

Workshop: React + Flux

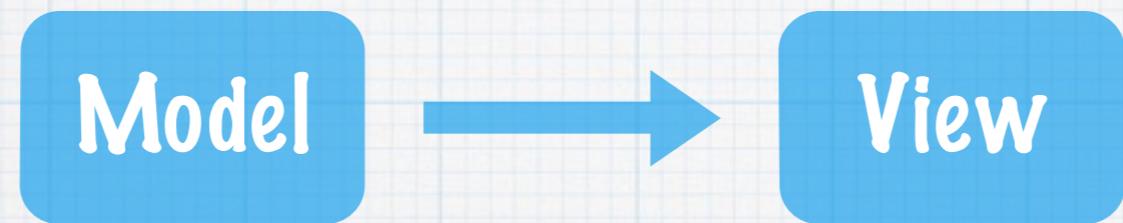
Tim Coppeters

React?

React = JavaScript **library** that provides **one-way**
data-binding from model to view.

React?

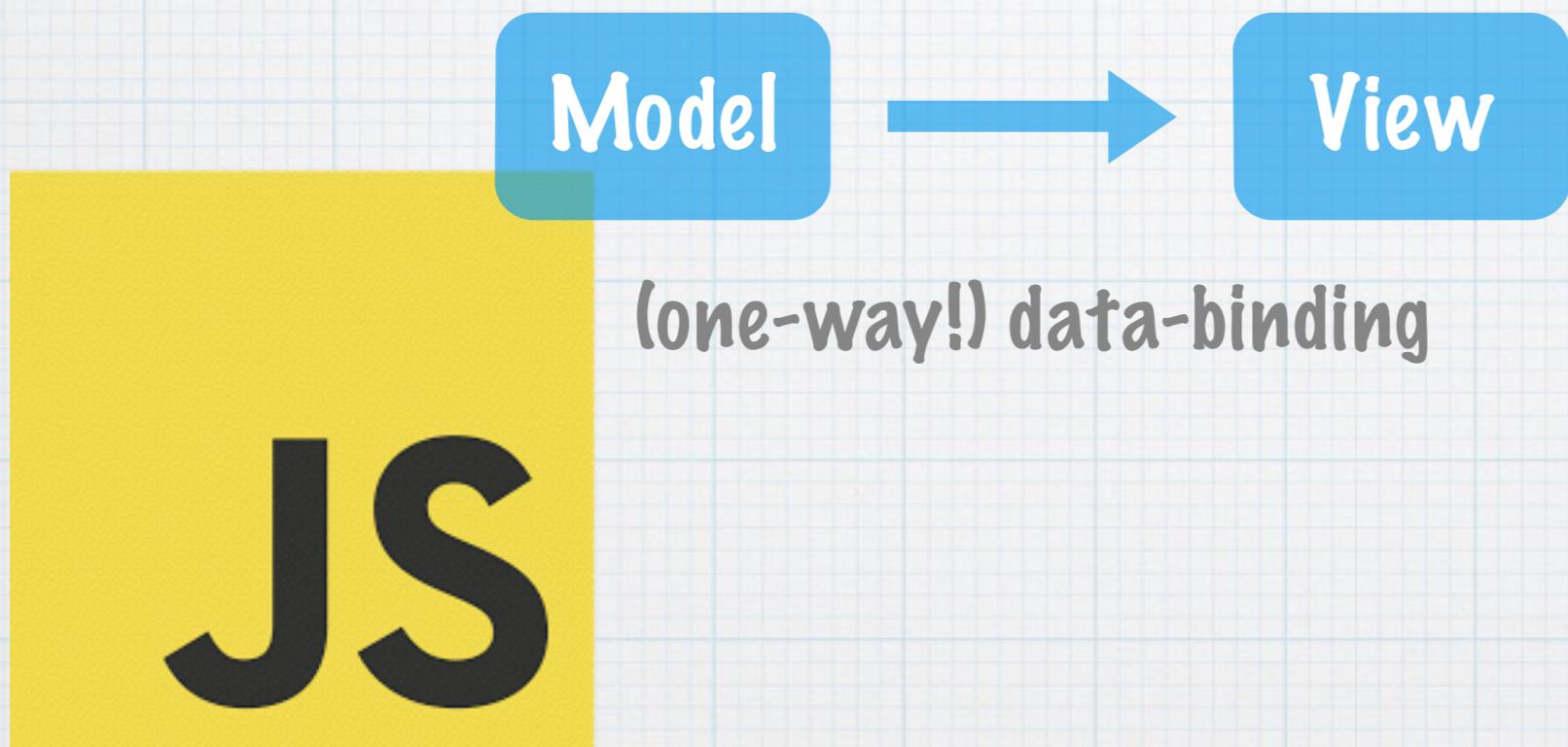
MVC



(one-way!) data-binding

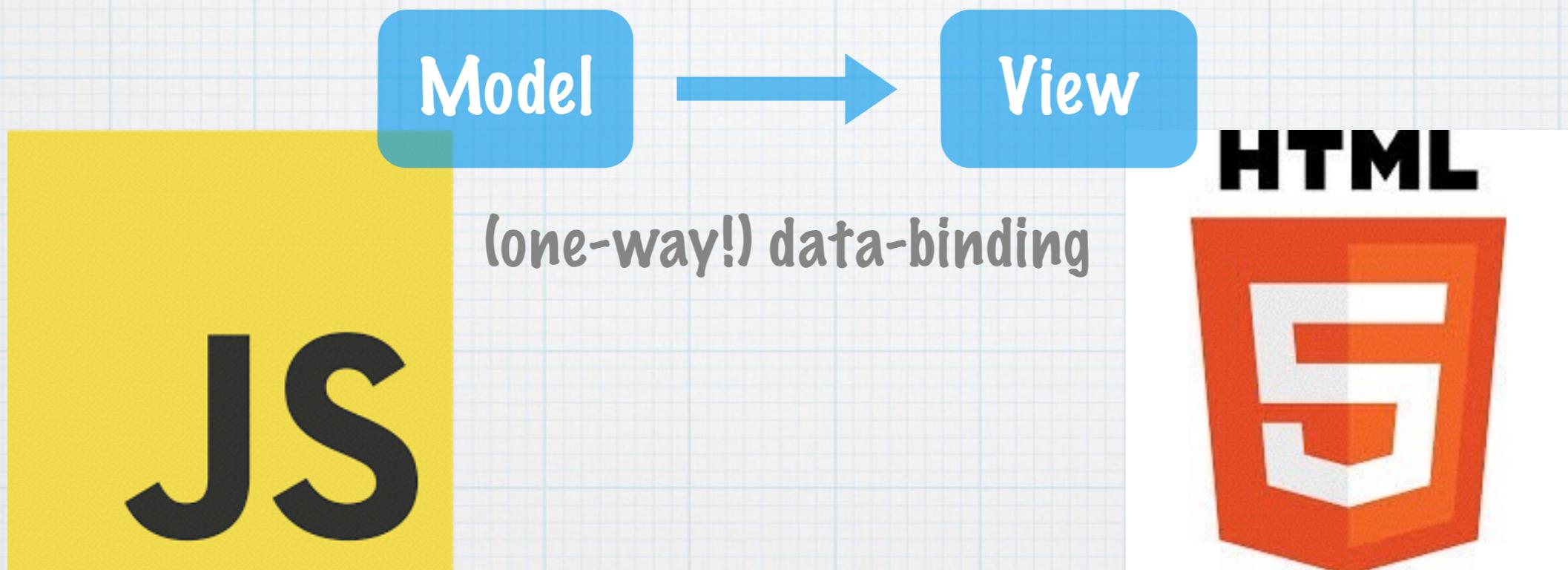
React?

MVC



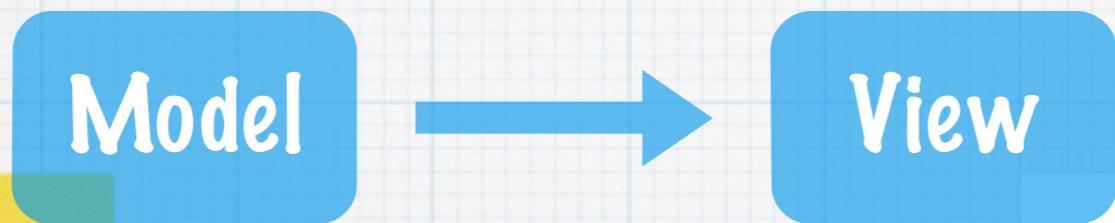
React?

MVC



React?

MVC



(one-way!) data-binding

IS

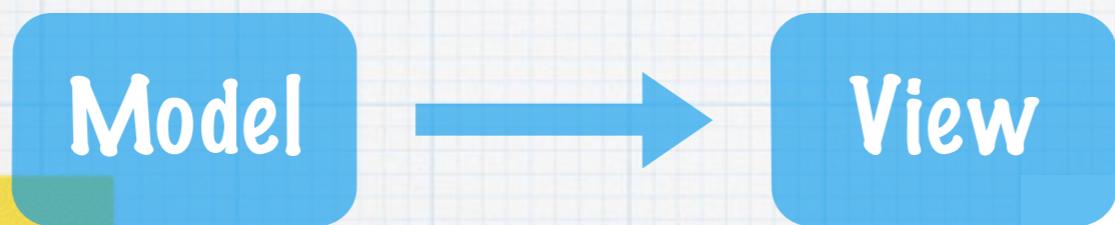
```
var cat = {  
  name: "kitty",  
  color: "pink"  
}
```

HTML



React?

MVC



(one-way!) data-binding

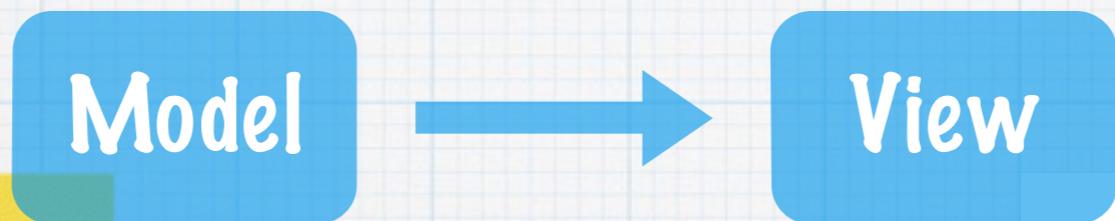
IS

```
var cat = {  
  name: "kitty",  
  color: "pink"  
}
```

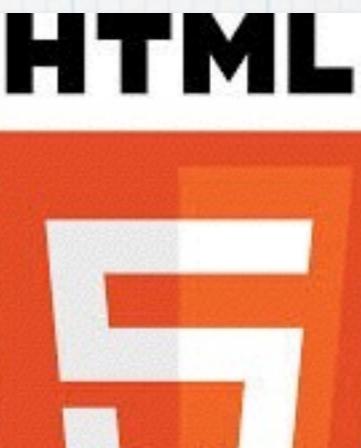


React?

MVC

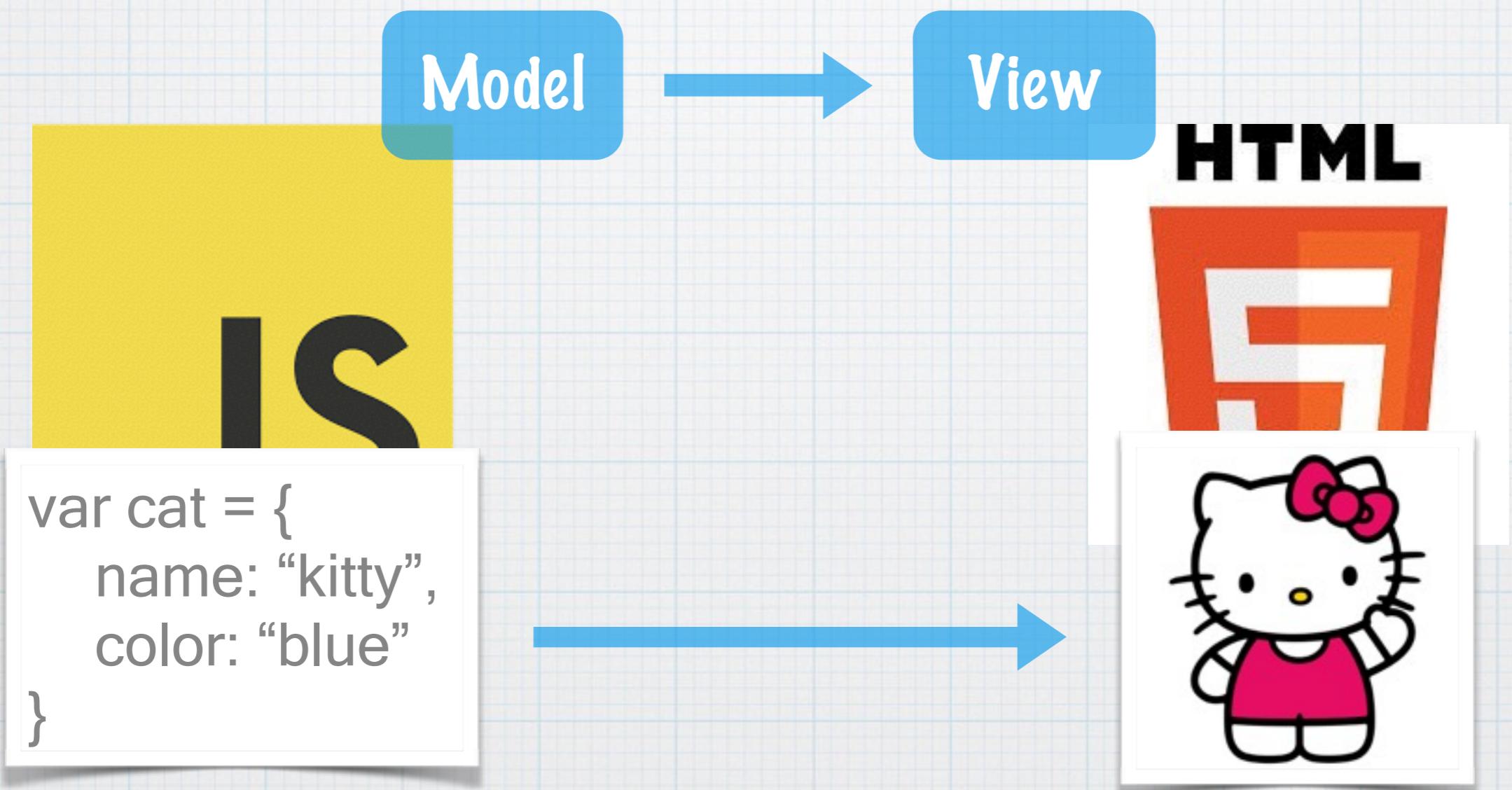


```
var cat = {  
  name: "kitty",  
  color: "blue"  
}
```



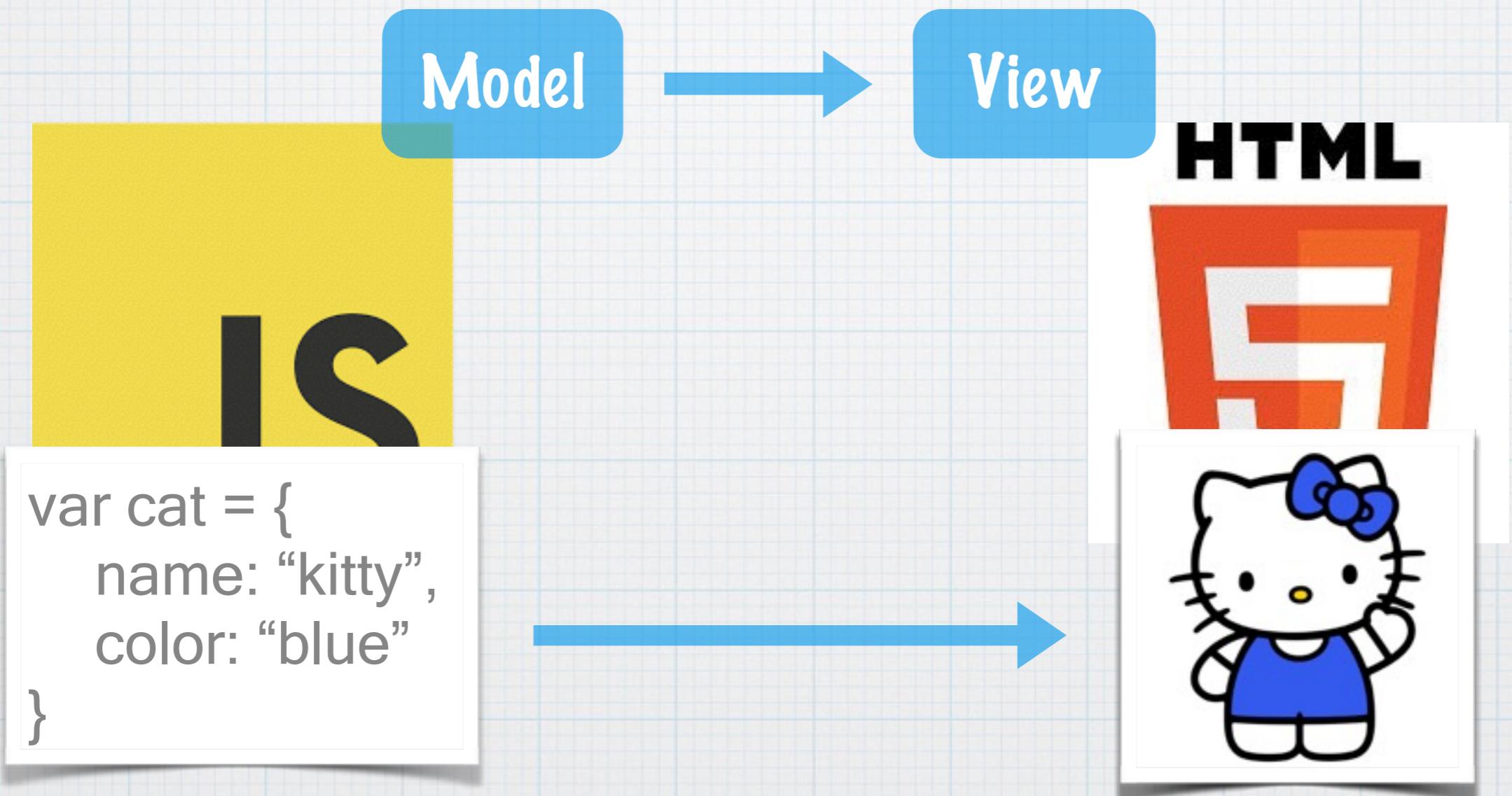
React?

MVC



React?

MVC



When?

“Large applications with data that (frequently) changes over time”

When?

“Large applications with data that (frequently) changes over time”



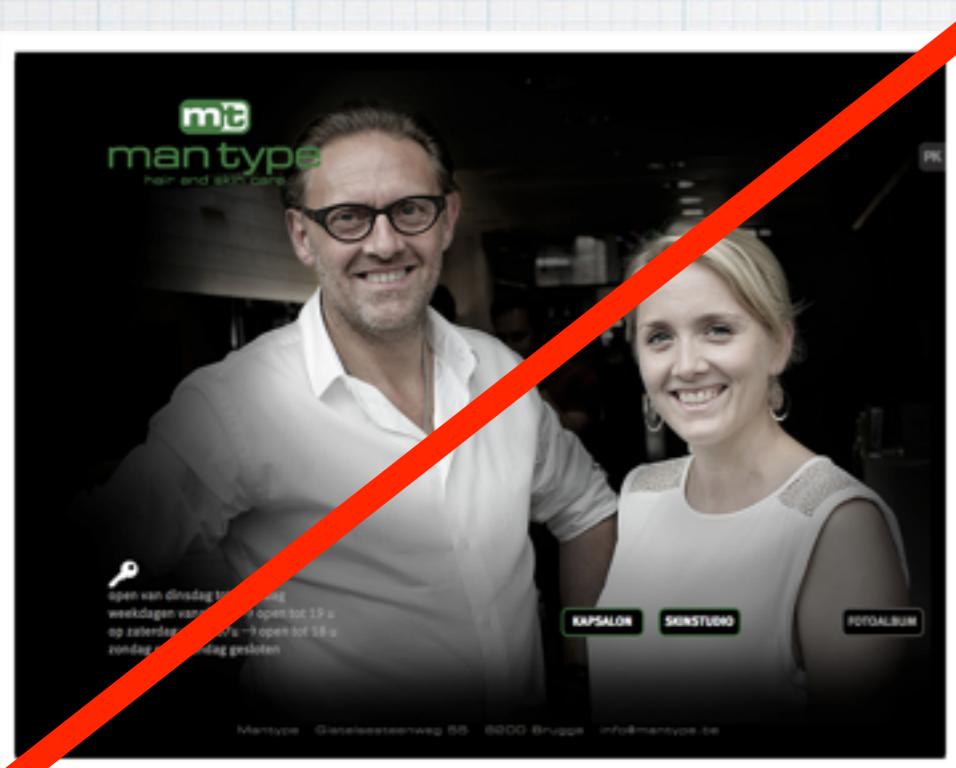
When?

“Large applications with data that (frequently) changes over time”



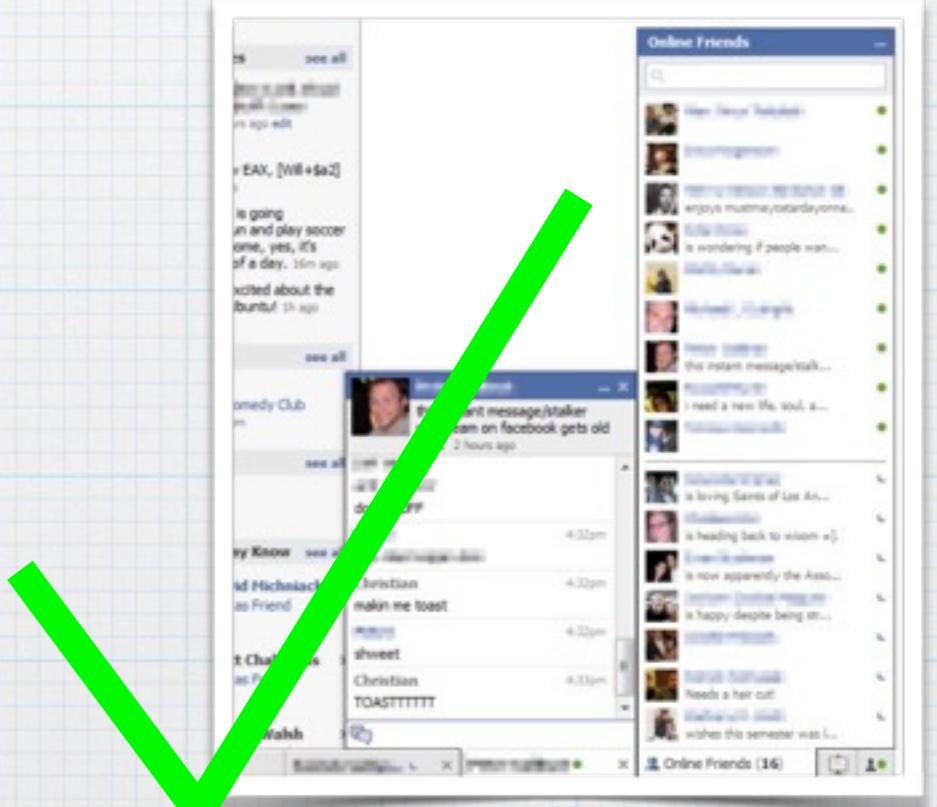
When?

“Large applications with data that (frequently) changes over time”



When?

“Large applications with data that (frequently) changes over time”



Imperative/Stateful

JS



Imperative/Stateful

JS

```
function createKitten(name, color) {  
  var kitten = {name: name, color: color};  
  showDOMKitten(kitten);  
  return kitten;  
}
```



Imperative/Stateful

JS

```
function createKitten(name, color) {  
  var kitten = {name: name, color: color};  
  showDOMKitten(kitten);  
  return kitten;  
}
```

```
function showDOMKitten(kitten) {  
  var dress = $("<div class=\"" + kitten.color + "\">");  
  .append(...)  
  $("<div id=\"" + kitten.name + "\">")  
  .append(dress)  
  .appendTo('#kittens');  
}
```



Imperative/Stateful

JS

```
function createKitten(name, color) {  
  var kitten = {name: name, color: color};  
  showDOMKitten(kitten);  
  return kitten;  
}
```

```
function updateColor(kitten, color) {  
  kitten.color = color;  
  updateDOMKitten(kitten);  
}
```

```
function showDOMKitten(kitten) {  
  var dress = $("<div class=\"" + kitten.color + "\">>");  
  .append(...)  
  $("<div id=\"" + kitten.name + "\">>")  
  .append(dress)  
  .appendTo('#kittens');  
}
```



Imperative/Stateful

JS

```
function createKitten(name, color) {  
  var kitten = {name: name, color: color};  
  showDOMKitten(kitten);  
  return kitten;  
}
```

```
function updateColor(kitten, color) {  
  kitten.color = color;  
  updateDOMKitten(kitten);  
}
```

```
function showDOMKitten(kitten) {  
  var dress = $('

' + ...);  
  $('<div id="' + kitten.name + '>').  
    append(dress).  
   appendTo('#kittens');  
}


```

```
function updateDOMKitten(kitten) {  
  var $domNode = $('#' + kitten.name + '.dress');  
  $domNode.attr('class', kitten.color);  
}
```



Imperative/Stateful

JS

```
function createKitten(name, color) {  
  var kitten = {name: name, color: color};  
  showDOMKitten(kitten);  
  return kitten;  
}
```

```
function updateColor(kitten, color) {  
  kitten.color = color;  
  updateDOMKitten(kitten);  
}
```



```
function showDOMKitten(kitten) {  
  var dress = $('

' +  
    '

' +  
      'is');


```

```
ten(kitten) {  
  '#' + kitten.name + ' .dress');  
ss', kitten.color');
```



Imperative/Stateful

JS

```
function createKitten(name, color) {  
  var kitten = {name: name, color: color};  
  showDOMKitten(kitten);  
  return kitten;  
}
```

```
function showDOMKitten(kitten) {  
  var dress = $('

' +  
    .append(...)  
    +'<div id="' + kitten.name + '>' +  
    is');


```

```
function updateColor(kitten) {  
  kitten.color = color;  
  updateDOMKitten(kitten);  
}
```



```
ten(kitten) {  
  '#'+kitten.name+'.dress');  
ss', kitten.color');
```



Fast

Imperative/Stateful

JS

```
function createKitten(name, color) {  
  var kitten = {name: name, color: color};  
  showDOMKitten(kitten);  
  return kitten;  
}
```



```
function showDOMKitten(kitten) {  
  var dress = $('

' +  
    .append(...)  
    +'<div id="' + kitten.name + '>' +  
    is');


```

```
function updateColor(kitten) {  
  kitten.color = color;  
  updateDOMKitten(kitten);  
}
```

```
ten(kitten) {  
  '#' + kitten.name + ' .dress');  
ss', kitten.color');
```

Fast



Terrible to
manage

Declarative/Stateless

JS

```
function render(kitten) {  
  return $('<div id=' + kitten.name '>')  
    .append(  
      $('<div class=' + kitten.color + '>');  
      .append(...));  
}
```

```
var kitty ={name: 'kitty', color: 'pink'};
```



Declarative/Stateless

JS

```
function render(kitten) {  
  return $('<div id=' + kitten.name '>')  
    .append(  
      $('<div class=' + kitten.color + '>');  
      .append(...));  
}
```

```
var kitty ={name: 'kitty', color: 'pink'};
```

```
$('#kittens').html(render(kitty))
```



Declarative/Stateless

JS

```
var kitty ={name: 'kitty', color: 'pink'};
```

```
$('#kittens').html(render(kitty))
```

```
function render(kitten) {  
  return $('<div id=' + kitten.name '>')  
    .append(  
      $('<div class=' + kitten.color + '>');  
      .append(...));  
}
```

HTML



Declarative/Stateless

JS

```
var kitty = {name: 'kitty', color: 'pink'};
```

```
$('#kittens').html(render(kitty))
```

```
kitty.color = 'blue';
```

```
function render(kitten) {
  return $('<div id=' + kitten.name '>')
    .append(
      $('<div class=' + kitten.color + '>');
      .append(...));
}
```

HTML



Declarative/Stateless

JS

```
function render(kitten) {  
  return $('<div id=' + kitten.name '>')  
    .append(  
      $('<div class=' + kitten.color + '>');  
      .append(...));  
}
```

```
var kitty ={name: 'kitty', color: 'pink'};
```

```
$('#kittens').html(render(kitty))
```

```
kitty.color = 'blue';
```

```
$('#kittens').html(render(kitty))
```

HTML



Declarative/Stateless

JS

```
function render(kitten) {  
  return $('<div id=' + kitten.name '>')  
    .append(  
      $('<div class=' + kitten.color + '>');  
      .append(...));  
}
```

```
var kitty ={name: 'kitty', color: 'pink'};
```

```
$('#kittens').html(render(kitty))
```

```
kitty.color = 'blue';
```

```
$('#kittens').html(render(kitty))
```

HTML



Declarative/Stateless

JS

```
var kitty = {name: 'kitty', color: 'pink'};
```

```
$('#kittens').html(render(kitty))
```

```
kitty.color = 'blue';
```

```
$('#kittens').html(render(kitty))
```

```
function render(kitten) {
  return $('<div id="' + kitten.name '>')
    .append(
      $('<div class="' + kitten.color + '>');
      .append(...));
}
```



Declarative/Stateless

JS

```
var kitty = {name: 'kitty', color: 'pink'};
```

```
$('#kittens').html(render(kitty))
```

```
kitty.color = 'blue';
```

```
$('#kittens').html(render(kitty))
```

Easy to
manage

```
function render(kitten) {
  return $('<div id="' + kitten.name '>')
    .append(
      $('<div class="' + kitten.color + '>');
      .append(...));
}
```



HTML



Declarative/Stateless

JS

```
var kitty = {name: 'kitty', color: 'pink'};
```

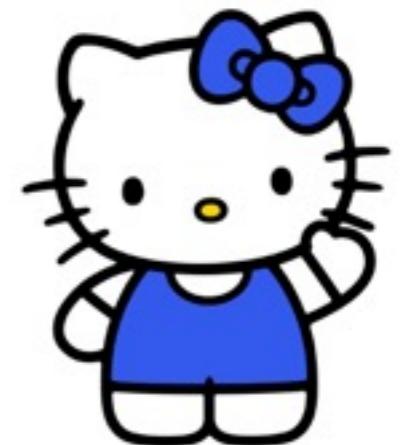
```
$('#kittens').html(render(kitty))
```

```
kitty.color = 'blue';
```

```
$('#kittens').html(render(kitty))
```

Easy to manage

```
function render(kitten) {
  return $('<div id="' + kitten.name '>')
    .append(
      $('<div class="' + kitten.color + '>');
      .append(...));
}
```



Slow

React: Virtual DOM

recalculation of
layout positioning, style
sheets, etc.

Accessing the DOM is slow!

React: Virtual DOM

recalculation of
layout positioning, style
sheets, etc.

Accessing the DOM is slow!

Keep a **JavaScript object representation** of the DOM
that you entirely re-create when data changes.

React: Virtual DOM

recalculation of
layout positioning, style
sheets, etc.

Accessing the DOM is slow!

Keep a **JavaScript object representation** of the DOM
that you entirely re-create when data changes.

Calculate **minimal differences** with previous version and
execute necessary DOM operations to reflect changes.

JS

React: Virtual DOM

fast



Virtual DOM

Real DOM

```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
    );  
  }  
});
```

JS

React: Virtual DOM

fast



Virtual DOM

Real DOM

```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
    );  
  }  
});
```

```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

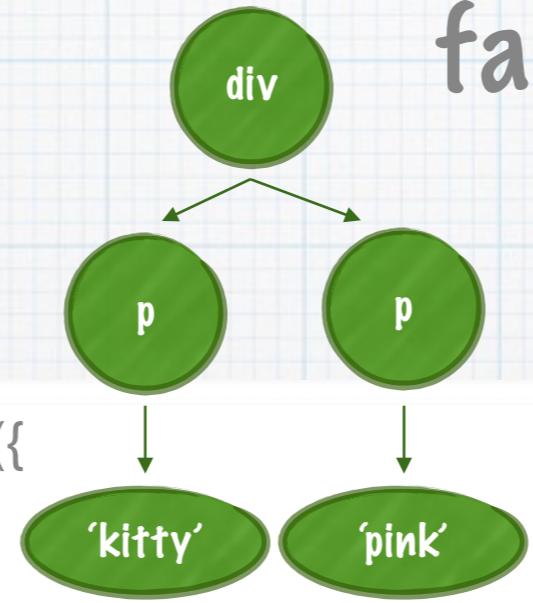
JS

React: Virtual DOM

fast



Virtual DOM



slow

Real DOM

```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```

```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

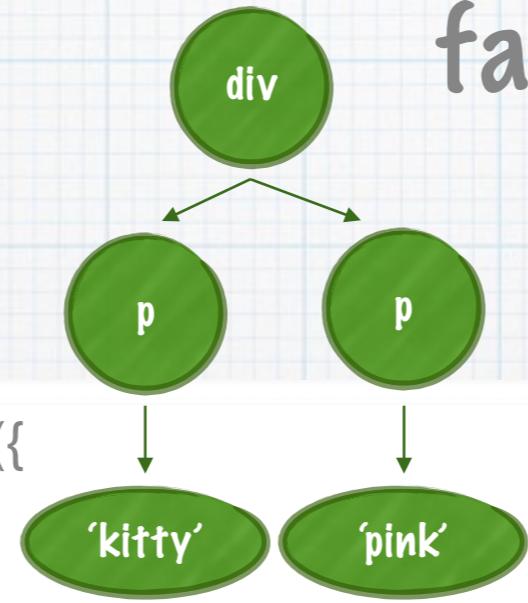
JS

React: Virtual DOM

fast



Virtual DOM



```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```

createElements()

```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

Real DOM

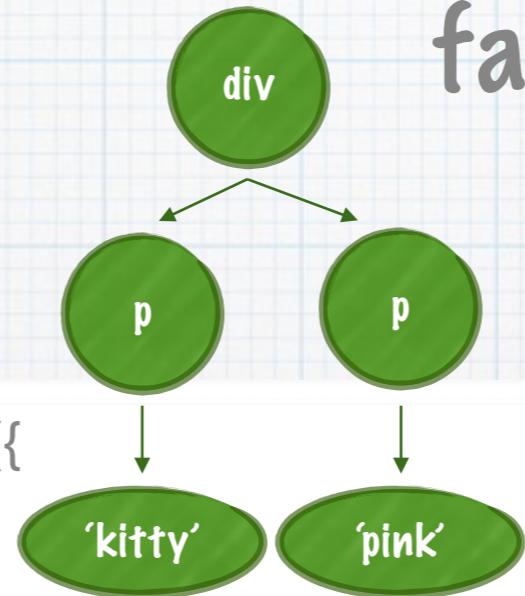
JS

React: Virtual DOM

fast



Virtual DOM

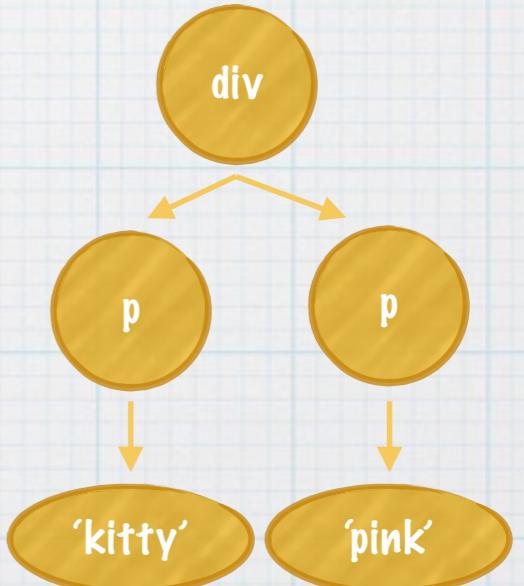


```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```

```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

Real DOM

createElements()



JS

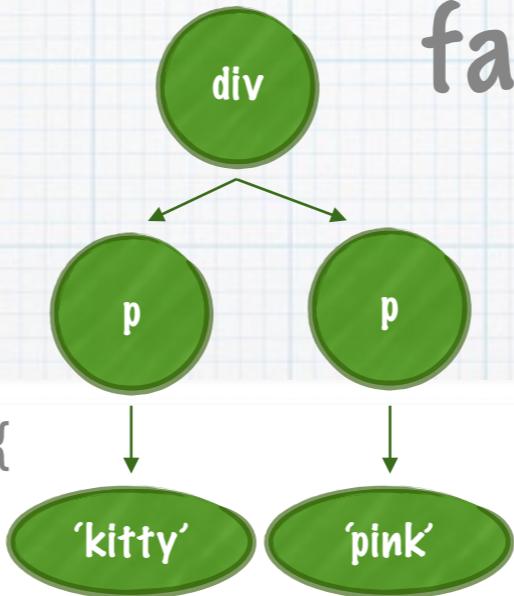
React: Virtual DOM

fast



Virtual DOM

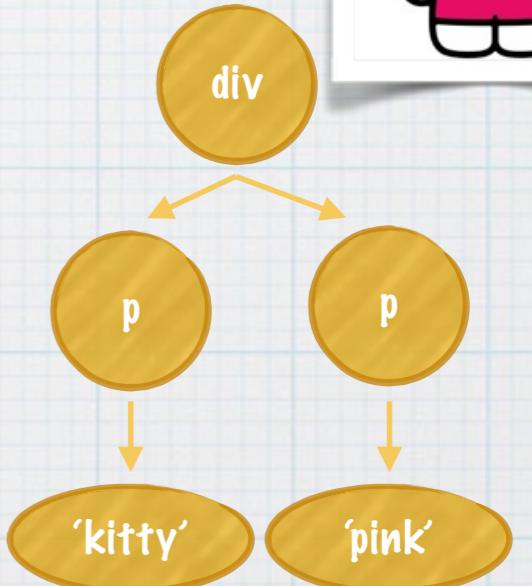
```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```



Real DOM



createElements()



```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

JS

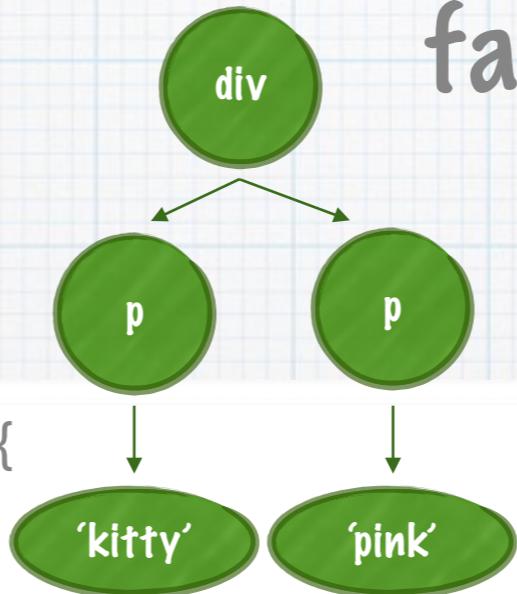
React: Virtual DOM

fast



Virtual DOM

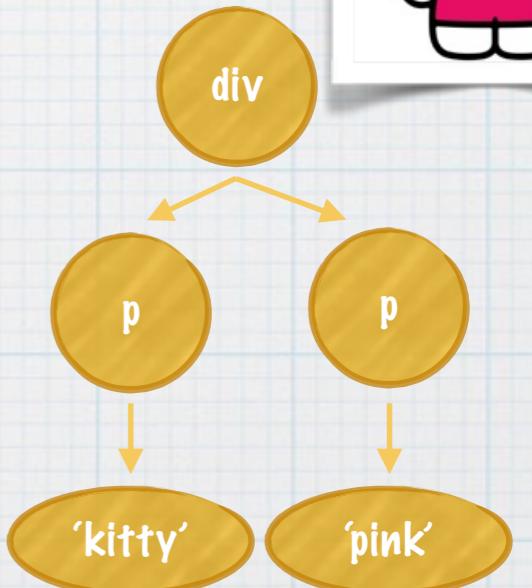
```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```



Real DOM



createElements()



```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

```
kitty.color = 'blue';  
React.render(KittenComponent, {kitten: kitty}, node);
```

JS

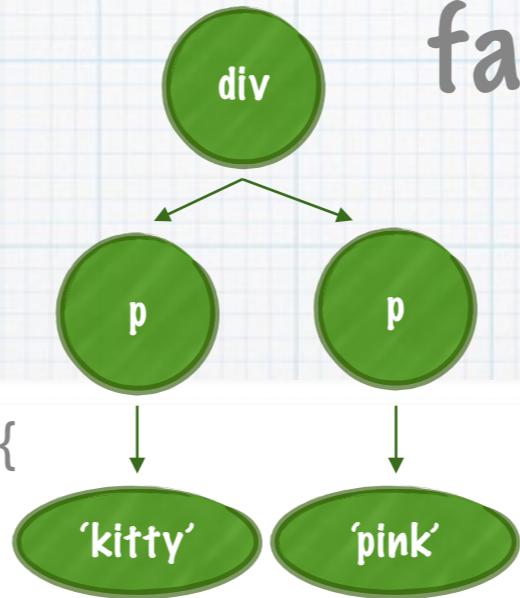
React: Virtual DOM

fast

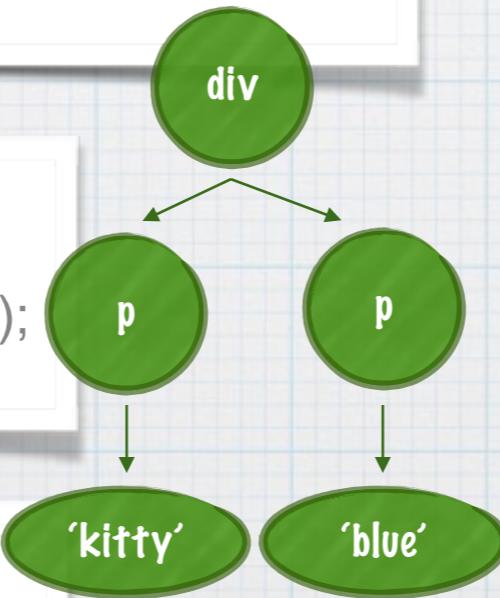


Virtual DOM

```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```



```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```



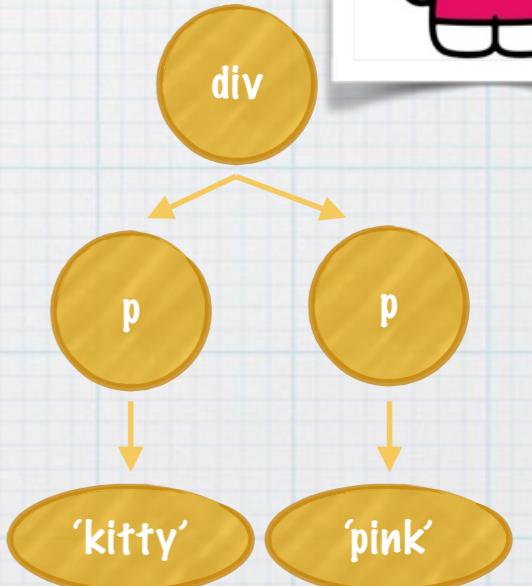
```
kitty.color = 'blue';  
React.render(KittenComponent, {kitten: kitty}, node);
```

slow

Real DOM



createElements()



JS

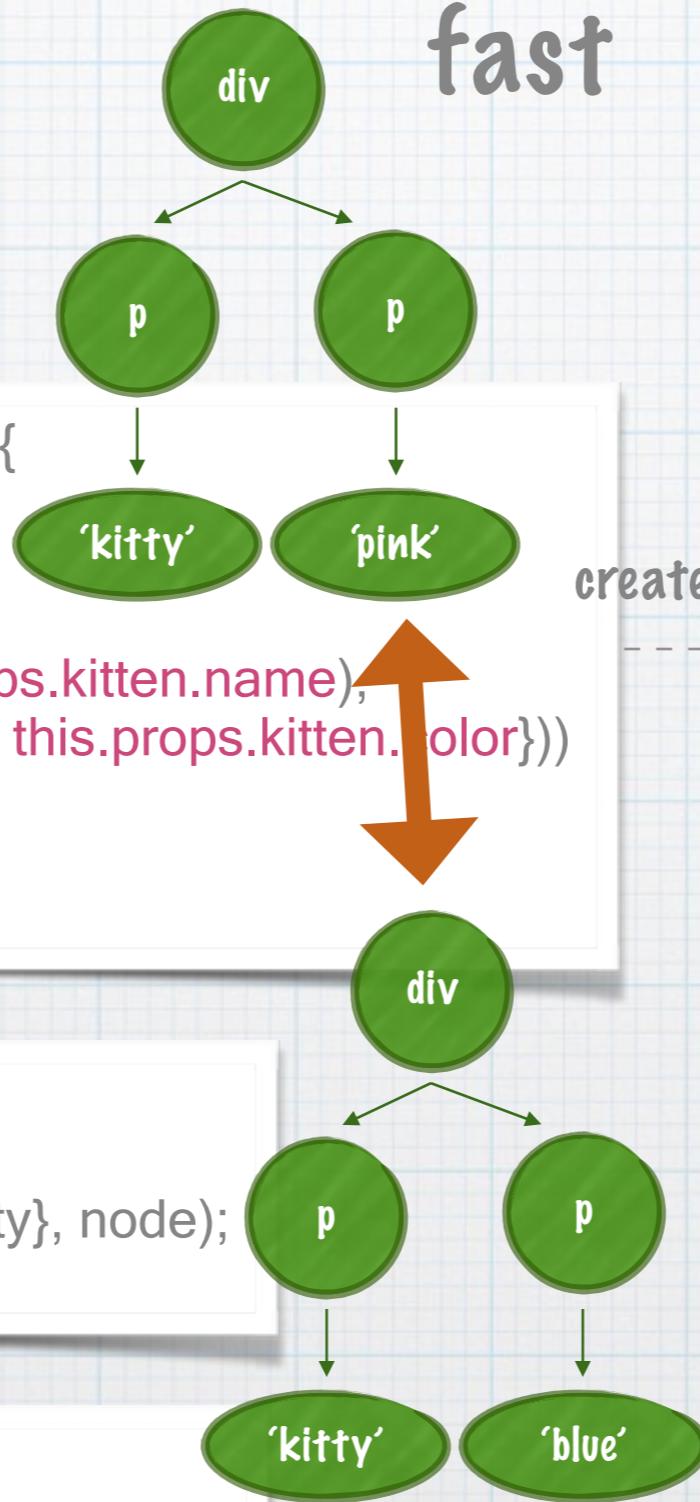
React: Virtual DOM

fast



Virtual DOM

```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```



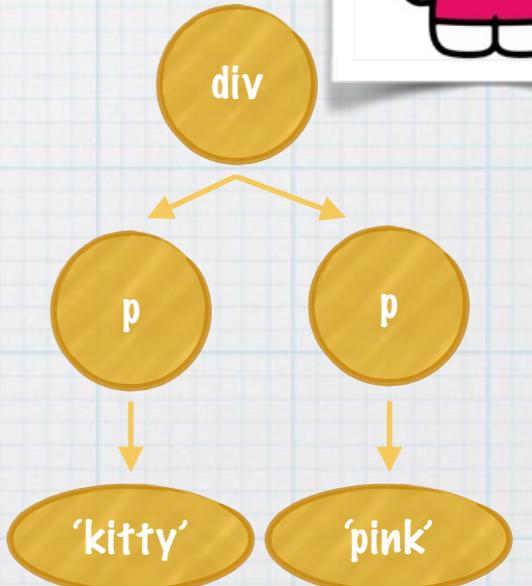
```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

```
kitty.color = 'blue';  
React.render(KittenComponent, {kitten: kitty}, node);
```

Real DOM



createElements()



JS

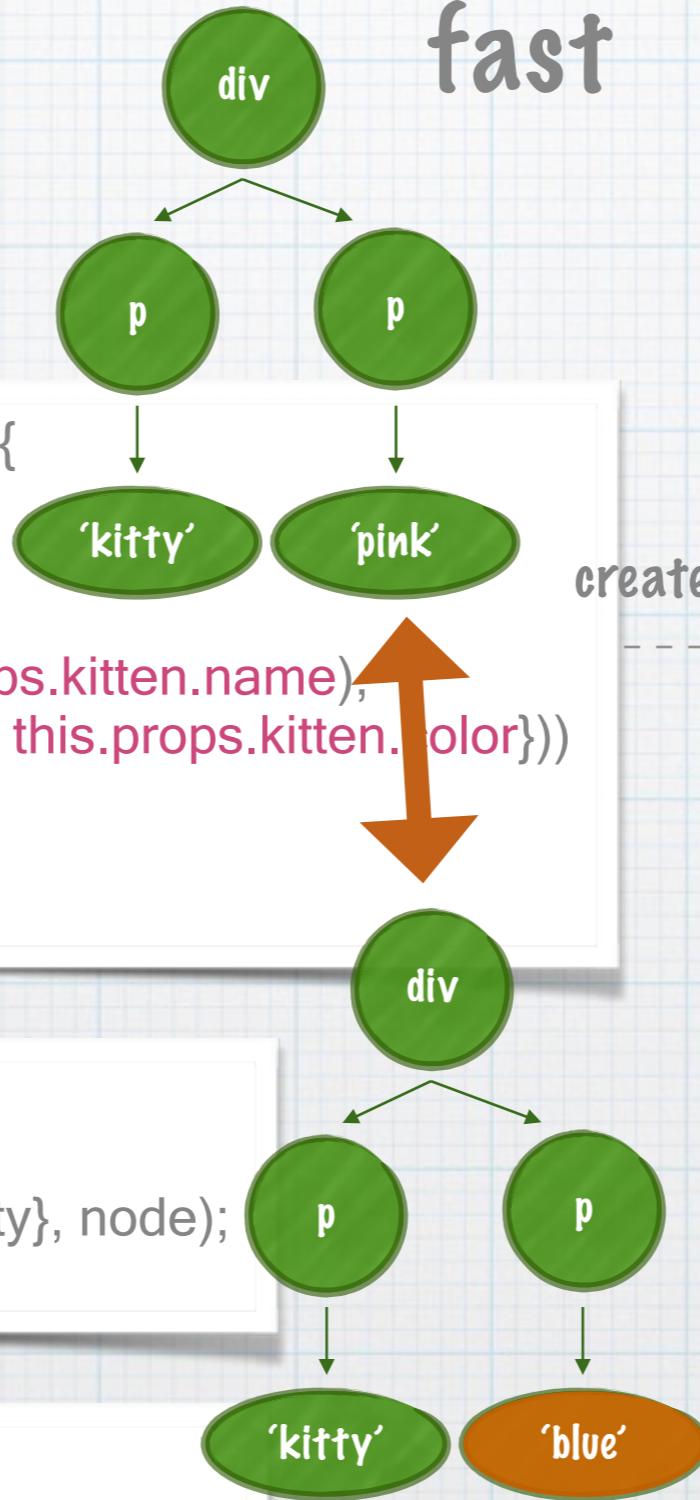
React: Virtual DOM

fast



Virtual DOM

```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```



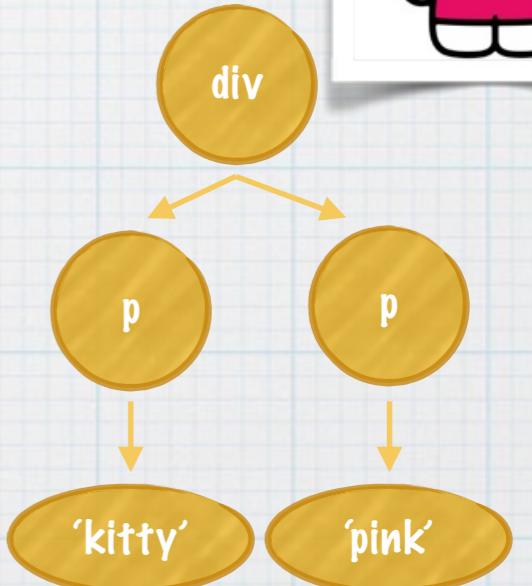
```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

```
kitty.color = 'blue';  
React.render(KittenComponent, {kitten: kitty}, node);
```

Real DOM



createElements()



JS

React: Virtual DOM

fast

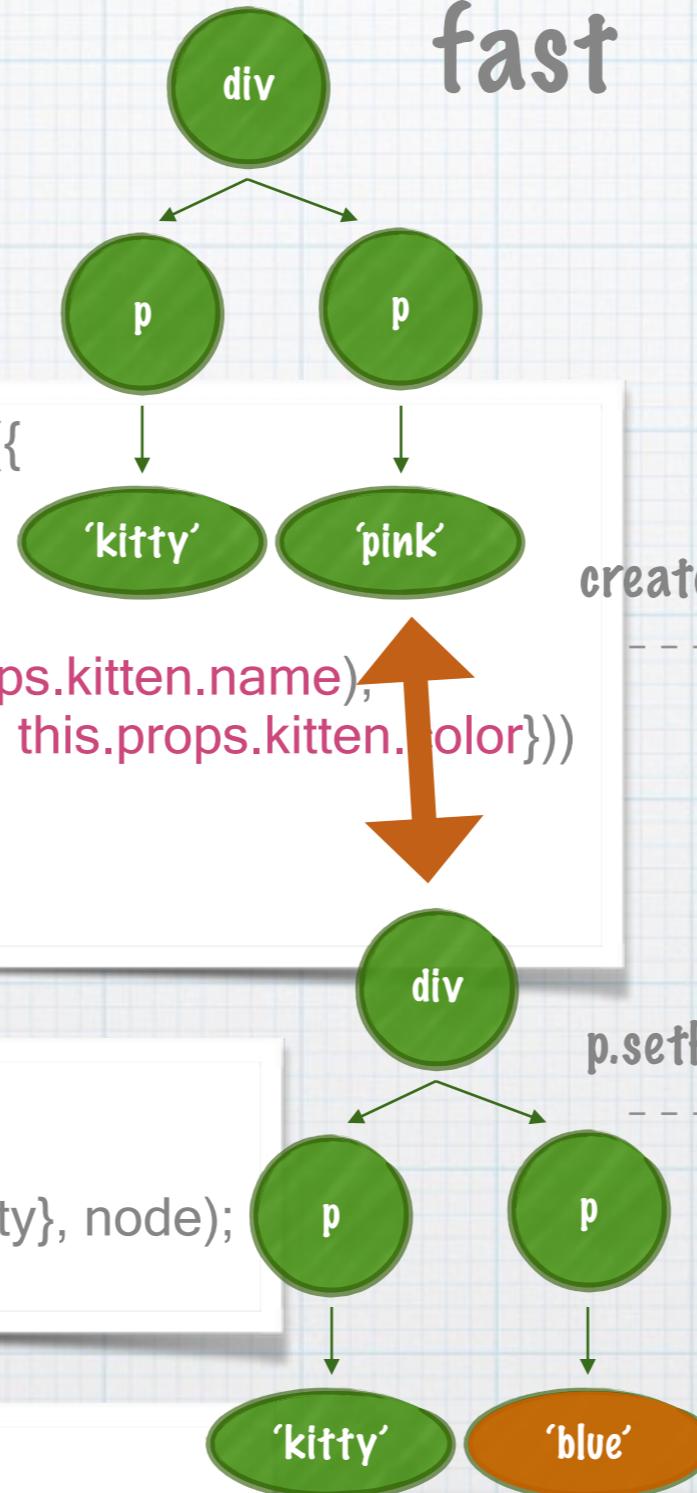


Virtual DOM

```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```

```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

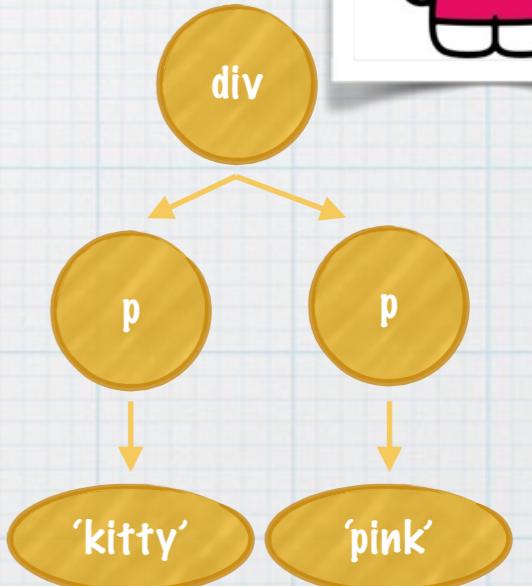
```
kitty.color = 'blue';  
React.render(KittenComponent, {kitten: kitty}, node);
```



createElements()

p.setHTML('blue')

Real DOM



JS

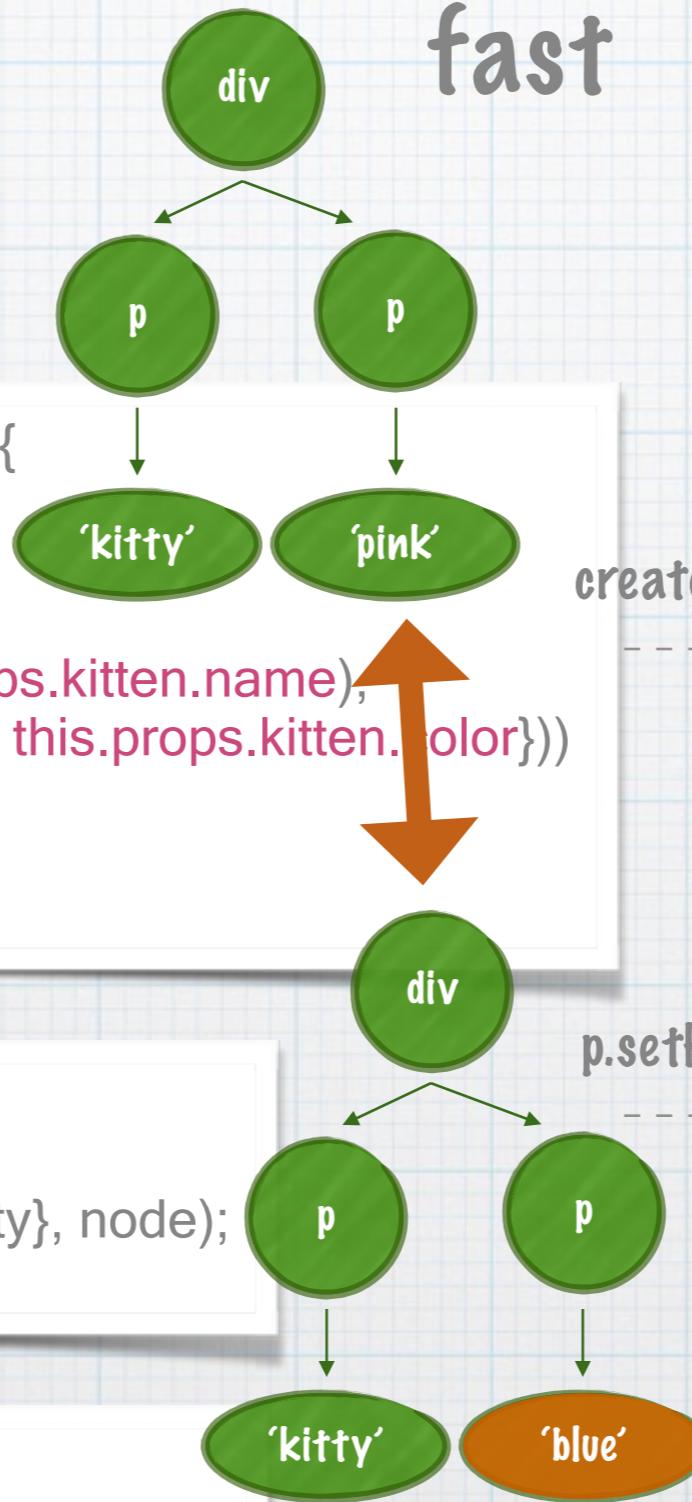
React: Virtual DOM

fast



Virtual DOM

```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```



```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

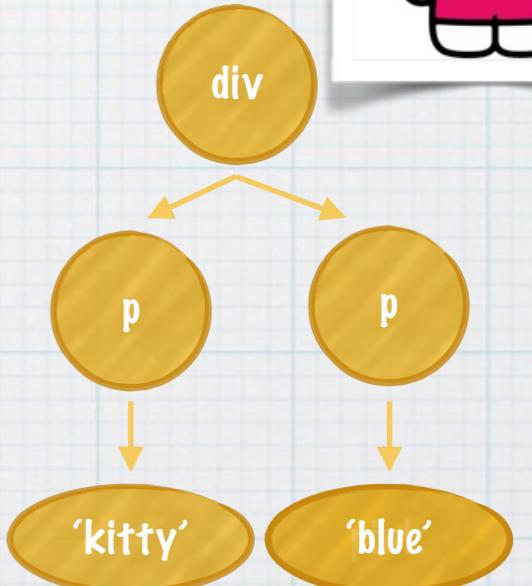
```
kitty.color = 'blue';  
React.render(KittenComponent, {kitten: kitty}, node);
```

slow

Real DOM



createElements()



p.setHtml('blue')

JS

React: Virtual DOM

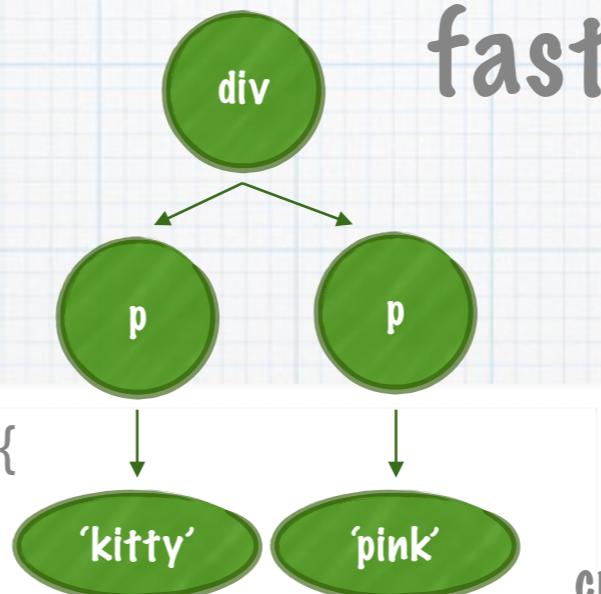


Virtual DOM

```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {className: this.props.kitten.color}))  
  }  
});
```

```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

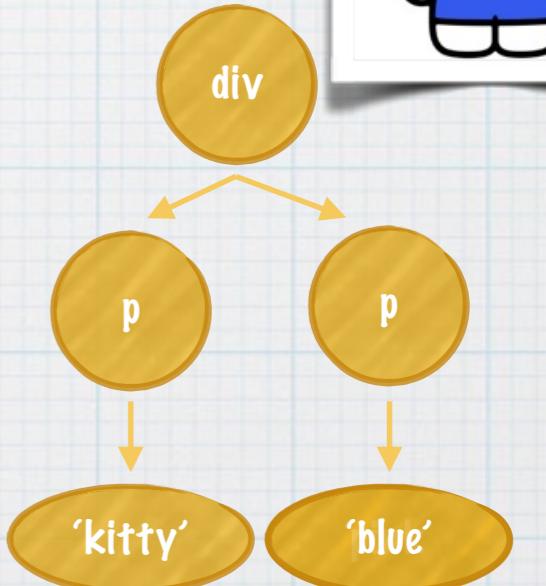
```
kitty.color = 'blue';  
React.render(KittenComponent, {kitten: kitty}, node);
```



fast
slow

Real DOM

createElements()



p.setHtml('blue')



JS

React: Virtual DOM

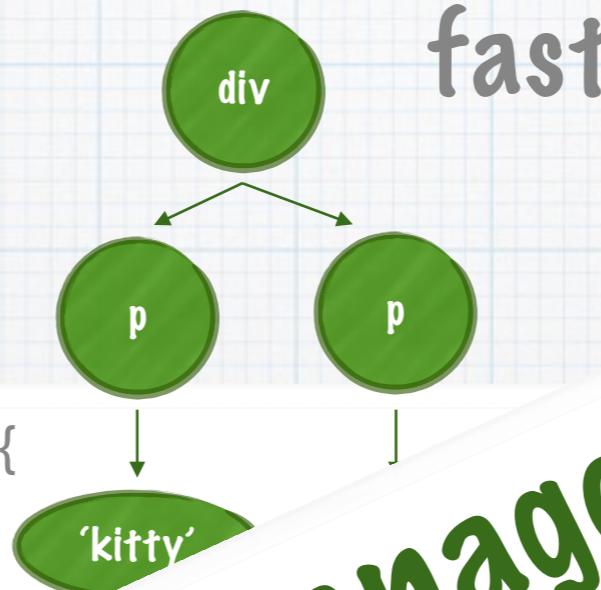


Virtual DOM

```
var KittenComponent = React.createClass({  
  render: function() {  
    return (  
      React.createElement('div', null,  
        React.createElement('p', null, this.props.kitten.name),  
        React.createElement('p', {class: 'color'}, this.props.kitten.color)  
    );  
  }  
});
```

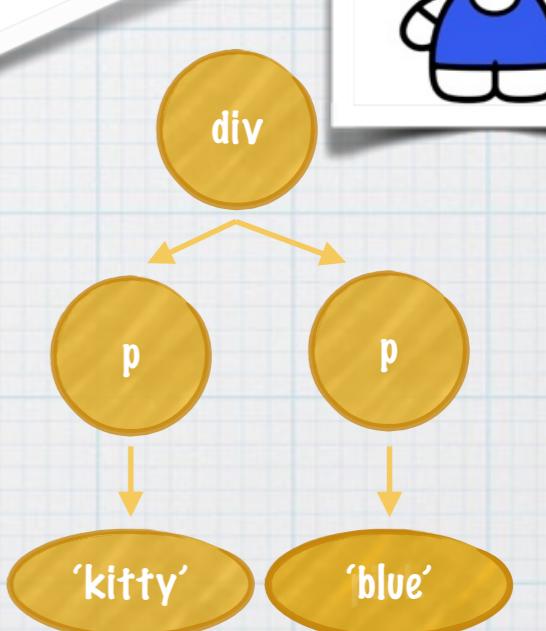
```
var kitty = {name: 'kitty', color: 'pink'};  
var node = document.body;  
React.render(KittenComponent, {kitten: kitty}, node);
```

```
kitty.color = 'blue';  
React.render(KittenComponent, {kitten: kitty}, node);
```

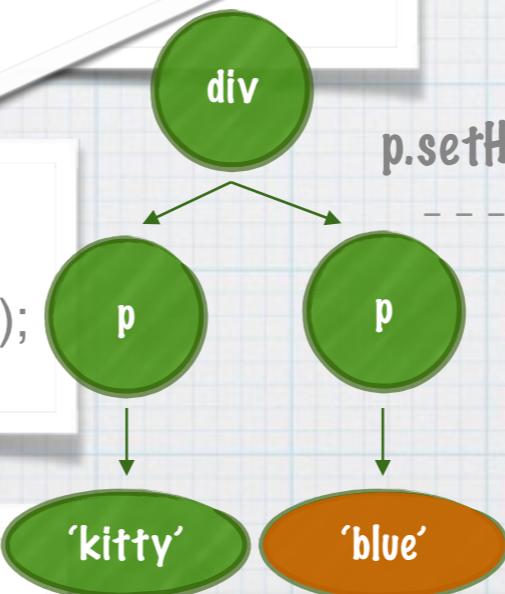


fast slow

Real DOM



Easy to manage * fast
= Win!



JSX Compiler

jsx (js + html)

```
var Kitten = React.createClass({  
  render: function() {  
    return (  
      <div>  
        <p> {this.props.name} </p>  
        <p className={this.props.color}>  
          {this.props.kitten.color}  
        </p>  
      </div>  
    );  
  }  
});  
  
React.render(<Kitten name="kitty"  
              color={getColor()} />,  
            document.body);
```

javascript

JSX Compiler

jsx (js + html)

```
var Kitten = React.createClass({  
  render: function() {  
    return (  
      <div>  
        <p> {this.props.name} </p>  
        <p className={this.props.color}>  
          {this.props.kitten.color}  
        </p>  
      </div>  
    );  
  }  
});
```

```
React.render(<Kitten name="kitty"  
              color={getColor()} />,  
            document.body);
```



javascript

```
var Kitten = React.createClass({  
  render: function() {  
    return (  
      React.createElement(  
        'div',  
        React.createElement(  
          'p',  
          this.props.kitten.name),  
        React.createElement(  
          'p',  
          {className: this.props.kitten.color},  
          this.props.kitten.color));  
    );  
  }  
});  
  
React.render(Kitten({kitten: kitty,  
                     color: getColor()}),  
            document.body);
```

JSX Compiler*

Client-Side (Development)

```
<script src="http://fb.me/JSXTransformer-0.13.2.js"></script>
```

Server-Side (Production)

```
> jsx src/KittenBox.js lib/KittenBox.js
```

Server-Side++ (Development + Production)

```
> watchify -o lib/bundle.js -v -d src/app.js
```

Rendering

lower-case

HTML elements

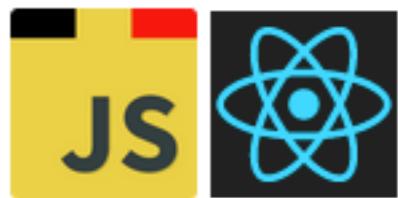
```
var myDiv = <div className="kitten" />;  
React.render(myDiv, document.body);
```

upper-case

Component

```
var MyComponent = React.createClass({ ... });  
React.render(<MyComponent property="foo" />, document.body);
```

Components



React + Flux Workshop

Tim Coppieters

Coppieters, Tim

People

Questions

Tasks

Ask your question here if you're really really really shy, but we prefer you just ask us!

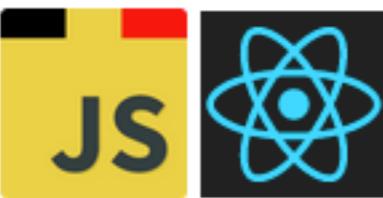
- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

When do we get cookies?

task 2

Ask!

Components



React + Flux Workshop

Tim Coppieters

People Questions Tasks

Coppieters, Tim

Ask your question here if you're really really really shy, but we prefer you just ask us!

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

When do we get cookies?

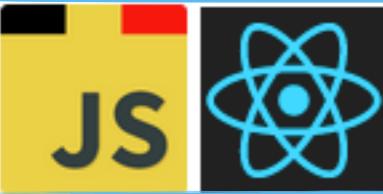
task 2 ▾

Ask!

WorkshopComponent

TitleComponent

Components



React + Flux Workshop

Tim Coppieters

People Questions Tasks Coppieters, Tim

Ask your question here if you're really really really shy, but we prefer you just ask us!

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

When do we get cookies?

task 2 ▾

Ask!

WorkshopComponent

TitleComponent

DividerComponent

The screenshot shows a user interface for a workshop. At the top left are icons for JavaScript (JS) and React. To the right, the title "React + Flux Workshop" and author "Tim Coppieters" are displayed. Below the title, there are three tabs: "People", "Questions", and "Tasks". The "Questions" tab is currently selected. A green bar spans across the screen below the tabs. On the right side, there is a button labeled "Coppieters, Tim". In the main content area, there is a message: "Ask your question here if you're really really really shy, but we prefer you just ask us!". Below this message is a list of questions:

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

At the bottom left, there is a text input field containing the question "When do we get cookies?". To its right is a dropdown menu set to "task 2". At the very bottom left is a blue "Ask!" button.

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

The screenshot shows a web application interface for a workshop. At the top left are two icons: one for JavaScript (JS) and another for React (an atom symbol). To the right of these is the title "React + Flux Workshop" and the author's name "Tim Coppieters". Below the title is a navigation bar with three tabs: "People", "Questions" (which is currently selected), and "Tasks". A sidebar on the right contains the author's name, "Coppieters, Tim". The main content area contains a message encouraging users to ask questions and a list of three questions from Tim Coppieters. At the bottom, there is a form with a question input field containing "When do we get cookies?", a dropdown menu showing "task 2", and a blue "Ask!" button.

React + Flux Workshop

Tim Coppieters

People Questions Tasks

Coppieters, Tim

Ask your question here if you're really really really shy, but we prefer you just ask us!

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

When do we get cookies?

task 2

Ask!

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

The screenshot shows a web application interface for a 'React + Flux Workshop' by Tim Coppieters. At the top left are icons for JavaScript (JS) and React. The title 'React + Flux Workshop' and author 'Tim Coppieters' are displayed prominently. Below the title is a navigation bar with tabs: 'People', 'Questions' (which is active), and 'Tasks'. A sidebar on the right contains a list of questions and a dropdown menu. A large text area in the center encourages users to ask questions, listing three specific ones from Tim Coppieters. At the bottom is a form with fields for asking a question and a dropdown for selecting a task.

JS React

React + Flux Workshop
Tim Coppieters

People Questions Tasks

Coppieters, Tim

Ask your question here if you're really really really shy, but we prefer you just ask us!

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

When do we get cookies?

task 2

Ask!

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

The screenshot shows a web application interface for a 'React + Flux Workshop' by Tim Coppieters. At the top left are icons for JavaScript (JS) and React. The title 'React + Flux Workshop' and author 'Tim Coppieters' are displayed prominently. Below the title is a navigation bar with three tabs: 'People', 'Questions' (which is currently selected), and 'Tasks'. A sidebar on the right contains a list of questions and a form for asking new ones. The main content area displays a list of questions from Tim Coppieters, followed by a question input field and an 'Ask!' button.

React + Flux Workshop
Tim Coppieters

People Questions Tasks

Coppieters, Tim

Ask your question here if you're really really really shy, but we prefer you just ask us!

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

When do we get cookies?

task 2

Ask!

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

TabMenuComponent

The screenshot shows a web application interface for a 'React + Flux Workshop' by Tim Coppieters. At the top left are icons for JavaScript (JS) and React. The title 'React + Flux Workshop' and author 'Tim Coppieters' are displayed prominently. Below the title is a navigation bar with tabs: 'People', 'Questions' (which is active), and 'Tasks'. A user profile box for 'Coppieters, Tim' is visible. The main content area contains a message encouraging users to ask questions and lists three questions from Tim Coppieters. At the bottom, there is a form with a question input ('When do we get cookies?'), a dropdown menu ('task 2'), and a blue 'Ask!' button.

React + Flux Workshop
Tim Coppieters

People Questions Tasks

Coppieters, Tim

Ask your question here if you're really really really shy, but we prefer you just ask us!

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

When do we get cookies?

task 2

Ask!

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

TabMenuComponent

TabMenuItemComp

The screenshot shows a user interface for a workshop. At the top, there are two icons: one for JavaScript (JS) and one for React (atom symbol). The title is "React + Flux Workshop" by Tim Coppieters. Below the title, there are three tabs: "People", "Questions", and "Tasks". The "Questions" tab is currently active. In the sidebar on the left, there is a list of questions from Tim Coppieters:

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

At the bottom of the sidebar, there is a text input field containing "When do we get cookies?" and a blue button labeled "Ask!".

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

TabMenuComponent

TabMenuItemComp

TabPageComponent

The screenshot shows a web application interface for a 'React + Flux Workshop' by Tim Coppieters. At the top left are icons for JavaScript (JS) and React. The title 'React + Flux Workshop' and author 'Tim Coppieters' are displayed. Below the title is a navigation bar with tabs: 'People', 'Questions' (which is active), and 'Tasks'. A user profile box for 'Coppieters, Tim' is visible. The main content area contains a message encouraging users to ask questions and a list of three questions from Tim Coppieters. At the bottom, there is a text input field with placeholder text 'When do we get cookies?' and a blue 'Ask!' button.

React + Flux Workshop
Tim Coppieters

People Questions Tasks

Coppieters, Tim

Ask your question here if you're really really really shy, but we prefer you just ask us!

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

When do we get cookies?

task 2

Ask!

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

TabMenuComponent

TabMenuItemComp

TabPageComponent

ExplanationComp

The screenshot shows a user interface for a workshop. At the top left are icons for JavaScript (JS) and React. The title "React + Flux Workshop" and subtitle "Tim Coppieters" are displayed. A navigation bar below has tabs for "People", "Questions" (which is active), and "Tasks". A sidebar on the right shows a list of questions and a dropdown menu. A main content area contains a text input field and a button.

React + Flux Workshop
Tim Coppieters

People Questions Tasks

Coppieters, Tim

Ask your question here if you're really really really shy, but we prefer you just ask us!

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

When do we get cookies?

task 2

Ask!

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

TabMenuComponent

TabMenuItemComp

TabPageComponent

ExplanationComp

QuestionsBox

The screenshot shows a workshop interface with the following components:

- Header:** JS logo and React logo, followed by the title "React + Flux Workshop" and author "Tim Coppieters".
- Header Buttons:** Three tabs labeled "People", "Questions", and "Tasks". The "Questions" tab is currently active.
- User Profile:** A box showing "Coppieters, Tim".
- Ask a Question:** A text input field containing "Ask your question here if you're really really really shy, but we prefer you just ask us!"
- List of Questions:** A list of three questions from Tim Coppieters:
 - How do I choose my components? --- Tim Coppieters (task 1)
 - When do you use props and when state? --- Tim Coppieters (task 3)
 - When do we get cookies? --- Tim Coppieters (task 2)
- Input Fields:** Two input fields: one with placeholder "When do we get cookies?" and another with placeholder "task 2".
- Buttons:** A blue "Ask!" button.

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

TabMenuComponent

TabMenuItemComp

TabPageComponent

ExplanationComp

QuestionsBox

QuestionsList

The screenshot shows a web application interface for a 'React + Flux Workshop' by Tim Coppieters. At the top left are icons for JS and React. The title 'React + Flux Workshop' and author 'Tim Coppieters' are displayed. Below the title is a navigation bar with tabs: 'People', 'Questions' (which is active), and 'Tasks'. A sidebar on the right contains a list of questions from Tim Coppieters: 'How do I choose my components? --- Tim Coppieters (task 1)', 'When do you use props and when state? --- Tim Coppieters (task 3)', and 'When do we get cookies? --- Tim Coppieters (task 2)'. In the main content area, there's a text input field with placeholder 'Ask your question here if you're really really really shy, but we prefer you just ask us!' and a blue 'Ask!' button. To the right of the input field is a dropdown menu showing 'task 2'.

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

TabMenuComponent

TabMenuItemComp

TabPageComponent

ExplanationComp

QuestionsBox

QuestionsList

QuestionItem

The screenshot shows a web application interface for a 'React + Flux Workshop' by Tim Coppieters. At the top left are icons for JS and React. The title 'React + Flux Workshop' and author 'Tim Coppieters' are displayed. Below the title is a navigation bar with tabs: 'People', 'Questions' (which is active), and 'Tasks'. A sidebar on the right shows a list of questions from Tim Coppieters: 'How do I choose my components? --- Tim Coppieters (task 1)', 'When do you use props and when state? --- Tim Coppieters (task 3)', and 'When do we get cookies? --- Tim Coppieters (task 2)'. In the main content area, there's a text input placeholder 'Ask your question here if you're really really really shy, but we prefer you just ask us!', a text input field containing 'When do we get cookies?', and a dropdown menu labeled 'task 2'. A blue button at the bottom left says 'Ask!'. The entire interface is styled with a clean, modern look using a grid system.

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

TabMenuComponent

TabMenuItemComp

TabPageComponent

ExplanationComp

QuestionsBox

QuestionsList

QuestionItem

The screenshot shows a user interface for a workshop. At the top left are icons for JavaScript (JS) and React. The title "React + Flux Workshop" and subtitle "Tim Coppieters" are displayed. A navigation bar below has tabs for "People", "Questions" (which is active), and "Tasks". A sidebar on the right lists three questions from Tim Coppieters: "How do I choose my components?", "When do you use props and when state?", and "When do we get cookies?". Below the sidebar is a text input field containing "When do we get cookies?" and a blue "Ask!" button. A dropdown menu labeled "task 2" is open.

JS React + Flux Workshop
Tim Coppieters

People Questions Tasks

Coppieters, Tim

Ask your question here if you're really really really shy, but we prefer you just ask us!

- How do I choose my components? --- Tim Coppieters (task 1)
- When do you use props and when state? --- Tim Coppieters (task 3)
- When do we get cookies? --- Tim Coppieters (task 2)

When do we get cookies?

task 2

Ask!

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

TabMenuComponent

TabMenuItemComp

TabPageComponent

ExplanationComp

QuestionsBox

QuestionsList

QuestionItem

The screenshot shows a workshop interface with the following components:

- Header:** JS logo and React logo.
- Title:** React + Flux Workshop by Tim Coppieters.
- User Profile:** A box showing "Coppieters, Tim".
- Tab Menu:** Tabs for People, Questions (selected), and Tasks.
- Ask Area:** A text input field with placeholder "Ask your question here if you're really really really shy, but we prefer you just ask us!" and a blue "Ask!" button.
- Questions List:** A list of questions from Tim Coppieters:
 - How do I choose my components? --- Tim Coppieters (task 1)
 - When do you use props and when state? --- Tim Coppieters (task 3)
 - When do we get cookies? --- Tim Coppieters (task 2)
- Task Input:** An input field with placeholder "When do we get cookies?" and a dropdown menu showing "task 2".

WorkshopComponent

TitleComponent

DividerComponent

MainComponent

ProfileComponent

MainContentComp

TabMenuComponent

TabMenuItemComp

TabPageComponent

ExplanationComp

QuestionsBox

QuestionsList

QuestionItem

QuestionForm

The screenshot shows a user interface for a workshop. At the top left are icons for JavaScript (JS) and React. The title "React + Flux Workshop" and subtitle "Tim Coppieters" are displayed. A navigation bar below has tabs for "People", "Questions" (which is active), and "Tasks". A sidebar on the right lists three questions from Tim Coppieters: "How do I choose my components?", "When do you use props and when state?", and "When do we get cookies?". Below the sidebar is a text input field with placeholder text "Ask your question here if you're really really really shy, but we prefer you just ask us!". A blue "Ask!" button is at the bottom left. On the right, there's a dropdown menu showing "task 2".

Components

WorkshopComponent

TitleComponent DividerComponent MainComponent

ProfileComponent MainContentComp

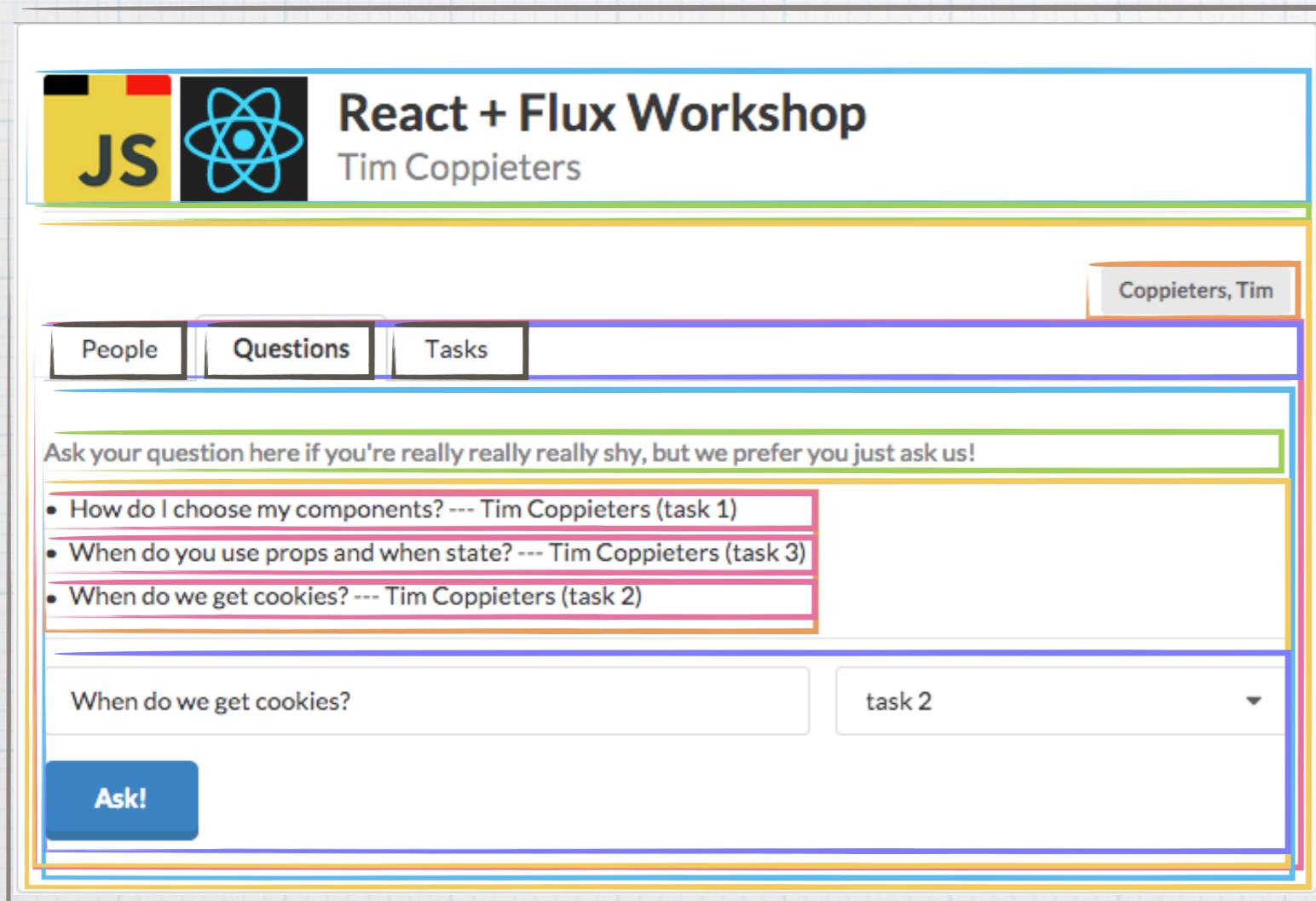
TabMenuComponent TabPageComponent

TabMenuItemComp

ExplanationComp QuestionsBox

QuestionsList QuestionForm

QuestionItem



Components = Composing

```
var WorkshopComponent = React.createClass({  
  render: function() {  
    return (  
      <div id="workshop">  
        <TitleComponent title="Workshop" subtitle="JSConfBE" />  
        <DividerComponent />  
        <MainComponent />  
      </div>  
    );  
  }  
});  
  
React.render(<WorkshopComponent />,  
  document.body);
```

```
var TitleComponent = React.createClass({  
  render: function() {  
    return (  
      <div class="title">  
        <h1> { this.props.title } </h1>  
        <h2> { this.props.subtitle } </h2>  
      </div>  
    );  
  }  
});
```

Components = Composing

```
var QuestionsList = React.createClass({  
  render: function() {  
    var questions = this.props.questions.map(function (question) {  
      return <QuestionItem question={question} />;  
    };  
    return (  
      <ul id="questions">  
        {questions}  
      </ul>  
    );  
  }  
});  
  
React.render(<QuestionsList />, node);
```

```
var QuestionItem = React.createClass({  
  render: function() {  
    return (  
      <li class="question">  
        <p> {this.props.question.text} </p>  
        <p> {this.props.question.author} </p>  
      </li>  
    );  
  }  
});
```

Component Data

props

immutable within the component

use as much as possible

```
var TitleComponent = React.createClass({  
  render: function() {  
    return (  
      <div class="title">  
        <h1> { this.props.title } </h1>  
        <h2> { this.props.subtitle } </h2>  
      </div>  
    );  
  }  
});
```

```
React.render(<TitleComponent  
  title="foo"  
  subtitle={getSubTitle()} />,  
  node);
```

Component Data

props

immutable within the component
use as much as possible

```
var TitleComponent = React.createClass({  
  render: function() {  
    return (  
      <div class="title">  
        <h1> { this.props.title } </h1>  
        <h2> { this.props.subtitle } </h2>  
      </div>  
    );  
  }  
});
```

```
React.render(<TitleComponent  
  title="foo"  
  subtitle={getSubTitle()} />,  
  node);
```

state

if you need to respond to changes from
user input, server responses...

Component Data

props

immutable within the component
use as much as possible

```
var TitleComponent = React.createClass({  
  render: function() {  
    return (  
      <div class="title">  
        <h1> { this.props.title } </h1>  
        <h2> { this.props.subtitle } </h2>  
      </div>  
    );  
  }  
});
```

```
React.render(<TitleComponent  
  title="foo"  
  subtitle={getSubTitle()} />,  
  node);
```

state

if you need to respond to changes from
user input, server responses...

```
var SwitchButton = React.createClass({  
  getInitialState: function () {  
    return {on: false}  
  },  
  handleClick: function () {  
    this.setState({on: !this.state.on});  
  },  
  render: function() {  
    var text = this.state.on ? 'off' : 'on';  
    return (  
      <button onClick={this.handleClick}>  
        turn { text }!  
      </button>  
    );  
  }  
});
```

Component Data

props

immutable within the component
use as much as possible

```
var TitleComponent = React.createClass({  
  render: function() {  
    return (  
      <div class="title">  
        <h1> { this.props.title } </h1>  
        <h2> { this.props.subtitle } </h2>  
      </div>  
    );  
  }  
});
```

```
React.render(<TitleComponent  
  title="foo"  
  subtitle={getSubTitle()} />,  
  node);
```

state

if you need to respond to changes from
user input, server responses...

```
var SwitchButton = React.createClass({  
  getInitialState: function () {  
    return {on: false}  
  },  
  handleClick: function () {  
    this.setState({on: !this.state.on});  
  },  
  render: function() {  
    var text = this.state.on ? 'off' : 'on';  
    return (  
      <button onClick={this.handleClick}>  
        turn { text }!  
      </button>  
    );  
  }  
});
```

Component Data



props

immutable within the component
use as much as possible

```
var TitleComponent = React.createClass({  
  render: function() {  
    return (  
      <div class="title">  
        <h1> { this.props.title } </h1>  
        <h2> { this.props.subtitle } </h2>  
      </div>  
    );  
  }  
});
```

```
React.render(<TitleComponent  
  title="foo"  
  subtitle={getSubTitle()} />,  
  node);
```

state

if you need to respond to changes from
user input, server responses...

```
var SwitchButton = React.createClass({  
  getInitialState: function () {  
    return {on: false}  
  },  
  handleClick: function () {  
    this.setState({on: !this.state.on});  
  },  
  render: function() {  
    var text = this.state.on ? 'off' : 'on';  
    return (  
      <button onClick={this.handleClick}>  
        turn { text }!  
      </button>  
    );  
  }  
});
```

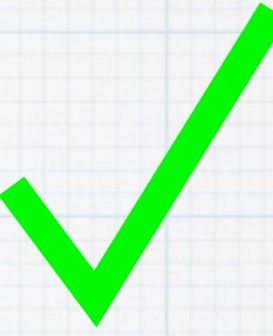
Use State

- * It is not passed in from a parent via props.
- * It can change over time.
- * You cannot compute it based on any other state or props in your component.

Using State

- * Data that a component's event handlers may change to trigger a UI update.
- * Small and JSON-serializable.
- * Minimal possible representation

Using State



- * Data that a component's event handlers may change to trigger a UI update.
- * Small and JSON-serializable.
- * Minimal possible representation

Using State

- * Computed data
- * React Components
- * Data from props

State in Component

- * As many components as possible stateless (props)
- * Common pattern: lots of stateless components with a stateful component above that passes state through props
- * Put state that is shared between components in a common ancestor

Reconciliation Strategy

- * = how react updates the DOM with a new render pass
- * Children according to order



Reconciliating State

```
var Kitten = React.createClass({
  getInitialState: function() {
    return {color: 'pink'};
  },
  handleClick: function () {
    this.setState({color: this.state.color == 'pink' ? 'blue' : 'pink'});
  },
  render: function() {
    return (
      <div onClick={this.handleClick}>
        <p> {this.props.kitten.name} </p>
        <p className={this.state.color} />
      </div>
    );
  }
});
```



```
<ul>
  <Kitten kitten={kitten1} />
  <Kitten kitten={kitten2} />
</ul>
```

```
<ul>
  <Kitten kitten={kitten2} />
</ul>
```



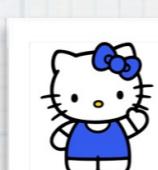
```
ul.removeChild(1);
ul.children(0).setHtml("kitten 2");
```

Reconciliating State

```
var Kitten = React.createClass({
  getInitialState: function() {
    return {color: 'pink'};
  },
  handleClick: function () {
    this.setState({color: this.state.color == 'pink' ? 'blue' : 'pink'});
  },
  render: function() {
    return (
      <div onClick={this.handleClick}>
        <p> {this.props.kitten.name} </p>
        <p className={this.state.color} />
      </div>
    );
  }
});
```



```
<ul>
  <Kitten kitten={kitten1} />
  <Kitten kitten={kitten2} />
</ul>
```



```
<ul>
  <Kitten kitten={kitten2} />
</ul>
```



```
ul.removeChild(1);
ul.children(0).setHtml("kitten 2");
```

Reconciliating State

```
var Kitten = React.createClass({
  getInitialState: function() {
    return {color: 'pink'};
  },
  handleClick: function () {
    this.setState({color: this.state.color == 'pink' ? 'blue' : 'pink'});
  },
  render: function() {
    return (
      <div onClick={this.handleClick}>
        <p> {this.props.kitten.name} </p>
        <p className={this.state.color} />
      </div>
    );
  }
});
```

```
<ul>
  <Kitten kitten={kitten1} />
  <Kitten kitten={kitten2} />
</ul>
```



```
<ul>
  <Kitten kitten={kitten2} />
</ul>
```



```
ul.removeChild(1);
ul.children(0).setHtml("kitten 2");
```



Reconciliating State

```
var Kitten = React.createClass({
  getInitialState: function() {
    return {color: 'pink'};
  },
  handleClick: function () {
    this.setState({color: this.state.color == 'pink' ? 'blue' : 'pink'});
  },
  render: function() {
    return (
      <div onClick={this.handleClick}>
        <p> {this.props.kitten.name} </p>
        <p className={this.state.color} />
      </div>
    );
  }
});
```

```
<ul>
  <Kitten kitten={kitten1} />
  <Kitten kitten={kitten2} />
</ul>
```



```
<ul>
  <Kitten kitten={kitten2} />
</ul>
```



~~ul.removeChild(1);
ul.children(0).innerHTML("kitten 2");~~

Reconciliating State

```
var Kitten = React.createClass({
  getInitialState: function() {
    return {color: 'pink'};
  },
  handleClick: function () {
    this.setState({color: this.state.color == 'pink' ? 'blue' : 'pink'});
  },
  render: function() {
    return (
      <div onClick={this.handleClick}>
        <p> {this.props.kitten.name} </p>
        <p className={this.state.color} />
      </div>
    );
  }
});
```



```
<ul>
  <Kitten key={kitten1.name} kitten={kitten1} />
  <Kitten key={kitten2.name} kitten={kitten2} />
</ul>
```

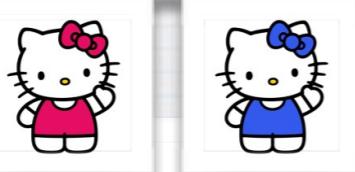
```
<ul>
  <Kitten key={kitten2.name} kitten={kitten2} />
</ul>
```

→

```
ul.removeChild(0);
```

Reconciliating State

```
var Kitten = React.createClass({
  getInitialState: function() {
    return {color: 'pink'};
  },
  handleClick: function () {
    this.setState({color: this.state.color == 'pink' ? 'blue' : 'pink'});
  },
  render: function() {
    return (
      <div onClick={this.handleClick}>
        <p> {this.props.kitten.name} </p>
        <p className={this.state.color} />
      </div>
    );
  }
});
```



```
<ul>
  <Kitten key={kitten1.name} kitten={kitten1} />
  <Kitten key={kitten2.name} kitten={kitten2} />
</ul>
```



```
<ul>
  <Kitten key={kitten2.name} kitten={kitten2} />
</ul>
```

→

```
ul.removeChild(0);
```

Reconciliating State

```
var Kitten = React.createClass({
  getInitialState: function() {
    return {color: 'pink'};
  },
  handleClick: function () {
    this.setState({color: this.state.color == 'pink' ? 'blue' : 'pink'});
  },
  render: function() {
    return (
      <div onClick={this.handleClick}>
        <p> {this.props.kitten.name} </p>
        <p className={this.state.color} />
      </div>
    );
  }
});
```



```
<ul>
  <Kitten key={kitten1.name} kitten={kitten1} />
  <Kitten key={kitten2.name} kitten={kitten2} />
</ul>
```



```
<ul>
  <Kitten key={kitten2.name} kitten={kitten2} />
</ul>
```

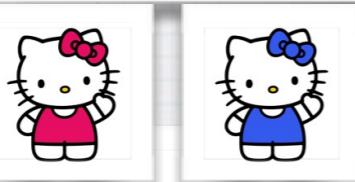
→

```
ul.removeChild(0);
```



Reconciliating State

```
var Kitten = React.createClass({
  getInitialState: function() {
    return {color: 'pink'};
  },
  handleClick: function () {
    this.setState({color: this.state.color == 'pink' ? 'blue' : 'pink'});
  },
  render: function() {
    return (
      <div onClick={this.handleClick}>
        <p> {this.props.kitten.name} </p>
        <p className={this.state.color} />
      </div>
    );
  }
});
```



```
<ul>
  <Kitten key={kitten1.name} kitten={kitten1} />
  <Kitten key={kitten2.name} kitten={kitten2} />
</ul>
```



```
<ul>
  <Kitten key={kitten2.name} kitten={kitten2} />
</ul>
```



```
ul.removeChild(0);
```

Component API

```
var Box = React.createClass({  
  getInitialState: function() {  
    return {foo: 1, bar: 2};  
  },  
  doSomething: function () {  
    this.setState({foo: 10});  
  }  
});
```

```
var Box = React.createClass({  
  getInitialState: function() {  
    return {...};  
  },  
  getDefaultProps: function() {  
    return {...};  
  },  
  componentDidMount: function () {  
  },  
  componentWillUnmount: function () {  
  },  
  render: function() {  
    return ( ... );  
  }  
});
```

After DOM is removed

Merges with provided props

After DOM is generated

e.g. when this Component is owned by another component and it no longer generates this component.

Component & DOM

```
var Box = React.createClass({
  componentDidMount: function () {
    var node = this.getDOMNode();
    React.findDOMNode(this);
  },
  submitForm: function () {
    var name = React.findDOMNode(this.refs.name).value;
    var color = React.findDOMNode(this.refs.color).value;
  },
  render: function() {
    return (
      <form onSubmit={this.submitForm}>
        <input ref="name" />
        <input ref="color" />
        <input type="submit" value="submit"/>
      </form>
    );
  }
});
```

Prop Validation

```
var KittenBox = React.createClass({  
  propTypes: {  
    name: React.PropTypes.string.isRequired,  
    color: React.PropTypes.string  
  },  
  getDefaultProps: {  
    return {color: 'pink'};  
  },  
  render: function() {  
    return (  
      <div>  
        <div> {this.props.name} </div>  
        <div> {this.props.color} </div>  
      </div>  
    );  
  }  
});
```

optional

provide
default

Prop Validation

```
function Kitten(name, color) {  
  this.name = name;  
  this.color = color;  
}
```

```
var KittenBox = React.createClass({  
  propTypes: {  
    kitten: React.PropTypes.instanceOf(Kitten).isRequired  
  },  
  render: function() {  
    return (  
      <div>  
        <div> {this.props.kitten.name} </div>  
        <div> {this.props.kitten.color} </div>  
      </div>  
    );  
  }  
});
```

And many more

More...

- * Addons

two-way data binding

className
manipulation

- * Server-Side Rendering

```
var body = React.renderToString(WorkshopComponent(props));
```

React vs other MV(C)s

- * Just a rendering library
vs Full-fledged frameworks
- * Full JavaScript expressiveness for rendering
vs Restrictive GUI Constructs (e.g. ng-repeat)
- * Virtual DOM Comparison
vs Explicit Linking
- * Overall Relatively Fast and memory efficient.
- * Server-Side Rendering

Enough React already,
what about Flux?

Flux = Design pattern that eschews MVC over a uni-directional data flow.

Case: KittenModel

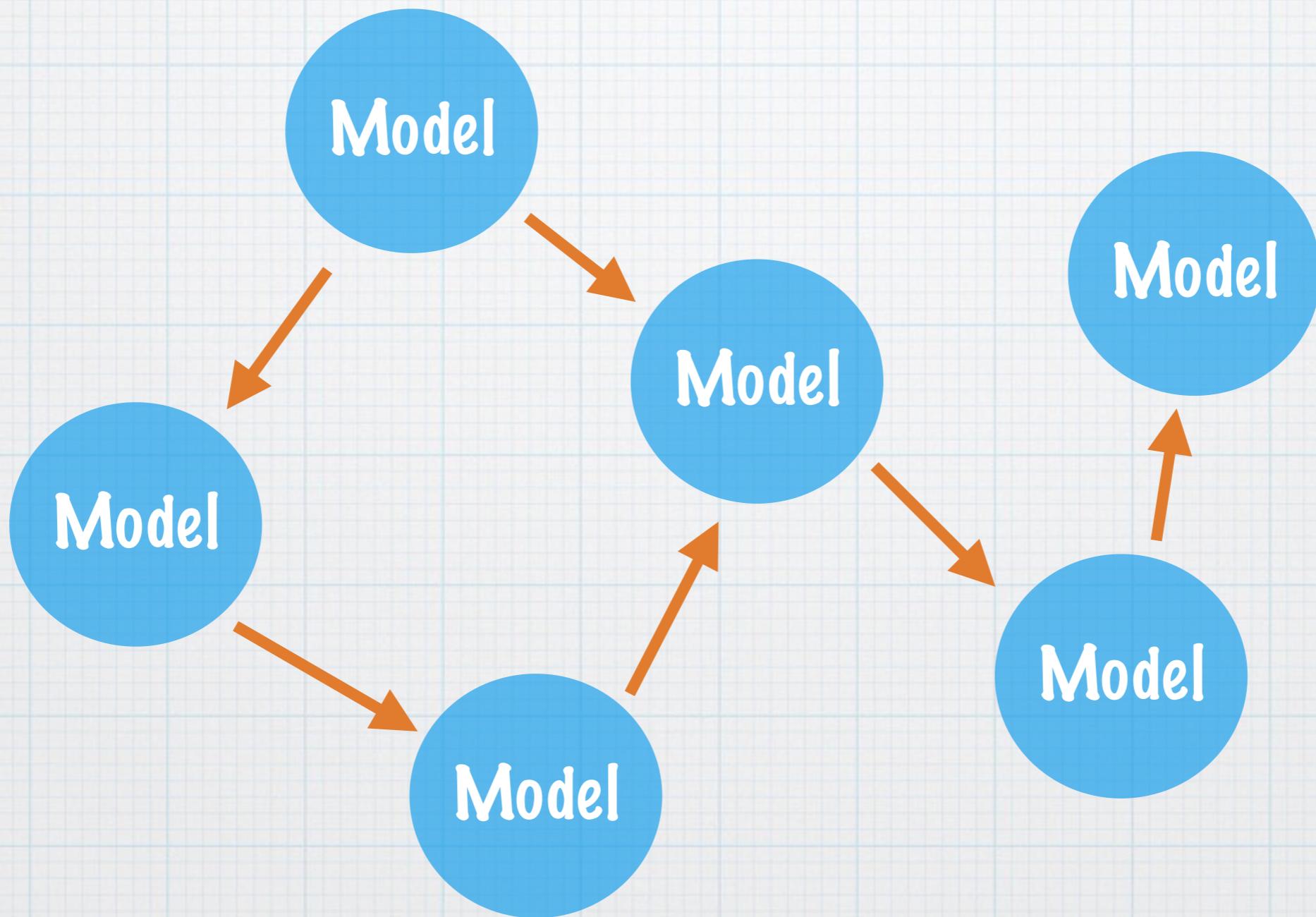
which user
actions can change my
kitten's color?

```
function KittenModel(name, color) {  
    this.name = name;  
    this.color = color;  
}  
KittenModel.prototype.updateColor = function (color) {  
    this.color = color;  
}
```

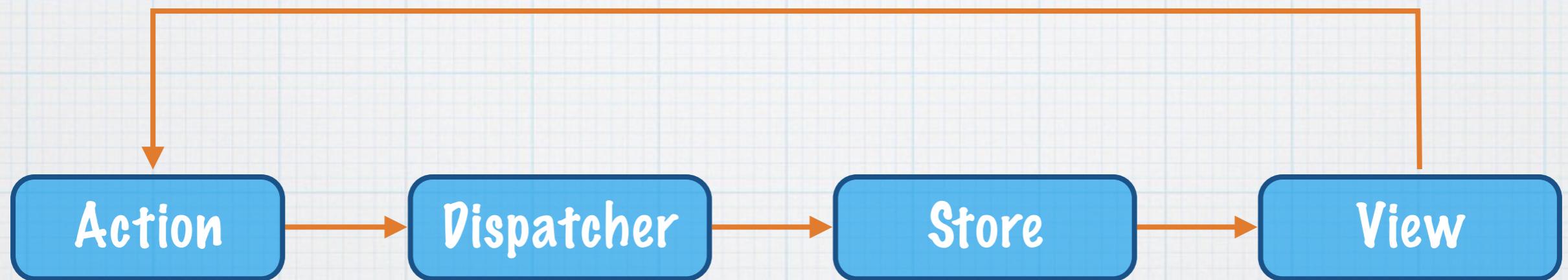
In MVC, you have to go and
look at all of your controllers

The M in MVC is not self-
contained

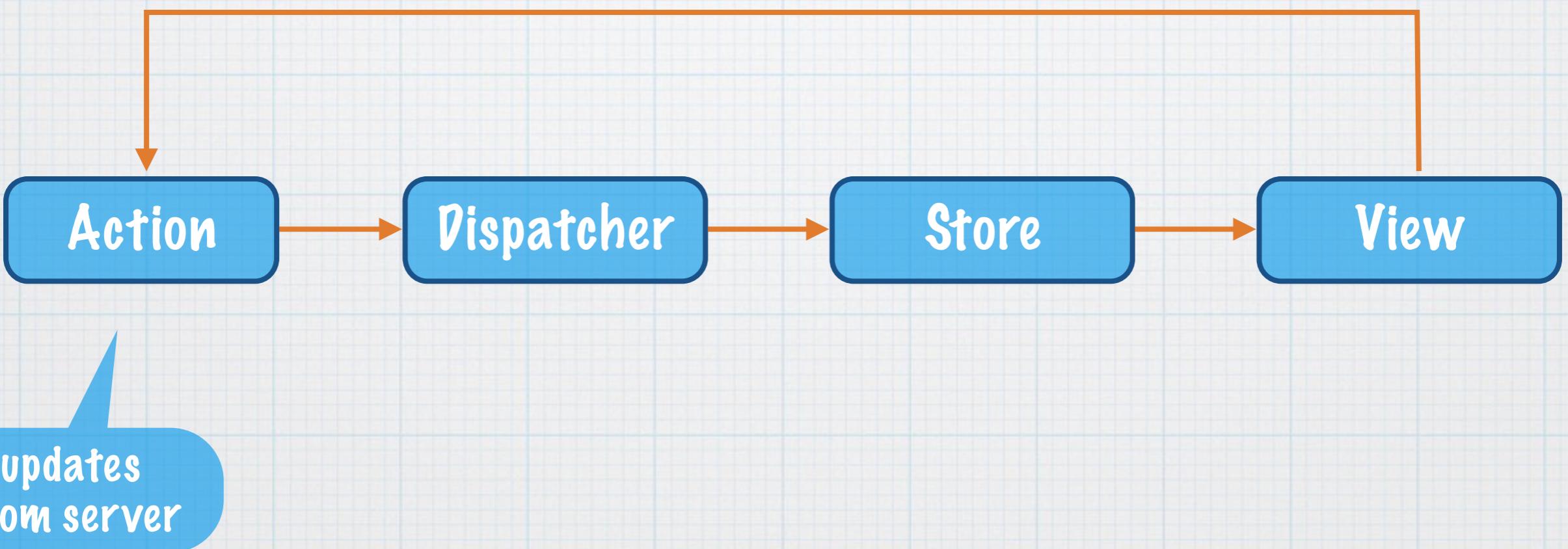
Dependencies of Models



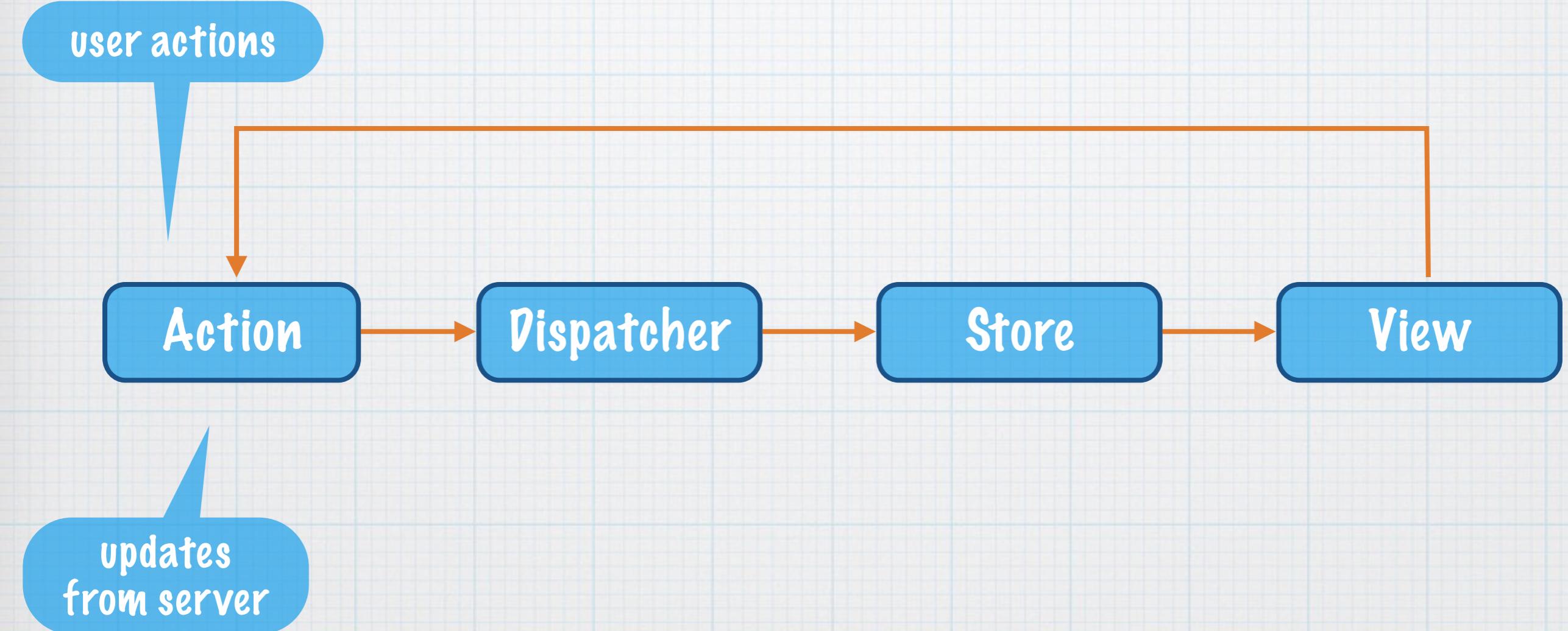
Flux: uni-directional data flow



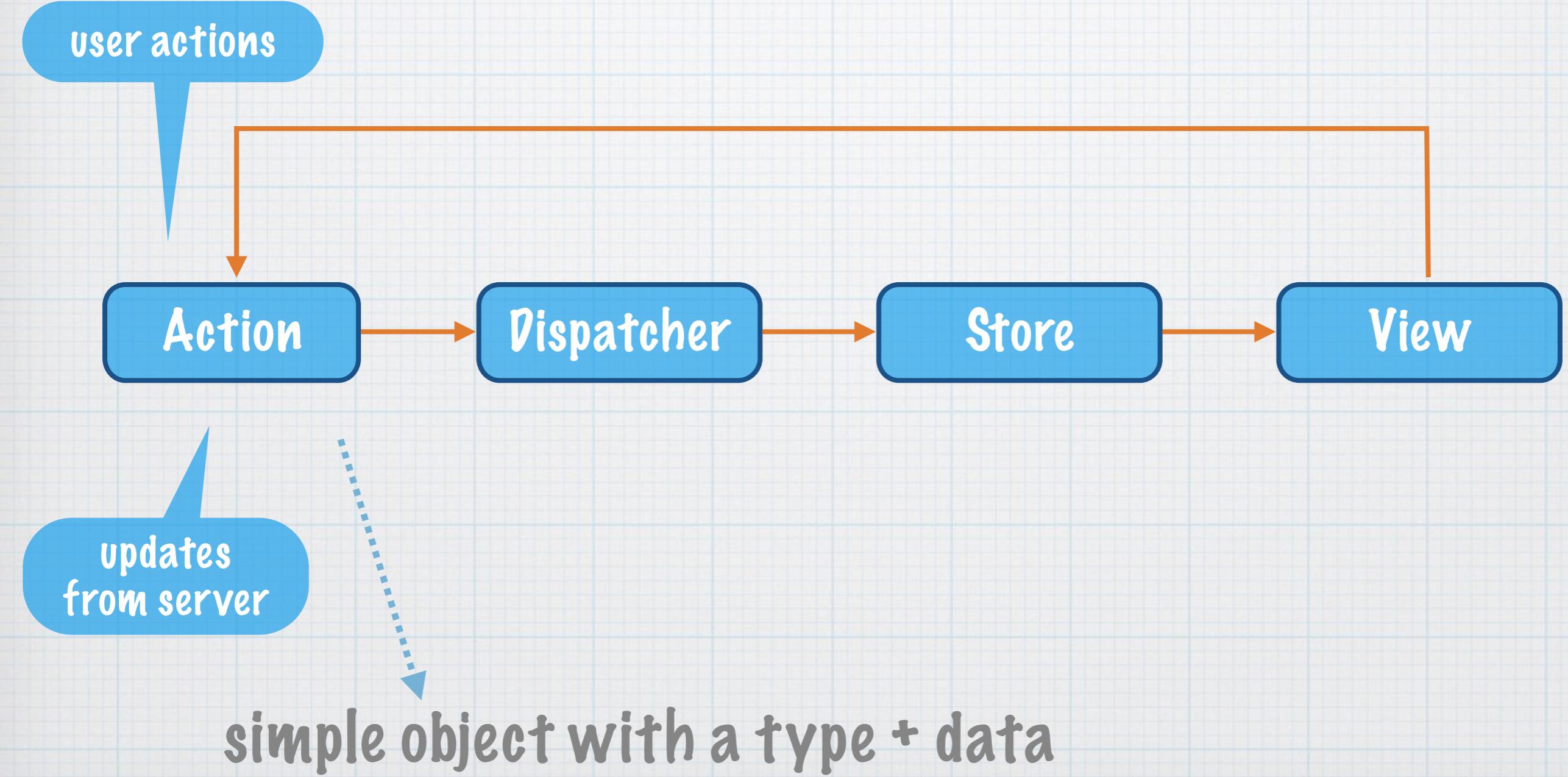
Flux: uni-directional data flow



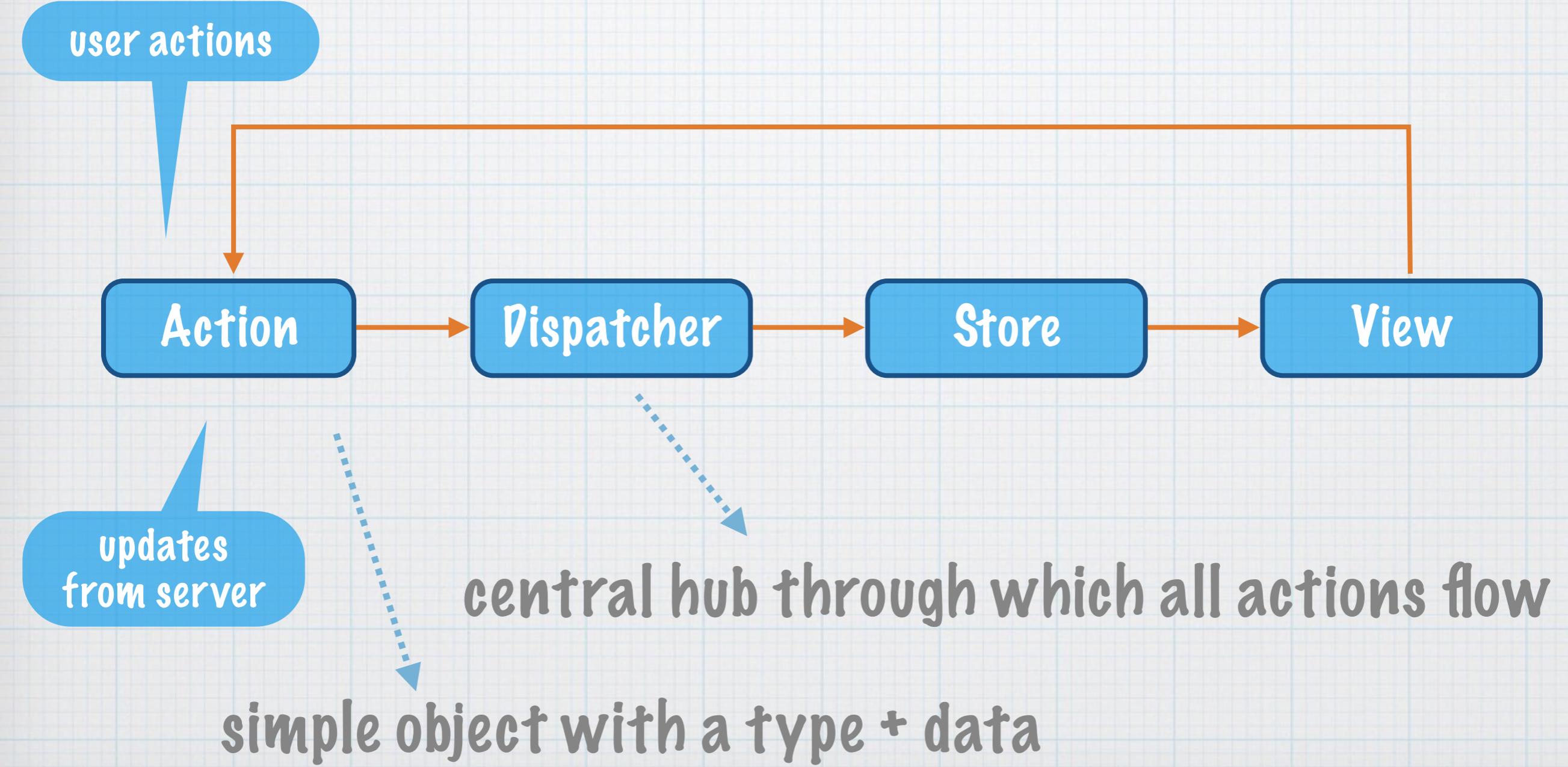
Flux: uni-directional data flow



Flux: uni-directional data flow



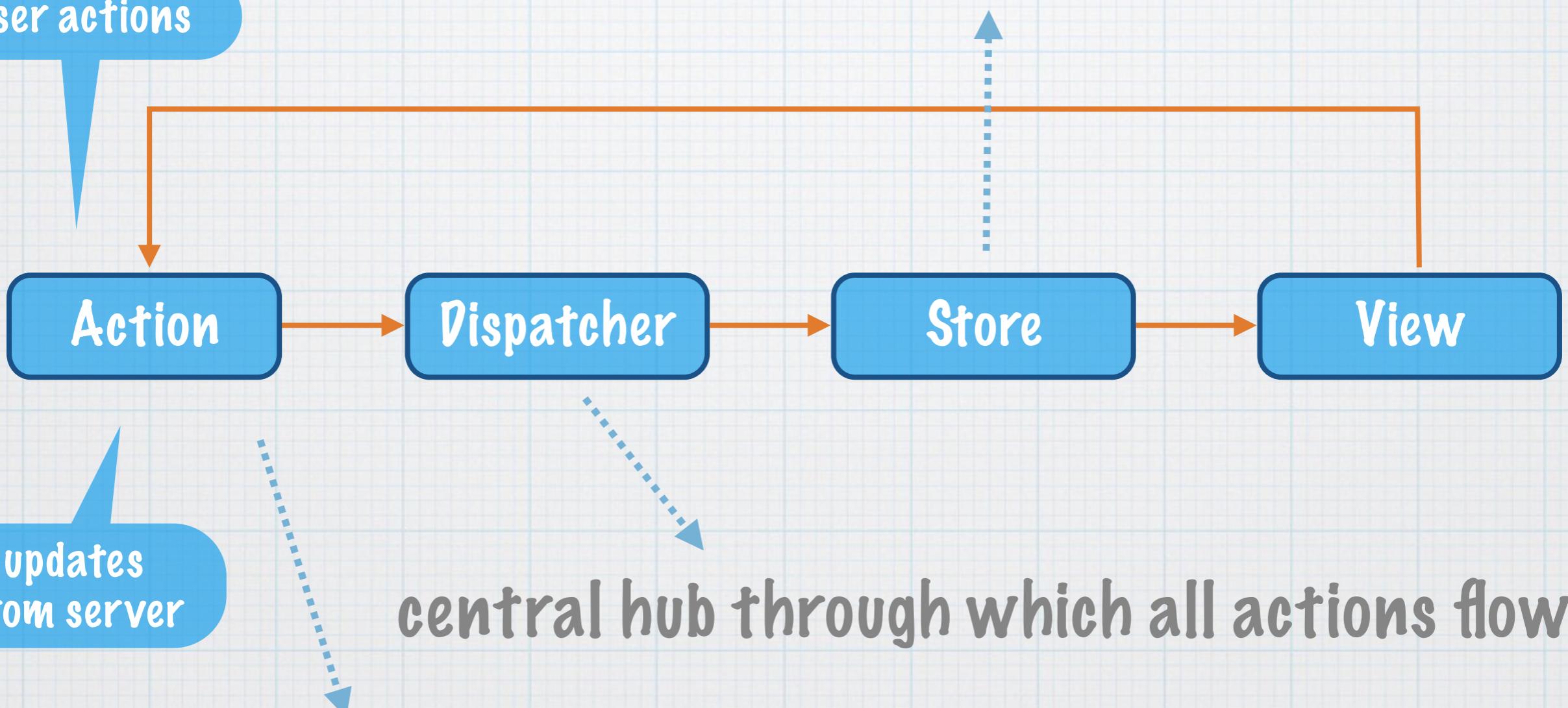
Flux: uni-directional data flow



Flux: uni-directional data flow

registers callback on dispatcher to receive actions

user actions



updates
from server

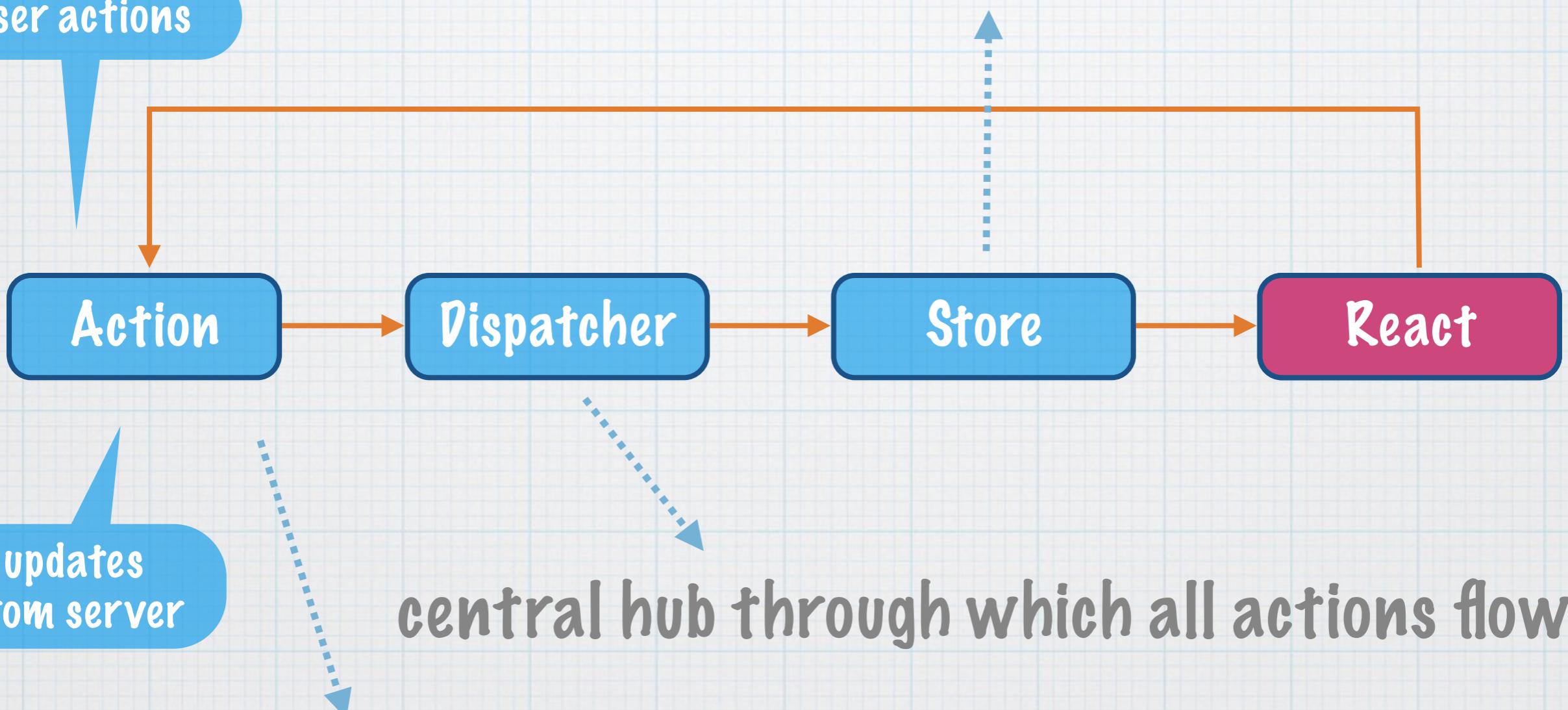
central hub through which all actions flow

simple object with a type + data

Flux: uni-directional data flow

registers callback on dispatcher to receive actions

user actions

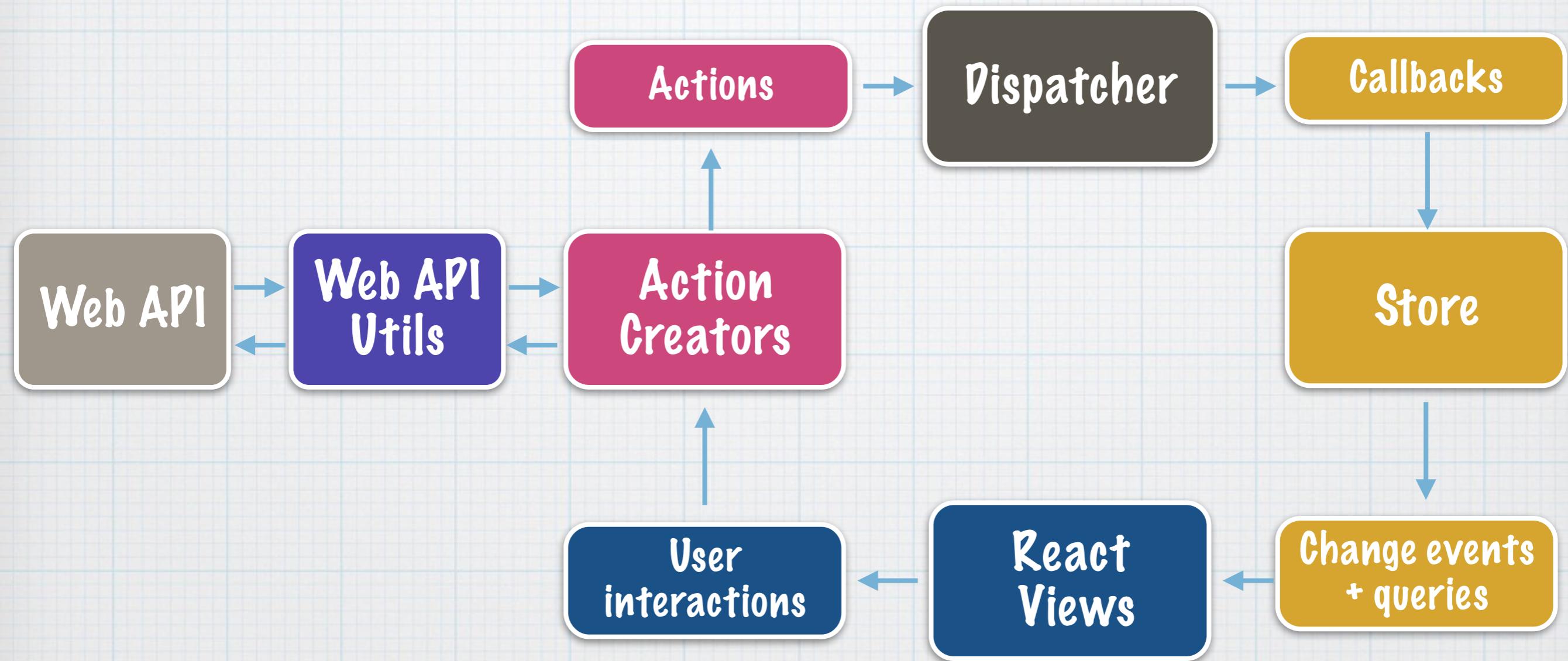


updates
from server

central hub through which all actions flow

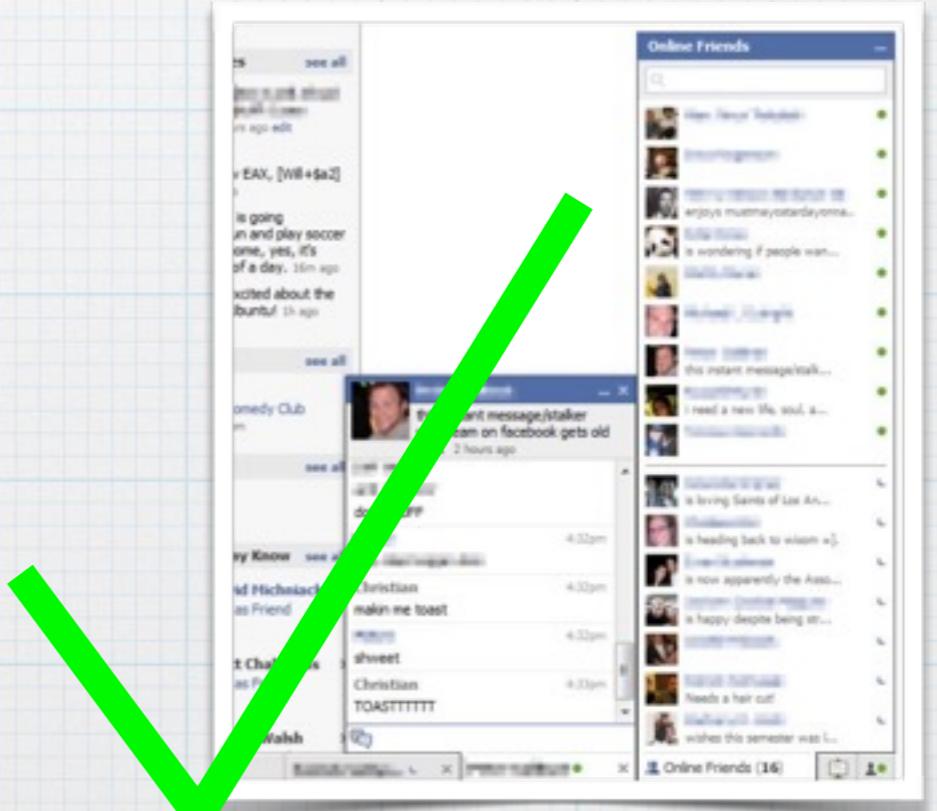
simple object with a type + data

Flux: uni-directional data flow



When?

“Large applications with data that (frequently) changes over time”



in MVC the controller
does the correct
propagation

Stores

- * Store contains data + logic about a certain **domain**. They can manage multiple “records”, so not always like models.
- * Accepts updates and appropriately reconciles them = **inversion of control**
- * All **update patterns are internal** to the store, nobody knows about it.
- * No external update methods like “`setColor(color)`”, only the callback registered to dispatcher.

Flux Pattern Implementation

- * Stores -> EventEmitter
- * Views -> React
- * Action -> just an object
- * Action Creators -> Lib of functions
- * Web API (utils) -> Lib of functions
- * Dispatcher -> require('flux').Dispatcher

File Structure

actions

KittenActionCreators.js

components

KittenApp.js

Header.js

Footer.js

KittenList.js

constants

AppConstants.js

utils

KittenWebAPI.js

KittenWebUtils.js

dispatcher

AppDispatcher.js

app.js

stores

KittenStore.js

Intermezzo

foo.js

```
var private = 42;
function privateFunction() { ... }
module.exports = {
  exportedFunction: function () { privateFunction(); }
};
```

bar.js

```
var foo = require('./foo.js');
foo.exportedFunction()
```

* node.js require module system

* browserify

```
> browserify src/app.js public/bundle.js
```

utils/KittenWebAPIUtils.js

```
socket.on('kitten:new', function (kitten) {
  KittenActionCreators.receiveKitten(kitten);
});
```

actions/KittenActionCreators.js

```
module.exports = {
  receiveKitten: function (kitten) {
    AppDispatcher.dispatch({
      type: ActionTypes.RECEIVE_RAW_KITTEN,
      rawKitten: kitten
    });
  }
};
```

constants/AppConstants.js

```
module.exports = {
  ActionTypes: {
    RECEIVE_RAW_KITTEN: "RECEIVE_RAW_KITTEN"
  }
};
```

AppDispatcher.js

```
var Dispatcher = require('flux').Dispatcher;
module.exports = new Dispatcher();
```

utils/KittenWebUtils.js

```
module.exports = {
  convertRawKitten: function (rawKitten) {
    return {
      name: rawKitten.name,
      color: rawKitten.color,
      DateOfBirth: new Date(rawKitten.DateOfBirth)
    };
  }
};
```

stores/KittenStore.js

```
var assign          = require('object-assign');
var EventEmitter = require('events').EventEmitter;
var utils          = require('../utils/KittenWebUtils');
var AppDispatcher = require('../dispatcher/AppDispatcher');
var ActionTypes   = require('../constants/AppConstants').ActionTypes;

var kittens = [];
var CHANGE_EVENT = 'change';

function _addRawKitten(rawKitten) {
  kittens.push(utils.convertRawKitten(kitten));
}

var KittenStore = assign({}, EventEmitter.prototype, {
  ...
  getAll: function () {
    return kittens;
  },
  getSortedByDateOfBirth: function () {
    return kittens.sortBy(function (kitten) { return kitten.DateOfBirth; });
  }
});

KittenStore.dispatchToken = AppDispatcher.register(function (action) {
  switch(action.type) {
    case ActionTypes.RECEIVE_RAW_KITTEN:
      _addRawKitten(action.rawKitten);
      KittenStore.emitChange();
      break;
  }
});
module.exports = KittenStore;
```

```
var KittenStore = ... {
  emitChange: function () {
    this.emit(CHANGE_EVENT);
  },
  addChangeListener: function (cb) {
    this.on(CHANGE_EVENT, cb);
  },
  removeChangeListener: function (cb) {
    this.removeListener(CHANGE_EVENT, cb);
  }
}
```

components/KittenList.js

```
var KittensBox = React.createClass({
  getInitialState: function () {
    return {kittens: KittenStore.getSortedByDateOfBirth()};
  },
  componentDidMount: function () {
    KittenStore.addChangeListener(this._onChange);
  },
  componentWillUnmount: function () {
    KittenStore.removeChangeListener(this._onChange);
  },
  _onChange: function () {
    this.setState({kittens: KittenStore.getSortedByDateOfBirth()});
  },
  render: function () {
    return (
      <div id="kittens">
        <h1> These are my kittens </h1>
        <KittenList kittens={this.state.kittens} />
      </div>
    );
  }
});
```

Dispatch: WaitFor*

```
KittenStore.dispatchToken = AppDispatcher.register(function (action) {  
  switch(action.type) {  
    case ActionTypes.RECEIVE_RAW_KITTEN:  
      AppDispatcher.waitFor(PawStore.dispatchToken);  
      _addRawKitten(action.rawKitten);  
      KittenStore.emitChange();  
      break;  
  }  
});  
module.exports = KittenStore;
```

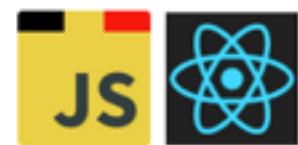
To work!

* <https://github.com/ticup/jsconfbe2015>

Common Component Problems

- * attributes: “class” -> “className”
- * Uncaught TypeError: type.toUpperCase is not a function
-> One of your component files isn’t exporting a Component
- * is render returning something? (one element)

State Update: Inversion Of Control



React + Flux Workshop

JSConfBe

Enter Your Name...

□ 1

First one to finish this page in proper React + Flux style gets a price (and obviously fame)

Task 1: Determine your Stores

Task 10: The protocol

Task 11: A Dynamic Page with State Updated by the Server

```
var MainComponent = React.createClass({
  getInitialState: function () {
    return {taskFilter: "", hideQuestions: false};
  },
  _setTaskFilter: function (val) {
    this.setState({taskFilter: val});
  },
  _setHideQuestions: function (val) {
    this.setState({hideQuestions: val});
  },
  render: function () {
    <Options setTaskFilter={this._setTaskFilter}
             setHideQuestion={this._setHideQuestions} />
    <Tasks taskFilter={this.state.taskFilter}
           hideQuestions={this.state.hideQuestions} />
  }
});
```

State Update: Inversion Of Control++

```
var MainComponent = React.createClass({
  getInitialState: function () {
    return {taskFilter: OptionStore.getTaskFilter(),
            hideQuestions: OptionStore.getHideQuestions()};
  },
  componentDidMount: function () {
    OptionStore.addChangeListener(this._onChange);
  },
  _onChange: function () {
    this.setState({OptionStore.getTaskFilter(), OptionStore.getHideQuestions()});
  },
  render: function() {
    <Options />
    <Tasks taskFilter={this.state.taskFilter}
          hideQuestions={this.state.hideQuestions} />
  }
});
```



React

JSCom

Enter Your Name...

First one to finish this page in proper

 Task 1: Determine y

});

 Task 10: The protocol Task 11: A Dynamic Page with State Updated by the Server