TODO: Prepping for react-fiber

Top Priority

change these asap

refs

BEFORE

```
…
render () {
  return (
    <div ref=”special”>
      ...
    </div>
  );
}


…
this.refs.special
…
```

AFTER

```
…
render () {
  return (
    <div ref={ref => { this.SPECIAL = ref; }}>
      ...
    </div>
  );
}


…
this.SPECIAL
…
```

React.PropTypes

BEFORE

```
import React, { Component } from 'react';


class Example extends Component {
  render() {
    return <div>{this.props.text}</div>;
  }
}

Example.propTypes = {
  text:
React.PropTypes.string.isRequired,
}
```

AFTER

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';

class Example extends React.Component {
  render() {
    return <div>{this.props.text}</div>;
  }
}

Example.propTypes = {
  text: PropTypes.string.isRequired,
};
```

React.createClass

BEFORE

```
import React from ‘react’;

var Component = React.createClass({
  mixins: [MixinA],
  render() {
    return <Child />;
  }
});
```

AFTER

```
import React from 'react';
import createReactClass from
'create-react-class';

var Component = createReactClass({
  mixins: [MixinA],
  render() {
    return <Child />;
  }
});
```

ADDONS

https://github.com/facebook/react/issues/9207

- Don’t Do Anything
- Fix and Forget
- Deprecate and Forget
- Undecided

Addons: Don’t Do Anything

- react-addons-perf
    - :(

Addons: Fix and Forget

- react-addons-create-fragment
- react-addons-linked-state-mixin
- react-addons-pure-render-mixin (use React.PureComponent)
- react-addons-shallow-compare (use React.PureComponent)
- react-addons-update
- react-linked-input

Addons: Deprecate and Forget

- react-addons-css-transition-group
- react-addons-transition-group

Addons: Undecided

- react-addons-test-utils

Extras

mandatory for ninjas

Updater Style `setState()`

BEFORE

```
constructor() {
  this.state = {
    foo: 0,
  };
}

componentWillReceiveProps(nextProps) {
  this.setState({
    foo: this.state.foo + 2,
  });

  this.setState({
    foo: this.state.foo + 1,
  });
}
```

AFTER

```
componentWillReceiveProps(nextProps) {
  this.setState(prevState => ({
    foo: prevState.foo + 2,
  }));

  this.setState(prevState => ({
    foo: prevState.foo + 1,
  }));
}
```

React.PropTypes: Flow

BEFORE

```
import React, { Component } from 'react';


class Example extends Component {
  render() {
    return <div>{this.props.text}</div>;
  }
}

Example.propTypes = {
  text:
React.PropTypes.string.isRequired,
}
```

AFTER

```
import React, { Component } from 'react';

type Props = { text: string };

class Example extends Component {
  props: Props;

  render() {
    return <div>{this.props.text}</div>;
  }
}
```

React.PropTypes: TypeScript

BEFORE

```
import React, { Component } from 'react';


class Example extends Component {
  render() {
    return <div>{this.props.text}</div>;
  }
}

Example.propTypes = {
  text: React.PropTypes.string.isRequired,
}
```

AFTER

```
import React, { Component } from 'react';

interface Props { text: string; }

class Example extends Component<Props, {}> {

  render() {
    return <div>{this.props.text}</div>;
  }
}
```