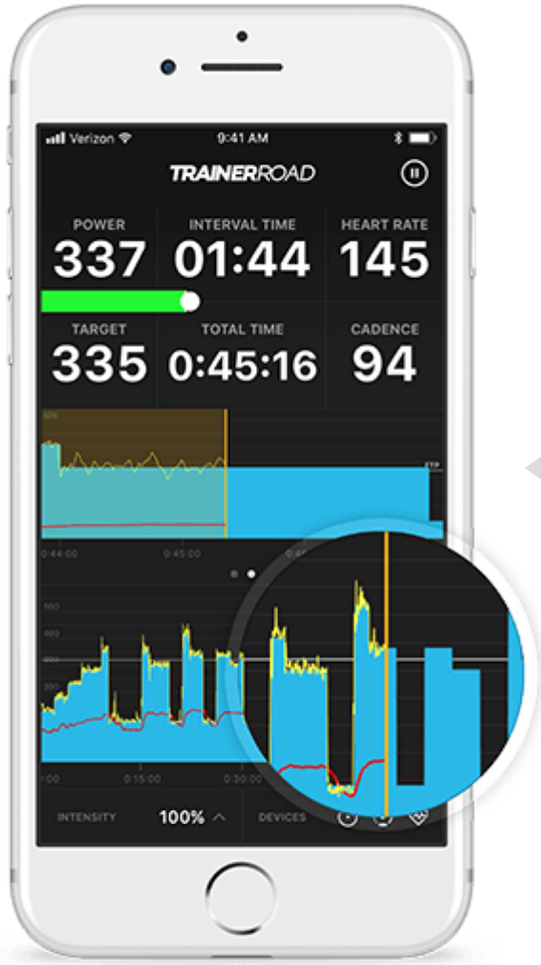
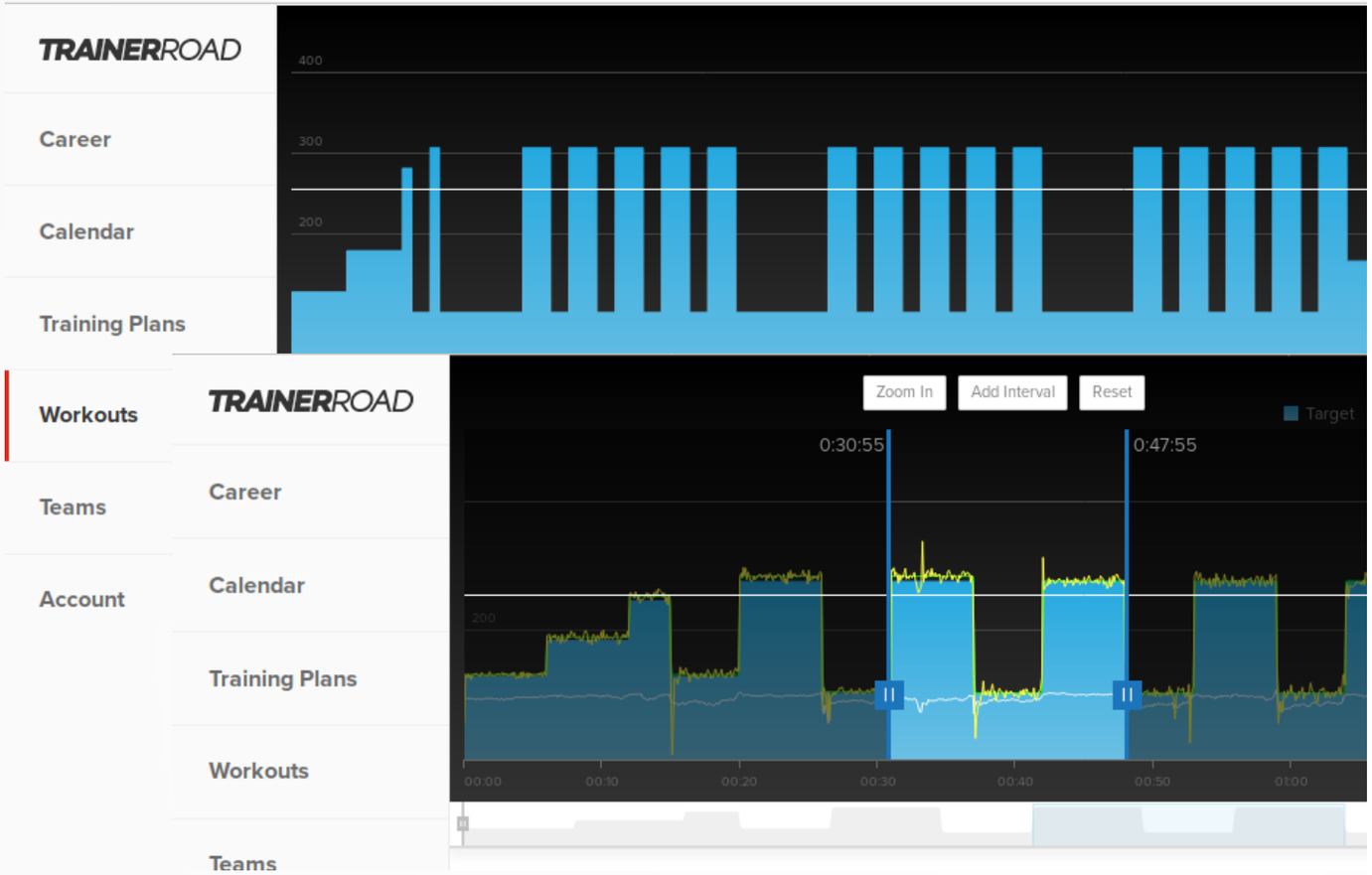
React / React Native

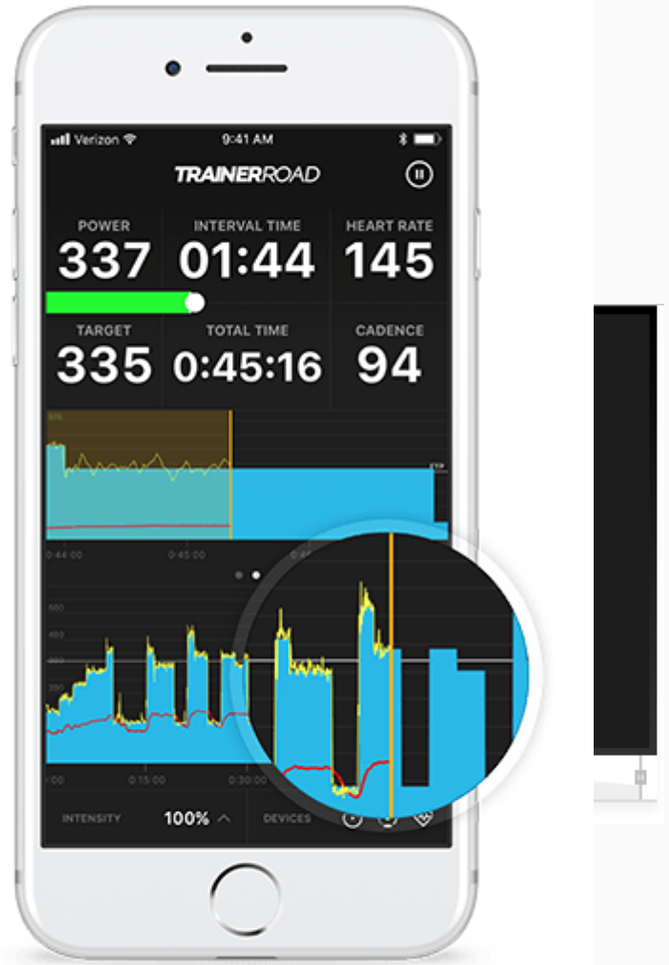


Communicate over
Bluetooth / ANT+

Trainerroad
Cycling Workout / Training App

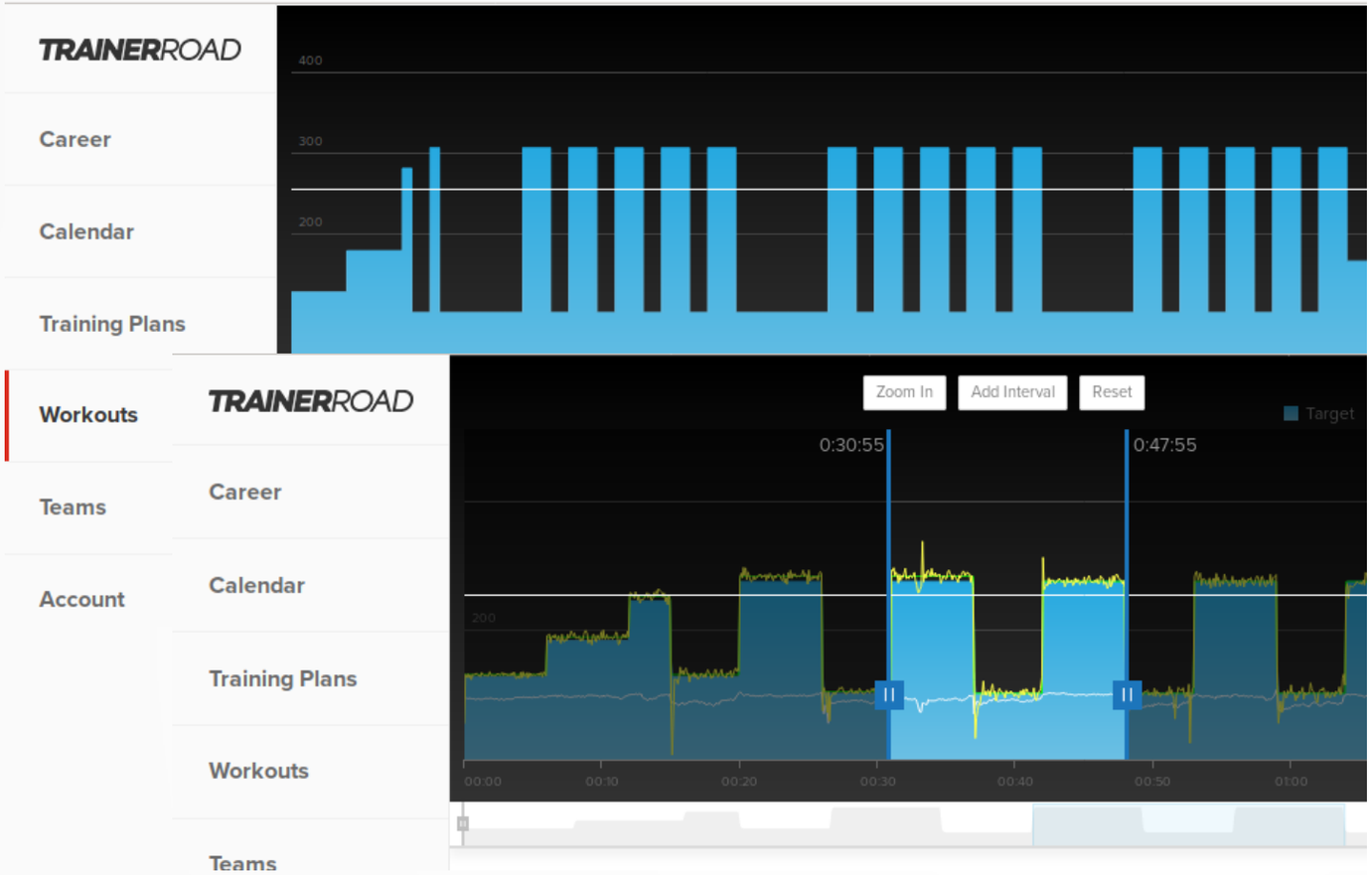
Web provides the “platform”

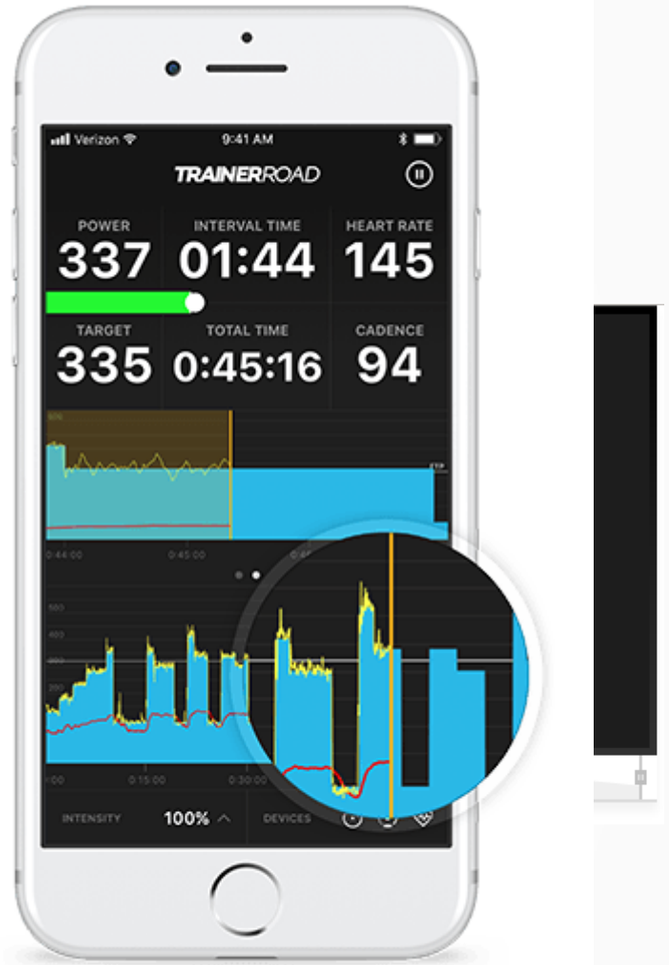




Web provides the “platform”

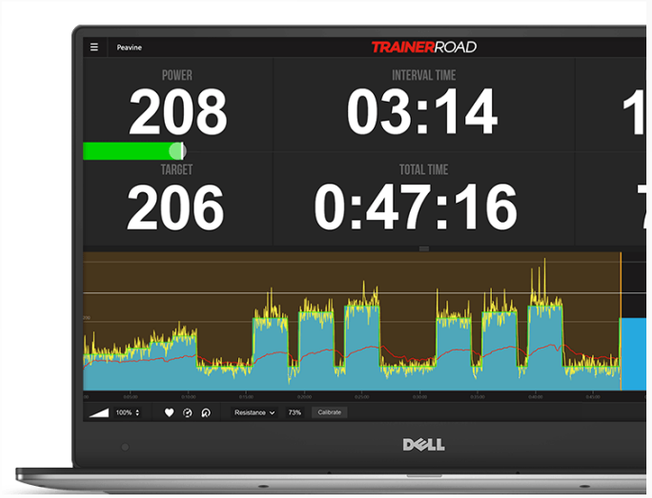
Phone interfaces to device





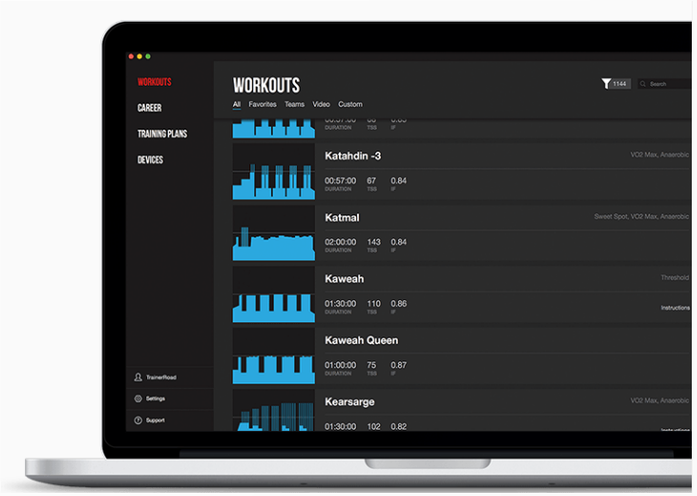
Web provides the “platform”

Phone interfaces to device

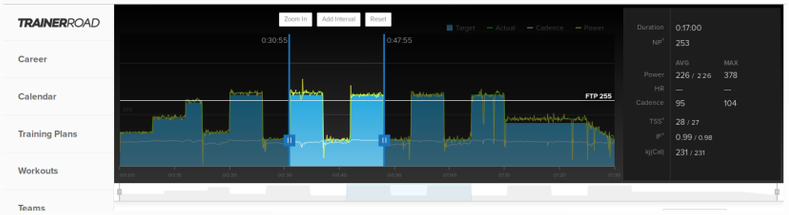


Challenge 1: Multiple Devices

Challenge 2: Half of platform web app anyways



Solution: Work with a
platform for web and native, if possible



Web Based GUI

Core UI framework in JS

Reuse framework with Native interface

The Emerging Platform

- We care about **React + React Native**
- But we need to understand React first
- In isolation, React is a fun framework. If it interests you, I encourage you to pursue it further.

React Hello World

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

React Element

Container

Displays “Hello, word!” in the browser

ReactDOM is a link between React and the outer world

React Hello World

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

React Element

Container

Displays “Hello, word!” in the browser

ReactDOM is a link between React and the outer world

JSX

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

React Element

Container

Displays “Hello, word!” in the browser

ReactDOM is a link between React and the outer world

JSX

Embed JS in { }

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

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

Pass JSX around like any other value

JSX syntax extension for javascript

Produces React elements from HTML-like syntax

JSX

```
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);
```

```
const element = React.createElement(
  'h1',
  {className: 'greeting'},
  'Hello, world!'
);
```

```
const element = {
  type: 'h1',
  props: {
    className: 'greeting',
    children: 'Hello, world!'
  }
};
```

JSX compiles down to React Elements for ease of use

React Components

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Components are reusable UI pieces

Build up React elements for rendering

React Components

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Take properties (props) as input, return JSX

Write them as functions, use them as HTML-like

React Components

- React / JS has a nice side effect of live coding
- https://codepen.io/acanino1/pen/bJKPRa?editors=0010

Exercise

- https://codepen.io/acanino1/pen/BEPmwd
- Make the following changes to Person list
  - 1. Create a Person component that encapsulates the <li> for a single person, add **age**
  - 2. Refactor people array into a array of JSON objects that represents people
  - 3. Refactor PersonList to create <Person/> for each person in the JSON object
  - 4. Create an a Pet component which will have a **name** and a **kind**
  - 5. Refactor JSON to include an array of pets for each person
  - 6. Refactor Person to render a sublist for each pet, per person

React Components

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  } }
```

Create a class component by extending React.Component

render() method drives the rendering

React Components

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  } }
```

Done mostly for **state**

React Components

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.timerID = setInterval(() => this.tick(), 1000);
  }

  tick() { this.setState({ date: new Date() }) };

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  } }
```

Fired every 1s, triggers re-render

Update state with **setState**

React State

- Props and State drive rendering of the app
- Changes to props and state cause component to get re-rendered, and has implication performances
- As such, state updates have semantics optimized for UI
  - 1. Updates may be asynchronous
  - 2. Independent updates are merged
  - 3. State local to component (top-down flow)

React State (Async Updates)

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

Not guaranteed both are “current”

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

Function will receive previous state, plus a snapshot of props

Note on Syntax

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



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

React State (Independent Updates)

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

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

Updates merged back into state, but do not affect one another

Could be any of...
{old.posts,old.comments}
{old.posts,response.comments}
{response.posts,old.comments}
{response.posts,response.comments}

If state has independent variables, update independently

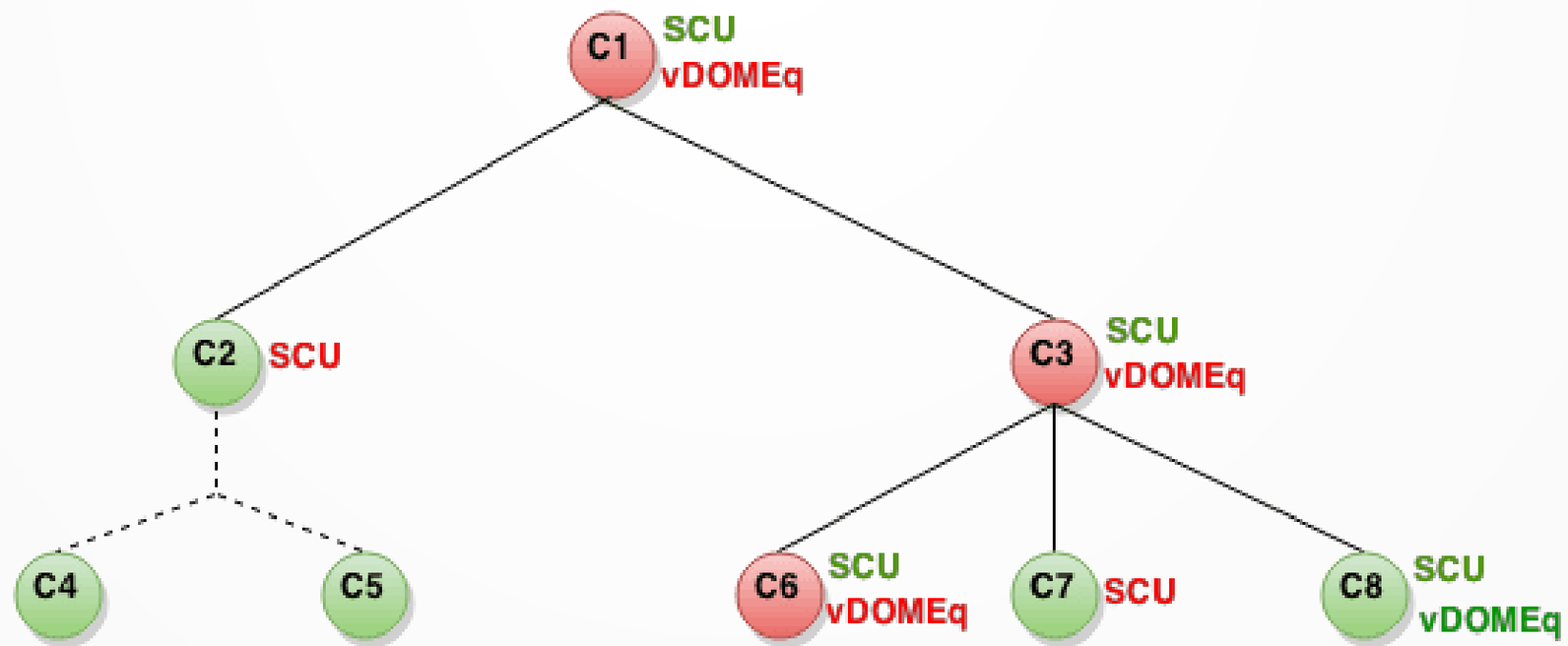
React State

- https://codepen.io/acanino1/pen/bJKPRa?editors=0010

Exercise

- https://codepen.io/acanino1/pen/VNBrze
- Try and complete the todo app
  - State is held in TodoApp
  - Create a TodoList, similar to lists demonstrated
    - Use JSON to build an item, which consists of text, and a date
    - Render both
  - Update **items** in handleSubmit
    - Date.now()
    - items.concat(...)

Why state matters?



No Reconciliation needed

Reconciliation needed

SCU shouldComponentUpdate?
SCU

vDOMEq are virtual DOMs equivalent?
vDOMEq

Some more fun

- https://codepen.io/acanino1/pen/gydYbq

Acknowledgments

- https://reactjs.org/
- https://www.trainerroad.com/
-