Facebook
Open Source

Corso ReactJS

libreria javascript per lo sviluppo di
moderne user interface

“A JavaScript library for building user interfaces”

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Facebook Open Source

Cos’è ReactJS

ReactJS è una libreria Javascript creata da Facebook che consente la realizzazione di applicazioni web ***single page*** (***SPA***) ,attraverso una struttura composta di **componenti web** dinamicie e riutilizzabili.
Componenti componibili, a seconda della complessità dell applicazione a cui stiamo lavorando e che si desidera ottenere.

In maniera approssimativa, possiamo dire che un'applicazione realizzata con **React** è composta da tanti componenti quanti sono i vari elementi che compongono l'interfaccia.
È nostro compito decidere quali componenti creare e come strutturarli.

Autore: Christian Chiama

Vantaggi di ReactJS

Facebook Open Source

- Semplice design, “declarative view” per ogni stato dell applicazione
- Componenti incapsulati e Data-Binding (unidirectional data-flow)
- Proprietà (**props**) e stato (**state**) dinamici
- JSX in replace di HTML e Javascript
- Virtual Dom come astrazione del DOM reale
- Può “renderizzare" sia lato client che lato server (SSR - server side rendering)
- SEO Friendly
- Testabile e manutenibile con grande efficienza

Facebook
Open Source

Svantaggi di ReactJS

- View-Oriented, è solamente il View layer
- Menuali non facili da comprendere per i “newbies”
- Dimensione della libreria
- In molti casi il codice da scrivere è imponente

Facebook Open Source

Perché Utilizzare ReactJS

- Virtual DOM
- Componenti riutilizzabili e componibili
- Declarative Approach

"Declarative views make your code more predictable and easier to debug"

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Facebook Open Source

Strumenti di Sviluppo

- Visual Studio Code
- Atom
- Bracket
- Extension Chrome per ReactJS (developer tools) e Redux Tools
- terminale

JSX, Babel, Webpack, and NPM necessari per lo sviluppo con React

Autore: Christian Chiama

INNOVAFORMAZIONE Tecnologia e Formazione

Facebook
Open Source

piccola review….

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

ECMASCRIPT 6 / ECMASCRIPT 2015

Facebook Open Source

- Dichiarazione di variabili e constanti: **const** e **let**
- Arrow function
- Template string
- Moduli ES6
- destrutturazione, classi, gestione parametri …..

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

ECMASCRIPT 6 / ECMASCRIPT 2015

const = dichiara una variabile **immutabile**, che non può essere modificata successivamente. La sua visibilità (**scope**) è a livello di “blocco” (**block-scoped**)

```
function bar() {
    const tmp = 4;
    if (true) {
        const tmp = 3;
        console.log(tmp); // 3
    }
    console.log(tmp); // 4
}
```

Autore: Christian Chiama

Facebook Open Source

ECMASCRIPT 6 / ECMASCRIPT 2015

let = dichiara una variabile, equivalente della “vecchia” **var**,
ma con visibilità (scope) a livello di “blocco” (**block-scoped**), diversamente da var in cui la visibilità è a livello di funzione (**function-scoped**)

```
function order(x, y) {
  if (x > y) {
    let tmp = x;
    x = y;
    y = tmp;
  }
  console.log(tmp === x); // ReferenceError: tmp is not defined
  return [x, y];
}
```

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Facebook Open Source

ECMASCRIPT 6 / ECMASCRIPT 2015

Arrow Function: In Javascript le funzioni anonime sono uno dei costrutti più utilizzati ma ogni volta richiedono un minimo di boilerplate, ossia codice ripetitivo.Con le arrow function scrivere funzioni anonime diventa molto semplice.

```
const number = [0,1,2,3,4];

const inc = number.map(num => ++num); // print [1,2,3,4,5]

const fn = () => {};
```

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

ECMASCRIPT 6 / ECMASCRIPT 2015

Template string: ES6 introduce template string, ossia stringhe che permettono l inclusione di variabili, evitando la concatenazione di stringhe

```
let reactJS = `
  reactjs course
  in multiline string
`

let template = `${reactJS}`

console.log(template)
```

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Facebook Open Source

ECMASCRIPT 6 / ECMASCRIPT 2015

Moduli: ES6 introduce il sistema per importare ed esportare moduli.

```
// file js/util.js
export function sum(x, y) {return x + y}
export const SUM_ADDENDUM = 10

// file test.js
import * as util from "js/util"
import {sum, SUM_ADDENDUM} from "js/util"

console.log(util.sum(10, 20)) // 30
console.log(sum(SUM_ADDENDUM, SUM_ADDENDUM)) // 20
```

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

ECMASCRIPT 6 / ECMASCRIPT 2015

named export

```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}

//------ main.js ------//
import { square, diag } from 'lib';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5

//oppure
// main.js ------//
import * as lib from 'lib';
console.log(lib.square(11)); // 121
console.log(lib.diag(4, 3)); // 5
```

il nome del modulo importato deve essere uguale a quello exportato solo con l operatore ***as*** si può modificare

export default

```
//------ myClass.js ------
export default class MyClass { };

//------ main.js ------
import MyClass from 'myClass';
const myCls = new MyClass();

import Pippo from 'myClass';
const pippo = new Pippo();
```

puo cambiare il nome dell import con il default export

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

classi, destrutturazione, nuove firme di metodi, gestione dei parametri e molto altro ancora ha introdotto ES6, che sarà visto nel dettaglio nel proseguo del corso…..

Autore: Christian Chiama

Virtual DOM

Facebook Open Source

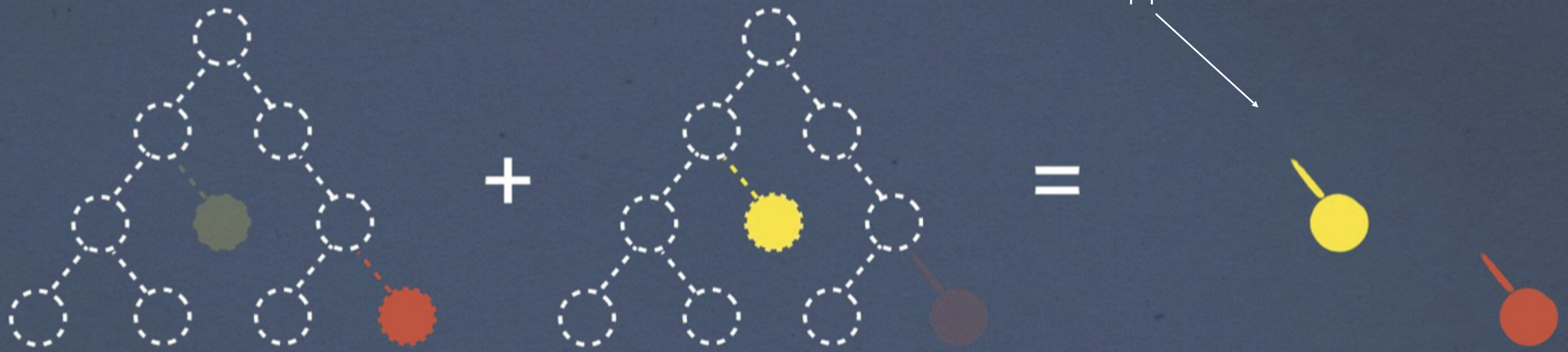
Una delle problematiche che spesso affligge gli sviluppatori web sono le **performance**.
A causa della natura dei browser, che eseguono le logiche JavaScript nello stesso thread dove eseguono il rendering della pagina, le performance delle applicazioni web possono ridursi drasticamente sui dispositivi che hanno a disposizione poca capacità di calcolo o poca memoria.
Il problema non è dovuto solitamente alla lentezza delle chiamate, *alla manipolazione degli elementi del DOM della pagina, i quali sono lenti e risultano bloccanti nell'esecuzione del codice*.

ReactJS introduce il **Virtual DOM**, come astrazione del **DOM**.

Autore: Christian Chiama

Virtual DOM

state change ⟶ compute diff ⟶ re-renderer



The diff algorithm generates a list of DOM mutations, the same way version controls output text mutations

Complessità: O(n)

Autore: Christian Chiama

Componenti ReactJS

Facebook Open Source

Un **Component o React.Component** è concettualmente una funzione Javascript e ti permette di suddividere la UI del progetto in parti sempre più piccole. Lo scopo è quello di creare componenti **riutilizzabili** da componenti più grandi, creando delle vere e proprie **gerarchie di components**.

Un **component** accetta come input le **props**. Le props arrivano al component come un oggetto contenente diverse proprietà. Per esempio posso passare ad un component che mostra il profilo di un utente più oggetti contenenti informazioni sull'utente così come passare solo un identificativo numerico atto poi al recupero dell'utente in un secondo momento.

In **ReactJS** un componente è formato da un tag HTML (o DOM tag) **utilizzando un nome che inizia con lettera maiuscola.** Un componente che definisce un utente "user" avrà la seguente forma: *<User /> e **deve sempre ritornare un elemento root** ,attraverso la funzione principale di **render**, funzione che non può mancare nella definizione di un componente, il cui scopo è quello di "ritornare" un pezzo di codice HTML ad altri components.*

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

React.createClass Vs React.Component

```
export default React.createClass({
  render() {
    return <div>Hello, InnovaFormazione React App!</div>
  }
})
```

due vie per creare un componente

```
export default class App extends React.Component{
  render() {
    return <div>Hello, InnovaFormazione React App!</div>
  }
}
```

```
render(<App/>, document.getElementById('app'))
```

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Analisi di un Componente ReactJS

```
class App extends React.Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <h1 className="App-title">Welcome to React</h1>
        </header>
      </div>
    );
  }
}

export default App;
```

funzione di render

definizione componente

root element

responsabile di restituire il contenuto
che si vuole mostrare nella pagina, ottenuto specificandone il markup HTML
o richiamando altri componenti React.

viene utilizzato **className**, perché class è una keyword riservata del javascript

esportazione moduli ES6

elemento root ritornato dalla funzione di render

INNOVAFORMAZIONE
Tecnologia e Formazione

Autore: Christian Chiama

React Render

Facebook Open Source

Per produrre un contenuto che sia visibile sulla pagina, è necessario introdurre nell'oggetto una funzione **render()** che avrà la responsabilità di restituire il contenuto che si vuole mostrare nella pagina, ottenuto specificandone il markup HTML o richiamando altri componenti React.

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

JSX

Facebook Open Source

JSX (JavaScript eXtension) è un estensione sintattica del JavaScript, che consente la scrittura dei **componenti** , utilizzando una scrittura simile all HTML "mischiato" con JavaScript, con un approccio dichiarativo. In **JSX** il codice rimane più leggibile e semplifica l'indicazione degli elementi e dei loro attributi.

```
/*  crea un oggetto di tipo React Element
*   La sintassi usata <h1>...</h1> NON è una Stringa.
*   <h1>...</h1> NON è racchiusa tra virgolette
*   è JSX
*/
const reactElement = <h1>Ciao... Un saluto da React!</h1>;
```

E' necessario l utilizzo di Babel per compilare JSX in plain JavaScript

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

JS Vs JSX

Facebook Open Source

```
const label = 'Corso ReactJS di InnovaFormazione';

const reactNode = React.createElement(
    'label',
    null,
    label
);
```

senza jsx

creazione di un nodo HTML, una label, senza l utilizzo di jsx

creazione identica con l utilizzo di jsx. Notare la compattezza della sistassi e la leggibilità

in jsx

```
const label = 'Corso ReactJS di InnovaFormazione';

const reactNode = <label>{label}</label>;
```

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Espressioni JavaScript in JSX

In **React/JSX** possiamo creare dei mix, mescolare JavaScript in JSX, ***racchiudendo semplicemente il js tra parantesi graffe***, come l esempio sottostante:

```
const label = 'Corso ReactJS di InnovaFormazione in data ';

const data = () => { return Date() };

const inputType = 'text';

//Notare l utilizzo di espressioni e/o valori JavaScript tramite {} mescolati tra JSX
const reactNode = <label>{label} {data()}=
                        <input type={inputType} value={ Math.pow(6 , 2) } />
                  </label>;
```

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Installazione e Setup React

- Installazione tool cli React 1
- Creazione App React 2
- Lancio App React 3

da terminale lanciare i seguenti comandi

```
$ npm install -g create-react-app

$ create-react-app innovaformazione

$ cd innovaformazione

$ npm start
```

utilizzare il terminale del SO ed aprire in seguito
I IDE a creazione avvenuta

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Struttura progetto

Facebook Open Source

cartella root che prende il nome dell app

root folder

```
innovaformazione-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── registerServiceWorker.js
```

package.json

file contenete tutte le dipendenze del progetto
e gli script che
eseguono l app ed altri comandi

sorgenti app

Tutti i sorgenti dell App andranno sotto questa cartella

public folder

In questa cartella vi sono i file eseguiti sul browser e dove la nostra App sarà caricata

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Nested Component e Props

```
class WrapperComponent extends React.Component {
    render() {
      return (
        <div className="wrapper">
          <h1>{this.props.title}</h1>
          {this.props.children}
        </div>
      );
    }
  }

  class ChildComponent extends React.Component {
    render() {
      return (
        <p>{this.props.body}</p>
      );
    }
  }
```

Componente padre

Le **props** sono immutabili

passaggio di parametri
al componente via **props**
(vedere slide seguente)

Componente figlio

```
class App extends React.Component {
  render() {
    return (
      <WrapperComponent title="I am the wrapper">
        <ChildComponent body="Hello from child component" />
      </WrapperComponent>
    );
  }
}
```

definizione ed annidazione

Ciclo di vita dei Componenti

- componentWillMount (@d)
- componentWillReceiveProps(nextProps) (@d)
- shouldComponentUpdate(nextProps, nextState, nextContext)
- componentWillUpdate(nextProps, nextState) (@d)
- componentDidUpdate(prevProps, prevState, prevContext)
- componentDidCatch(errorString, errorInfo)
- componentDidMount
- componentWillUnmount

(@d) = deprecated

Autore: Christian Chiama

Component State

Facebook Open Source

Lo stato è rappresentato da un oggetto accessibile nel codice del componente tramite la proprietà ***state*** che può essere inizializzato a proprio piacimento; a differenza di quanto avviene con le proprietà , fornite dall'oggetto props, lo stato può essere variato chiamando il metodo ***setState()***.

A fronte di una variazione dello stato, generalmente tramite la gestione di un evento che si verifica a seguito di un'azione dell'utente, **React** invoca la funzione *render()* e verifica la necessità di apportare modifiche alla parte del DOM che costituisce la rappresentazione del componente all'interno della pagina Web.

Component State

Facebook Open Source

```
class LikeButton extends React.Component {

  getInitialState() {
      return { liked: false };
  }
  render() {
      if (this.state.liked)
          return <div>Ti piace!</div>
      else
          return <a href="#" onClick={this.doLike}>Mi piace</a>
  }
  doLike() {
      this.setState({ liked: true });
  }
};
```

1

La funzione **`getInitialState()`** ha la responsabilità di restituire un oggetto JavaScript che contiene lo stato iniziale del componente; nell'oggetto possiamo inserire tutte le proprietà che vogliamo a seconda dei requisiti del componente che vogliamo realizzare.

2

fil metodo **`setState()`** permette di passare un oggetto al componente **React** con l'obiettivo di specificare quali proprietà dello stato hanno subito un cambiamento: nel nostro caso, il flag liked è stato impostato al valore true. Quando lo stato cambia, React chiama nuovamente il metodo ***render()*** per ottenere la rappresentazione aggiornata del contenuto che terrà conto dello stato attuale.

3

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

React Router

Facebook Open Source

React Router è una libreria che permette di creare applicazioni **React** con più pagine in cui la *transizione* fra una pagina e l'altra avviene in maniera dinamica tramite **Javascript**, senza dover ogni volta ricaricare la pagina.Questa è certamente una delle funzionalità indispensabili per una *Single Page Application*. Potremmo creare il nostro *Router* da zero usando per esempio la *History API di HTML5*, ma **React Router** ci agevola il lavoro mettendoci a disposizione un'API completa e semplice da usare. E' un insieme di **componenti**, come vedremo.

elemento principale (React componente)

React Router mette a disposizione diversi tipi di ***<Router />*** a seconda del genere di applicazione che vogliamo realizzare o delle funzionalità di cui abbiamo bisogno. Possiamo aggiungere React Router al nostro progetto con il seguente comando npm:

$ npm i react-router-dom —save
$ yarn add react-router-dom —dev (alternativa)

questo comando aggiunge la libreria,
che esponendo le proprie API, la rende disponibile a tutto il progetto,
ed inserisce nel package.json nella sezione "***dependencies***"
il "pacchetto"composto dal nome della libreria e versione installata

React Router

Facebook Open Source

```
export class BrowserRouter extends React.Component<BrowserRouterProps, any> {}
```

typescript definition

Progetti browser based

http://localhost:3000/home

<BrowserRouter />

dovrebbe essere usato in un contesto
dove il server dovrà
gestire principalmente richieste dinamiche
(conoscendo come rispondere a determinati URI).
Url stile RESTFul, senza hash

```
render(
    <BrowserRouter>
        <App/>
    </BrowserRouter>,
    document.getElementById('root')
)
```

Progetti statici come siti web

http://localhost:3000/#/home

<HashRouter />

```
export class HashRouter extends React.Component<HashRouterProps, any> {}
```

dovrebbe essere usato in un contesto
come siti web, risorse statiche. Gli url hanno l *hash*

```
render(
    <HashRouter>
        <App/>
    </HashRouter>,
    document.getElementById('root')
)
```

INNOVAFORMAZIONE
Tecnologia e Formazione

App con React Router

Facebook Open Source

```
import React from 'react';
import {
  Route,
  NavLink,
  HashRouter
} from "react-dom-router";
import Home from "./Home";
import Stuff from "./Stuff";
import Contact from "./Contact";
```

```
class Home extends React.Component {}

class Stuff extends React.Component {}

class Contact extends React.Component {}
```

tipo di router

Home Component

Stuff Component

Contact Component

```
class App extends React.Component {
  render() {
    return (
      <HashRouter>
      <div>
        <h1>Simple SPA</h1>
        <ul className="header">
          <li><NavLink to="/home">Home</NavLink></li>
          <li><NavLink to="/stuff">Stuff</NavLink></li>
          <li><NavLink to="/contact">Contact</NavLink></li>
        </ul>
        <div className="content">
          <Route path="/home" component={Home}/>
          <Route path="/stuff" component={Stuff}/>
          <Route path="/contact" component={Contact}/>
        </div>
      </div>
    </HashRouter>
    );
  }
}

export default App;
```

- proprietà Route: **RouteProps**
- path
- exact
- location
- component
- render
- children
- strict

INNOVAFORMAZIONE Tecnologia e Formazione

Flux: Architettura

Facebook Open Source

Flux è un pattern architetturale che *si pone come alternativa del ben più noto* **Model View Controller**. Flux è il pattern che regge il front-end web di Facebook, ed è quindi il pattern di riferimento per applicazioni enterprise sviluppate tramite **React**. Nulla vieta però di utilizzarlo con altre tecnologie.

Di base questi pattern hanno il loro cuore e filosofia nei DDD, Domain Driven Designg, ovvero nei sistemi ad eventi, dove vi sono degli eventi che girano in un contenitore ed i “subscriber”, che aspettano di essere “triggerati”. In Flux potremo associare un evento ad un Action , il Dispatcher all EventBus, ed i subscriber o client alle view o components.

Autore: Christian Chiama

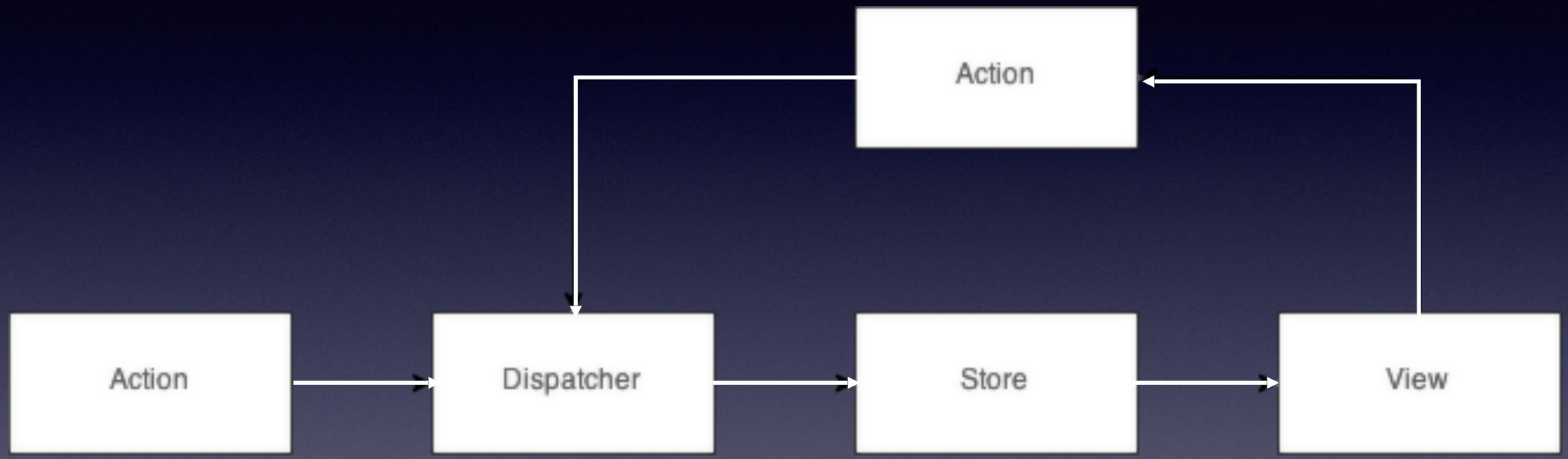
Flux: Gli attori coinvolti

Facebook Open Source

- **Actions**: Sono degli Helpers che passano dati al Dispatcher;
- **Dispatcher** : Riceve le action e trasmette i payload alle callback di chi si è registrato (listener).
- **Stores**: Sono come contenitori per lo stato e la logica dell'applicazione. Il vero lavoro nell'applicazione è fatto negli store. Gli store che si sono registrati ad i listener registrati nel Dispatcher eseguiranno di conseguenza e aggiorneranno le View.
- **View:** Sono i componenti **React** che prendono lo stato dagli store e lo propagano ad i figli (ricordate props dal padre ad i nested)

Flux: Architettura

Facebook Open Source



Concettualmente simile al Domain-Driven-Design (DDD) o Sistemi ad Eventi

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Flux React: Integrazione

Facebook Open Source

Per integrare **Flux** in un nostro porgetto, dobbiamo per prima cosa importare la libreria, tramite npm o yarn, come gia ampiamente visto nel corso, per cui procediamo con i classici comandi:

$ npm i flux —save
$ yarn add flux —dev (alternativa)

ricordare: *con la versione di npm > 5 non è più necessario l opzione —save, di default verrà messo nelle dependencies del package.json*

Redux

Facebook Open Source

fLa definizione ufficiale recita che **Redux** è ***“un contenitore di stati prevedibili per le applicazioni JavaScript”***. Al posto del termine *“prevedibile”* avremmo potuto usare *“deterministico”*, ma forse questo non rende più chiara la definizione a chi parte da zero. Il fatto è che Redux nasce per risolvere un problema a cui siamo talmente abituati nello sviluppo delle nostre applicazioni che quasi non ci facciamo più caso: **la gestione dello stato o state management.**

dati provenienti dal server e memorizzati in una cache locale;

Da un punto di vista puramente tecnico, **lo stato di un sistema** (per esempio di un’applicazione) è l’insieme delle condizioni interne in uno specifico istante che determinano il risultato delle interazioni con l’esterno.
In altre parole, lo stato di un’applicazione è l’insieme delle informazioni che determinano l’output in corrispondenza di un dato input in uno specifico istante.

“Redux is a predictable state container for JavaScript apps.”

INNOVAFORMAZIONE Tecnologia e Formazione

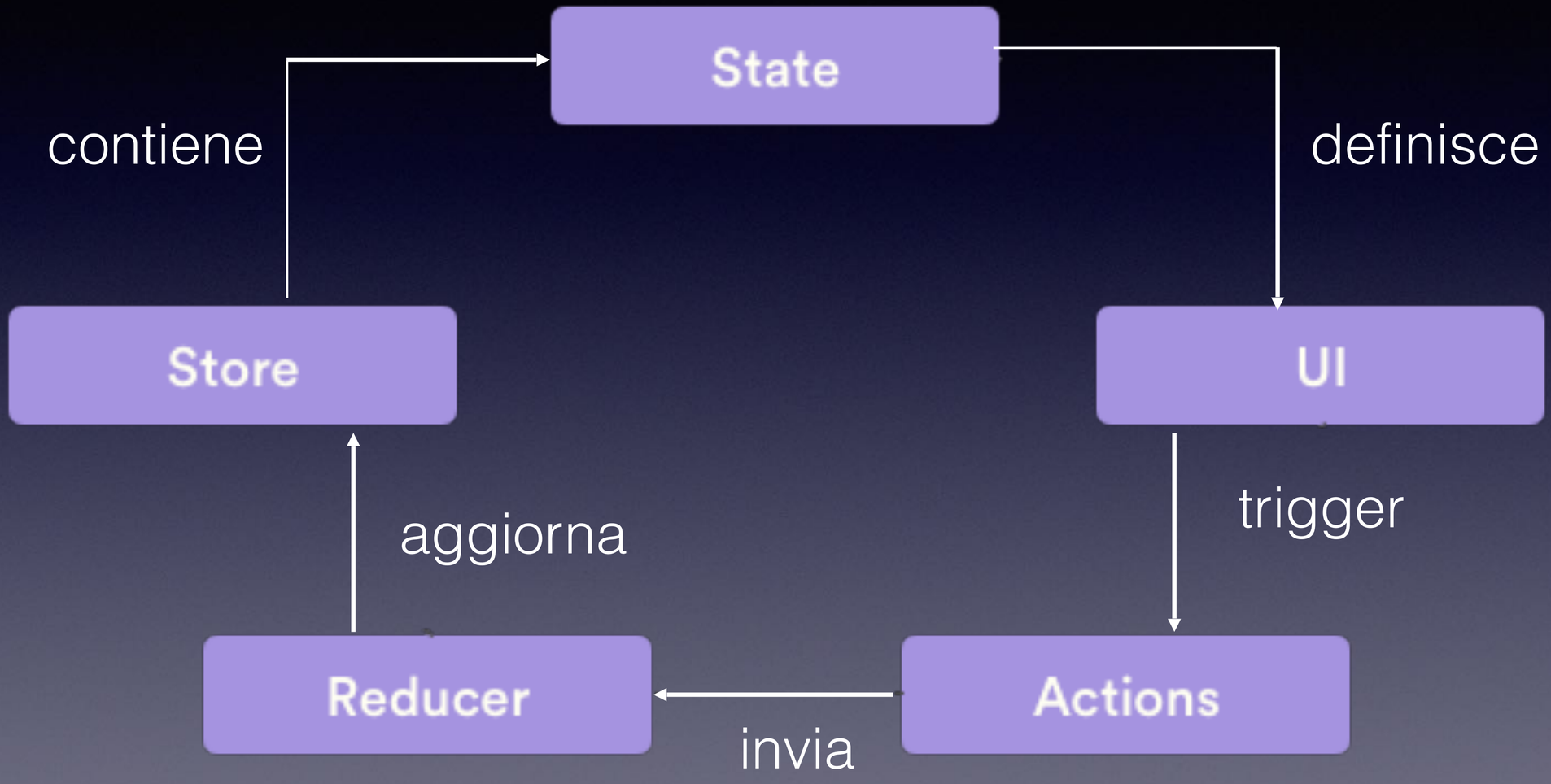
Autore: Christian Chiama

Redux: Gli attori coinvolti

- **Actions**: E' un oggetto JavaScript che rappresenta un'azione in grado di cambiare lo stato corrente, cioè di sostituire l'attuale stato con un nuovo stato;
- **State**: Rappresenta lo stato dell'applicazione ed è costituito da un oggetto con struttura arbitraria, dipendente dalle nostre esigenze applicative; in linea di massima lo stato dell'applicazione ha una struttura ad albero, ma questo non è assolutamente richiesto da Redux
- **Dispatcher** : È il componente che ha il compito di inviare una *Action* allo *Store* il quale, tramite un opportuno *Reducer*, effettua la transizione di stato prevista dal tipo di *Action*
- **Stores**: È il contenitore di Redux che custodisce lo stato dell'applicazione e ne consente l'accesso e la manipolazione dall'esterno; come esiste un unico oggetto che rappresenta lo stato dell'applicazione, così esiste un unico *store* per la sua gestione;
- **Reducer:** Un *reducer* è una normale funzione JavaScript che prende lo stato corrente e un'azione e restituisce lo stato successivo; esso è responsabile della transizione tra uno stato e l'altro nel flusso operativo della nostra applicazione ed è fondamentale che sia una funzione pura, cioè una funzione senza effetti collaterali (side-effects), che dato uno specifico input restituisce sempre lo stesso output

Redux: Architettura e flow

Facebook Open Source



INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Redux React: Integrazione

Facebook Open Source

Per integrare **Redux** in un nostro progetto, dobbiamo per prima cosa importare la libreria, tramite npm o yarn, come gia ampiamente visto nel corso, per cui procediamo con i classici comandi:

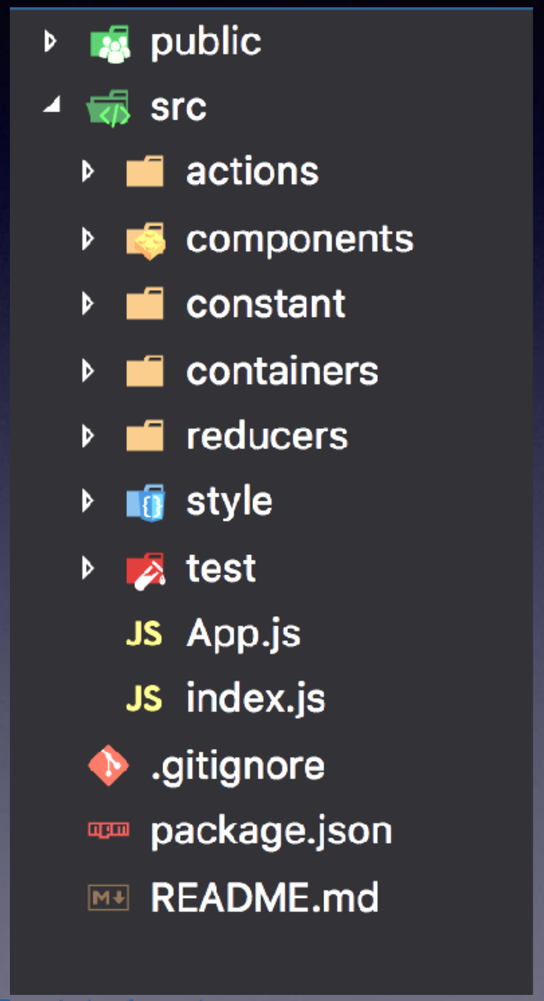
$ npm i react-redux redux —save
$ yarn add react-redux redux —dev (alternativa)

Vi sono alter librerie molto spesso associate come **redux-thunk** (middleware) redux-logger .....

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Sviluppare un App con Redux: struttura



- Actions
- components
- constant
- containers
- reducers
- styles
- test

INNOVAFORMAZIONE Tecnologia e Formazione

Autore: Christian Chiama

Facebook Open Source

Conclusione

Spesso **Redux** viene associato a **React**, la nota libreria di Facebook per la creazione di interfacce utente. Molti sviluppatori pensano che questo connubio sia indissolubile, ovvero che se usi React devi necessariamente usare Redux (e viceversa). In realtà le due librerie sono totalmente scollegate: è possibile usare **React** senza **Redux**, come è pure possibile usare **Redux** con JavaScript puro o con altre librerie e framework come **Angular** o **jQuery**.
Il motivo per cui spesso si tende ad accoppiare **Redux** con **React** dipende probabilmente dalla natura dei componenti di **React**: elementi il cui rendering visivo deriva dalle variazioni di stato.
Questo ***reagire*** alle variazioni di stato si presta bene ad essere interfacciato con il modello ***publisher/subscriber*** proposto da **Redux**, come abbiamo visto quando abbiamo illustrato l'integrazione con l'interfaccia grafica della nostra applicazione. In altre parole, possiamo registrare un componente React tramite `store.subscribe()` in modo che venga ***avvertito*** quando si verifica un cambio di stato in Redux per effettuare il rendering automatico sulla base della nuova situazione. Naturalmente non c'è nulla di particolarmente complesso in questo, che potrebbe quindi essere un approccio praticabile. Ma se vogliamo sfruttare al meglio le prestazioni di **React** dovremmo effettuare alcune analisi delle condizioni dello stato attuale dello ***store*** di **Redux** prima di avviare il rendering. Si pensi, ad esempio, al caso in cui lo stato di Redux ha subito una modifica, ma nessuno dei dati che interessano il nostro componente è stato cambiato. In questa situazione rischieremmo di fare dei rendering inutili che possono avere un impatto non indifferente sulla nostra applicazione.

Autore: Christian Chiama