AFS
Agile Full Stack
Developer Bootcamp
Thoughtworks ©

Hello, React!

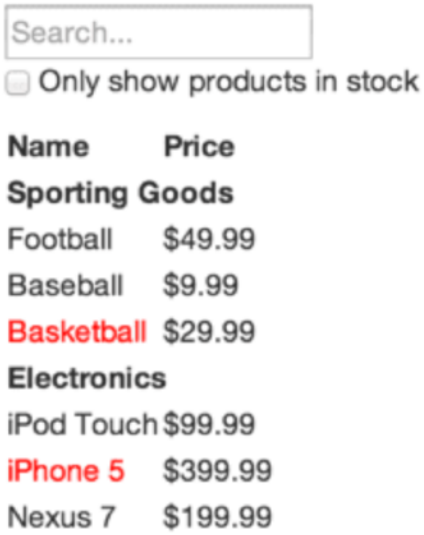
thoughtworks

What is React 

A JavaScript library for building **user interfaces**

What is React

Component-Based



https://reactjs.org/docs/thinking-in-react.html

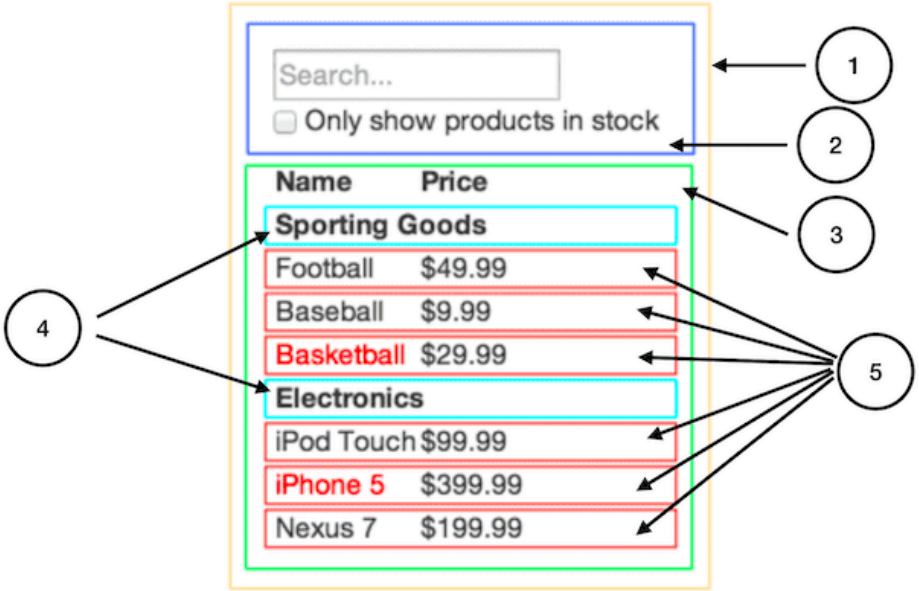
What is React

Component

Search...

Only show products in stock

Name	Price
Sporting Goods	
Football	$49.99
Baseball	$9.99
Basketball	$29.99
Electronics	
iPod Touch	$99.99
iPhone 5	$399.99
Nexus 7	$199.99



1. **`FilterableProductTable` (orange):** contains the entirety of the example
2. **`SearchBar` (blue):** receives all *user input*
3. **`ProductTable` (green):** displays and filters the *data collection* based on *user input*
4. **`ProductCategoryRow` (turquoise):** displays a heading for each *category*
5. **`ProductRow` (red):** displays a row for each *product*

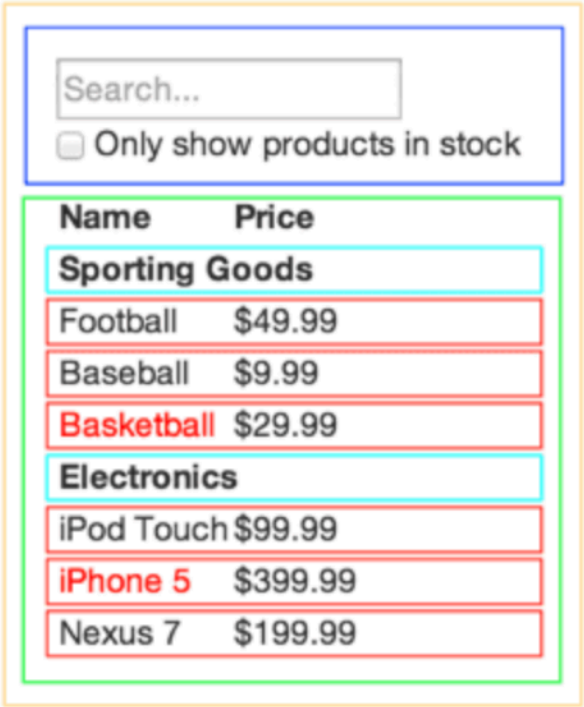
https://reactjs.org/docs/thinking-in-react.html

React Component

Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.

- Components can be combined with each other

- Components can be reused

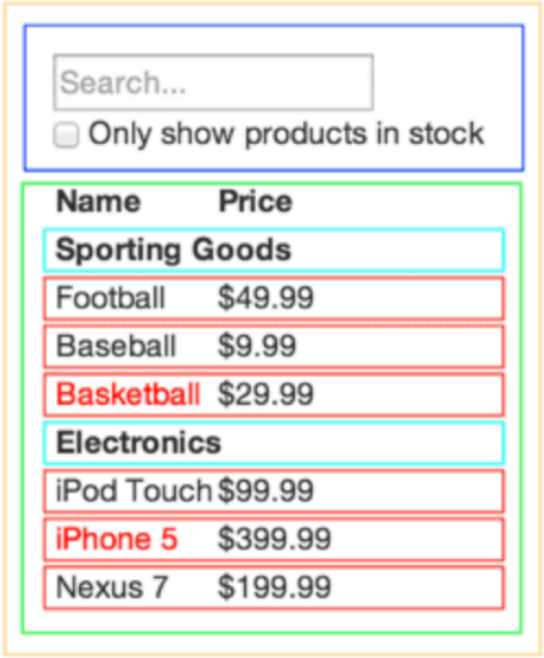


Building React App Steps

Step 1: Break The UI Into A Component Hierarchy

Step 2: Build A Static Version in React

Step 3: Identify data sources, determine if it's props or state

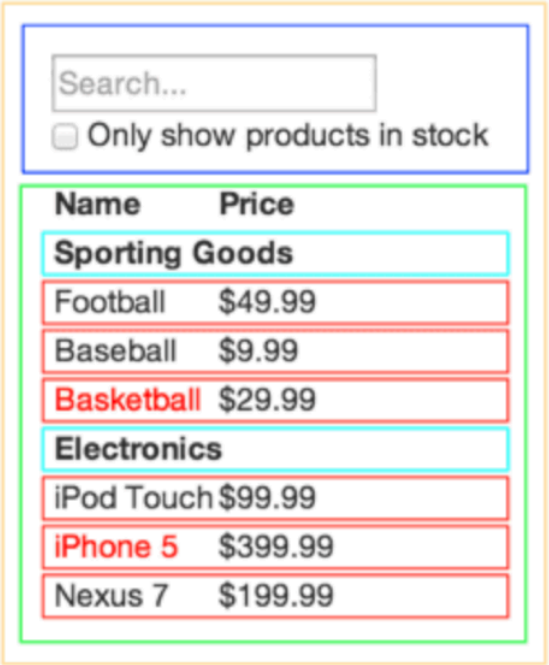


https://reactjs.org/docs/thinking-in-react.html

Component Props

Props are mainly used for communication between parent and child components:

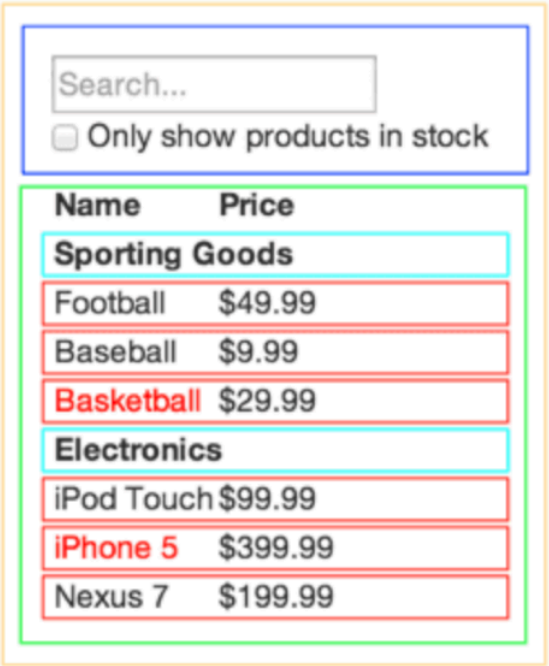
- The parent component passes props to the child components.
- Child components can use props directly
- Child components cannot modify props, props are read-only!



Component State

State is similar to props, but it is private and fully controlled by the component.

- Do not modify state directly
- State updates may be asynchronous
- State updates are merged



JSX

JSX is a syntax extension to JavaScript.

We recommend using it with React to describe what the UI should look like.

https://reactjs.org/docs/introducing-jsx.html

https://reactjs.org/docs/jsx-in-depth.html#gatsby-focus-wrapper

HTML VS JSX

HTML	JSX
<input maxlength="10" />	<input maxLength={10} />
<form novalidate />	<form noValidate />
Draggable	<p draggable>Draggable Draggable

- prop names follow **camelCase**
- wrap **numbers** attributes with **curly braces**
- boolean "true" can be written just property name and "false" should be written with curly braces

HTML VS JSX

HTML	JSX
`<h2 class="city">Chengdu</h2>`	`<h2 className="city">Chengdu</h2>`
`<h1 style="color:blue;text-align:center">header</h1>`	`<h1 style={{color:"blue",textAlign:"center"}}>header</h1>`

- "class" attributes is written as "className"
- Inline styles are written as objects

HTML VS JSX

Fix the syntax error of the following JSX element

```
const element = <input readonly="true" maxlength="10" style="background-color: blue;" />
```

❌

```
const element = <input readOnly maxLength={10} style={{backgroundColor: "blue"}} />
```

✅

JSX - javascript expressions

```
const name = "World";
const element = <h1>Hello, {name}</h1>;
```

Embedding expression of string

```
const user = {name: "Lily", age: 20};
const element = <h1>Hello, {user.name}</h1>;
```

Embedding expression of object

```
const imageUrl = "some link";
const userAvatar = <img src={imageUrl} />;
```

Specifying attributes

```
const imageUrl = "some link";
const userAvatar = <img src={imageUrl} />;

const elementWrapper = <div className="imageWrapper">{userAvatar}</div>
```

Specifying children

HTML VS JSX

Fix the syntax error of the following JSX element

```
const user = {name: "Lily", age: 20};
const element = <h1>Hello, {user}</h1>;
```

❌

```
const user = {name: "Lily", age: 20};
const element = <h1>Hello, {user.name}</h1>;
const element = <h1>Hello, {user.age}</h1>;
```

✅

JSX - Component

HTML

<UserDefined />

JSX

```
const UserDefined = <div>My name is {user.name}, I am {user.age}</div>

const user = {name: "", age: 20}
<UserDefined  user={user}/>
```

JSX - Component

Component Definition

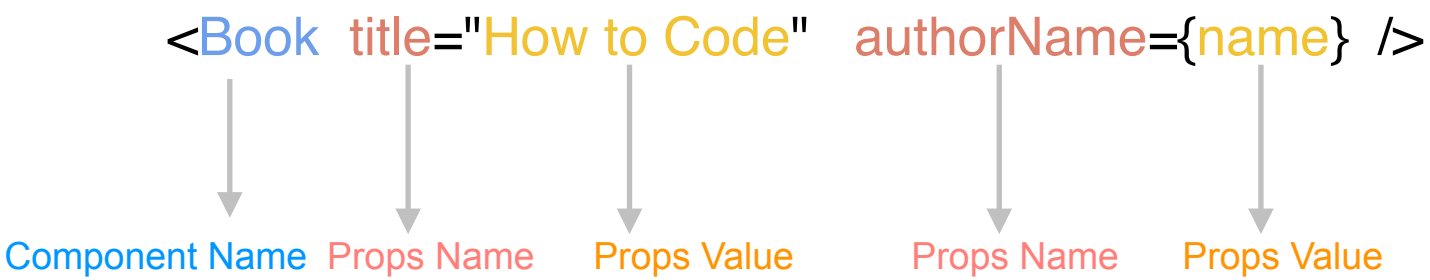
```
function Book(props) {

      return (
            <section>
                  <div className="title">{props.title}</div>
                  <div className="author-name">{props.authorName}</div>
            </section>
      );
}

export default Book;
```

JSX - Component

Component Definition

```jsx
function Book(props) {

      return (
            <section>
                  <div className="title">{props.title}</div>
                  <div className="author-name">{props.authorName}</div>
            </section>
      );
}

export default Book;
```

Component Usage



JSX - Component

Component Render

```
<div id="root" ></div>
```

```
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<Book title="" authorName="" />);
```

JSX - Component

Component Definition

```
function Book(props) {

      return (
            <section>
                  <div className="title">{props.title}</div>
                  <div className="author-name">{props.authorName}</div>
            </section>
      );
}

export default Book;
```

Component Usage

Component Render

```
<div id="root" ></div>
```

```
const root =
ReactDOM.createRoot(document.getElementById("root"));
root.render(<Book title="" authorName="" />);
```

Practice - Counter

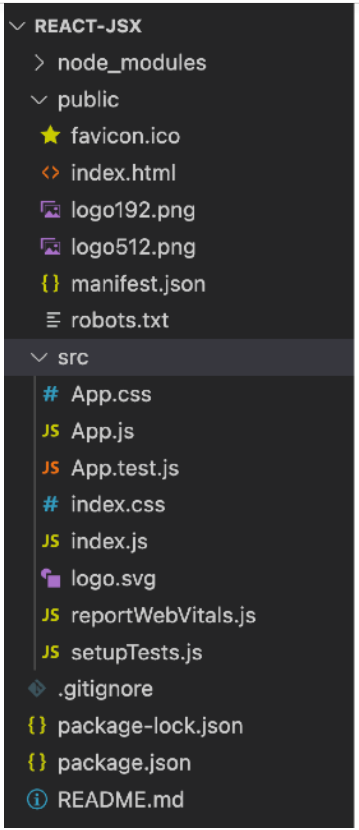
Env: Node 16.17.0 & npm 8.15.0

```
npx create-react-app react-counter
cd react-counter
npm start
```

Will start with localhost:3000

https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi

React Project



node_modules: all dependencies, should be ignored by git

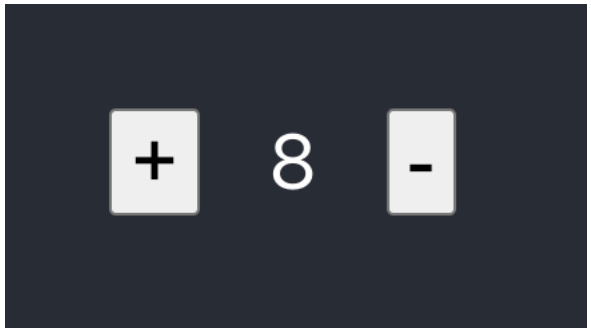
public: static files, which can be referenced from the HTML

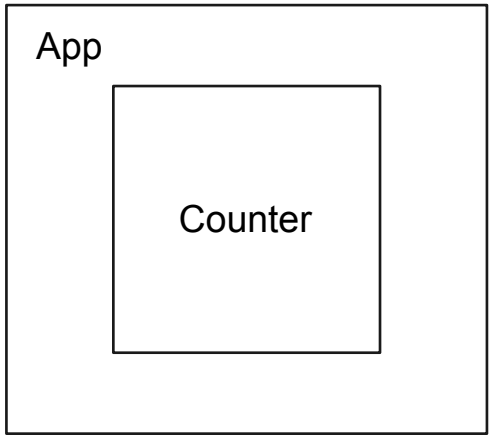
src: main folder in react project

package.json: dependencies and scripts required for the project

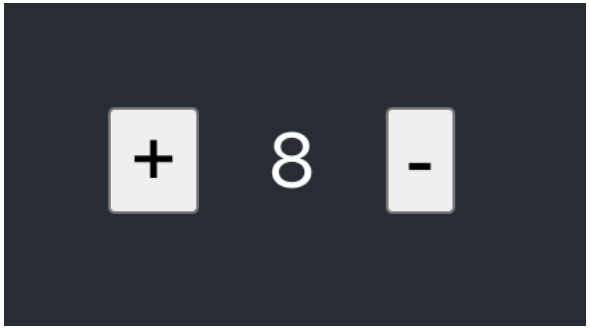
package-lock.json: keep track of the exact version of every package that is installed

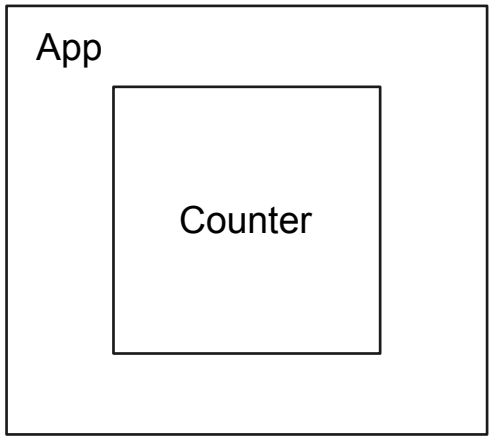
Practice - Counter





Practice - Counter



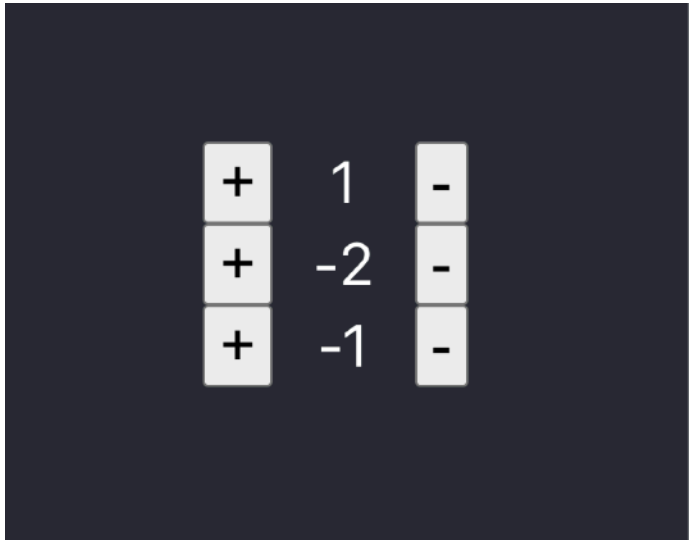


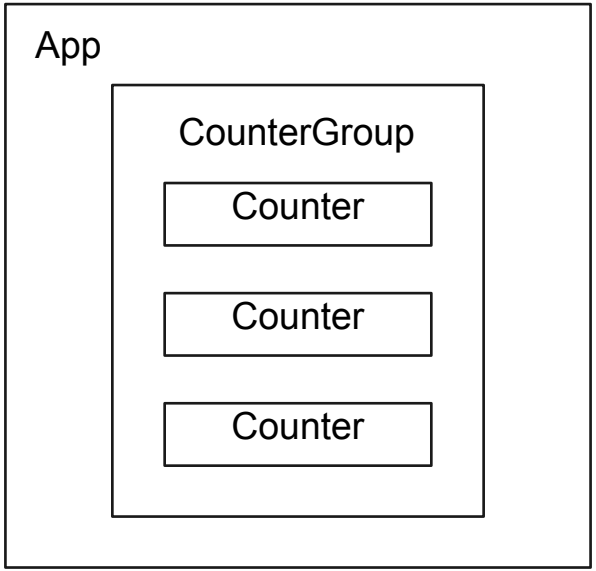
Pair Programming

1	Thomas	Jenny
2	Heinrich	Alvin
3	Alan	Michael
	Marie	

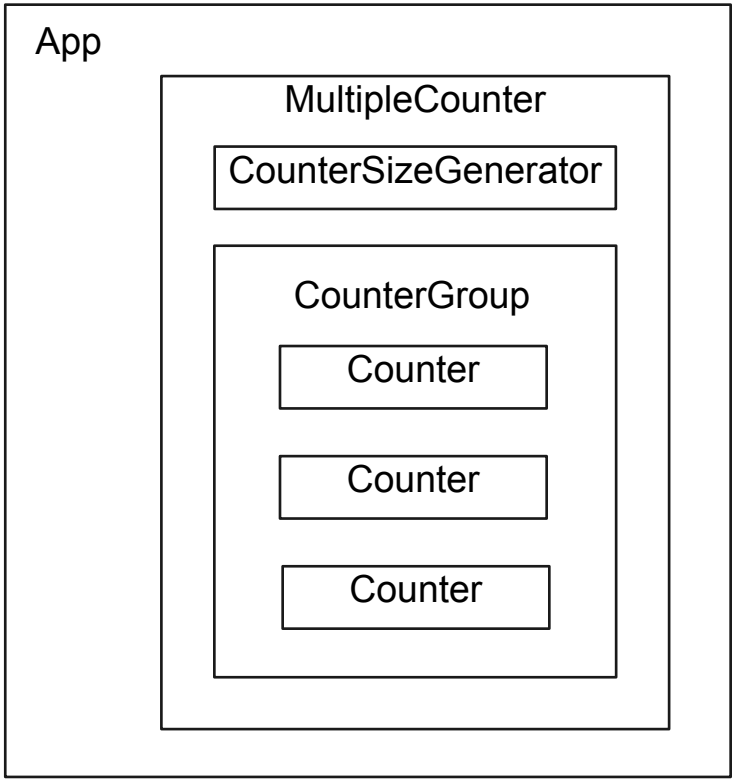
4	Kelvin	Vincent
5	Polly	Chris
6	Joyce	Antony

Practice - CounterGroup

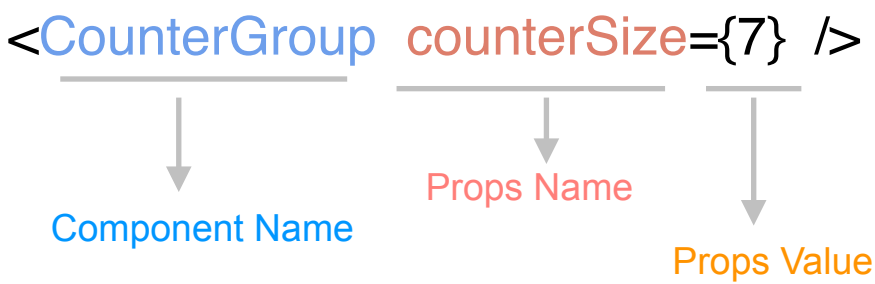




Practice - MultipleCounter



How to use Props




```
return (
  <div className="multiple-counter">
    <CounterGroup counterSize={7} />
  </div>
);
```

```
return (
    <div>Here are {props.counterSize} counters</div>
);
```

CounterGroup

How to use Props

<CounterSizeGenerator updateCounterSize={updateCounterSize} />

Component Name

Props Name

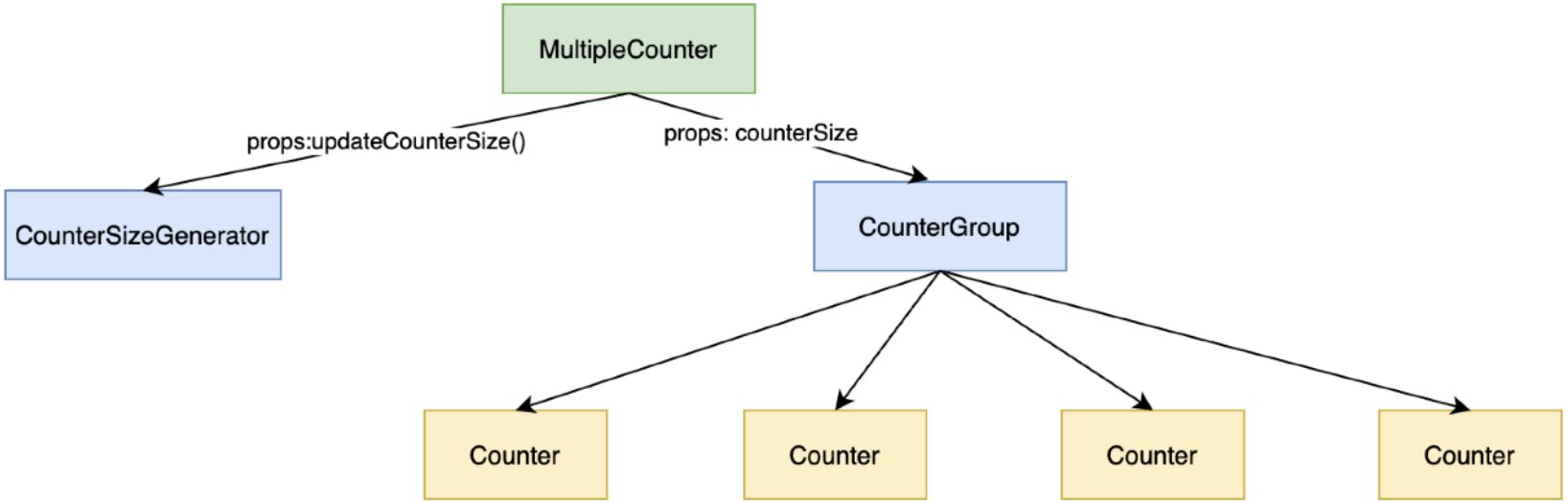
Props Value

```
function updateCounterSize(counterSize) {
  setCounterSize(counterSize);
}

return (
  <div className="multiple-counter">
    <CounterSizeGenerator updateCounterSize={updateCounterSize}/>
  </div>
);
```

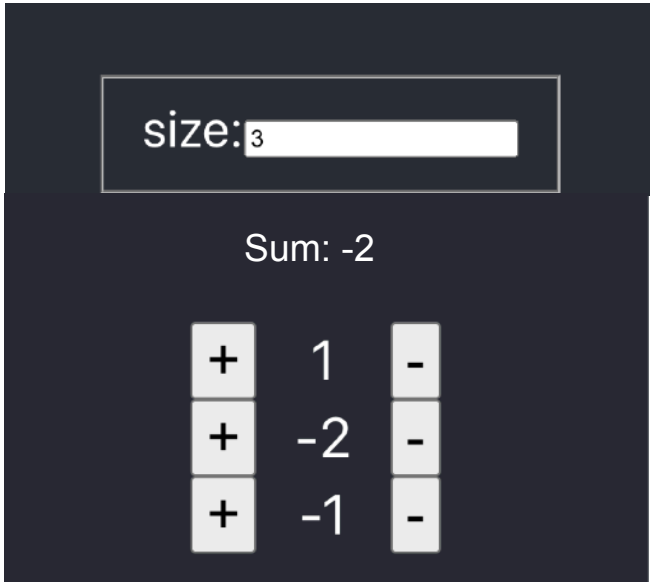
```
function generateCounters() {
  props.updateCounterSize(size);
}

return (
  <div>
    <button className="btn" onClick={generateCounters}>
      generate
    </button>
  </div>
);
```

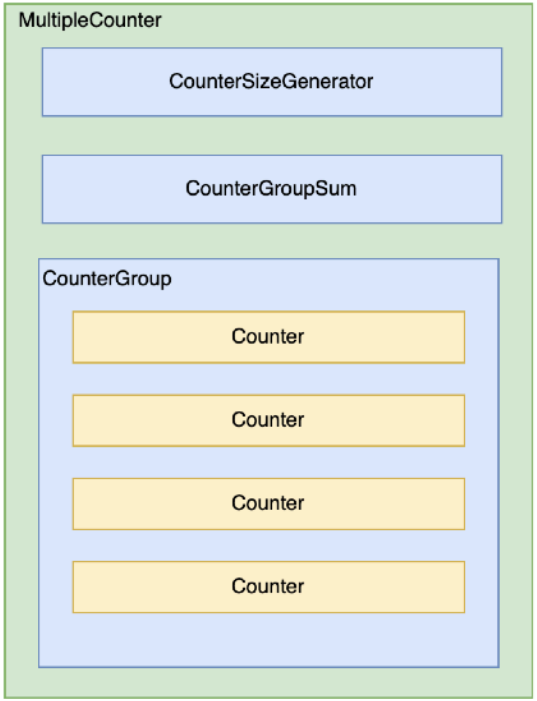
CounterSizeGenerator

Structure of MultipleCounter

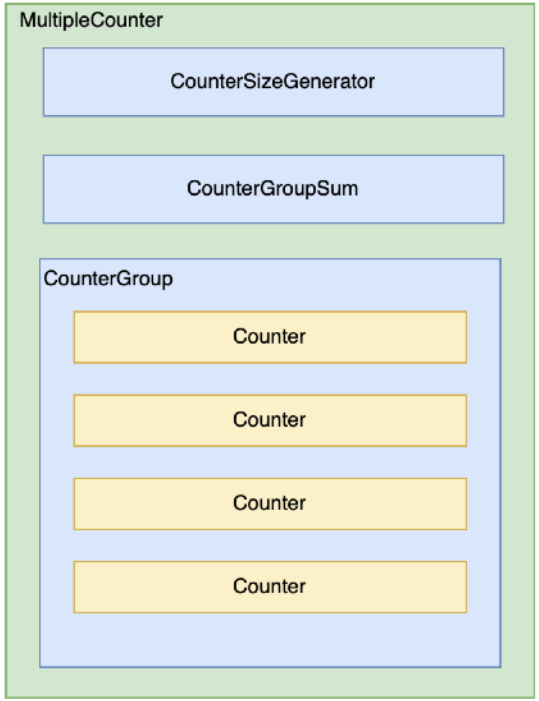


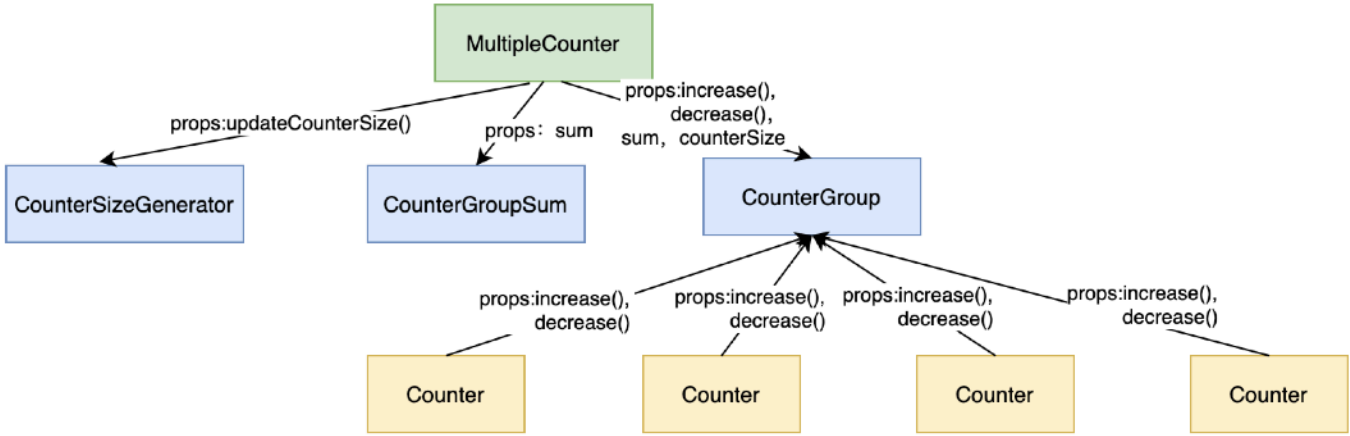
Practice - MultipleCounter





Structure of MultipleCounter

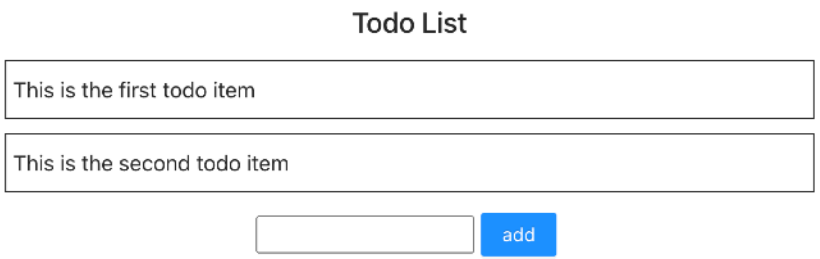


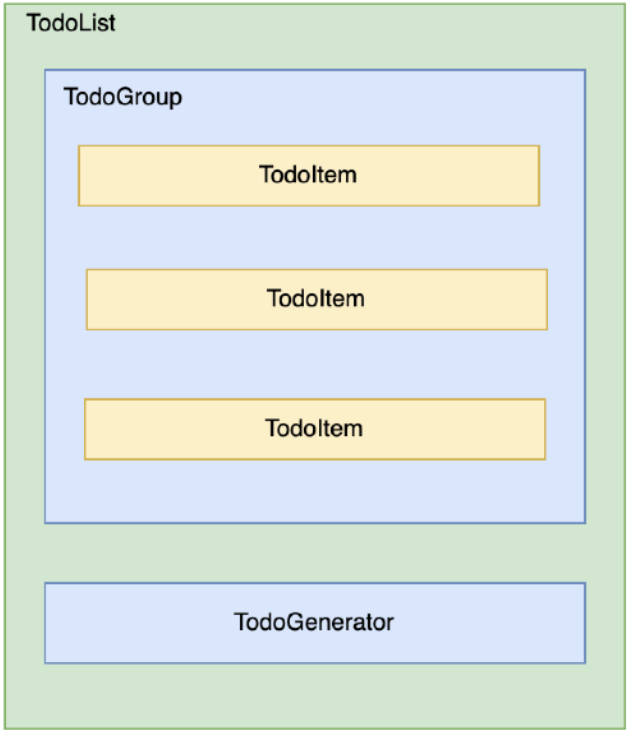


Homework

Deadline 22:00

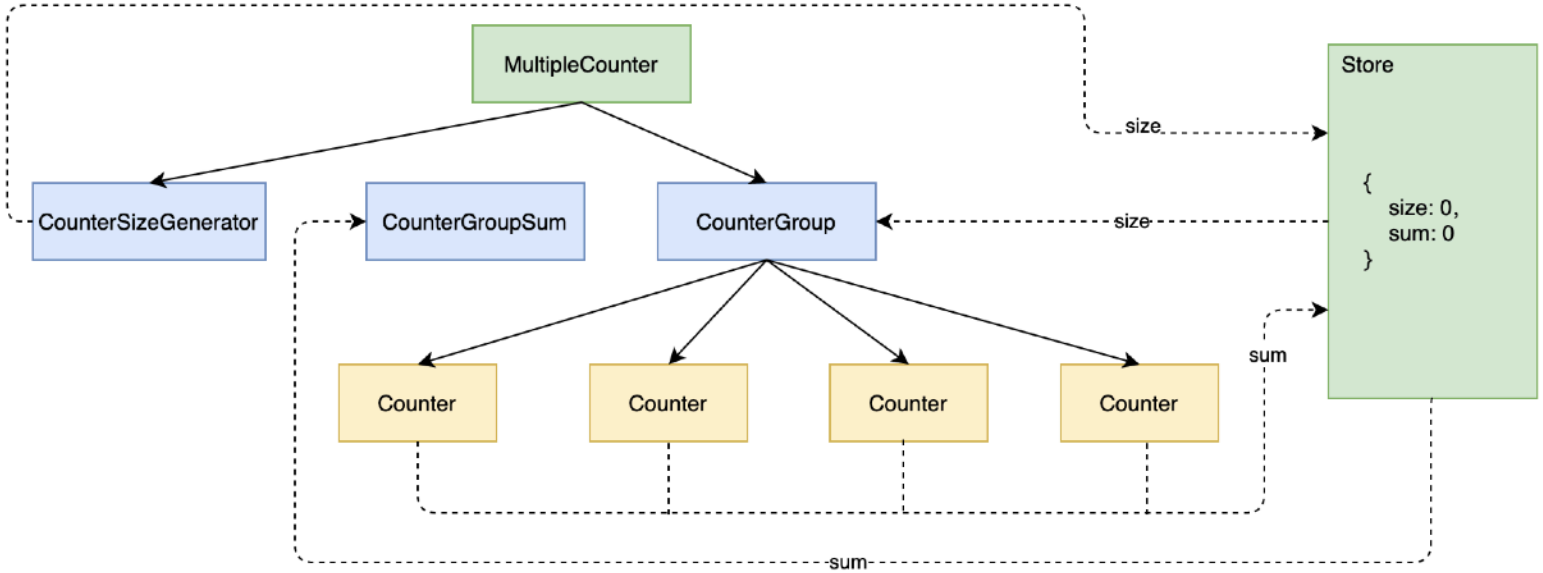
- **ORID Diary**
- **Finish Todo list individually with a new react project**
  - Input box
  - Click ‘add’ button and show todo in the list
  - Fork https://github.com/afs-public-202211/todo-react-day11-homework
  - Submit on https://school.thoughtworks.cn/learn/program-center/student/index.html#/program/404/task/10579





Homework

- Preview React-redux (Optional)

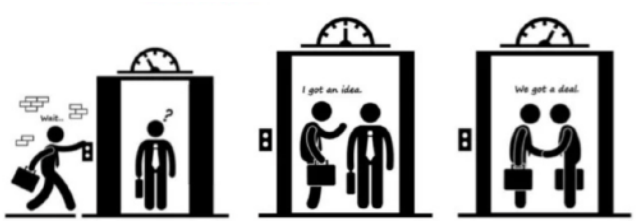


Self Learning - Agile Activities

Deadline Day 13
22:00pm

- **Learning Output**
    - **Draw concept map on 1 topic**
        - Concept map should contain the questions listed inside each topic.
    - **Showcase time: Day 14 afternoon or Day 15**

Topic 1: Elevator Speech



1. For ... (Target user group)
2. They want ... (Pain point)
3. This ... (Product name)
4. Is a ... (Product features)
5. It can ... (Irresistible advantages)
6. Different from ... (Other competing products)
7. Our product ... (Core differentiated competitiveness)

- **What is it ?**
- **When do we need it ?**
- **What is the purpose and core value of it ?**
- **How to use it ?**

Topic 2: User Story

Story Name: User can park and fetch a car

Description:

As a parking lot manager, I would like to provide a parking lot, so that the customer can park his/her car and fetch it.

AC1:

GIVEN a car and a parking lot with available space

WHEN customer park a car into the parking lot

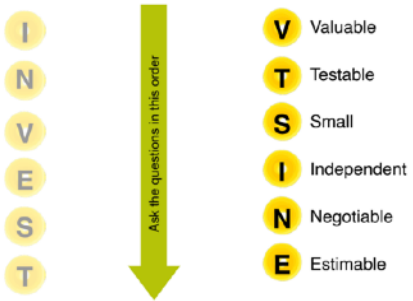
THEN the customer can park successfully and receive a parking ticket.

AC2:

GIVEN a parking ticket and a parking lot that parked the car

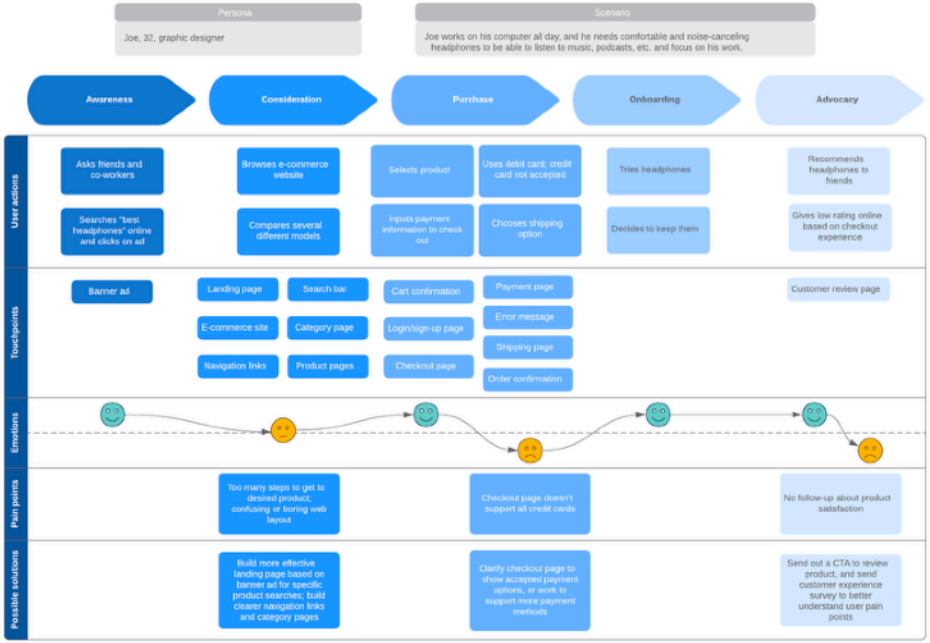
WHEN customer fetch car

THEN the customer can get his/her car



- **What is it ?**
- **What principles should a good user story follow ?**
- **What are the components of user story?**
- **What is the purpose and core value of it ?**

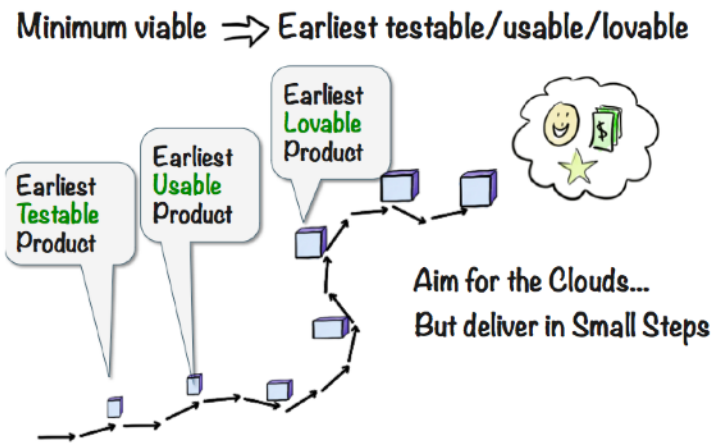
Topic 3: User Journey



- **What is it ?**
- **When do we need it ?**
- **What is the purpose and core value of it ?**
- **How to use it ?**

Topic 4: MVP(Minimum Viable Product)

"The Minimum Viable Product (MVP) is that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort"
— Eric Ries



- **What is it ?**
- **When do we need it ?**
- **What is the purpose and core value of it ?**
- **How to use it ?**

An MVP it is not necessarily the smallest product imaginable; it is simply the fastest way to get through the Build-Measure-Learn feedback loop with the minimum amount of effort.

https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp

Topic Assignment

	Group A	Group B	Topic
1	ALAN WAN	ANTONY CHOI	Elevator Speech
2	ALVIN LEUNG	CHRIS T C WONG	User Story
3	HEINRICH SIU	JOYCE LI	User Journey
4	JENNY WONG	KELVIN TO	MVP
5	MARIE CHOW	POLLY LEUNG	Elevator Speech
6	MICHAEL NAM	VINCENT TAM	User Story
7	THOMAS K Y KWOK		User Journey

Thank You !

thoughtworks