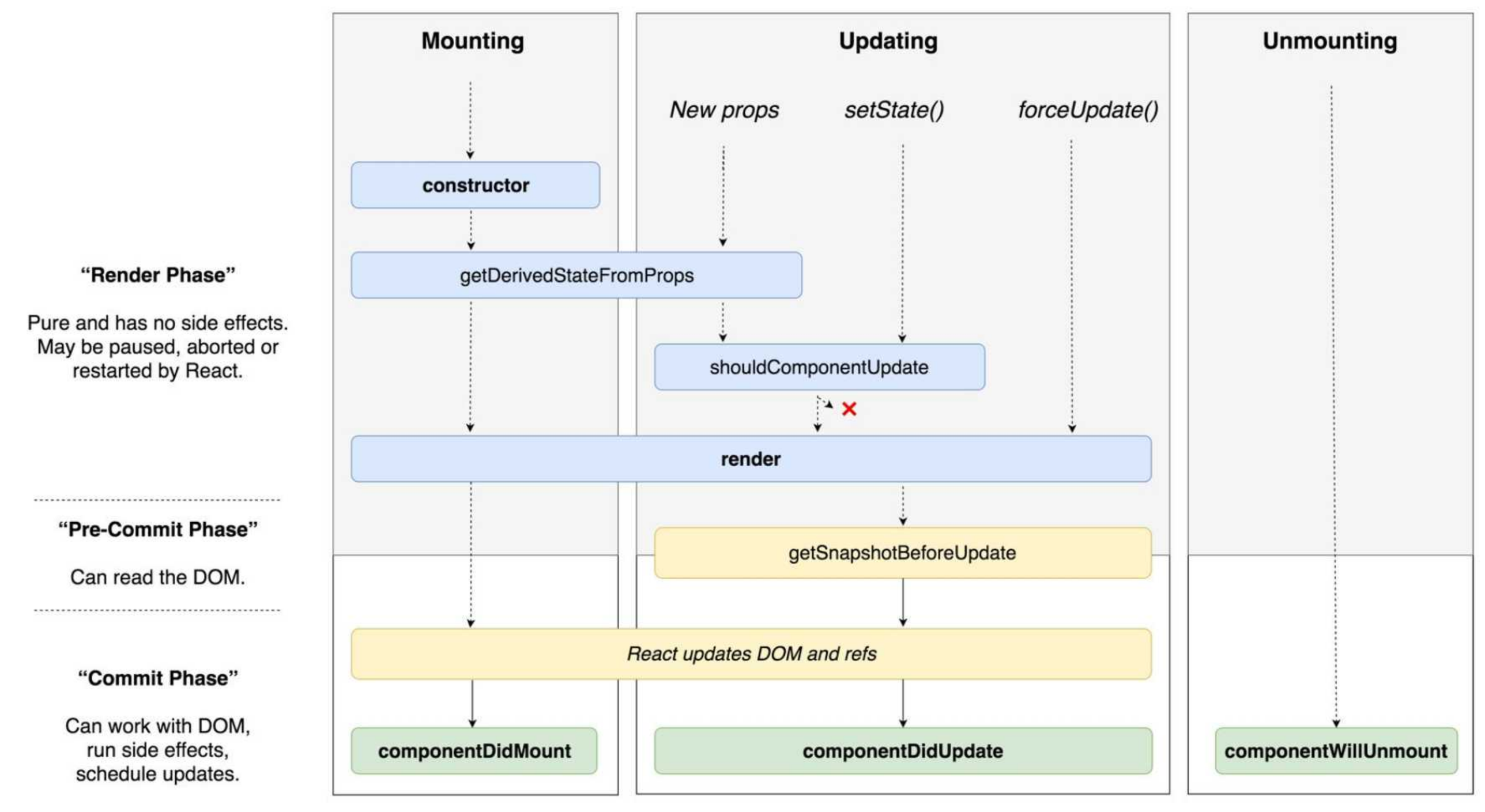
Mounting
Updating
Unmounting
New props
setState()
forceUpdate()
constructor
getDerivedStateFromProps
shouldComponentUpdate
render
getSnapshotBeforeUpdate
React updates DOM and refs
componentDidMount
componentDidUpdate
componentWillUnmount
"Render Phase"
Pure and has no side effects. May be paused, aborted or restarted by React.
"Pre-Commit Phase"
Can read the DOM.
"Commit Phase"
Can work with DOM, run side effects, schedule updates.

egghead.io

React Cheat Sheet (0.14.7)


```
import React from 'react'
```

```
class ExampleComponent extends React.Component { ... } ⇒ ReactClass
```

Create a component class, given a specification. A component implements a render method which returns one single child.

```
React.createClass( ReactComponent|Specification ) ⇒ ReactClass
```

Equivalent to above ES6 class notation. **NOTE:** Prefer the above ES6 usage

```
React.createElement( HTMLTag String|ReactClass , {PROPS}? , [CHILDREN...]? ) ⇒ ReactElement
```

Create and return a new ReactElement of the given type.

↳ JSX nodes desugar into createElement() calls, e.g. <Node /> becomes React.createElement(Node ...)

```
React.cloneElement( ReactElement , {PROPS}? , [CHILDREN...]? ) ⇒ ReactElement
```

Clone and return a new ReactElement using element as the starting point with shallow merged props.

```
React.isValidElement( ReactElement ) ⇒ Boolean
```

Verifies the object is a ReactElement.

```
import ReactDOM from 'react-dom'
```

```
ReactDOM.render( ReactElement , DOMElement , CALLBACK? ) ⇒ ReactComponent
```

Render a ReactElement into the DOM into supplied DOMElement.

↳ ReactDOM.render(<ExampleComponent />, document.getElementById('react-app'))

```
ReactDOM.findDOMNode( ReactComponent ) ⇒ DOMElement
```

If this component has been mounted into the DOM, this returns the corresponding native browser DOM element.

```
ReactDOM.unmountComponentAtNode( DOMElement ) ⇒ Boolean
```

Remove a mounted React component from the DOM and clean up its event handlers and state.

```
import ReactDOMServer from 'react-dom/server'
```

```
ReactDOMServer.renderToString( ReactElement ) ⇒ String
```

Render a ReactElement to its initial HTML.

```
ReactDOMServer.renderToStaticMarkup( ReactElement ) ⇒ String
```

Similar to renderToString, except this doesn't create extra DOM attributes such as data-react-id, that React uses internally.

Component API `ExampleComponent extends React.Component {...}`

METHODS

```
setState( Function *|{nextState} , CALLBACK? ) ⇒ void
```

Performs a shallow merge of nextState into current state and triggers UI update. Callback after update. NEVER mutate *this.state*.

↳ * **Function Signature:** (previousState, currentProps) => {stateVariable: newValue, ...}

CORE

```
forceUpdate( CALLBACK? ) ⇒ void
```

Calling forceUpdate() will cause render() to be called on the component, skipping shouldComponentUpdate(). Avoid usage.

```
render() ⇒ ReactElement|void|null
```

A pure function that returns a ReactElement which relies upon props and state. REQUIRED.

```
constructor( props ) { super(props); this.state = {...} } ⇒ StateObject
```

Invoked once before the component is mounted, returns this.state.

LIFECYCLE METHODS

```
componentWillMount() ⇒ void
```

Invoked once, both on the client and server, immediately before the initial rendering occurs.

```
componentDidMount() ⇒ void
```

Invoked once, only on the client (not on the server), immediately after the initial rendering occurs.

```
componentWillReceiveProps( {NextProps} ) ⇒ void
```

Invoked when a component is receiving new props. This method is not called for the initial render.

```
shouldComponentUpdate( {NextProps} , {NextState} ) ⇒ Boolean
```

Invoked before rendering when new props or state are being received. Not called on initial render or when forceUpdate is used.

```
componentWillUpdate( {NextProps} , {NextState} ) ⇒ void
```

! Cannot use this.setState()

Invoked immediately before rendering when new props or state are being received. This method is not called for the initial render.

```
componentDidUpdate( {PreviousProps} , {PreviousState} ) ⇒ void
```

Invoked immediately after the component's updates are flushed to the DOM. This method is not called for the initial render.

```
componentWillUnmount() ⇒ void
```

Invoked immediately before a component is unmounted from the DOM.

Supported Tags in JSX

HTML Elements

a abbr address area article aside audio b base bdi bdo big blockquote body br button canvas caption cite code col colgroup data datalist dd del details dfn dialog div dl dt em embed fieldset figcaption figure footer form h1 h2 h3 h4 h5 h6 head header hgroup hr html i iframe img input ins kbd keygen label legend li link main map mark menu menuitem meta meter nav noscript object ol optgroup option output p param picture pre progress q rp rt ruby s samp script sec vvbvtion select small source span strong style sub summary sup table tbody td textarea tfoot th thead time title tr track u ul var video wb

HTML Attributes

data -* aria -* accept acceptCharset accessKey action allowFullScreen allowTransparency alt async autoComplete autoFocus autoPlay capture cellPadding cellSpacing challenge charSet checked classID className colSpan cols content contentEditable contextMenu controls coords crossOrigin data dateTime default defer dir disabled download draggable encType form formAction formEncType formMethod formNoValidate formTarget frameBorder headers height hidden high href hrefLang htmlFor httpEquiv icon id inputMode integrity is keyParams keyType kind label lang list loop low manifest marginHeight marginWidth max maxLength media mediaGroup method min minLength multiple muted name noValidate nonce open optimum pattern placeholder poster preload radioGroup readOnly rel required reversed role rowSpan rows sandbox scope scoped scrolling seamless selected shape size sizes span spellCheck src srcDoc srcLang srcSet start step style summary tabIndex target title type useMap value width wmode wrap

RDFa: about datatype inlist prefix property resource typeof vocab

SVG Elements

circle clipPath defs ellipse g image line linearGradient mask path pattern polygon polyline radialGradient rect stop svg text tspan

SVG Attributes

clipPath cx cy d dx dy fill fillOpacity fontFamily fontSize fx fy gradientTransform gradientUnits markerEnd markerMid markerStart offset opacity patternContentUnits patternUnits points preserveAspectRatio r rx ry spreadMethod stopColor stopOpacity stroke strokeDasharray strokeLinecap strokeOpacity strokeWidth textAnchor transform version viewBox x1 x2 x xlinkAcvtuate xlinkArcrole xlinkHref xlinkRole xlinkShow xlinkTitle xlinkType xmlBase xmlLang xmlSpace y1 y2 y

Component API *(cont'd)*

NON-DOM TAGS

key	`<ExampleComponent key="uniqueValue" />`	An optional, unique identifier. When your component shuffles around during render passes, it might be destroyed and recreated
ref	`<ExampleComponent ref={String | Callback} />`	Reference to the React Component. ReactDOM.FindDOMNode(ref). If a callback is used, the component will be passed to the function.
dangerouslySetInnerHTML	`<span dangerouslySetInnerHTML={__HTML: String} />`	Provides the ability to insert raw HTML, mainly for cooperating with DOM string manipulation libraries.

USEFUL PROPERTIES AND FEATURES

this.props.children	`<Component>{this.props.children}</Component>`	Will contain any nested children passed in from the parent component.
...	`<ExampleComponent {...this.props} />`	The Spread Operator (...) can be used to extract the entirety of an object without the need to define every key.
Stateless Syntax	`var HelloMsg ⇒ function(props) { return <div>Hello {props.name}</div> }`	This defines a stateless functional component. Can ReactDOM.render(<HelloMsg name="John" />.

PROPERTIES

`ReactComponentClass.defaultProps = DefaultPropertiesObject`	This object defines the initial props values. It is cached and invoked once when a class is instantiated.
`ReactComponentClass.propTypes = PropertiesSpecificationObject`	The PropertiesSpecificationObject defines the contract a parent component must comply with when providing properties.

↳ The PropertiesSpecificationObject can define the following property types (they are optional by default):

- → React.PropTypes.**array**
- → React.PropTypes.**bool**
- → React.PropTypes.**func**
- → React.PropTypes.**number**
- → React.PropTypes.**object**
- → React.PropTypes.**string**
- → React.PropTypes.**node** (ANYTHING THAT CAN BE RENDERED)
- → React.PropTypes.**element** (REACTELEMENT)
- → React.PropTypes.**instanceOf**(*Message*) (MUST BE OF JAVASCRIPT TYPE)
- → React.PropTypes.**oneOf**(['News', 'Photos']) (SPECIFY ENUMERATED VALUES)
- → React.PropTypes.**oneOfType**([React.PropTypes.string, React.PropTypes.number]) (LIMIT PROPERTY TYPES)
- → React.PropTypes.**arrayOf**(React.PropTypes.number) (LIMIT TO A TYPED ARRAY)
- → React.PropTypes.**objectOf**(React.PropTypes.number) (LIMIT TO A TYPED OBJECT)
- → React.PropTypes.**shape**({color: React.PropTypes.string, fontSize: React.PropTypes.number}) (LIMIT TO OBJECT WITH SPECIFIC KEYS/TYPES)
- → React.PropTypes.**func.isRequired** (PRODUCE AN ERROR IF THE PROPERTY ISN'T PASSED TO THE CHILD)
- → React.PropTypes.**any.isRequired** (CAN BE ANY OBJECT BUT MUST BE REQUIRED)
- → (props, propName, componentName) => Boolean (CREATE A CUSTOM PROPERTY WITH THE FOLLOWING FUNCTION SIGNATURE)

↳ Example: *ReactComponent*.**propTypes** = { optionalArray: React.PropTypes.array, requiredFunction: React.PropTypes.func.isRequired };

JSX Events

Synthetic Event *(default callback arg)*
{
- Boolean bubbles
- Boolean cancelable
- DOMEventTarget currentTarget
- Boolean defaultPrevented
- Number eventPhase
- Boolean isTrusted
- DOMEvent nativeEvent
- void preventDefault()
- Boolean isDefaultPrevented()
- void stopPropagation()
- Boolean isPropagationStopped()
- DOMEventTarget target
- Number timeStamp
- String type

}

Clipboard onCopy onCut onPaste
(DOMDataTransfer clipboardData)

Composition onCompositionEnd onCompositionStart onCompositionUpdate
(String data)

Keyboard onKeyDown onKeyPress onKeyUp
(Boolean altKey, Number charCode, Boolean ctrlKey, Boolean getModifierState(key), String key, Number keyCode, String locale, Number location, Boolean metaKey, Boolean repeat, Boolean shiftKey, Number which)

Focus onFocus onBlur
(DOMEventTarget relatedTarget)

Form onChange onInput onSubmit

Mouse onClick onContextMenu onDoubleClick onDrag onDragEnd onDragEnter onDragExit onDragLeave onDragOver onDragStart onDrop onMouseDown onMouseEnter onMouseLeave onMouseMove onMouseOut onMouseOver onMouseUp
(Boolean altKey, Number button, Boolean buttons, Number clientX, Number clientY, Boolean ctrlKey, Boolean getModifierState(key), Boolean metaKey, Number pageX, Number pageY, DOMEventTarget relatedTarget, Number screenX, Number screenY, Boolean shiftKey)

Selection onSelect

Touch onTouchCancel onTouchEnd onTouchMove onTouchStart
(Boolean altKey, DOMTouchList changedTouches, Boolean ctrlKey, Boolean getModifierState(key), Boolean metaKey, Boolean shiftKey, DOMTouchList targetTouches, DOMTouchList touches)

UI onScroll (Number detail, DOMAbstractView view)

Wheel onWheel (Number deltaMode, Number deltaX, Number deltaY, Number deltaZ)

Media onAbort onCanPlay onCanPlayThrough onDurationChange onEmptied onEncrypted onEnded onError onLoadedData onLoadedMetadata onLoadStart onPause onPlay onPlaying onProgress onRateChange onSeeked onSeeking onStalled onSuspend onTimeUpdate onVolumeChange onWaiting

Image onLoad onError

egghead.io

Redux Cheat Sheet (3.2.1)

```
import React from 'react'
import ReactDOM from 'react-dom'
import { createStore, combineReducers,
        applyMiddleware, bindActionCreators } from 'redux'

const greetingReducer = (state='' ,  action) => {
  switch (action.type) {
    case 'SAY_HELLO': return 'Hello '
    case 'SAY_GOODBYE': return 'Goodbye '
  }
  return state
}

const nameReducer = (state='John', action) => {
  switch (action.type) {
    case 'CHANGE_NAME': return 'Joel'
  }
  return state
}

const actionLogger = ({dispatch, getState}) =>
  (next) => (action) =>
    { console.log(action); return next(action) }

const reducers = combineReducers({
  greeting: greetingReducer,
  name: nameReducer
})

const middleware = applyMiddleware(actionLogger)
const store = createStore(
  reducers,
  { greeting: '(Roll over me) '},
  middleware
)

const changeName = () => {return { type: 'CHANGE_NAME' }}
const hello = () => {return { type: 'SAY_HELLO' }}
const goodbye = () => {return { type: 'SAY_GOODBYE' }}

const Hello = (props) =>
  <div
    onMouseOver={props.hello}
    onMouseOut={props.goodbye}
    onClick={props.changeName}>
    {props.greeting}{props.name}
  </div>

const render = () => {
  ReactDOM.render(
    <Hello
      greeting={store.getState().greeting}
      name={store.getState().name}
      {...bindActionCreators({changeName, hello, goodbye},
                             store.dispatch)}
    />,
    document.getElementById('root')
  )
}

render()
store.subscribe(render)
```

Welcome to the egghead.io Redux cheat sheat! On your left you will find a full-fledged Redux application with a React.js front-end (React is not required).

`function reducer(State, Action) ⇒ State`

Takes the previous state and an action, and returns the next state.

Splitting your app into multiple reducers (`greetingsReducer`, `nameReducer`) allows for a clean separation of concerns when modifying your application's state.

`function middleware({Dispatch, getState}) ⇒ next ⇒ action`

Receives Store's `dispatch` and `getState` functions as named arguments, and returns a function. That function will be given the next middleware's dispatch method, and is expected to return a function of action calling `next(action)` with a potentially different argument, or at a different time, or maybe not calling it at all. The last middleware in the chain will receive the real store's `dispatch` method as the next parameter, thus ending the chain.

`combineReducers({Reducers}) ⇒ Function`

Combines multiple reducers into a single reducing function with each reducer as a key/value pair. Can then be passed to `createStore()`.

`applyMiddleware(...middleWares) ⇒ Function`

Extends Redux with custom functionality by wrapping the store's dispatch method.

`createStore(Reducer, ?initialState, ?enhancer) ⇒ Store`

Creates a Redux store that holds the complete state tree of your app.
There should only be a single store in your app.

`store = { ... }`

Brings together your application's state and has the following responsibilities:

- Allows access to state via `getState()`;
- Allows state to be updated via `dispatch(action)`;
- Registers listeners via `subscribe(listener)`;
- Handles unregistering of listeners via the function returned by `subscribe(listener)`.

`action = { type: String, ...payload: any }`

Holds action payloads in plain javascript objects. Must have a type property that indicates the performed action, typically be defined as string constants. All other properties are the action's payload.

`function actionCreator(?any) ⇒ Action|AsyncAction`

Creates an action with optional payload and bound dispatch.

`bindActionCreators(ActionCreators, Dispatch) ⇒ Fn | Obj`

Turns an object whose values are action creators, into an object with the same keys, but with every action creator wrapped into a dispatch call so they may be invoked directly.

Redux's Three Principles

- Single source of truth
- State is read-only
- Changes are made with pure functions

Glossary

State

```
type State = any
```

Action

```
type Action = {type: String, payload: any}
```

Reducer

```
type Reducer<State, Action> = (State, Action) => State
```

Dispatching Functions

```
type BaseDispatch = (Action) => Action
type Dispatch = (Action | AsyncAction) => any
```

Action Creator

```
type ActionCreator = (ANY) => Action | AsyncAction
```

Async Action

```
type AsyncAction = any
```

Middleware

```
type MiddlewareAPI = {dispatch: Dispatch, getState: () => State}
type Middleware = (MiddlewareAPI) => (Dispatch) => Dispatch
```

Store

```
type Store =
{
   dispatch(Action | AsyncAction) => any,
   getState() => State,
   subscribe(() => VOID) => () => void,
   replaceReducer(Reducer) => void
}
```

Store Creator

```
type StoreCreator = (Reducer, ?initialState, ?enhancer) => Store
```

Store Enhancer

```
type StoreEnhancer = (StoreCreator) => StoreCreator
```