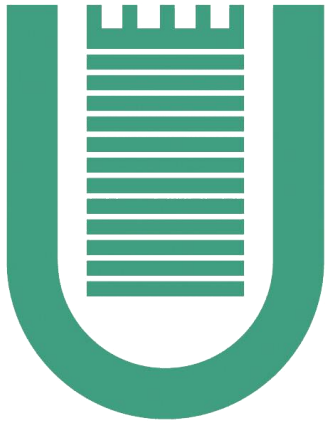
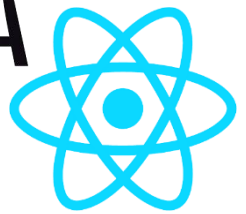


UNIVERSITA' degli STUDI di ROMA
TOR VERGATA

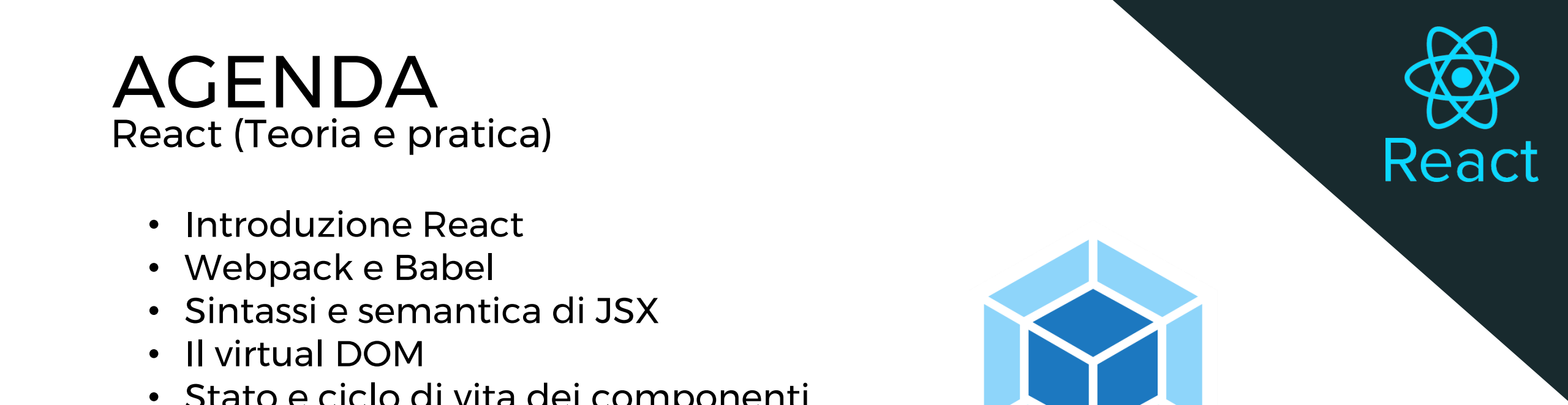


Presentazione finale ISSSR – React

Pelella, Tranzocchi, Trotta, Perrone, Pusceddu, Scarpitta, Scarlino

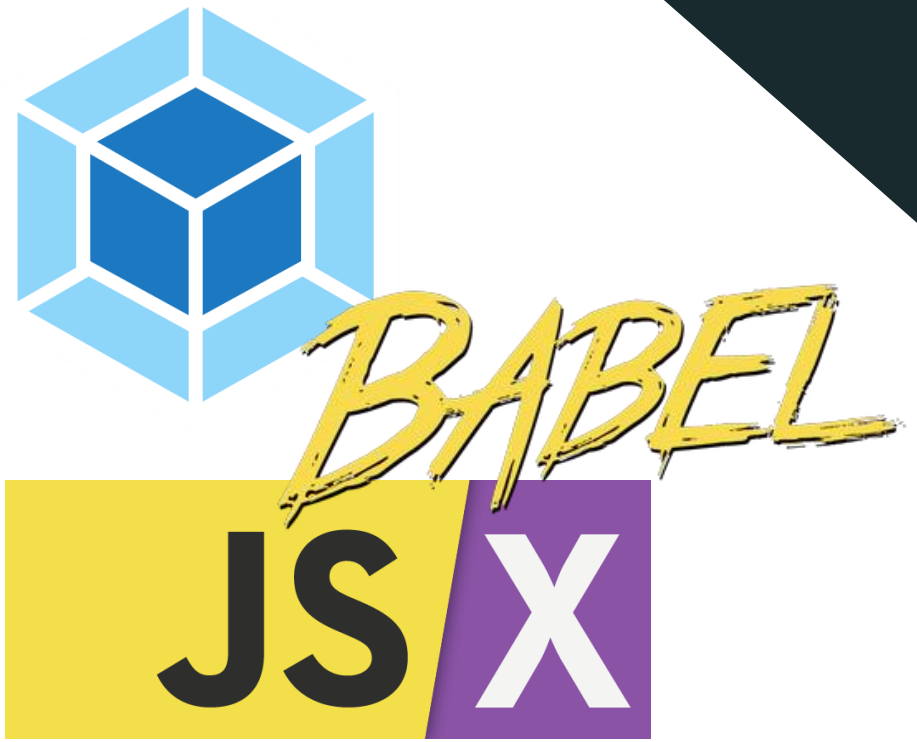
AGENDA

React (Teoria e pratica)



- Introduzione React
- Webpack e Babel
- Sintassi e semantica di JSX
- Il virtual DOM
- Stato e ciclo di vita dei componenti
  - Props
  - State
  - I metodi che governano il ciclo di vita
- Gestione degli eventi
  - onClick
- Confronto con AngularJS

- Hands-on React



Presentazione
Introduzione React

1

Declarative
React semplifica la creazione di interfacce utente interattive.
Codice più prevedibile e più semplice da capire.

2

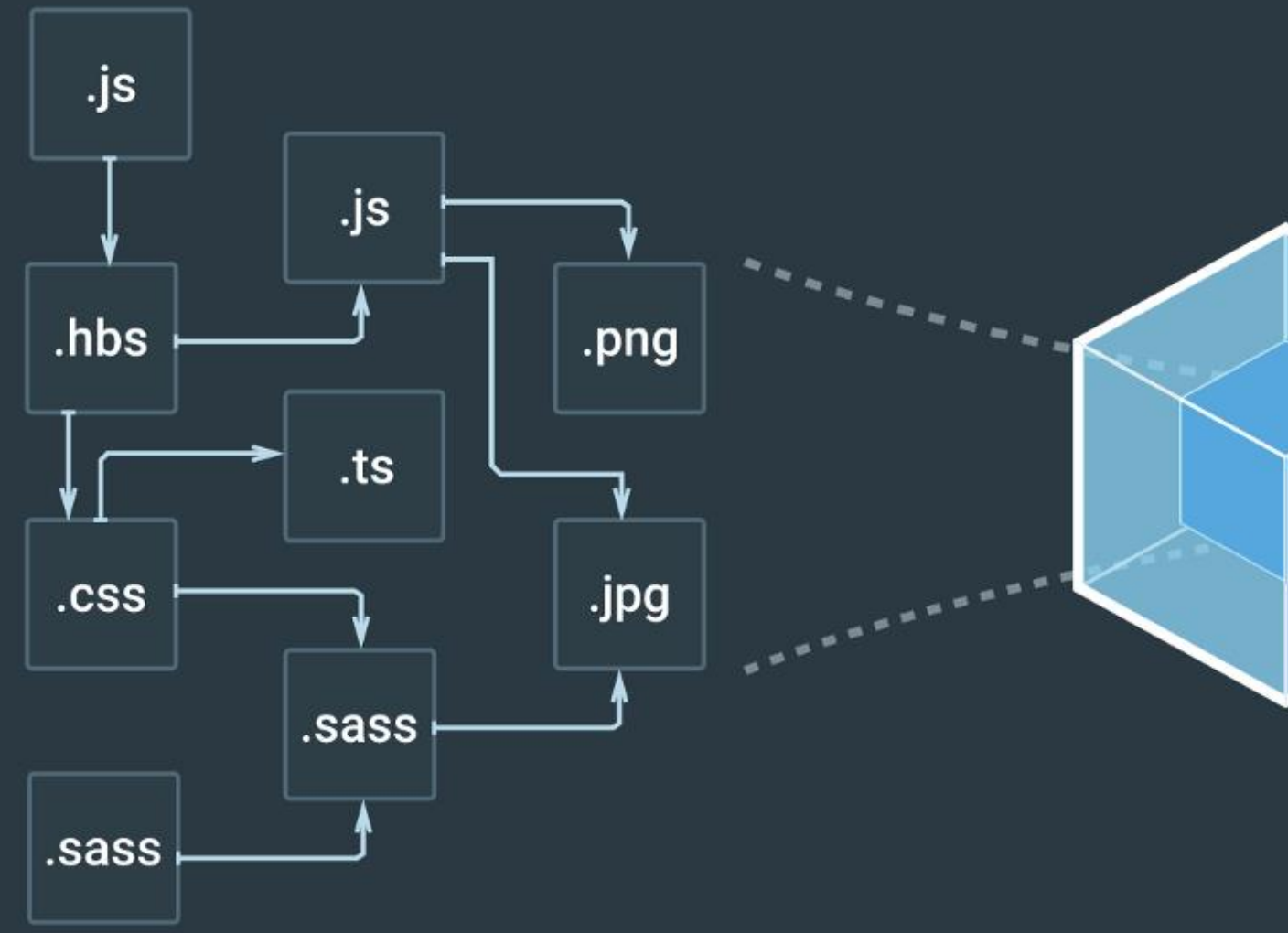
Component-based
Crea componenti incapsulati che gestiscono il proprio stato, quindi sfruttali per creare interfacce complesse.

3

Learn Once, Write Anywhere
React può essere sfruttato per progetto nuovi o inserito in progetti esistenti.
React-Native per app mobile.

A JavaScript library for building user interfaces.

A bundler for javascript and friends.



MODULES WITH DEPENDENCIES

- Webpack è un bundler di moduli statici per moderne applicazioni JavaScript.
- *npm init* per avviare la configurazione base del progetto
- Cartelle *app* e *public* per il codice sorgente e quello per il browser.

The compiler for next generation JavaScript

Babel come compilatore JS

Insieme di programmi (tools) generalmente utilizzato per convertire codice ECMAScript 2015+ in versioni precedenti di JS compatibili con vecchi browser o ambienti.

Babel e JSX

Babel può convertire la sintassi di JSX e rimuovere le type annotations.

JavaScript eXtended (JSX)

È un’estensione sintattica del linguaggio JavaScript.

Il linguaggio di React

- Descrive la GUI con un approccio dichiarativo volto alla leggibilità.
- Babel permette la compilazione da JSX a puro JS.
- Un **elemento** JSX è un oggetto che descrive cosa vogliamo mostrare sullo schermo.

Hello, world!

```
var element = <h1>Hello, world!</h1>;

ReactDOM.render(
      element,
      document.getElementById('root')
);
```

Il metodo render()

Renderizza un elemento JSX nel DOM, alla posizione specificata dal container.

Unire logica applicativa di Front End e markup garantisce alta coesione.

Essi cooperano nel mostrare la UI.

JSX permette di definire unità debolmente accoppiate dette Component.

Un **Component** aggrega elementi JSX.

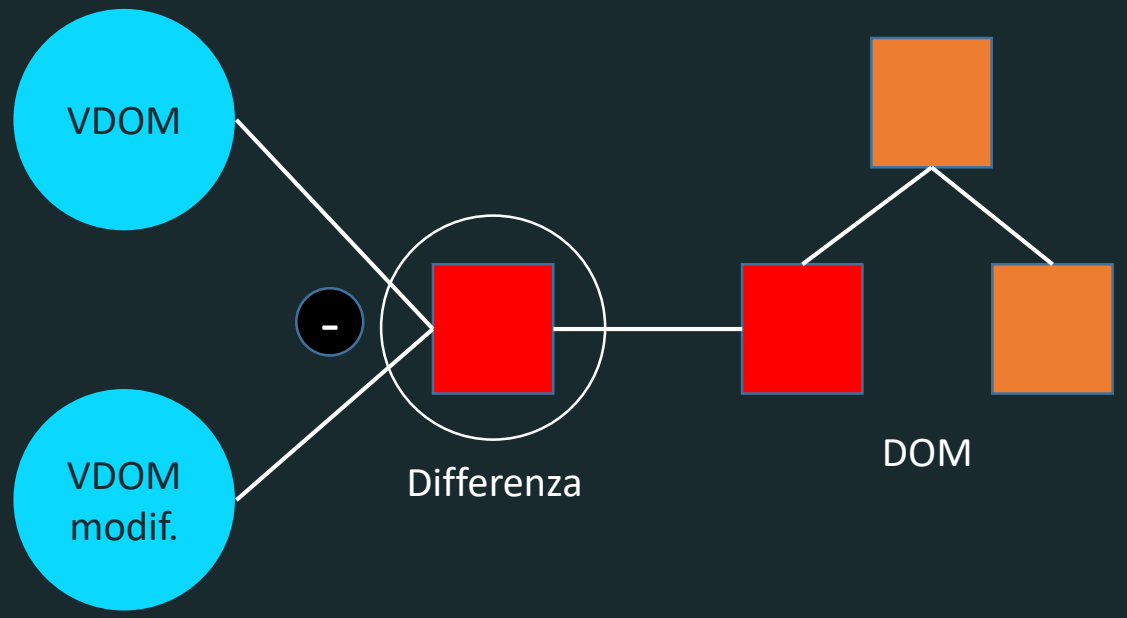
```
Class Intro extends React.Component {
   render() {
      return <h2>Benvenuti</h2>;
   }
}
```

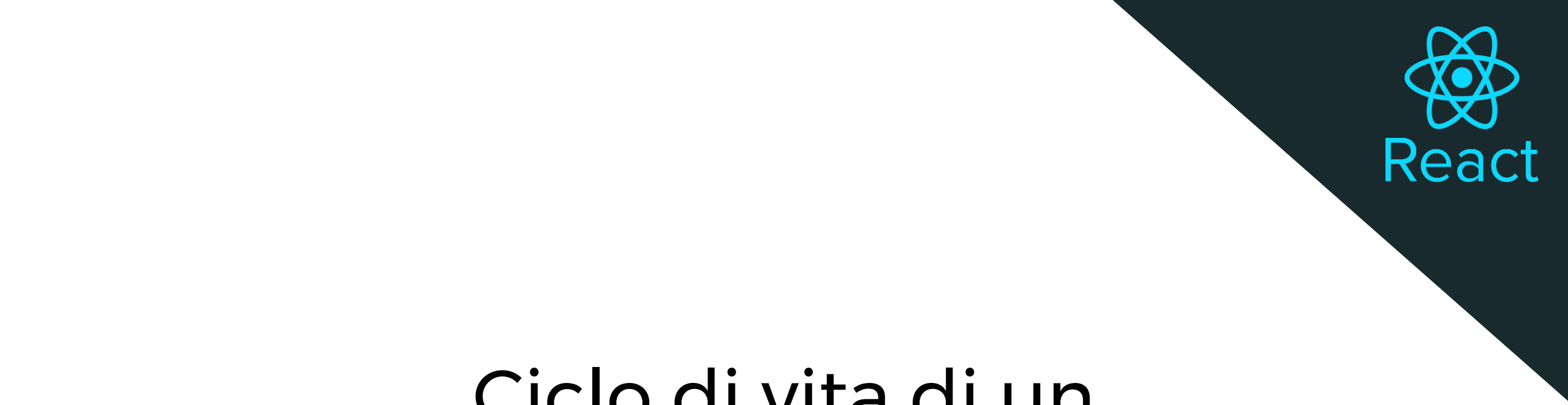
Il Virtual DOM (VDOM)

È una rappresentazione virtuale della struttura della pagina Web in memoria centrale.

Virtual DOM

- Quando l'utente interagisce con la UI, React dapprima applica le modifiche al VDOM.
- Un algoritmo di «diffing» calcola in tempo lineare le differenze tra gli stati del VDOM prima e dopo l'azione dell'utente.
- Si re-renderizzano solo le parti del DOM effettivamente modificate.
- Performance eccellenti.

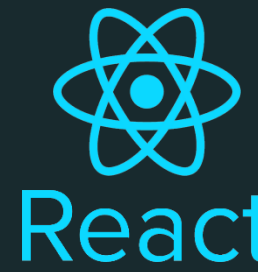




Ciclo di vita di un componente

Ciclo di vita di un componente

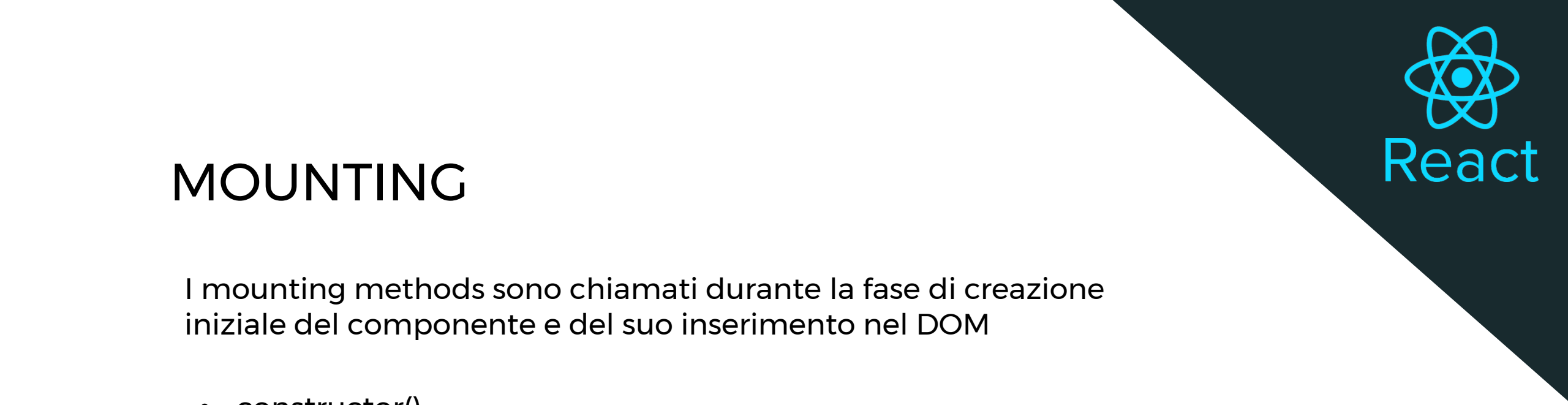
Dal *Mounting* all'*Unmounting*



Ad ogni componente vengono associati dei metodi che regolano il loro ciclo di vita. Ovviamente su questi metodi è possibile eseguire l'Override per poterne personalizzare il loro comportamento.

Tutti i metodi che regolano il ciclo di vita di un componente possono essere racchiusi in queste quattro categorie:

1) Mounting 2) Updating 3) Unmounting 4) Error Handling



MOUNTING

I mounting methods sono chiamati durante la fase di creazione iniziale del componente e del suo inserimento nel DOM

- constructor()
- getDerivedStateFromProps()
- render()
- componentDidMount()

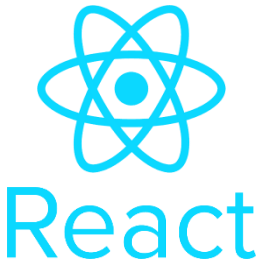
Constructor

```
Constructor(props) {
  super(props) ;
  this.state = {props : 0};
  this.handleClick = this.handleClick.bind(this) ;
}
```

Il metodo constructor()

Viene chiamato prima che il componente venga montato

- La prima cosa da fare è inizializzare la classe React.component attraverso il super(props)
- Tipicamente nei costruttori di React bisognerebbe fare esclusivamente due cose:
  - Inizializzare lo stato locale (solamente nel costruttore si può cambiare lo stato senza usufruire del metodo setState())
  - Fare il binding con gli event handler per poterli poi assegnare ai React element



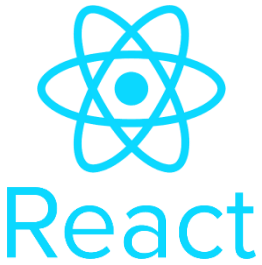
Render

```
render() {
  return(
    <button onClick = {this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
    </button>
  ) ;
}
```

Il metodo render()

È l'unico metodo obbligatorio richiesto in una class component

- Questo metodo renderizza i React elements tramite JSX e indica a React come renderizzarli in nodi del DOM
- Dovrebbe essere una funzione "pura" in cui si renderizzano gli elementi e nient'altro (nessuna chiamata al backend o cambiamento di stato).



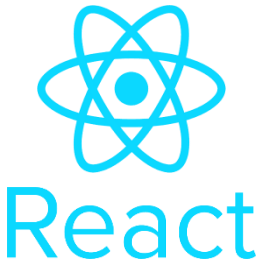
ComponentDidMount

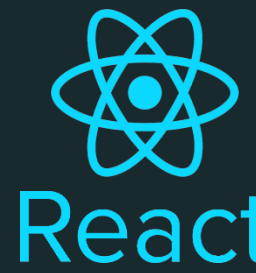
```
getData(){
  setTimeout(() => {
    console.log('Our data is fetched');
    this.setState({
      data: 'Hello WallStreet'
    })
  }, 1000)
}

componentDidMount(){
  this.getData();
}
```

Il metodo componentDidMount()

È l'ultimo metodo chiamato dei mounting methods

- Viene invocato immediatamente dopo che il componente viene montato
- Indicato per aggiornare lo stato (con il setState()) controllando il props o facendo una chiamata al backend
- Solo alla fine del componentDidMount il componente apparirà sulla UI del browser





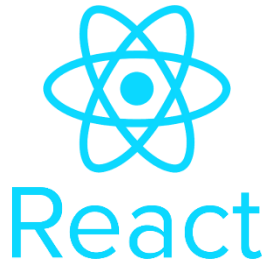
UPDATING

I metodi update vengono usati per aggiornare i component già montati e renderizzati nel DOM. Tipicamente vengono usati per cambiare lo state del component.
Sono chiamati in quest'ordine quando viene effettuato un nuovo render del component a seguito di un aggiornamento:

- getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- componentDidUpdate()

componentDidUpdate()

Metodo di aggiornamento del componente



Viene invocato immediatamente dopo che il componente viene montato.
Indicato per aggiornare lo stato (con il setState()) controllando il props o facendo una chiamata al backend.
Solo alla fine del componentDidMount il componente apparirà sulla UI del browser

```
componentDidUpdate(prevProps) {
  // Typical usage (don't forget to compare props):
  if (this.props.userID !== prevProps.userID) {
    this.fetchData(this.props.userID);
  }
}
```

Ciclo di vita di un componente

Unmounting e Error Handling

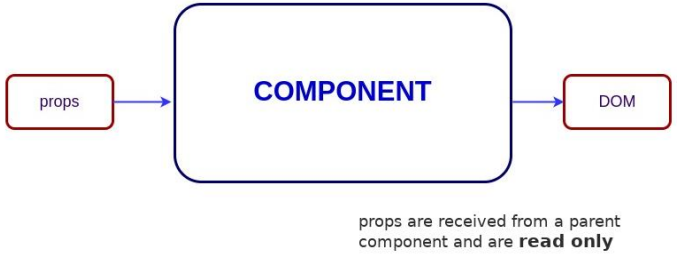
Sono chiamati quando un component viene rimosso dal DOM e quando avviene un errore durante la fase di rendering del component

componentWillUnmount()

componentDidCatch()

LO STATO DEI COMPONENTI

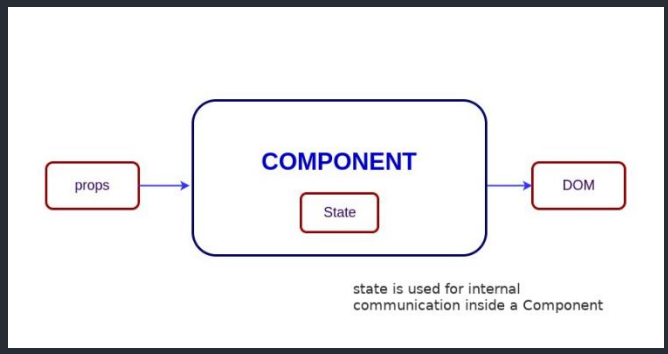
PROPS



Il props è lo stato proveniente dal component padre, cioè il component dal quale viene istanziato. Il props è read only ed in genere viene usato per far ereditare lo stato interno del componente.

```
const Button = (props) => {
    // props are read only
    props.count =21;
.
.
}
```

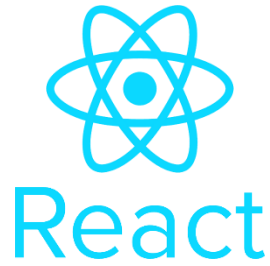
STATE



Lo state è un oggetto che appartiene unicamente a quel singolo component e non viene passato a nessun altro.

```
constructor(props) {
  super(props);
  this.state = {
    posts: [],
    comments: []
  };
}
```

AGGIORNAMENTO DELLO STATO



Inizializzazione dello stato:

```
constructor(props) {
  super(props);
  this.state = {
    posts: [],
    comments: []
  };
}
```

Set dello stato:

```
// Wrong
this.state.comment = 'Hello';
```

```
// Correct
this.setState({comment: 'Hello'});
```

Aggiornamento dello stato (asincrona):

```
// Wrong
this.setState({
  counter: this.state.counter + this.props.increment,
});
```

```
// Correct
this.setState((prevState, props) => ({
  counter: prevState.counter + props.increment
}));
```

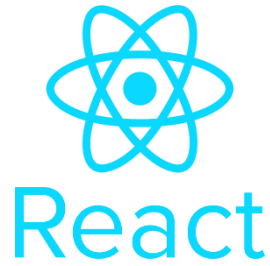
Merge degli stati:

```
componentDidMount() {
  fetchPosts().then(response => {
    this.setState({
      posts: response.posts
    });
  });

  fetchComments().then(response => {
    this.setState({
      comments: response.comments
    });
  });
}
```

GESTIONE DEGLI EVENTI

Event handling

Modificare il comportamento dei componenti



E' possibile personalizzare il comportamento dei React elements in relazione a determinati eventi esterni.
L'assegnazione degli handler risulta essere molto simile a quella di HTML.

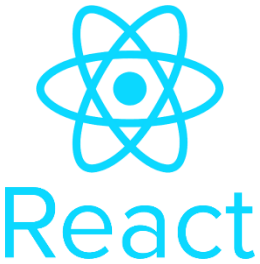
Non è necessario chiamare la addEventListener per aggiungere un listener ad un elemento del DOM. Bisogna solamente fornire una funzione listener (sfruttando il JSX) quando un elemento viene inizialmente renderizzato.

```
<button onclick="activateLasers()">
  Activate Lasers
</button>
```

```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

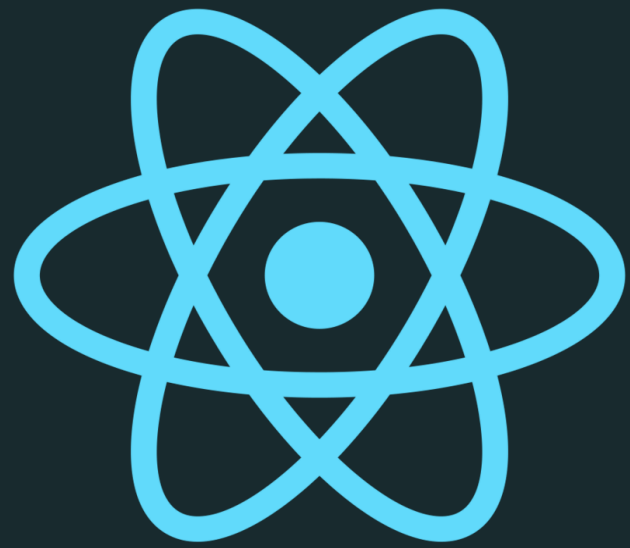
Event handling

Modificare il comportamento dei componenti




```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};

    // This binding is necessary to make `this` work in the callback
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(prevState => ({
      isToggleOn: !prevState.isToggleOn
    }));
  }

  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}

ReactDOM.render(
  <Toggle />,
  document.getElementById('root')
);
```

Quando definiamo un componente usando le classi di javascript ES6, un pattern comune è quello di settare come event handler un metodo della classe.
Ad esempio, questo Toggle component renderizza un bottone che lascia decidere all'utente, cliccando, se settare ON oppure OFF.



React vs AngularJS

1

Cos’è React?

React è una tecnologia di Facebook, in particolare è una libreria.

2

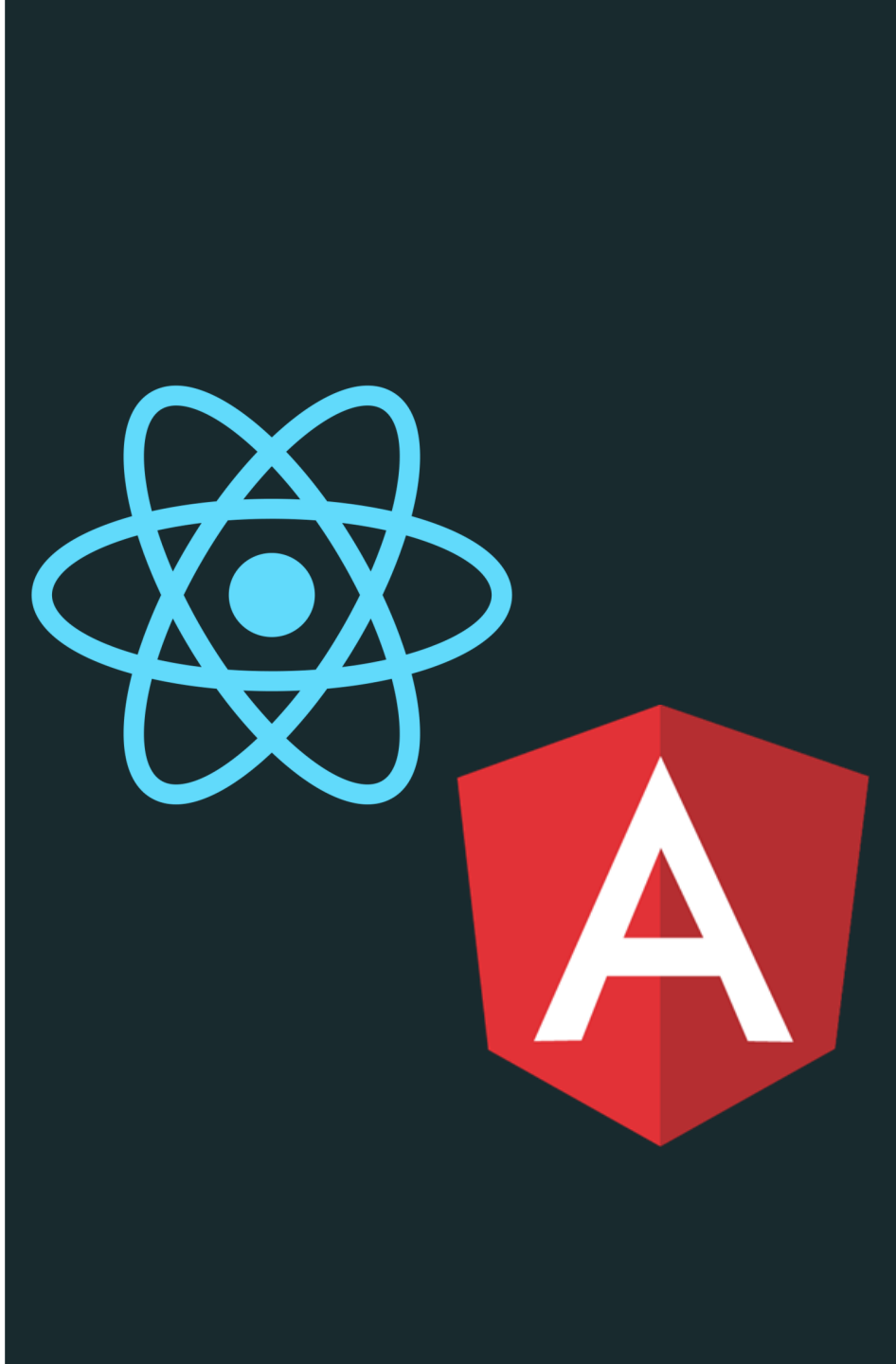
Cos'è AngularJS?

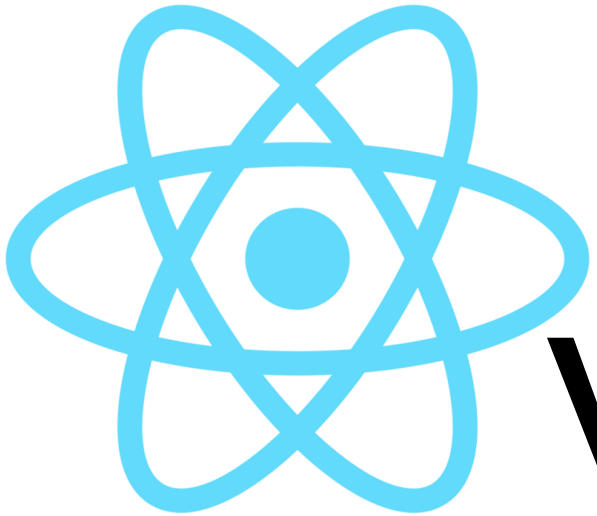
Com'è ben noto AngularJS è un è un framework JavaScript per lo sviluppo di applicazioni Web client side, nato in casa Google

3

Quale scegliere?

La scelta è demandata all’approccio di sviluppo che lo sviluppatore preferisce.





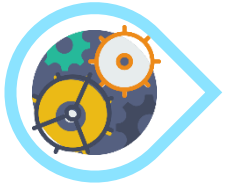
VS

Confronto delle caratteristiche

Punti di forza e debolezze

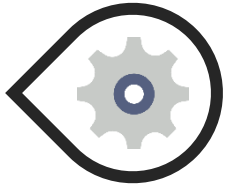
1 **Architettura**
Component VS AngularJS

DOM
Document Object Model 2



3 **Curva d’apprendimento**
React più veloce da apprendere

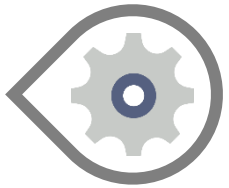
Dimensioni
Data package dimensions 4



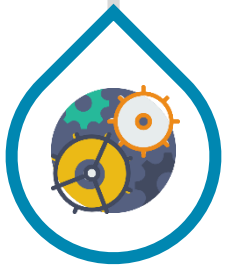
5 **V vs MVC**
View vs Model View Control

Data binding
1-Way vs 2-Way

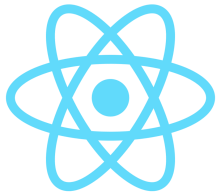
6



7 **Dependency**
ReactDI vs DI



Rendering
Client VS Server side



React

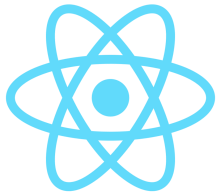
Architettura

- In React scriviamo applicazioni in termini di “Componenti”.
- Un Componente troppo grande può essere diviso in componenti più piccoli.
- È responsabile di renderizzare il contenuto dell’elemento e di gestire gli eventi in esso presenti.
- **Un componente può inoltre essere combinato con altri componenti.** Può essere innestato all’interno di un altro componente o avere altri componenti figli al suo interno.

AngularJS

Architettura

- Angular è un framework e offre molte funzionalità predefinite.
- Dependency Injection
- Two-Way-Binding
- Supporto MVC
- Angular propone una strutturazione modulare composta da componenti che hanno ciascuno una ben precisa funzione(View,Controller,Filtri,Direttive,Servizi ecc..)



React

Virtual DOM

- Il Virtual DOM è un albero logico che rappresenta la struttura DOM effettiva. In altre parole, React mantiene una sorta di "copia" del DOM in memoria consumando memoria.
- Come già detto il Virtual DOM si occupa di applicare le giuste modifiche per aggiornare la struttura dell' applicazione.
- React determinerà cosa cambiare dopo ogni evento, modificherà il Virtual DOM e poi eseguirà le operazioni più veloci per regolare di conseguenza il DOM reale.
- La velocità dipende dalla struttura dell'applicazione.

AngularJS

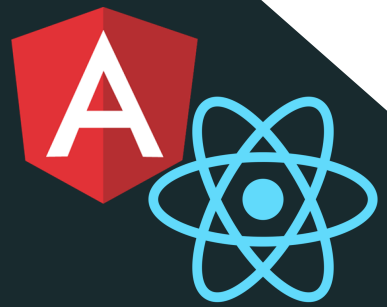
Regular DOM

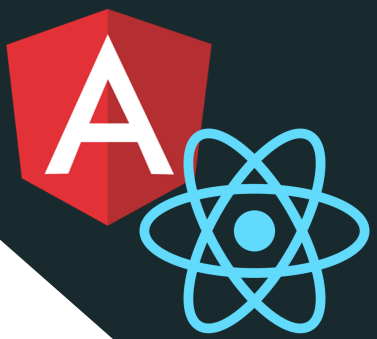
- AngularJs non utilizza un DOM virtuale
- utilizza il proprio meccanismo per Change Detection.
- Tale meccanismo aiuta Angular a passare attraverso l'albero di Change Detection dalla sua radice alle sue foglie.

Curva di apprendimento

Padroneggiare React o AngularJS

- Il tempo che occorre per raggiungere un buon livello di conoscenza della piattaforma React è più breve rispetto a quello richiesto da AngularJs.
- L'uso di JSX permette di utilizzare il codice HTML insieme al codice JavaScript rendendo più semplice la comprensione dei Component.
- Per usare React basta conoscere JavaScript e HTML.
- Il Framework mette a disposizione più strumenti a discapito della facilità d'uso.
- AngularJs può contare si una comunità più robusta.

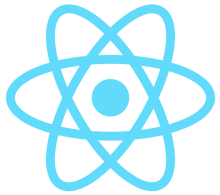




Dimensioni

Data package dimensions

- E' stato effettuato un test in cui vengono misurate le dimensioni dei pacchetti trasferiti al browser.
- Intestazione e corpo della risposta.
- Il risultato del test ha dimostrato che il pacchetto dati di React inviato è più leggero di quello Angular.



React

View

- Nasce per gestire la parte relativa alle view,ossia alla presentazione, e all'intercettazione degli eventi di input dell'utente.
- Rispetto alla "separazione dei compiti" React opta per una "separazione di interessi".
- React tende a isolare codice e stato dei componenti in base al loro ambito di utilizzo, rendendoli il più possibile autonomi e riutilizzabili in un medesimo contesto.
- L'obiettivo di avere un codice distinto per compito è comunque raggiungibile adottando alcune *best practices* e *pattern* di sviluppo.

AngularJS

MVC

- Il framework supporta il pattern Model View Controller.
- L'associazione tra una view e un controller avviene sull'HTML tramite la direttiva ng-controller.
- Lo scope è consente la definizione del modello dei dati e la sua esposizione alla view.
- Consente separazione delle competenze.

React

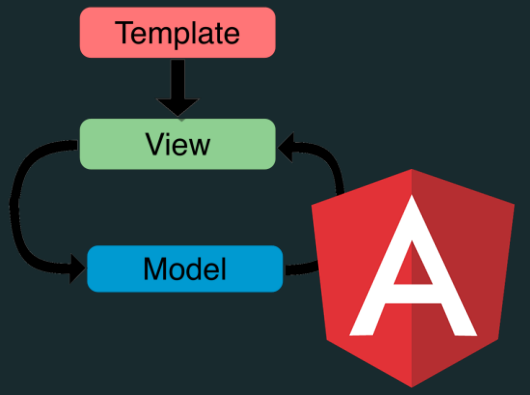
One-Way Binding

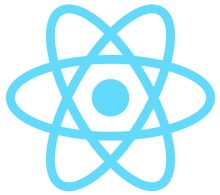
- Il Virtual DOM è un albero logico che rappresenta la struttura DOM effettiva. In altre parole, React mantiene una sorta di "copia" del DOM in memoria consumando memoria.
- Come già detto il Virtual DOM si occupa di applicare le giuste modifiche per aggiornare la struttura dell' applicazione.
- React determinerà cosa cambiare dopo ogni evento, modificherà il Virtual DOM e poi eseguirà le operazioni più veloci per regolare di conseguenza il DOM reale.
- La velocità dipende dalla struttura dell'applicazione.

AngularJS

Two-way-Binding

- Meccanismo di sincronizzazione automatica dei dati tra il modello e la view.
- Il data binding di AngularJS invece è bidirezionale (two-way binding), cioè ogni modifica al modello dei dati si riflette automaticamente sulla view e ogni modifica alla view viene riportata sul modello dei dati.





React

ReactDI

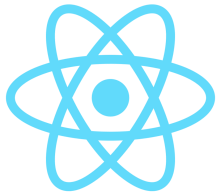
- Reagire per impostazione predefinita non distribuisce DI.
- In React, potremmo includere una libreria speciale chiamata "ReactDI" per implementare DI.
- Tuttavia, per creare componenti riutilizzabili, ogni volta che un componente dipende da una funzione, tale funzione viene passata come "prop".

AngularJS

Dependency Injection

- Il framework consente di dichiarare che all'interno di un componente verrà utilizzato un altro componente

- Dependency Injection(DI) è una caratteristica intrinseca esclusiva di Angular e viene utilizzata per condividere servizi tra i componenti.



React

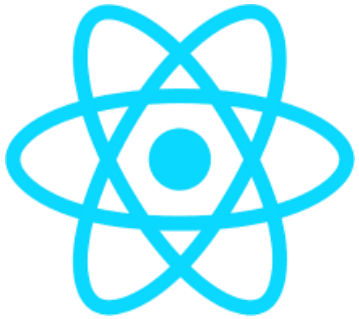
Rendering server side

- React effettua il render delle pagine lato server inviando così al browser l'HTML, che rappresenta lo stato iniziale dell'applicazione.
- Ciò comporta un ridotto carico di lavoro da parte del Browser ed al tempo stesso che l'applicativo sia SEO friendly.
- SEO friendly: il rendering lato server aiuta i crawler dei motori di ricerca a trovare i contenuti.
- Il miglioramento è percepito nel caso in cui il JavaScript sia molto grande.
- In questo caso il Browser può cominciare a mostrare l'HTML mentre scarica ed elabora il JavaScript.

AngularJS

Rendering client-side

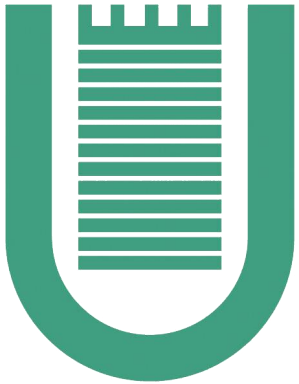
- AngularJs effettua il rendering delle pagina lato client.
- Il server invia i dati all'interno di file JSON.
- Meno carico di lavoro per il server.
- La quantità di traffico tra client e server è ridotta.
- Il Browser deve eseguire il lavoro extra di manipolazione del DOM ed eseguire richieste REST aggiuntive.
- I computer moderni e dispositivi mobili riesco a reggere tale carico.



Hands-on React

Dal boilerplate alle chiamte HTTP

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



UNIVERSITA' degli STUDI di ROMA
TOR VERGATA

Grazie per l'attenzione

Gruppo 3