React Cheat Sheet

A javascript library for building user interfaces.

DEMO: https://s.codepen.io/ericnakagawa/debug/ALxakj
GITHUB: https://github.com/facebook/react
DOCUMENTATION: https://facebook.github.io/react/docs/
CDN: https://cdnjs.com/libraries/react/

Hello World

```
// Import React and ReactDOM
import React from 'react'
import ReactDOM from 'react-dom'

// Render component into the DOM - only once per app
ReactDOM.render(
  <h1>Hello, world!</h1>,
  document.getElementById('root')
);
```

Stateless Components

```
// Stateless React Component
const Headline = () => {
  return <h1>React Cheat Sheet</h1>
}

// Component that receives props
const Greetings = (props) => {
  return <p>You will love it {props.name}.</p>
}

// Component must only return ONE element (eg. DIV)
const Intro = () => {
  return (
    <div>
      <Headline />
      <p>Welcome to the React world!</p>
      <Greetings name="Petr" />
    </div>
  )
}

ReactDOM.render(
  <Intro />,
  document.getElementById('root')
);

// Components and Props API - http://bit.ly/react-props
// CodePen Demo: http://bit.ly/react-simple
```

ES6 Class

```
// use class for local state and lifecycle hooks
class App extends React.Component {

  // Core
  constructor(props) {
    // fires before component is mounted
    super(props); // makes this refer to this component
    this.state = {date: new Date()}; // set state
  }
  render() {
    return (
      <h1>
        It is {this.state.date.toLocaleTimeString()}.
      </h1>
    )
  }

  // Lifecycle Methods
  componentWillMount() {
   // fires immediately before the initial render
  }
  componentDidMount() {
   // fires immediately after the initial render
  }
  componentWillReceiveProps() {
   // fires when component is receiving new props
  }
  shouldComponentUpdate() {
   // fires before rendering with new props or state
  }
  componentWillUpdate() {
   // fires immediately before rendering
   // with new props or state
  }
  componentDidUpdate() {
   // fires immediately after rendering with new P or S
  }
  componentWillUnmount() {
   // fires immediately before component is unmounted
   // from DOM (removed)
  }
}
      // CodePen Demo: http://bit.ly/react-es6-class
```

Conditional Rendering

```
// conditional rendering of elements and CSS class
render() {
  const {isLoggedIn, username} = this.state;
  return (
    <div className={`login ${isLoggedIn ? 'is-in'
: 'is-out'}`}>
      {
        !!isLoggedIn ?
        <p>Logged in as {username}.</p>
        :
        <p>Logged out.</p>
      }
    </div>
  )
}
// CodePen Demo: http://bit.ly/react-if-statements
```

Tools and Resources

```
// Create React App
   http://bit.ly/react-app
// React Dev Tools for Chrome
   http://bit.ly/react-dev-tools
// React Code Snippets for Visual Studio Code
   http://bit.ly/react-es6-snippets-for-vscode
// Babel for Sublime Text 3
   http://bit.ly/babel-sublime
```

VUE ESSENTIALS CHEAT SHEET

Vue Mastery

EXPRESSIONS

```
<div id="app">
  <p>I have a {{ product }}</p>
  <p>{{ product + 's' }}</p>
  <p>{{ isWorking ? 'YES' : 'NO' }}</p>
  <p>{{ product.getSalePrice() }}</p>
</div>
```

DIRECTIVES

Element inserted/removed based on truthiness:

```
<p v-if="inStock">{{ product }}</p>
```

```
<p v-else-if="onSale">...</p>
<p v-else>...</p>
```

Toggles the display: none CSS property:

```
<p v-show="showProductDetails">...</p>
```

Two-way data binding:

```
<input v-model="firstName" >
```

`v-model.lazy="..."`	Syncs input after change event
`v-model.number="..."`	Always returns a number
`v-model.trim="..."`	Strips whitespace

LIST RENDERING

```
<li v-for="item in items" :key="item.id">
  {{ item }}
</li>
```

key always recommended

To access the position in the array:

```
<li v-for="(item, index) in items">...
```

To iterate through objects:

```
<li v-for="(value, key) in object">...
```

Using v-for with a component:

```
<cart-product v-for="item in products"
              :product="item"
              :key="item.id" />
```


BINDING

```
<a v-bind:href="url">...</a>
```

shorthand →

```
<a :href="url">...</a>
```

True or false will add or remove attribute:

```
<button :disabled="isButtonDisabled”>...
```

If isActive is truthy, the class 'active' will appear:

```
<div :class="{ active: isActive }">...
```

Style color set to value of activeColor:

```
<div :style="{ color: activeColor }">
```

ACTIONS / EVENTS

Calls addToCart method on component:

```
<button v-on:click="addToCart">...
```

shorthand →

```
<button @click="addToCart">...
```

Arguments can be passed:

```
<button @click="addToCart(product)">...
```

To prevent default behavior (e.g. page reload):

```
<form @submit.prevent="addProduct">...
```

Only trigger once:

```
<img @mouseover.once="showImage">...
```

`.stop`	Stop all event propagation
`.self`	Only trigger if event.target is element itself

Keyboard entry example:

```
<input @keyup.enter="submit">
```

Call onCopy when control-c is pressed:

```
<input @keyup.ctrl.c="onCopy">
```

Key modifiers:

`.tab`	`.up`	`.ctrl`
`.delete`	`.down`	`.alt`
`.esc`	`.left`	`.shift`
`.space`	`.right`	`.meta`

Mouse modifiers:

`.left`	`.right`	`.middle`

VUE ESSENTIALS CHEAT SHEET

Vue Mastery

COMPONENT ANATOMY

```
Vue.component('my-component', {
  components: { Components that can be used in the template
    ProductComponent, ReviewComponent
  },
  props: {  → The parameters the component accepts
    message: String,
    product: Object,
    email: {
      type: String,
      required: true,
      default: 'none'
      validator: function (value) {
        Should return true if value is valid
      }
    }
  },
  data: function() {  Must be a function
    return {
      firstName: 'Vue',
      lastName: 'Mastery'
    }
  },
  computed: {     Return cached values until
    fullName: function () { dependencies change
      return this.firstName + ' ' + this.lastName
    }
  },
  watch: {  Called when firstName changes value
    firstName: function (value, oldValue) { ... }
  },
  methods: { ... },
  template: '<span>{{ message }}</span>',
})       Can also use backticks for multi-line
```

CUSTOM EVENTS

Use props (above) to pass data into child components, custom events to pass data to parent elements.

Set listener on component, within its parent:

```
<button-counter v-on:incrementBy="incWithVal">
```

Inside parent component:

```
methods: {
  incWithVal: function (toAdd) { ... }
}
```

Inside button-counter template:

```
this.$emit('incrementBy', 5)   Custom event name; Data sent up to parent
```


LIFECYCLE HOOKS

beforeCreate	beforeUpdate
created	updated
beforeMount	beforeDestroy
mounted	destroyed

USING A SINGLE SLOT

Component template:

```
<div>
  <h2>I'm a title</h2>
    <slot>
      Only displayed if no slot content
    </slot>
</div>
```

Use of component with data for slot:

```
<my-component>
  <p>This will go in the slot</p>
</my-component>
```

MULTIPLE SLOTS

Component template:

```
<div class="container">
  <header>
    <slot name="header"></slot>
  </header>
  <main>
    <slot>Default content</slot>
  </main>
  <footer>
    <slot name="footer"></slot>
  </footer>
</div>
```

Use of component with data for slot:

```
<app-layout>
  <template v-slot:header><h1>Title</h1></template>
  <p>The main content.</p>
  <template v-slot:footer><p>Footer</p></template>
</app-layout>
```

LIBRARIES YOU SHOULD KNOW

Vue CLI
Command line interface for rapid Vue development.

Vue Router
Navigation for a Single-Page Application.

Vue DevTools
Browser extension for debugging Vue applications.

Nuxt.js
Library for server side rendering, code-splitting, hot-reloading, static generation and more.