono essere scomposti in componenti più piccoli, semplici e riutilizzabili.
- **V** in **MVC**

- React utilizza il **Virtual DOM** invece di operare direttamente sul **DOM** reale.
- Il **Virtual DOM** è un'astrazione del DOM. Si tratta di una rappresentazione in memoria del DOM.
- È veloce e indipendente dalle specifiche implementazioni del browser.
- Possiamo pensare al Virtual DOM come una copia in memoria del DOM reale.

- React effettua le modifiche sul **Virtual DOM** e lo aggiorna per rispecchiare i cambiamenti avvenuti.
- React calcola poi la differenza tra le due rappresentazioni del Virtual DOM, ovvero fra la rappresentazione del Virtual DOM prima che i dati venissero modificati e l'attuale rappresentazione del Virtual DOM
- A questo punto, React effettua le modifiche nel DOM reale, aggiornando solo ed esclusivamente quello che deve essere cambiato.

IDE per lo Sviluppo

- **Visual Studio Code – Gratuito**

(https://code.visualstudio.com/)

- **ATOM – Gratuito**

(https://atom.io/)

- **Sublime Text - A pagamento**

(http://www.sublimetext.com/3)

- **WebStorm - A pagamento**

(https://www.jetbrains.com/webstorm/download/#section=windows-version)

Installare NodeJS e NPM

Quando si lavora con **React** è necessario utilizzare un **Package Manager** (npm) con **NodeJS** abbiamo **NPM** (Node Package Manager) che ci servirà per gestire ed installare le dipendenze.

Verificare il funzionamento
node –v
npm – v
npm install npm@latest –g (aggiornare NPM)

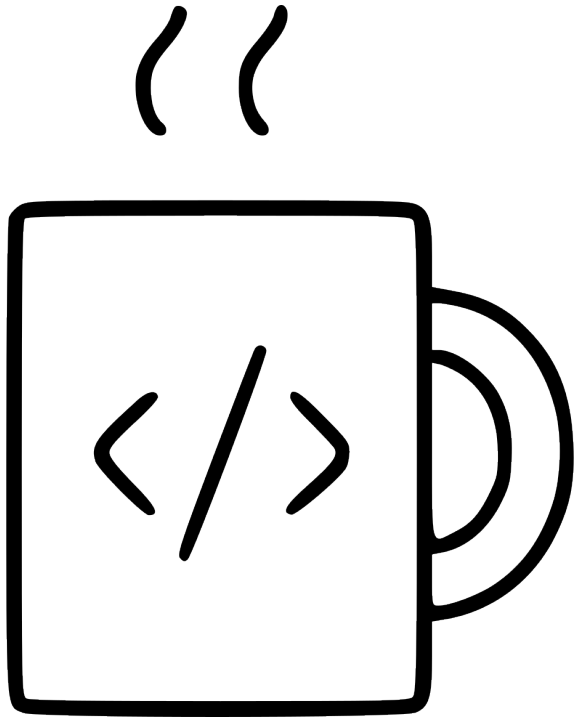
https://nodejs.org/en/

Create React App
Installiamo tramite NPM

Create React App è un boilerplate con molti strumenti già integrati. Fornisce un buon punto di partenza per le nostre applicazioni ReactJS. Open-source, continuamente aggiornato da Facebook

npm install -g create-react-app

https://github.com/facebook/create-react-app



PAUSA

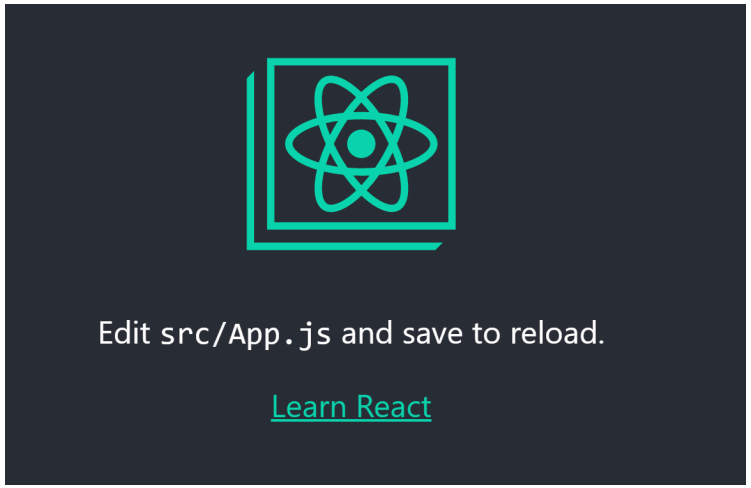
Ci vediamo alle ore 11.20

Creiamo il nostro progetto REACT

npx create-react-app my-app (npx è un esecutore di pacchetti incluso in npm 5.2+)

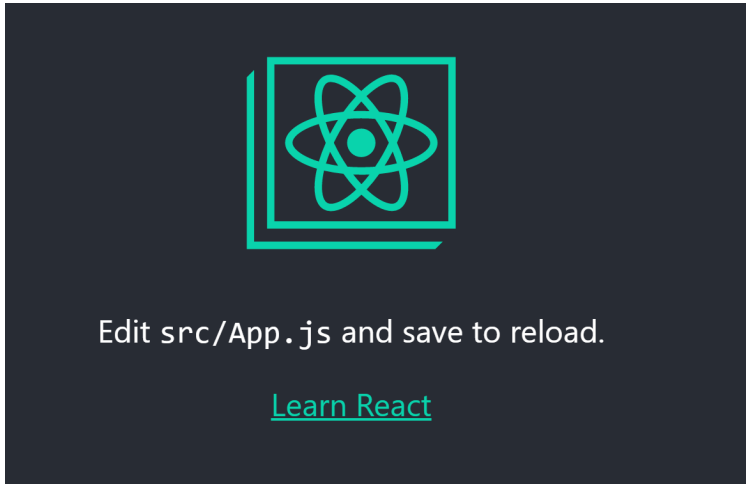
npm init react-app my-app (*is available in **npm** 6+)*

npm start (la nostra applicazione verrà compilata e lanciata su **localhost:3000**)



Building di un applicazione REACT

Quando sei pronto a rilasciare in produzione, esegui il comando **npm run build** e verrà creata una build ottimizzata della tua applicazione nella **cartella build**.



Creazione e struttura di un app React


```
npx create-react-app mia-app
cd mia-app
npm start
```

https://it.reactjs.org/docs/create-a-new-react-app.html

Struttura di una app React

Creare elementi React utilizzando React puro

Renderizzare un Elemento nel DOM

Supponiamo di avere un `<div>` da qualche parte nel tuo file HTML:

```
<div id="root"></div>
```

Lo chiameremo nodo DOM "radice" (o root) in quanto ogni cosa al suo interno verrà gestita dal DOM di React.

Per renderizzare un elemento React nel nodo DOM radice, bisogna passare entrambi a `ReactDOM.render()`:

```
const element = <h1>Hello, world</h1>;
ReactDOM.render(element, document.getElementById('root'));
```

Struttura di una app React

Prendiamo in considerazione il prossimo esempio, nel quale abbiamo un orologio:

```
function tick() {
  const element = (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {new Date().toLocaleTimeString()}.</h2>
    </div>
  );
  ReactDOM.render(element, document.getElementById('root'));
}

setInterval(tick, 1000);
```

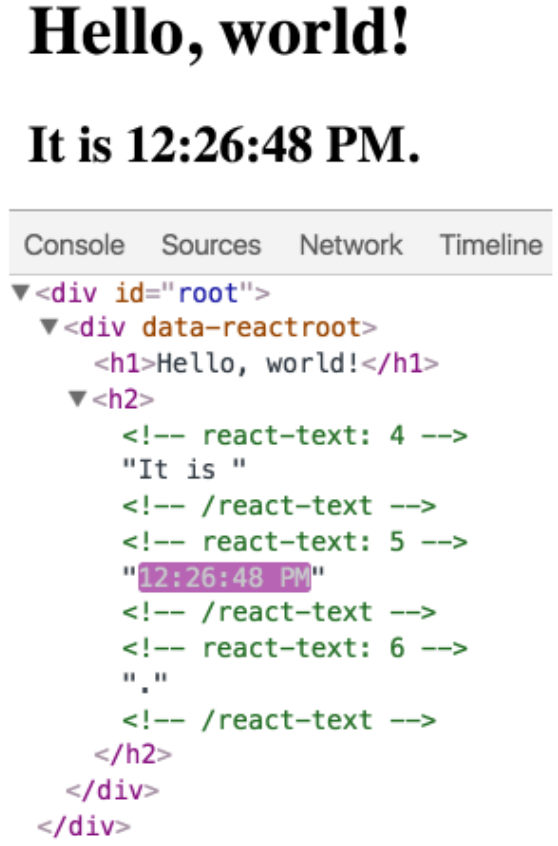
ReactDOM.render() dovrebbe esser chiamato solo una volta.

React Aggiorna Solo Quanto Necessario

Il DOM di React confronta l'elemento ed i suoi figli con il precedente, applicando solo gli aggiornamenti al DOM del browser necessari a renderlo consistente con lo stato desiderato.

Puoi verificare questo fatto ispezionando l'ultimo esempio usando i developer tools:

Anche se abbiamo creato un elemento che descrive l'intero albero della UI ad ogni tick (ogni qual volta la callback viene richiamata, nell'esempio, ogni secondo), solo il nodo testo il quale contenuto è stato modificato viene aggiornato dal DOM di React.



Creare elementi React utilizzando React puro

https://it.reactjs.org/docs/react-api.html

Creare elementi React utilizzando React puro

React

`createElement()`

```
React.createElement(
  type,
  [props],
  [...children]
)
```

Crea e restituisce un nuovo elemento React del tipo indicato. L'argomento `type` può essere un nome di tag (ad esempio `'div'` o `'span'`), un tipo di componente React (una classe o una funzione), o un tipo di frammento React.

Creare elementi React utilizzando React puro

React

createElement()

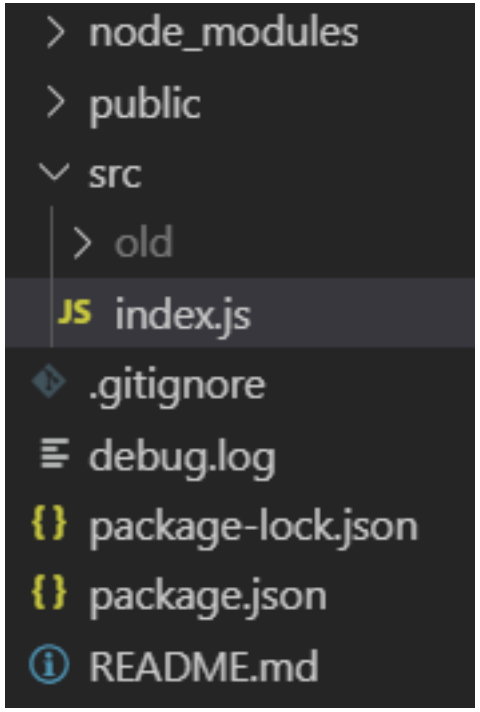
```
React.createElement(
  type,
  [props],
  [...children]
)
```

type: stringa o un elemento React;

props: null o un oggetto;

content: stringa, un elemento React o un componente React;

Creare elementi React utilizzando React puro



src > index.js > ...

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  const el = React.createElement(
5      'h1',
6      {className: 'abc'}, // null
7      'Hello World');
8  ReactDOM.render(el, document.getElementById('root'));
```

Cosa Presentare

Dove Presentare

Hello World

```
<body>
  <noscript>You need to enable JavaScr
  <div id="root">
    <h1 class="abc">Hello World</h1>
  </div>
```

React

Elementi annidati

src > JS index.js > el

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3
4   const elementi = ['Primo', 'Secondo', 'Terzo'];
5
6   const el = React.createElement(
7       'ul',
8       null,
9       /*React.createElement('li', null, 'Primo'),
10      React.createElement('li', null, 'Secondo'),
11      React.createElement('li', null, 'Terzo')*/
12      elementi.map(e => React.createElement('li', null, e))
13      );
14  ReactDOM.render(el, document.getElementById('root'));
```

createElement()

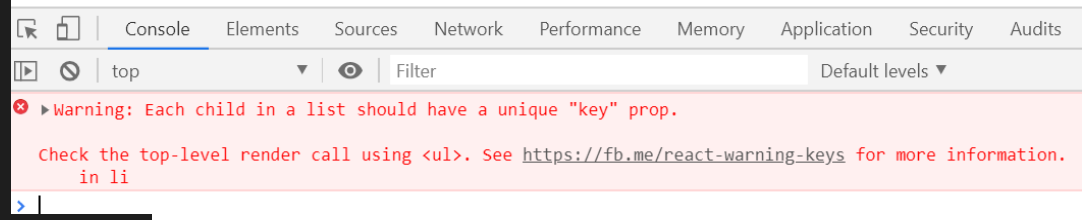
```
React.createElement(
  type,
  [props],
  [...children]
)
```

Per creare più elementi annidati, posso creare singolarmente gli elementi o Iterare un Array.

Elementi annidati

src > JS index.js > el

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3
4   const elementi = ['Primo', 'Secondo', 'Terzo'];
5
6   const el = React.createElement(
7       'ul',
8       null,
9       /*React.createElement('li', null, 'Primo'),
10      React.createElement('li', null, 'Secondo'),
11      React.createElement('li', null, 'Terzo')*/
12      elementi.map(e => React.createElement('li', null, e))
13      );
14  ReactDOM.render(el, document.getElementById('root'));
```



Iterando un Array React ci chiede una key unica. Le chiavi aiutano React a identificare quali elementi sono stati aggiornati, aggiunti o rimossi.

```
elementi.map((e, index) => React.createElement('li', {key: index}, e))
```

Creare elementi React utilizzando React puro

Esempio: Errato Utilizzo della Chiave

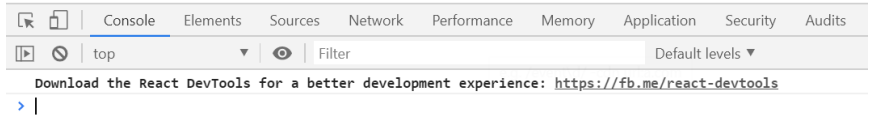
```
function Numero(props) {
  const valore = props.valore;
  return (
    // Sbagliato! Non è necessario specificare la chiave qui:
    <li key={valore.toString()}>
      {valore}
    </li>
  );
}

function ListaNumeri(props) {
  const numeri = props.numeri;
  const lista = numeri.map((numero) =>
    // Sbagliato! La chiave deve essere stata specificata qui:
    <Numero valore={numero} />
  );
  return (
    <ul>
      {lista}
    </ul>
  );
}

const numeri = [1, 2, 3, 4, 5];
ReactDOM.render(
  <ListaNumeri numeri={numeri} />,
  document.getElementById('root')
);
```

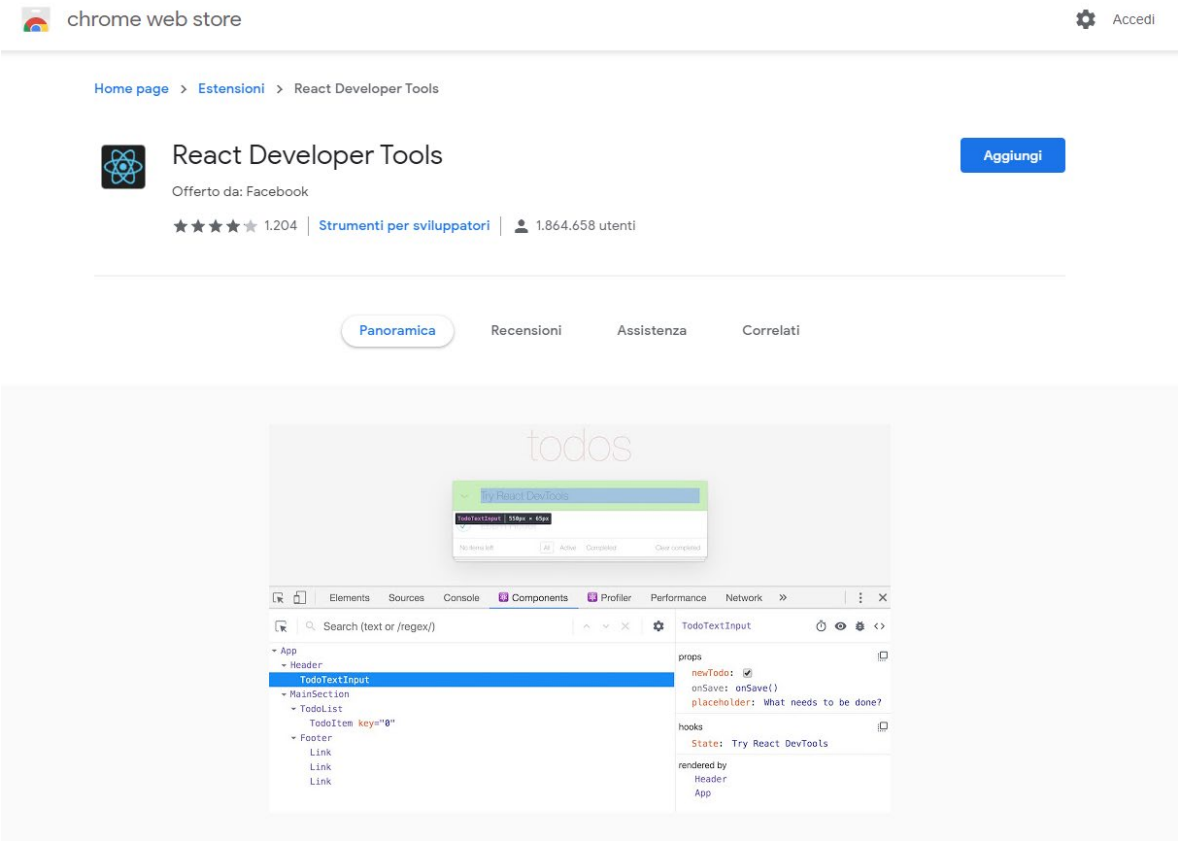
Esempio: Corretto Utilizzo della Chiave

```
function Numero(props) {
  // Corretto! Non è necessario specificare la chiave qui:
  return <li>{props.valore}</li>;
}

function ListaNumeri(props) {
  const numeri = props.numeri;
  const lista = numeri.map((numero) =>
    // Corretto! La chiave deve essere specificata all'interno dell'array.
    <Numero key={numero.toString()}
            valore={numero} />

  );
  return (
    <ul>
      {lista}
    </ul>
  );
}

const numeri = [1, 2, 3, 4, 5];
ReactDOM.render(
  <ListaNumeri numeri={numeri} />,
  document.getElementById('root')
);
```

È buona regola ricordarsi che gli elementi all’interno della chiamata map() hanno bisogno di chiavi.

Per ispezionare il codice React è utile installare su Chrome il **React Developer Tools**





https://fb.me/react-devtools

JSX

Jsx è un estensione della sintassi Javascript, che ci permette di scrivere codice **Javascript** che assomigli molto al codice **HTML**.

https://it.reactjs.org/docs/introducing-jsx.html

JSX

JSX sta per **J**ava**S**cript e**X**tension.

- Si tratta di un' **estensione sintattica di Javascript** che permette di scrivere la struttura dei vari componenti usando una sintassi simile al linguaggio HTML.
- La sintassi JSX viene convertita in semplice linguaggio Javascript.
- Attributi in JSX

  attributo è di tipo stringa -> virgolette

  espressioni Javascript -> parentesi graffe.
- Usare JSX all'interno del codice Javascript
- Annidare i tag in JSX
- Commenti in JSX

React

JSX Rappresenta Oggetti

Babel compila JSX in chiamate a `React.createElement()`.

Questi due esempi sono identici:

```jsx
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);
```

```jsx
const element = React.createElement(
  'h1',
  {className: 'greeting'},
  'Hello, world!'
);
```

JSX

Tramite **JSX** nelle parentesi graffe possiamo inserire qualsiasi espressione che vogliamo sia valutata, altrimenti viene considerata come una stringa.

src > JS index.js > ...

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3
4   const elementi = ['Primo', 'Secondo', 'Terzo'];
5
6   const el = <ol>
7       {elementi.map((e, index) => <li key={index}>{e}</li> )}
8   </ol>
9
10  ReactDOM.render(el, document.getElementById('root'));
11
```

React

JSX Previene gli Attacchi di Iniezione del Codice

Utilizzare l'input degli utenti in JSX è sicuro:

```
const title = response.contenutoPotenzialmentePericoloso;
// Questo è sicuro:
const element = <h1>{title}</h1>;
```

React DOM effettua automaticamente l'escape di qualsiasi valore inserito in JSX prima di renderizzarlo. In questo modo, garantisce che non sia possibile iniettare nulla che non sia esplicitamente scritto nella tua applicazione. Ogni cosa è convertita in stringa prima di essere renderizzata. Questo aiuta a prevenire gli attacchi XSS (cross-site-scripting).

React

JSX restituisce un solo elemento root

```jsx
const el =
<h1>Lista</h1>
<ol>
    {elementi.map((e, index) => <li key={index}>{e}</li> )}
</ol>
```

Failed to compile

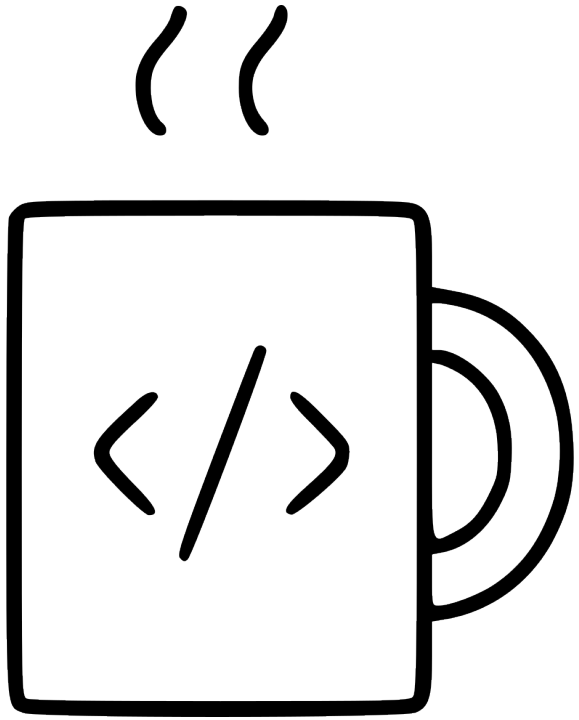
```
./src/index.js
  Line 8:  Parsing error: Adjacent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fragment <>...</>?

   6 | const el =
   7 | <h1>Lista</h1>
>  8 | <ol>
     |  ^
   9 |     {elementi.map((e, index) => <li key={index}>{e}</li> )}
  10 | </ol>
  11 |
```

This error occurred during the build time and cannot be dismissed.

```jsx
const el =
<div>
    <h1>Lista</h1>
    <ol>
        {elementi.map((e, index) => <li key={index}>{e}</li> )}
    </ol>
</div>
```

```jsx
return (
    <Reac.Fragment>
        <h1>Titolo</h1>
        <p>Paragrafo</p>
    </Reac.Fragment>
)
```

React.Fragment risolve il problema del return di più elementi JSX



PAUSA

Ci vediamo alle ore 14.00

challenge
NETWORK

shaping the skills of tomorrow

challengenetwork.it