INFINUM

Composing views with React

KRISTIJAN BAMBIR

REACT AT INFINUM

- First used in a project around a year and half ago
- Several projects have been done with React since
- Scalable, flexible, performant

THE BEGINNING

- MVC, MVVM, MV*
- Two-directional data binding
- Mutations

A JavaScript LIBRARY for building user interfaces

BASICS

- Not a “full” framework (like Backbone or Angular)
- Technology stack agnostic
- Component based
- All logic is written in JavaScript
- Can be rendered on the server
- Unidirectional (one-way) data binding
- Virtual DOM

HISTORY

- Created by Jordan Walke, software engineer at Facebook
- First deployed to Facebook’s newsfeed in 2011
- Open-sourced in May 2013

KEY CONCEPTS

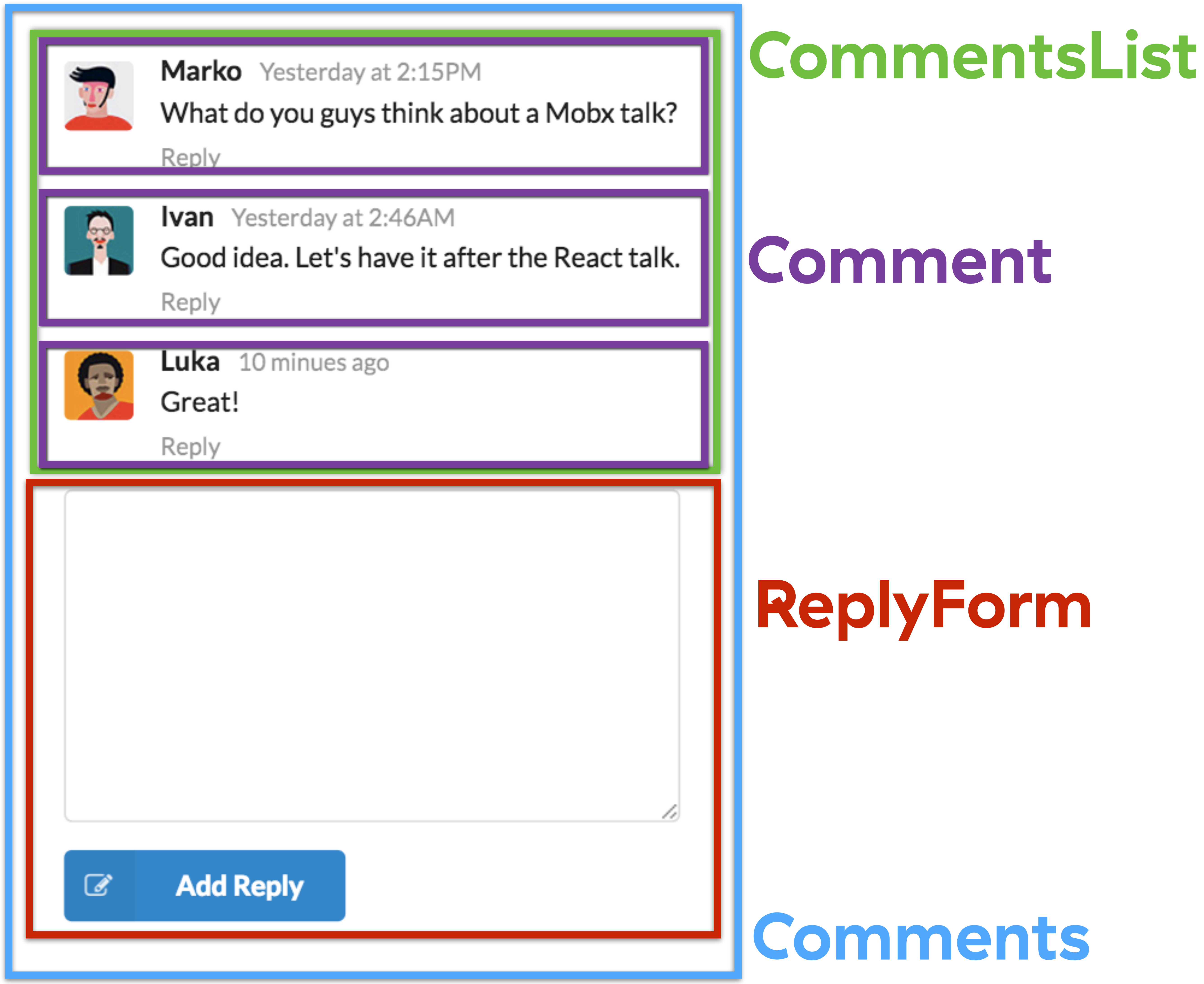
1. Components
2. JSX
3. Props & State
4. Component API
5. Component types

01

COMPONENTS

A COMPONENT

- Self-contained element containing the HTML output and all the logic needed to control that output
- Components split up the UI into independent, reusable pieces
- An application is big components calling smaller components



Marko Yesterday at 2:15PM
What do you guys think about a Mobx talk?
Reply

Ivan Yesterday at 2:46AM
Good idea. Let's have it after the React talk.
Reply

Luka 10 minues ago
Great!
Reply

02

JSX

JSX(ML)

- Both HTML and JavaScript live in the same file
- JSX makes this alliance possible
- Syntax extension to JavaScript
- Syntactic sugar over JavaScript functions
- Not natively supported (transpiler, like Babel)
- Produces React elements

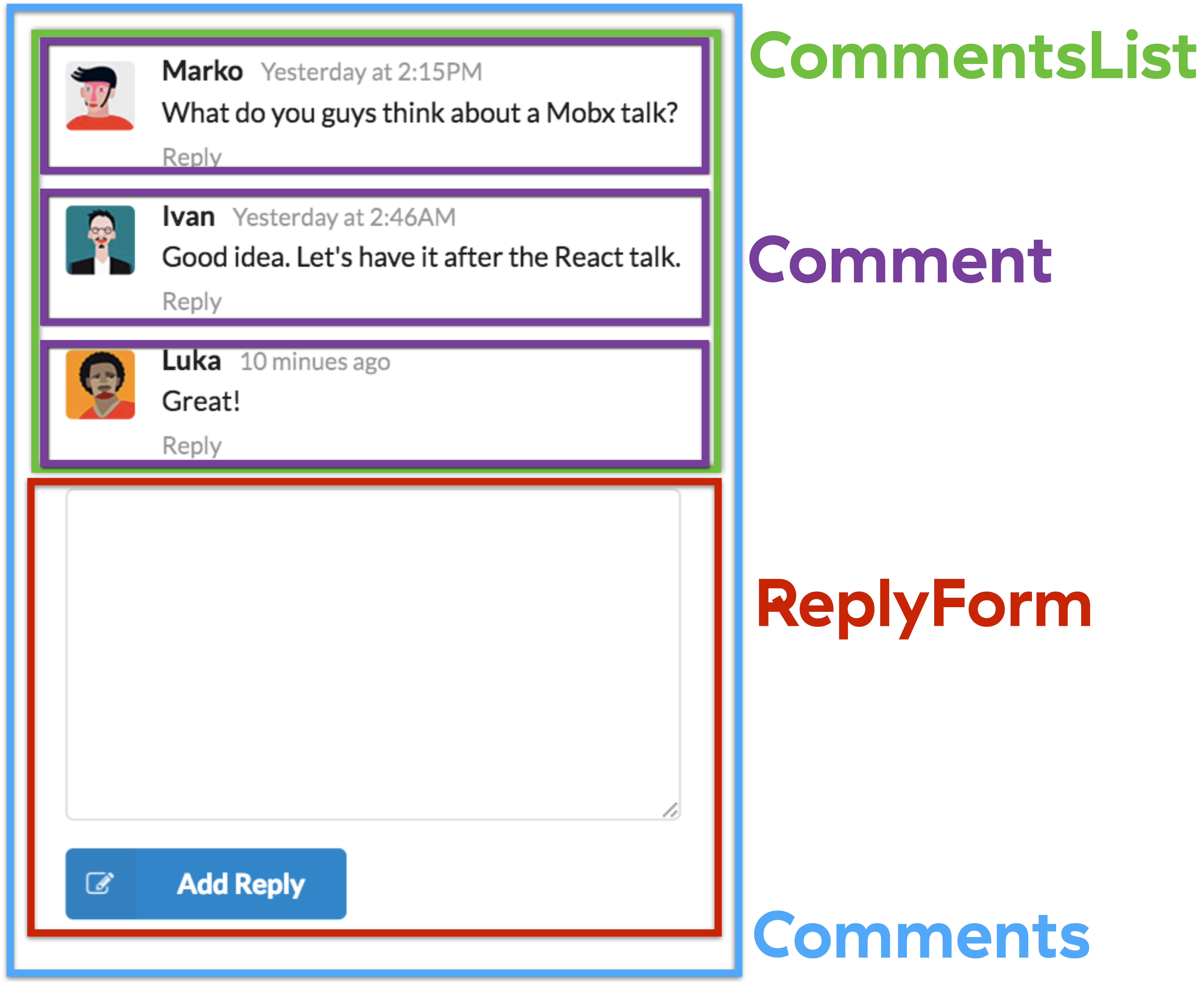
```
// JSX
const element = <h1>Hello, World!</h1>


// Compiled JS
const element = React.createElement('h1', null, 'Hello, World!');
```

```
function formatDate(date) {
  return date.toDateString();
}

const element = (
  <p>Current date is {formatDate(new Date())}</p>
);
```

```
const element = <textarea rows='5' />

const element = <a href={post.url} />
const element = (
  <div>
    <h1>JS talks</h1>
    <p>Interesting talks and mini lectures about JS development.</p>
  </div>
);
```

```
// Doesn’t compile!
const element = (
  <h1>JS talks</h1>
  <p>Interesting talks and mini lectures about JS development.</p>
);
```

```
const element = (
  <Comments>
    <CommentsList>
      <Comment />
      <Comment />
      <Comment />
    </CommentsList>
    <ReplyForm />
  </Comments>
);
```



Marko Yesterday at 2:15PM
What do you guys think about a Mobx talk?
Reply

Ivan Yesterday at 2:46AM
Good idea. Let's have it after the React talk.
Reply

Luka 10 minues ago
Great!
Reply

03

PROPS AND STATE

PROPS

- Short for “properties”
- They are how components talk to each other
- They represent the unidirectional data flow
- Data can only go from parent to child components
- Are read–only (like const)

```
const element = (
  <Comment
    author={user.name}
    text={comment.text}
    date={comment.date}
  />
);
```

STATE

- Data that is created and lives inside of a component
- It is local and encapsulated
- Accessible only by the component that owns it
- User input usually changes a component state
- Components can be stateful or stateless

04

COMPONENT API

FUNCTIONAL COMPONENTS

- Literally functions
- Simplest component form

```
function Greeting(props) {
  return <h1>Hi, {props.name}!</h1>;
}


const element = <Greeting name='Kristijan' />
```

CLASS COMPONENTS

- Have additional features
- Must have at least the “render” method

```
class Greeting extends React.Component {

  render() {
    return <h1>Hi, {this.props.name}!</h1>;
  }

}
```

COMPONENT LIFECYCLE

Mounting

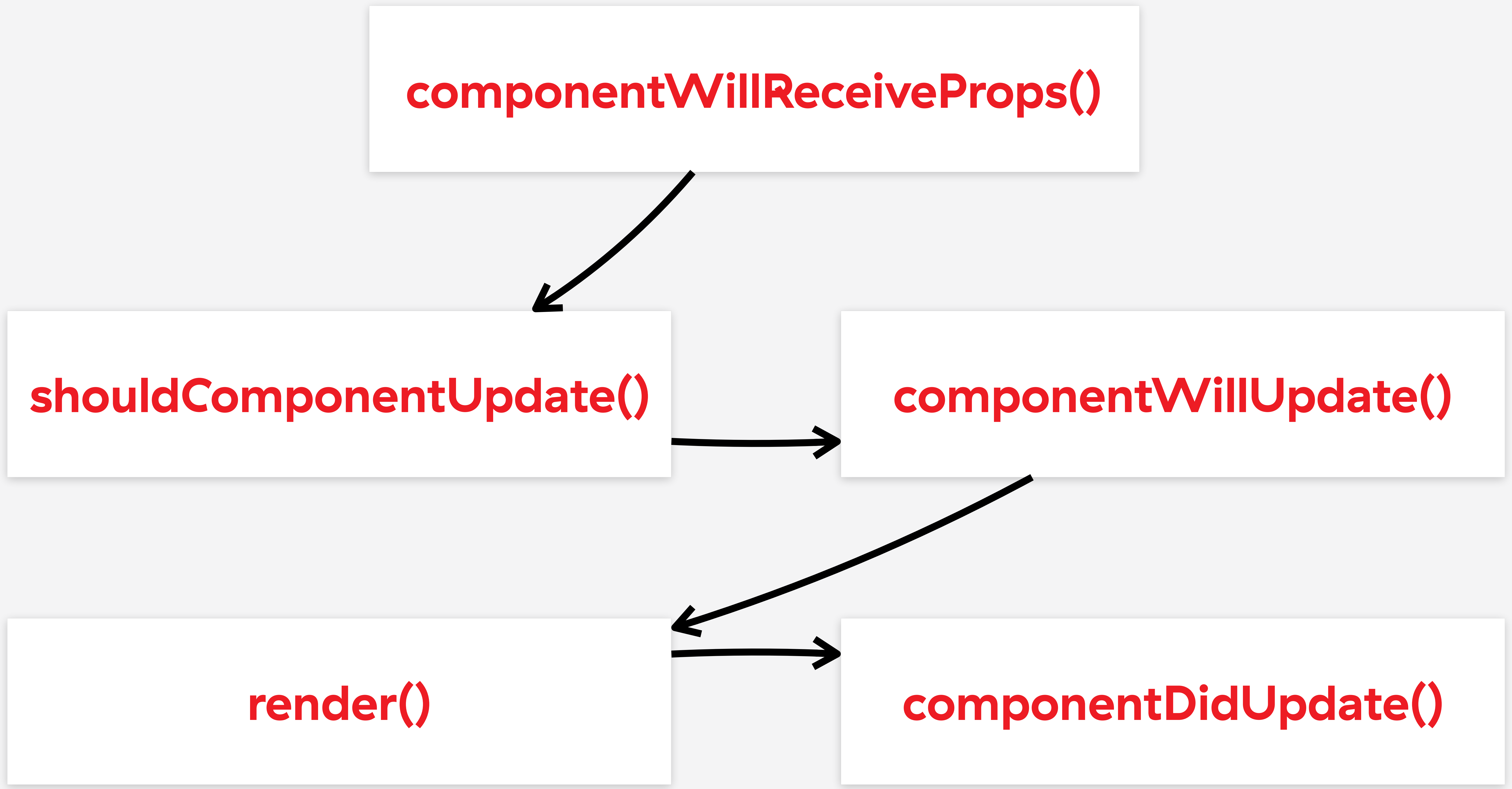
Updating

Unmounting

MOUNTING

$$\begin{tikzcd}
\text{constructor()} \arrow[r] & \text{componentWillMount()} \arrow[dl] \\
\text{render()} \arrow[r] & \text{componentDidMount()}
\end{tikzcd}$$

UPDATING


componentWillReceiveProps()
shouldComponentUpdate()
componentWillUpdate()
render()
componentDidUpdate()

UNMOUNTING

componentWillUnmount()

05

COMPONENT TYPES

COMPONENT ROLES

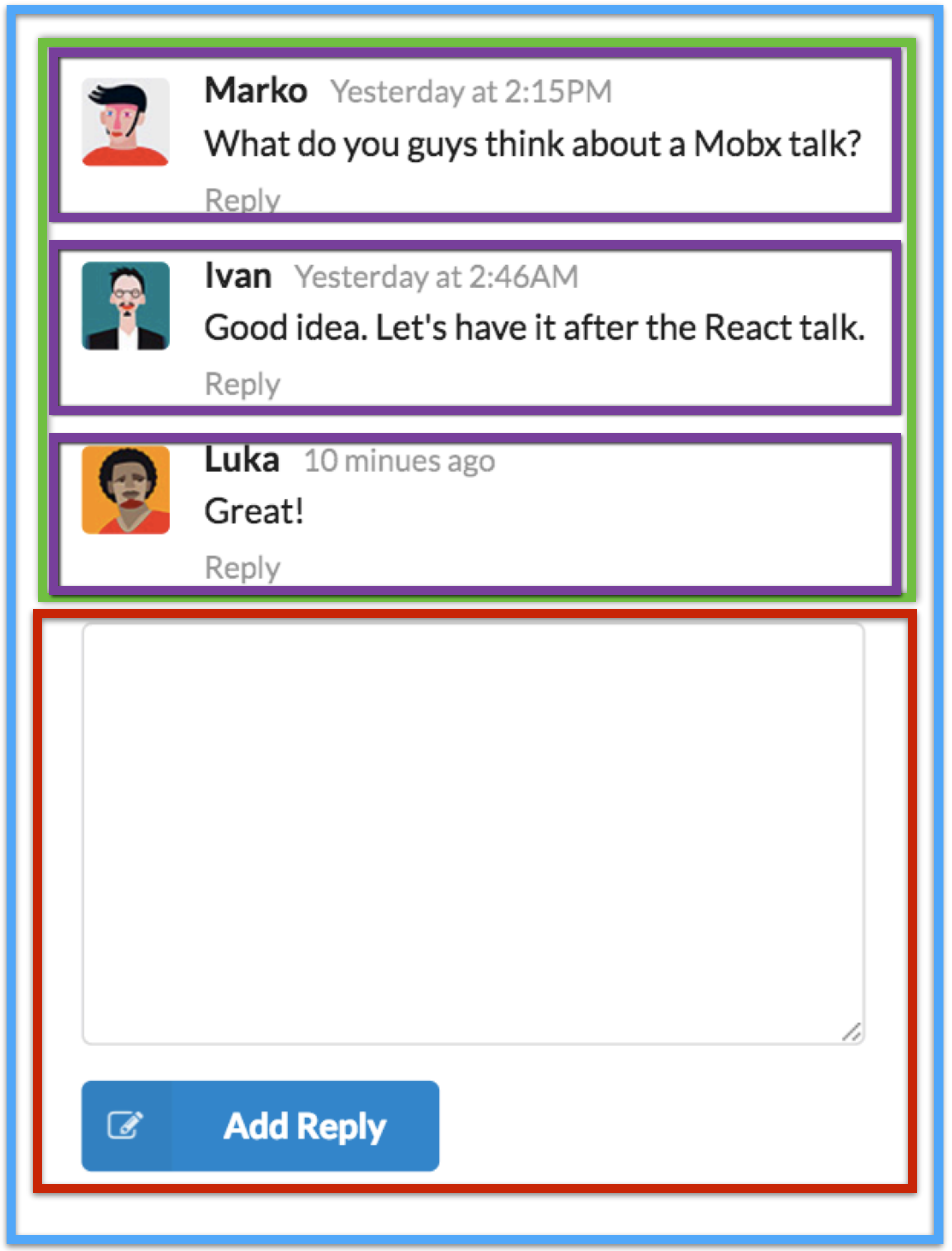
- Not all components serve the same purpose
- Two distinct flavours**:** **UI** and **data logic**
- These led to **container** and **presentational** components
- Or **dirty** and **reusable**

Container/
Presentational

Presentational

Presentational

Container



What do you guys think about a Mobx talk?

Good idea. Let's have it after the React talk.

Great!

CommentsList

Comment

ReplyForm

Comments

DEMO

https://github.com/kristijanbambir/my-blog

created using:

create-react-app

COMPONENT STATE

- Not all components can be stateless functional components
- Component state is not enough
- State containers such as Redux, MobX

MOBILE DEVELOPMENT?

- React Native!

CONCLUSION

- Server–side architectures are not what’s needed in the front–end
- Components works well
- They can be shared easily
- Scalable, flexible, performant
- Fun to write!

INFINUM

Thank you!

Visit infinum.co or find us on social networks:

infinum.co infinumco infinumco infinumco infinum