

Design & React

Asst.Prof.Drusawin Vongpramate

Department of Information Technology

Faculty of Science, BRU

Hint



Main Topic



Sub Topic

Web Design Process



Analyzing Requirements

We analyze your technical requirements to define the technical tools and resources needed.



Prototyping

We make a prototype that includes the features, content, homepage, and the initial designs for your website.



UI Design

Our team builds a UI design for your website that is attractive, unique, and easy to navigate.



UX Design

We build a flexible and responsive UX design using complex and modern technologies.



MVP Development

We develop an MVP of the website ready for testing and improving.



Testing

We examine the design on all levels and we test it through different devices.

MVP : Minimum Viable Product

A development approach where a product is built with the minimum features necessary to satisfy the early adopters or users and gather valuable feedback for further development. The primary goal of an MVP is to validate the core concept of the product while minimizing the time, resources, and costs involved in its development. By launching an MVP, developers can test the product idea in the market, identify its strengths and weaknesses, and make improvements based on user feedback before investing in developing a full-fledged product.



Web Design Types

Web Design Types

We choose the right layout or style based on your needs and objectives

Responsive Design

A website design that includes flexible images and layouts for smooth navigation on different screens and devices

Fixed Design

A design with a fixed sizes and dimensions even when viewed on different screens and devices.



Single Page

A single page website that includes one HTML page.



Static Website

A static website includes multiple HTML files and is delivered to web browsers as stored.



Dynamic Website

A database driven site that includes content and elements that change based on different factors.

1

2

3

4

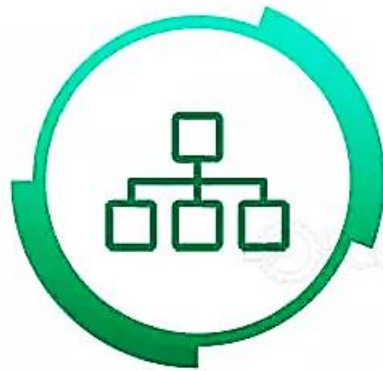
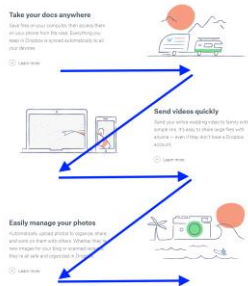
5

Website Layout Types



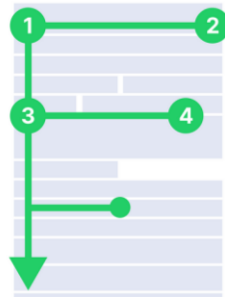
Zigzag Layout

Rows of images that are arranged over one another on one side while text is on the opposite.



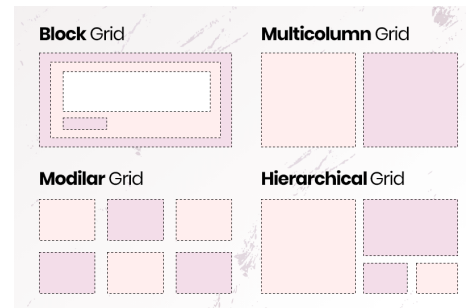
F - shape Layout

A design that establishes a visual hierarchy to get the visitors' attention to specific elements.



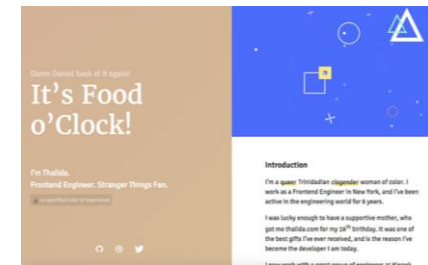
Grid Layout

A grid system is a series of boxes, or columns, that you divide your page into.



Split-screen

A web layout where the web components are divided into two vertical columns.



Web Component

HEADER



IMAGE SECTION



FOOTER



BLOG POST



FEATURES



TEXT



SIDE BLOCKS



Library/Framework



Page One | Page Two | Page Three | Search Sort by PRICE ▾ RU / EN

Submenu One / Submenu Two / Submenu Three / Submenu Four

Step 1 — Step 2 — Step 3 — Step 4 — Step 5

Tag One × Tag Two × Tag Three +

<input type="button" value="STATIC"/>	<input type="button" value="STATIC"/>	<input type="button" value="STATIC"/>
<input type="button" value="HOVER"/>	<input type="button" value="HOVER"/>	<input type="button" value="HOVER"/>
<input type="button" value="PRESSED"/>	<input type="button" value="PRESSED"/>	<input type="button" value="PRESSED"/>
<input type="button" value="LARGE"/>	<input type="button" value="LARGE"/>	<input type="button" value="LARGE"/>
<input type="button" value="NORMAL"/>	<input type="button" value="NORMAL"/>	<input type="button" value="NORMAL"/>
<input type="button" value="SMALL"/>	<input type="button" value="SMALL"/>	<input type="button" value="SMALL"/>

DROPDOWN ▾

Select One

Select Two

Select Three

Select Four

ON OFF

13.10.2016 ▾

< October 2016 >

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

LOGIN

You name

PASSWORD

●●●●●●●●

< 1 2 3 4 5 >

AUTOCOMPLITE

Ca

Cat

Category

Catholic

UI/UX Design Tools



Adobe XD

VS



InVision Studio

VS



Sketch

VS



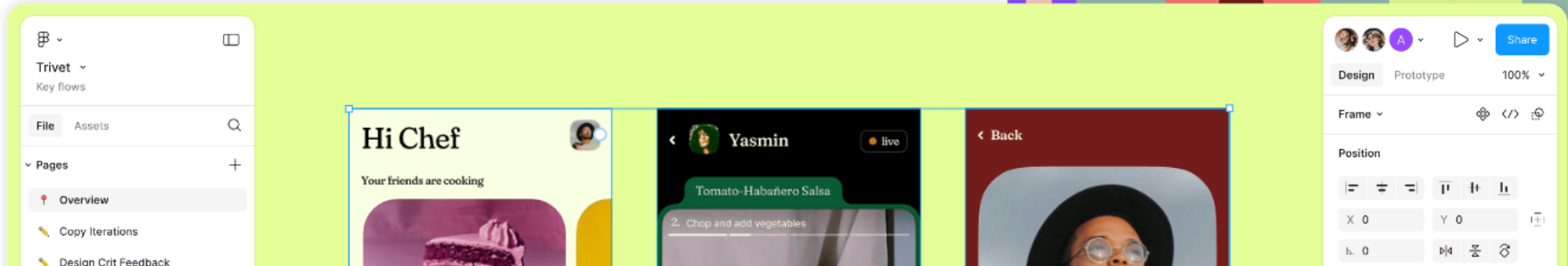
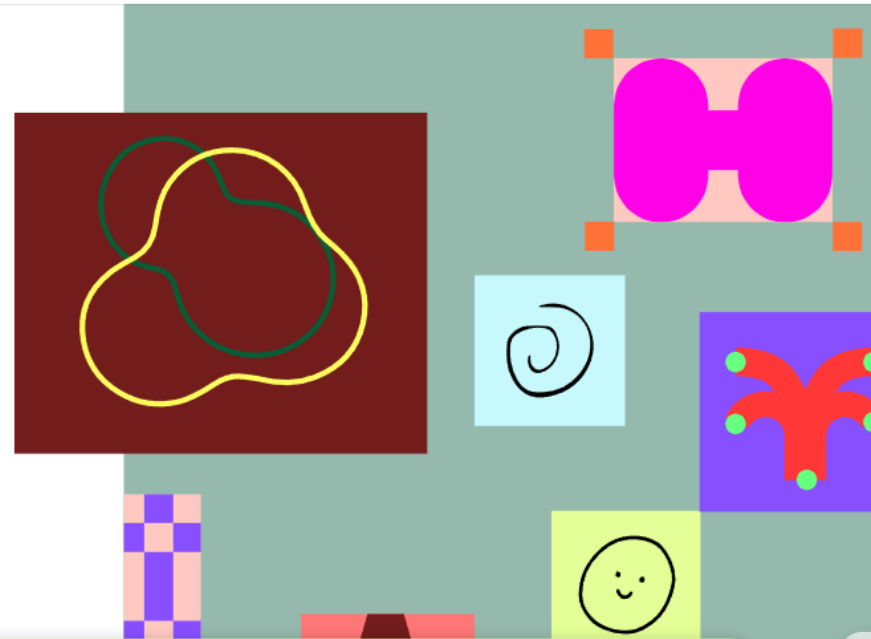
Figma

Figma

Think bigger. Build faster.

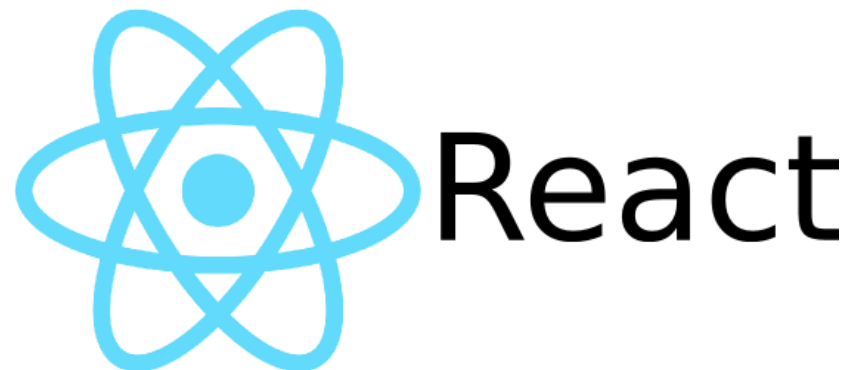
Figma helps design and development teams build great products, together.

Get started for free



React

- The library for web and native user interfaces
- React lets you build user interfaces out of individual pieces called **components**. Create your own React components like Thumbnail, LikeButton, and Video. Then combine them into entire screens, pages, and apps.



React

Class Component VS Function Component

```
1 import React from 'react';
2
3 class HelloWorld extends React.Component {
4   constructor(props) {
5     super(props);
6   }
7
8   sayHi(event) {
9     alert(`Hi ${this.props.name}`);
10  }
11
12  render() {
13    return (
14      <div>
15        <a
16          href="#"
17          onClick={this.sayHi.bind(this)}>Say Hi</a>
18      </div>
19    );
20  }
21 }
22
23 HelloWorld.propTypes = {
24   name: React.PropTypes.string.isRequired
25 };
26
27 export default HelloWorld;
```

```
1 import React from 'react';
2
3 const HelloWorld = ({name}) => {
4   const sayHi = (event) => {
5     alert(`Hi ${name}`);
6   };
7
8   return (
9     <div>
10      <a
11        href="#"
12        onClick={sayHi}>Say Hi</a>
13    </div>
14  );
15 };
16
17 HelloWorld.propTypes = {
18   name: React.PropTypes.string.isRequired
19 };
20
21 export default HelloWorld;
```

React

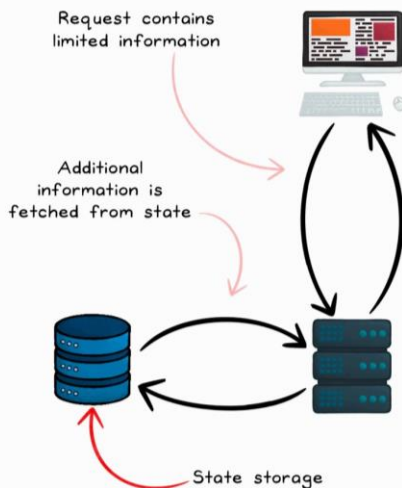
Functional Component vs Class Component

Functional vs Class

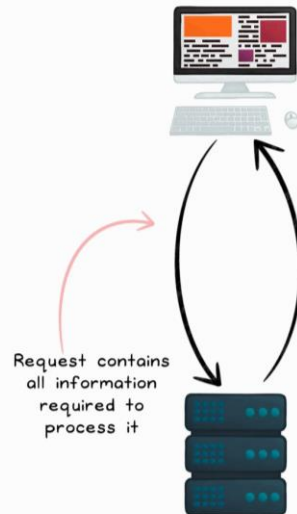
- Receive parameter (Props) - Optional
- **Stateless** or **dumb** component.
- Just Plain old JavaScript functions.
- Shorter to write
- For UI Components

- Has local State
- Receive parameter (Props) - Optional
- **Statefull** or **smart** component
- Has Lifecycle hooks.
- Can Handles fetching data via ajax calls

Stateful



Stateless



Stateless

Does not require the server to retain information about the state.

Server design, implementation and architecture is simple.

Handles crashes well, as we can fail over to a completely new server. Servers are regarded as cheap commodity machines.

Scaling architecture is easy.

Stateful

Requires a server to save information about a session.

Server design, implementation and architecture is complicated.

Does not handle crashes well. Servers are regarded as valuable and long-living. The user would probably be logged out and have to start from the beginning.

Scaling architectures is difficult and complex.

React

Go full-stack with a framework

React is a library. It lets you put components together, but it doesn't prescribe how to do routing and data fetching. To build an entire app with React, we recommend a full-stack React framework like [Next.js](#) or [Remix](#).

```
import { useState } from 'react';

function SearchableVideoList({ videos }) {
  const [searchText, setSearchText] = useState('');
  const foundVideos = filterVideos(videos, searchText);
  return (
    <>
      <SearchInput
        value={searchText}
        onChange={newText => setSearchText(newText)} />
      <VideoList
        videos={foundVideos}
        emptyHeading={`No matches for "${searchText}"`} />
    </>
  );
}
```

React Videos

A brief history of React

5 Videos



React: The Documentary
The origin story of React



Rethinking Best Practices
Pete Hunt (2013)



TypeScript

TypeScript is JavaScript with syntax for types.

TypeScript is a strongly typed programming language that builds on JavaScript, giving you better tooling at any scale.



TypeScript Style



```
'use client'

function MyButton({ title }: { title: string }) {

  const handleClick = () => {
    console.log('clicked');
  };

  return (
    <button onClick={handleClick}>{title}</button>
  );
}

export default function MyApp() {
  return (
    <div>
      <h1>Welcome to my app</h1>
      <MyButton title="I'm a button" />
    </div>
  );
}
```

```
import React from 'react';
import MyButton from '../components/MyButton';

const App: React.FC = () => {
  return (
    <div>
      <h1>Welcome to My Page</h1>
      <MyButton title="I'm a button" />
    </div>
  );
};

export default App;
```



```
"use client";

import React from "react";

interface Props {
  title: string;
  id?: string;
}

const MyButton: React.FC<Props> = ({ title }) => {
  const handleClick = () => {
    console.log("clicked");
  };

  return <button onClick={handleClick}>{title}</button>;
};

export default MyButton;
```

TS for Beginner

```
1 "use client";
2
3 import React from "react";
4
5 type Result = "pass" | "fail";
6 type underTen = 1 | 2 | 5 | 8;
7 type numberArray = Array<number>;
8
9 const numA: numberArray = [10, 5, -7];
10 // const numA: numberArray = [10, 5, '12']; //wrong type
11
12 interface Person {
13   firstName: string;
14   lastName: string;
15   single: boolean;
16 }
17
18 const person: Person = {
19   firstName: "Win",
20   lastName: "Vong",
21   single: true,
22 };
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50 const handleClick = () => {
51   console.log("clicked");
52   console.log(id);
53   console.log(cal(5, 7));
54 };
55
56
57
58
59
60
61
62
63
64
65
66 MyButton.defaultProps = {
67   title: "Win",
68   id: 15,
69 };
70
71 export default MyButton;
```

```
24 interface Props {
25   title: string;
26   id?: number;
27   test: (item: string) => void; //function
28   Item: {
29     //object
30     name: string;
31     price: number;
32   };
33   person: Person; //prop
34   car: string[]; //array
35 }
36
37 const MyButton: React.FC<Props> = ({ title, id }) => {
38   const verifyFn = (result: Result) => {
39     if (result === "fail") console.log("failed");
40   };
41
42   const cal = (a: number, b: number): number => {
43     return a + b;
44   };
45
46   const getLength = (obj: string | string[]): number => {
47     return obj.length;
48   };
49
50 }
```

```
50 const handleClick = () => {
51   console.log("clicked");
52   console.log(id);
53   console.log(cal(5, 7));
54 };
55
56
57
58
59
60
61
62
63
64
65
66 MyButton.defaultProps = {
67   title: "Win",
68   id: 15,
69 };
70
71 export default MyButton;
```

```
<>
  /* callback function */
  <button onClick={() => verifyFn("fail")}>{title}</button>
  /* call function */
  <button onClick={handleClick}>{title}</button>
</>
```

Event

onClick

```
import React from "react";

const MyComponent: React.FC = () => {
  const handleClick = () => {
    alert("Button clicked!");
  };

  return <button onClick={handleClick}>Click me</button>;
};
```

onScroll

```
import React from "react";

const MyComponent: React.FC = () => {
  const handleScroll = () => {
    console.log("Scrolled");
  };

  return <div onScroll={handleScroll}>Scroll me</div>;
};
```


Event

onChange

```
import React, { useState } from "react";

const MyComponent: React.FC = () => {
  const [inputValue, setInputValue] = useState("");

  const handleInputChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    setInputValue(event.target.value);
  };

  return <input type="text" value={inputValue} onChange={handleInputChange} />;
};
```

Event

onSubmit

```
import React from "react";

const MyComponent: React.FC = () => {
  const handleSubmit = (event: React.FormEvent<HTMLFormElement>) => {
    event.preventDefault();
    alert("Form submitted!");
  };

  return (
    <form onSubmit={handleSubmit}>
      { /* Form fields */ }
      <button type="submit">Submit</button>
    </form>
  );
};
```

Event

onMouseOver / onMouseOut

```
import React from "react";

const MyComponent: React.FC = () => {
  const handleMouseOver = () => {
    console.log("Mouse over");
  };

  const handleMouseOut = () => {
    console.log("Mouse out");
  };

  return (
    <div onMouseOver={handleMouseOver} onMouseOut={handleMouseOut}>
      | Hover over me
    </div>
  );
};
```

Event

onKeyDown / onKeyUp / onKeyPress

```
import React from "react";

const MyComponent: React.FC = () => {
  const handleKeyDown = (event: React.KeyboardEvent<HTMLInputElement>) => {
    console.log("Key down:", event.key);
  };

  const handleKeyUp = (event: React.KeyboardEvent<HTMLInputElement>) => {
    console.log("Key up:", event.key);
  };

  const handleKeyPress = (event: React.KeyboardEvent<HTMLInputElement>) => {
    console.log("Key pressed:", event.key);
  };

  return (
    <input
      onKeyDown={handleKeyDown}
      onKeyUp={handleKeyUp}
      onKeyPress={handleKeyPress}
      placeholder="Press a key"
    />
  );
};
```

Event

onTouchStart / onTouchEnd

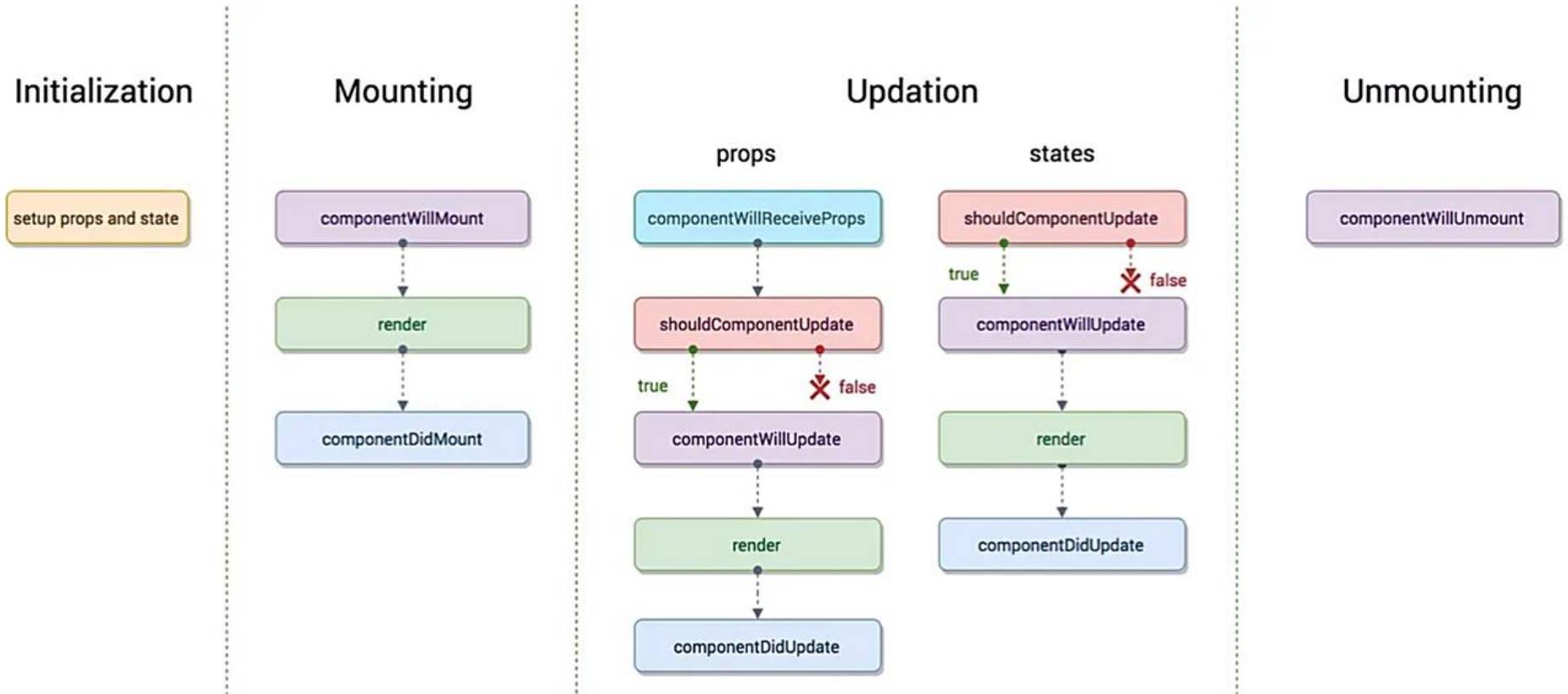
```
import React from "react";

const MyComponent: React.FC = () => {
  const handleTouchStart = () => {
    console.log("Touch started");
  };

  const handleTouchEnd = () => {
    console.log("Touch ended");
  };

  return (
    <div onTouchStart={handleTouchStart} onTouchEnd={handleTouchEnd}>
      Touch me
    </div>
  );
};
```

React component lifecycle



<https://projects.wojtekmaaj.pl/react-lifecycle-methods-diagram/>

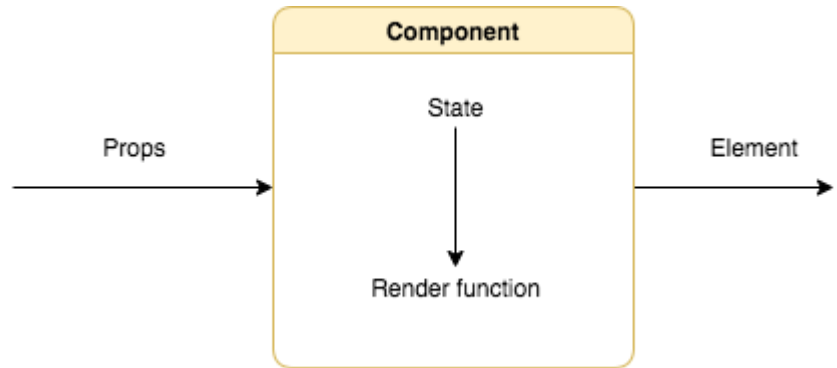
React Prop and State

In react components are responsible for generating html. To make html generate dynamically we need to pass data to our component so that our component can use variables and serve dynamic html.

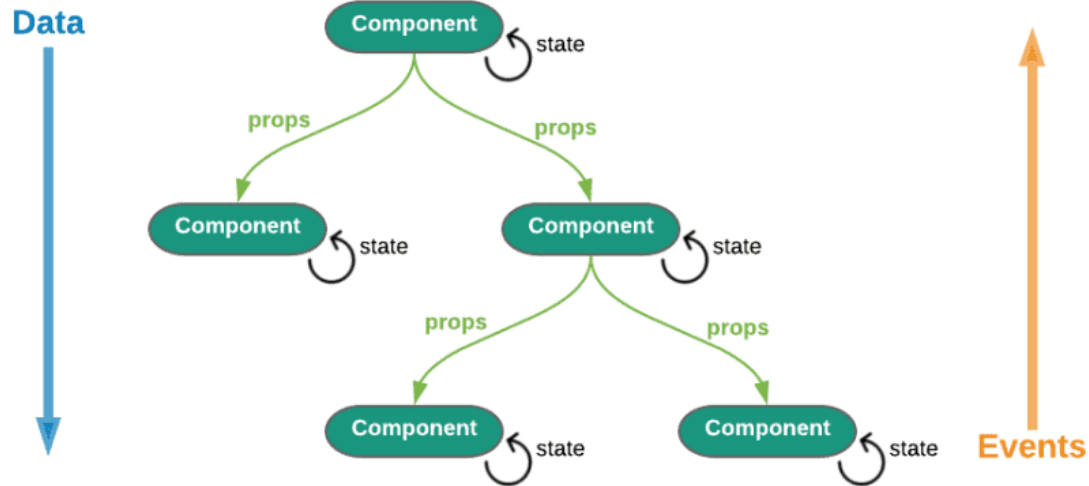
There are two types of data in react

state (private data available in component only)

props (public data can be passed to component from outside)

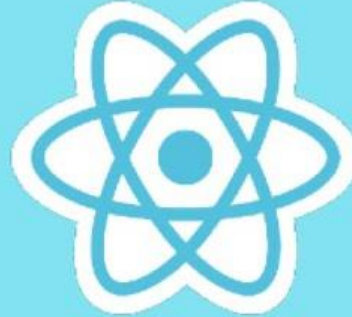


React Data flow



React Hook

Class Component



Function Component (Using Hooks)

`componentDidMount`
`componentDidUpdate`
`componentWillUnmount`



`useEffect`

`state`



`useState`

`ref={contentRef => ref}`



`useRef`

Rules of Hooks

- Only Call Hooks at the Top Level
 - Don't call Hooks inside loops, conditions, or nested functions.
- Only Call Hooks from React Functions
 - Don't call Hooks from regular JavaScript functions. Instead, you can:
 - Call Hooks from React function components.
 - Call Hooks from custom Hooks

React Hook

useState

```
import { useState } from "react";

function demo() {
  const [isVisible, setIsVisible] = useState(true);

  return <>{isVisible && <h1>I'm visible</h1>}</>;
}

export default demo;
```

To set its value use the dedicated function like this:

```
setIsVisible(false);
```

React Hook

<https://legacy.reactjs.org/docs/hooks-reference.html>

useEffect

```
1  "use client";
2
3  import { FC, useState, useEffect } from "react";
4
5  interface Num {
6    num: number;
7  }
8
9  const Counter: FC<Num> = ({ num }) => {
10   const [cnt, setCnt] = useState<number>(0);
11   const [cntEff, setCntEff] = useState<number>(0);
12
13   useEffect(() => {
14     document.title = `You clicked ${cnt} times`;
15     setCntEff(cnt);
16
17     return () => {
18       console.log("clean up");
19     };
20   }, [cnt]);
21
22   const inc = (num: number): number => {
23     return num + 1;
24   };
25
26   return (
27     <>
28       <div>{cnt}</div>
29       <div>Effect : {cntEff}</div>
30       <button onClick={() => setCnt(inc(cnt))}>Click Me</button>
31     </>
32   );
33 };
```

dependency array

If set to []

useEffect will run
only one time

React Hook Recommendation style

```
const Hook: FC<Props> = ({ initHook }) => {
  const [count, setCount] = useState<number>(initHook);
  const [data, setData] = useState({});

  const inc = (num: number): number => {
    return num + 1;
  };

  const dec = (num: number) => {
    return num - 1;
  };

  const getData = async (url: string) => {
    const res = await fetch(url);
    const dataRes = await res.json();

    console.log(dataRes);
    setData(dataRes);
  };

  useEffect(() => {
    const url = "https://dataapi.moc.go.th/products?keyword=นมพรวร์";

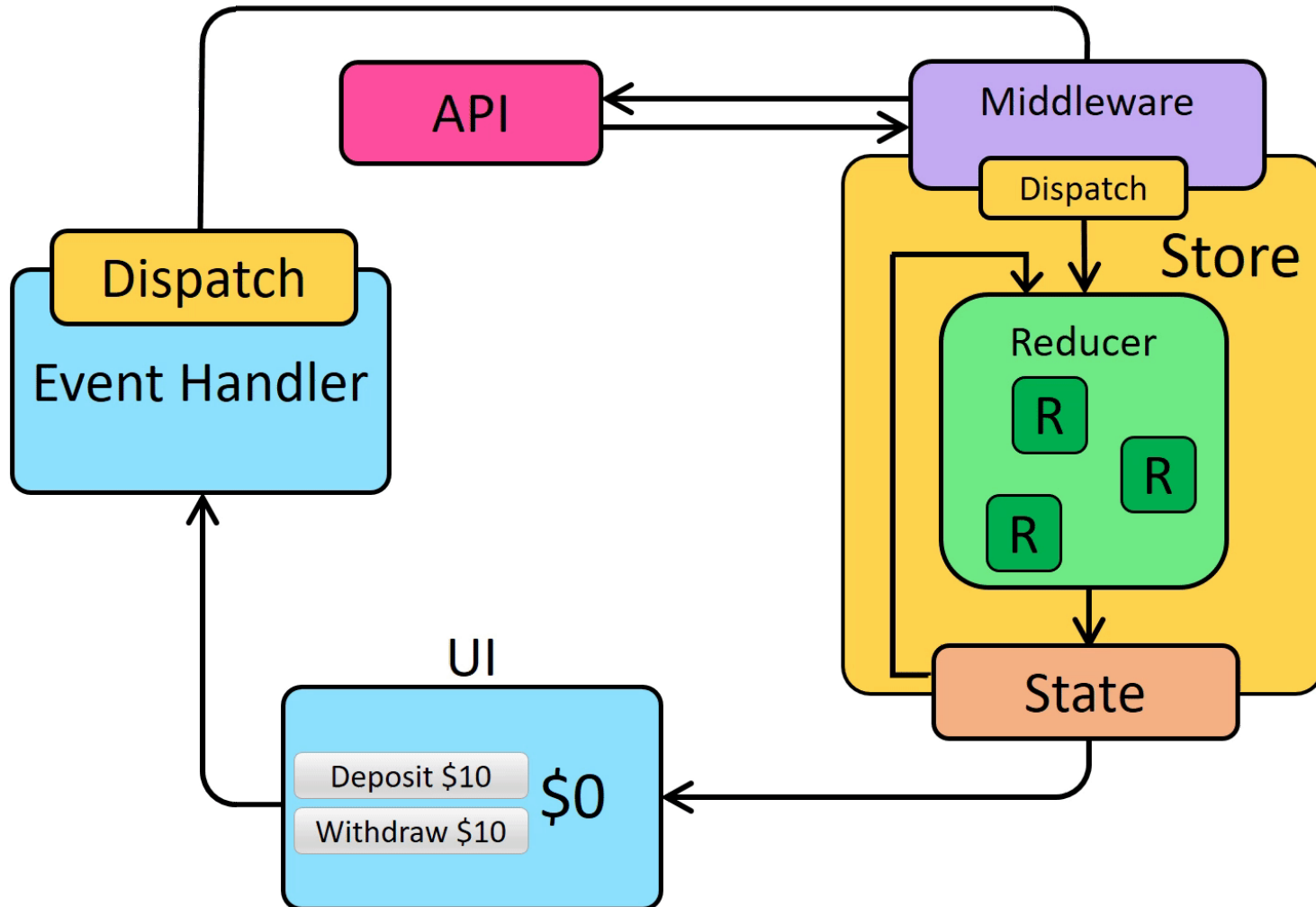
    // setCount(initHook);
    try {
      getData(url);
    } catch (err) {
      console.log("Can't fetch :", err);
    }
  }, []);
};
```

Create
function

Use
try and
catch

Redux

Predictable state container for JavaScript apps



Create Page



Let's Try

Guideline

Assemble components into a web page

1. What is component?
2. Where are position of components?

Q & A