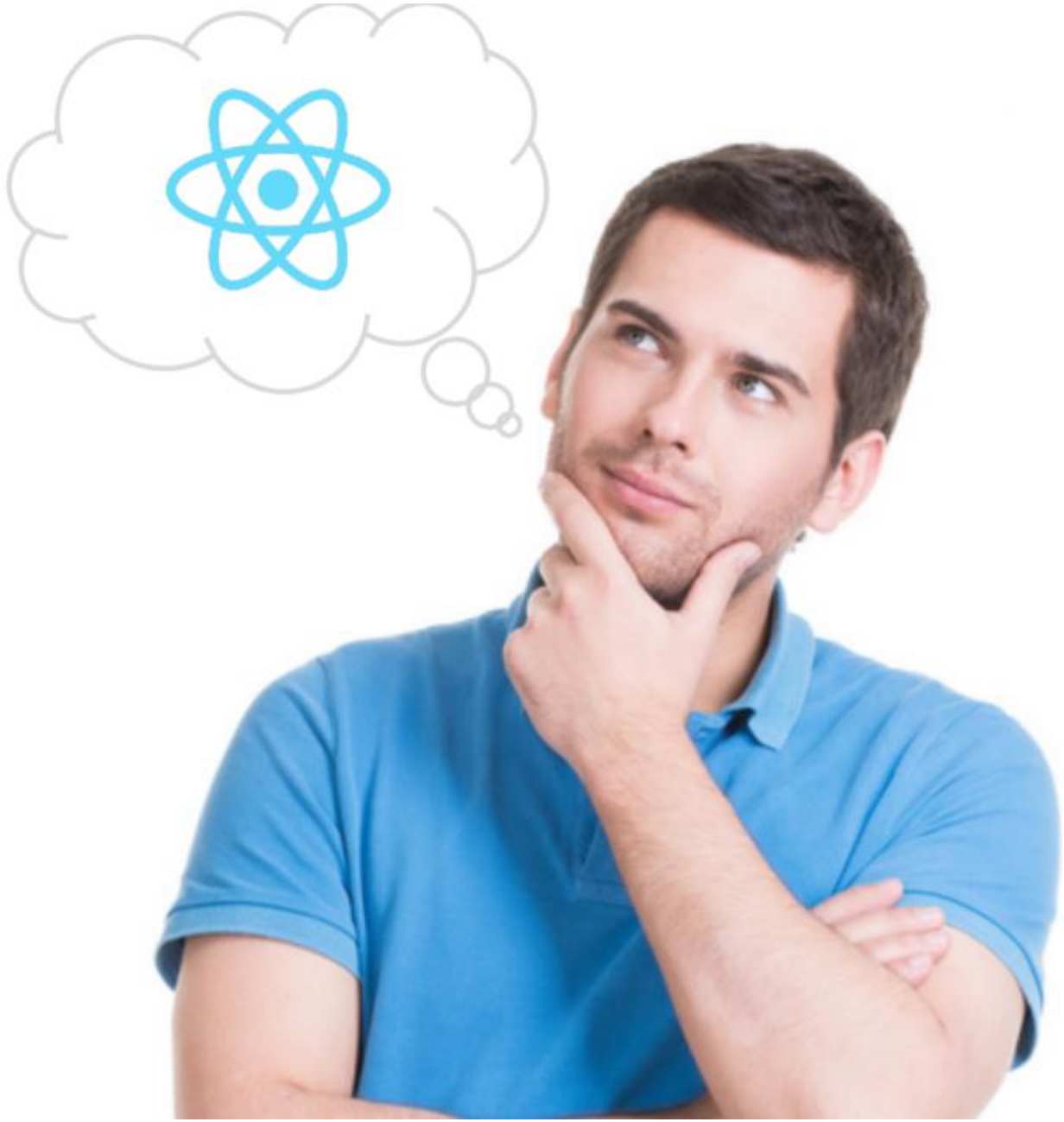
Why React?



History

2011	Created by Facebook
2012	Used on Instagram
2013	Open sourced
2014	Embraced by many large companies
2015	React Native released
2016	React 15 released (previous version was 0.14)
Today	Over 50k components at Facebook Full-time dev staff Used by many in Fortune 500

Where Can I Use React?

Web apps

Static sites

Mobile

Desktop

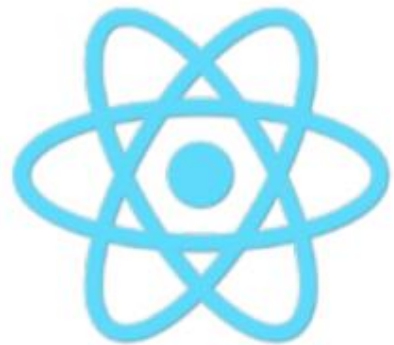
Server-rendered

Virtual Reality

Framework

Clear opinions
Less decision fatigue
Less setup overhead
More cross-team consistency

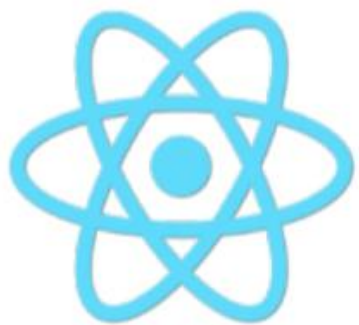
Library



Light-weight
Use on existing apps
Pick what you need
Choose best tech
Popular boilerplates exist

"JS" in HTML

```
<div *ngFor="let user of users">
<div v-for="user in users">
{{#each user in users}}
```



"HTML" in JS

```
{users.map(createUser)}
```

Conditional

```
<h1 *ngIf="isAdmin">Hi Admin</h1>
```

```
<h1 v-if="isAdmin">Hi Admin</h1>
```


```
<h1>{{if isAdmin 'Hi Admin'}}</h1>
```

```
{isAdmin && <h1>Hi Admin</h1>}
```

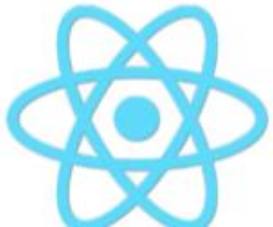
Since plain JS, you get:
1. Autocomplete support
2. Error messages

Loop

```
<div *ngFor="let user of users">{{user.name}}</div>
```

```
<div v-for="user in users">{{user.name}}</div>
```


```
{{#each users as |user|}}
<div>{{user.name}}</div>
{{/each}}
```



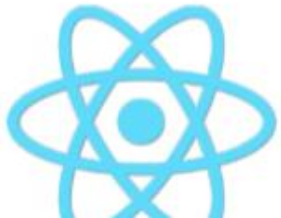
```
users.map(user => <div>{user.name}</div>)
```

Event

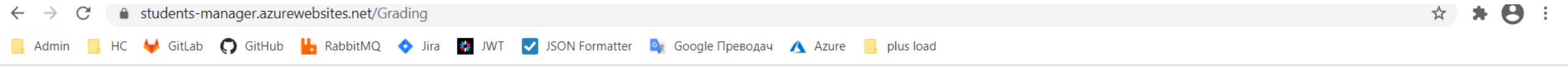
```
<button (click)="delete()">Delete</button>
```

```
<button v-on:click="delete">Delete</button>
```


```
<button onclick={{action 'delete'}}>Delete</button>
```



```
<button onClick={delete}>Delete</button>
```



Начало

Текущ Контрол

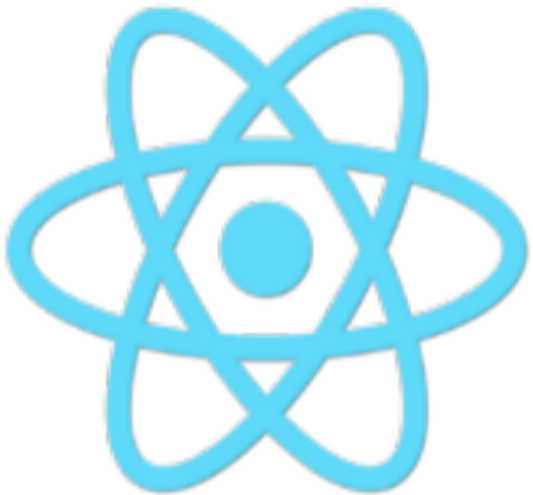
Формиране

Име	Фак №	Въведение в JS	Дом #1	Advance JS	Дом #2	Async JS	Дом #3	jQuery	Дом #4	Cordova	Дом #5	Onsen UI	Дом #6	React	Дом #7	Конт #1
Асистент Йорданов	JJJJJJJJJ	1 т.	Предадено	1 т.	Предадено	1 т.	Предадено	0.5 т.	Предадено	0.5 т.	Предадено	0.5 т.	Предадено	1 т.	Предадено	3 т.

Икономически Университет - Варна

Replace Grading page with React

Why Virtual DOM?



Updating the DOM is expensive

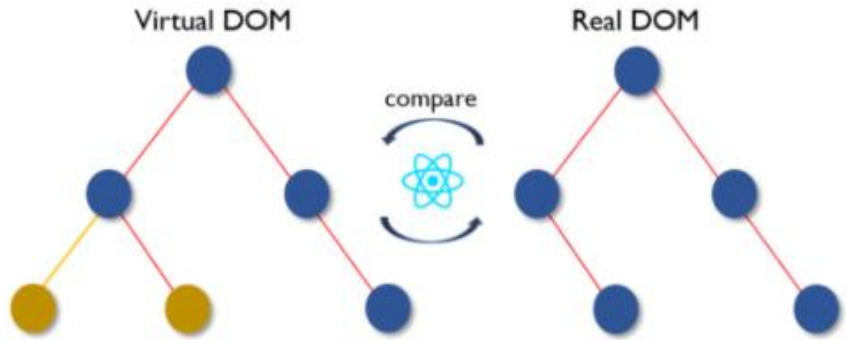
So React minimizes DOM changes

Real DOM	Virtual DOM
1. It updates slow.	1. It updates faster.
2. Can directly update HTML.	2. Can't directly update HTML.
3. Creates a new DOM if element updates.	3. Updates the JSX if element updates.
4. DOM manipulation is very expensive.	4. DOM manipulation is very easy.
5. Too much of memory wastage.	5. No memory wastage.

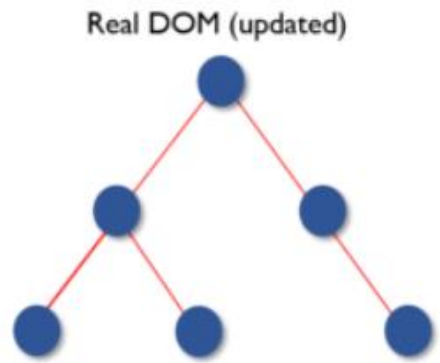
This Virtual DOM works in three simple steps.

1. Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.

2. Then the difference between the previous DOM representation and the new one is calculated.

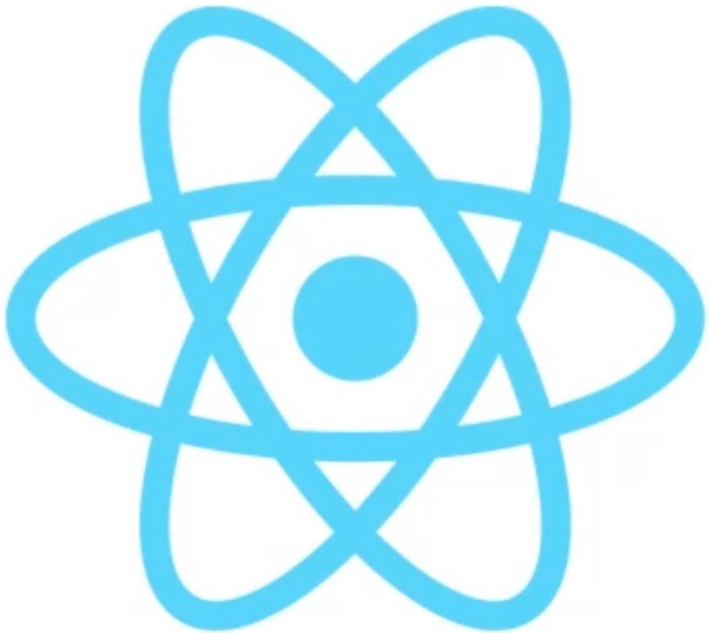


3. Once the calculations are done, the real DOM will be updated with only the things that have actually changed.



Features

	React	Angular
Components	✓	✓
Testing	Jest, Mocha	✓
HTTP library	Fetch, Axios	✓
Routing	React Router	✓
I18n	react-intl	✓
Animation	react-motion	✓
Form validation	react-forms	✓
CLI	create-react-app	angular-cli



1) Components

- Like functions
- Input: props, state | Output: UI
- Reusable and composable
- <Component />
- Can manage a private state

2) Reactive updates

- React will react
- Take updates to the browser

3) Virtual views in memory

- Generate HTML using JavaScript
- No HTML template language
- Tree reconciliation

Two-way binding

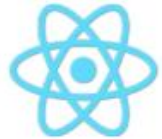
Less coding

Automatic

```
let user = 'Cory';

<input
  type="text"
  value={user}
/>
```

"Magically" kept in-sync

One-way binding



More control

More explicit

Easy to debug

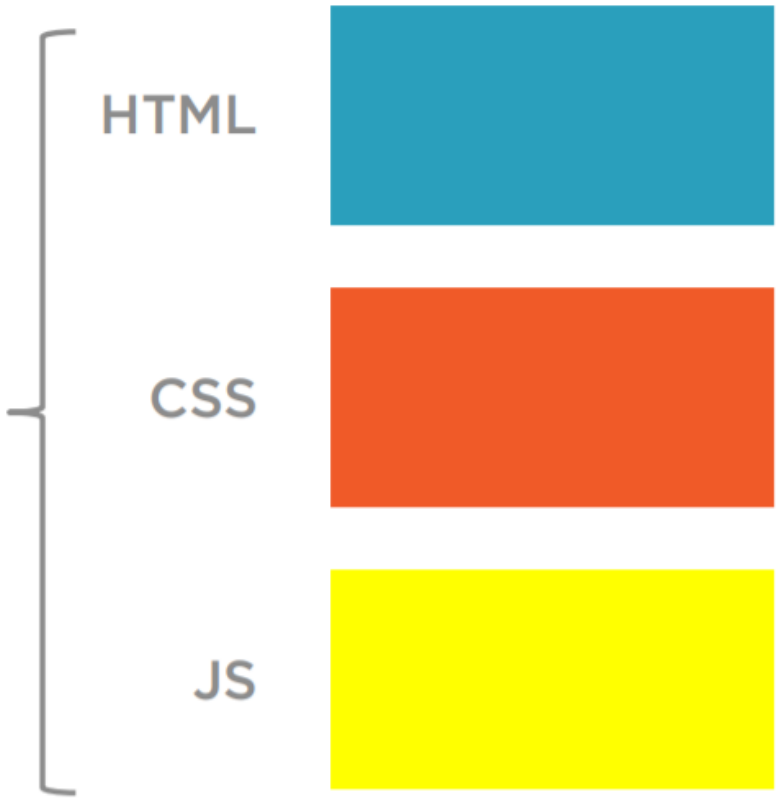
```
state = { user: 'Cory' };

function handleChange(event) {
  this.setState({
    user: event.target.value
  });
}

<input
  type="text"
  value={this.state.user}
  onChange={this.handleChange}
/>
```

Explicit change handler

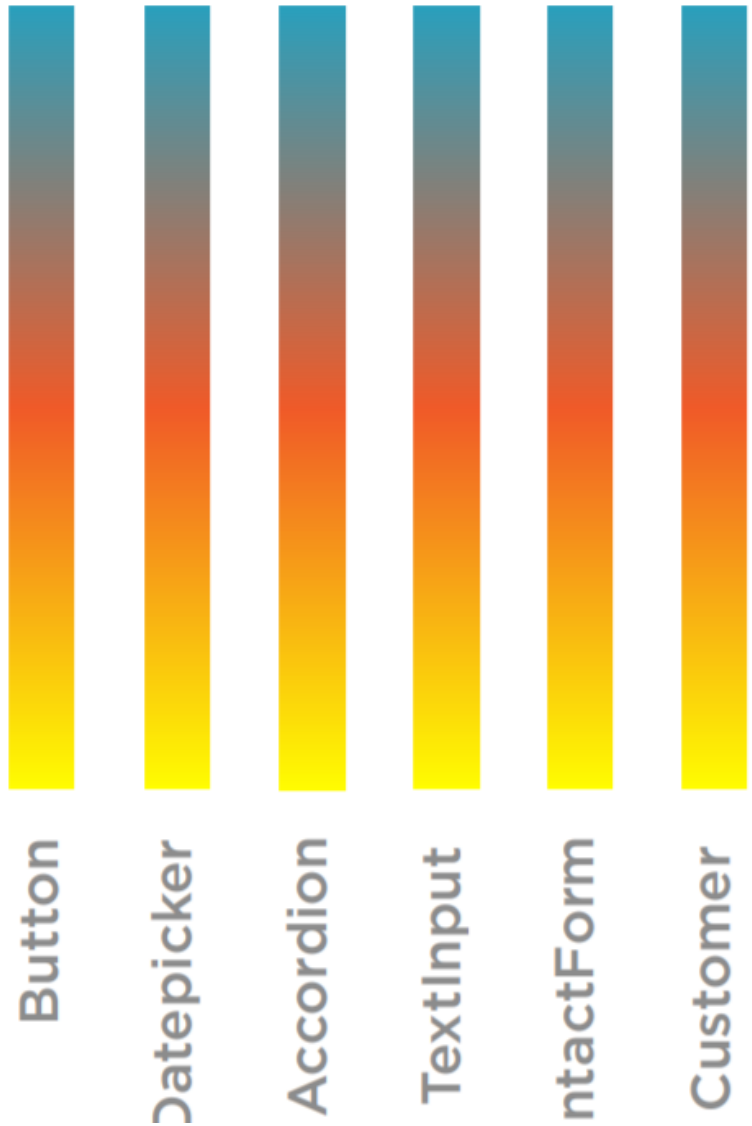
Conditions	State	Props
1. Receive initial value from parent component	Yes	Yes
2. Parent component can change value	No	Yes
3. Set default values inside component	Yes	Yes
4. Changes inside component	Yes	No
5. Set initial value for child components	Yes	Yes
6. Changes inside child components	No	Yes

Stateful Component	Stateless Component
1. Stores info about component's state change in memory	1. Calculates the internal state of the components
2. Have authority to change state	2. Do not have the authority to change state
3. Contains the knowledge of past, current and possible future changes in state	3. Contains no knowledge of past, current and possible future state changes
4. Stateless components notify them about the requirement of the state change, then they send down the props to them.	4. They receive the props from the Stateful components and treat them as callback functions.

Separate technologies, but *intertwined* concerns.



Each *component* is a separate concern.



Playground

- https://jscomplete.com/playground/rgs1.1
- https://jscomplete.com/playground/rgs1.2
- https://jscomplete.com/playground/rgs1.4
- https://jscomplete.com/playground/rgs1.3
- https://jscomplete.com/playground/rgs1.5
- https://jscomplete.com/playground/rgs1.6
- https://jscomplete.com/playground/rgs1.8
- https://jscomplete.com/playground/scopes
- https://jscomplete.com/playground/arrow-vs-regular-functions
- https://jscomplete.com/playground/object-literals
- https://jscomplete.com/playground/destructuring
- https://jscomplete.com/playground/template-strings
- https://jscomplete.com/playground/classes
- https://jscomplete.com/playground/promises