Design & React

Asst.Prof.Drusawin Vongpramate

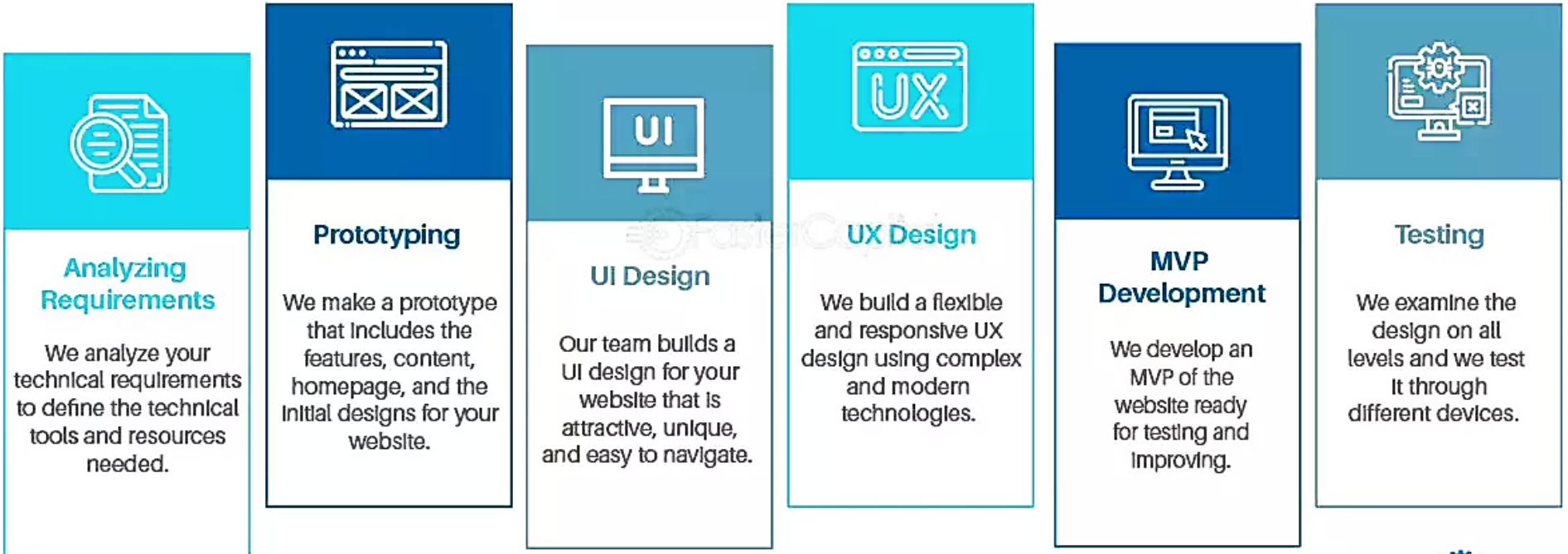
Department of Information Technology

Faculty of Science, BRU

Hint

Main Topic Sub Topic

Web Design Process

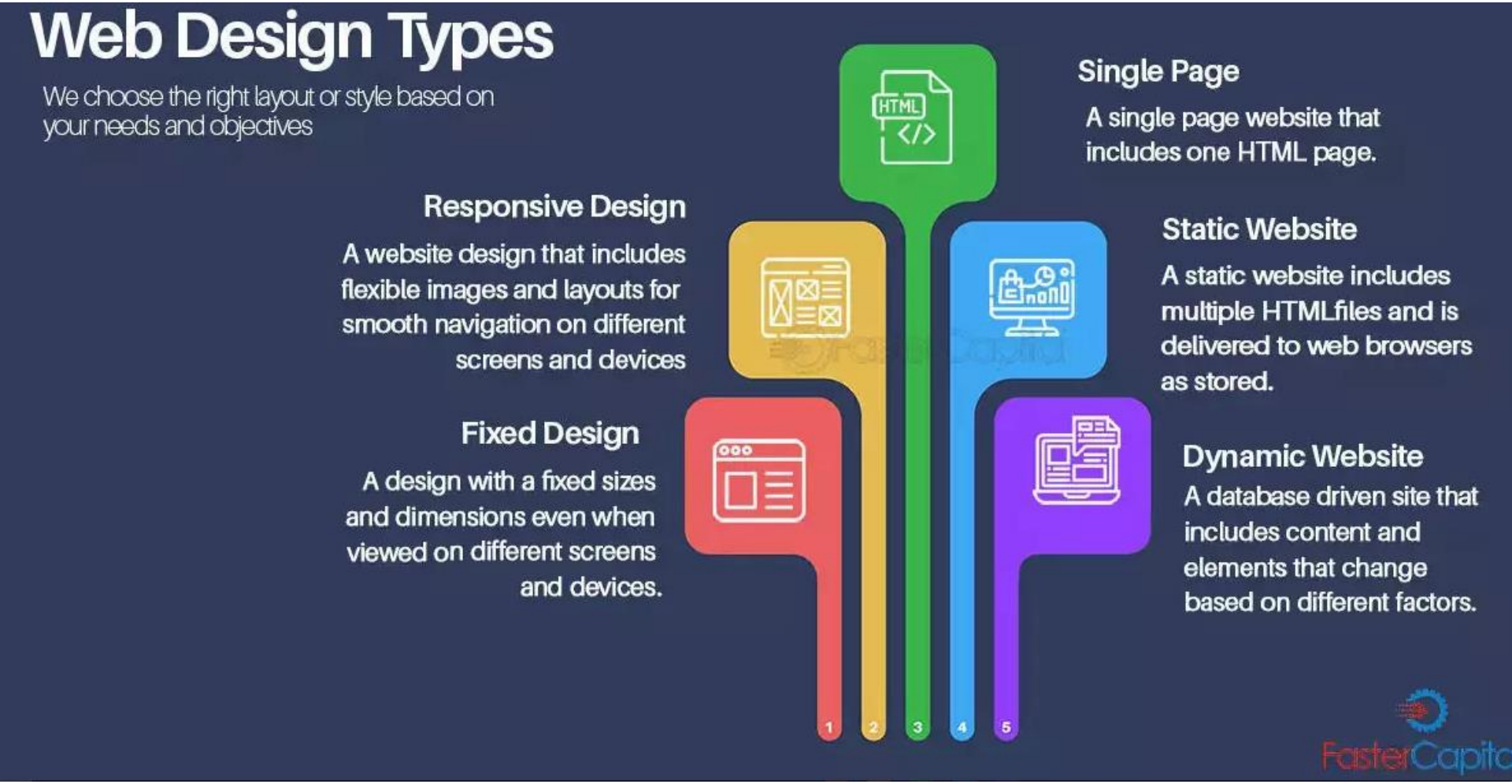


FasterCapital

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 minimising 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.

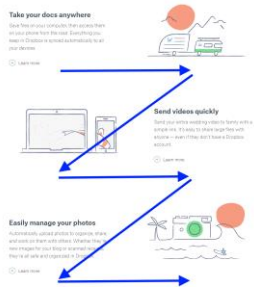
Web Design Types



Website Layout Types

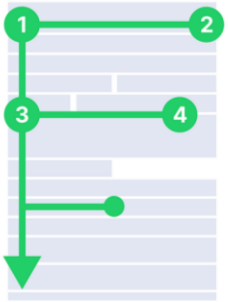
Zigzag Layout

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



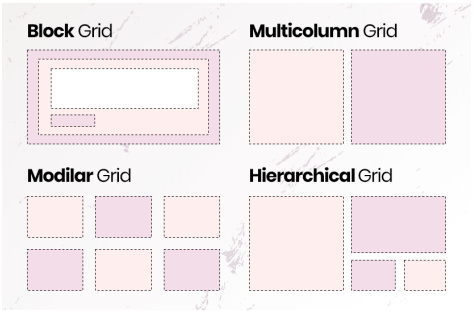
F - shape Layout

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



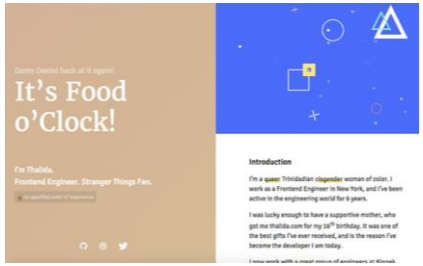
Grid Layout

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

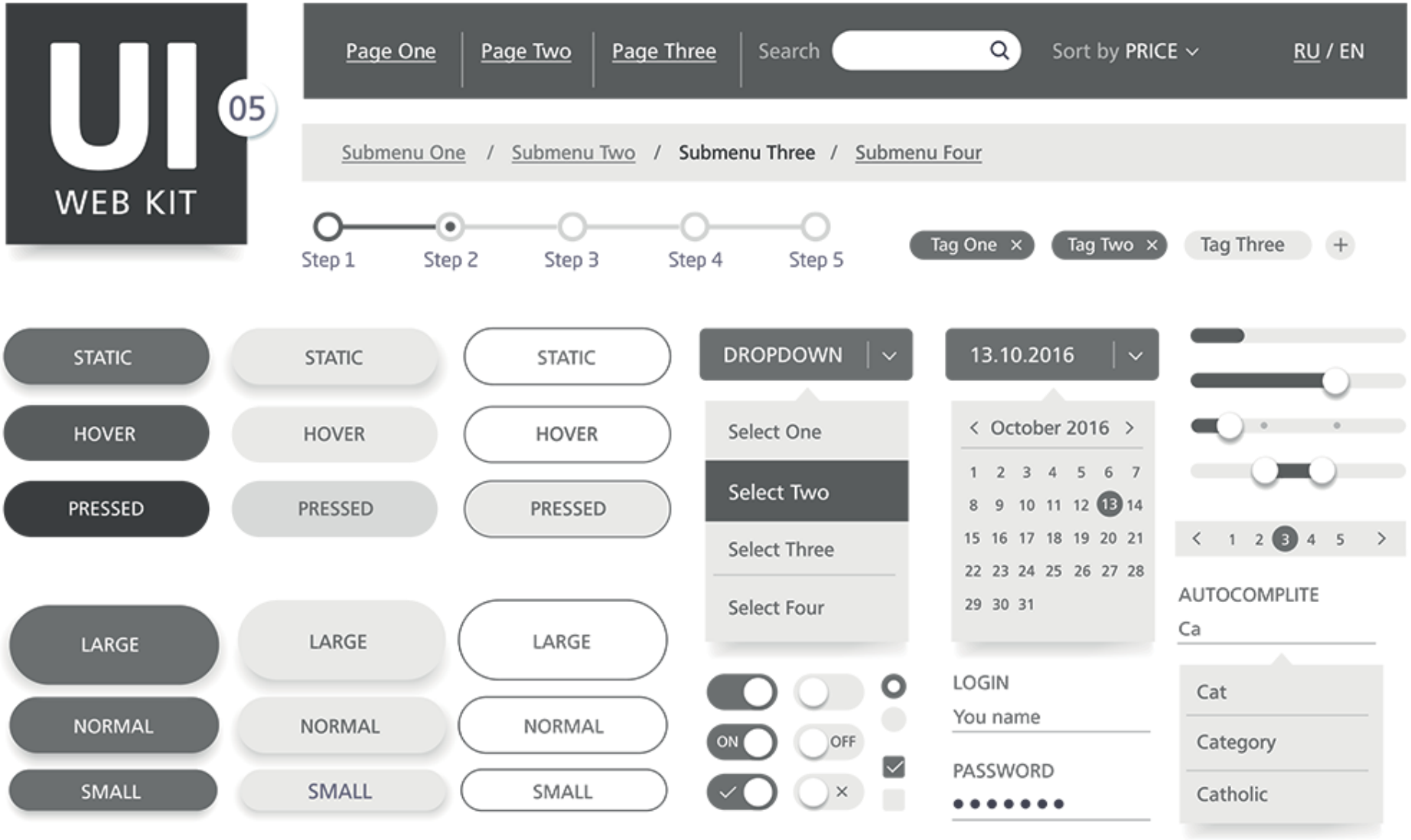
BLOG POST

FEATURES

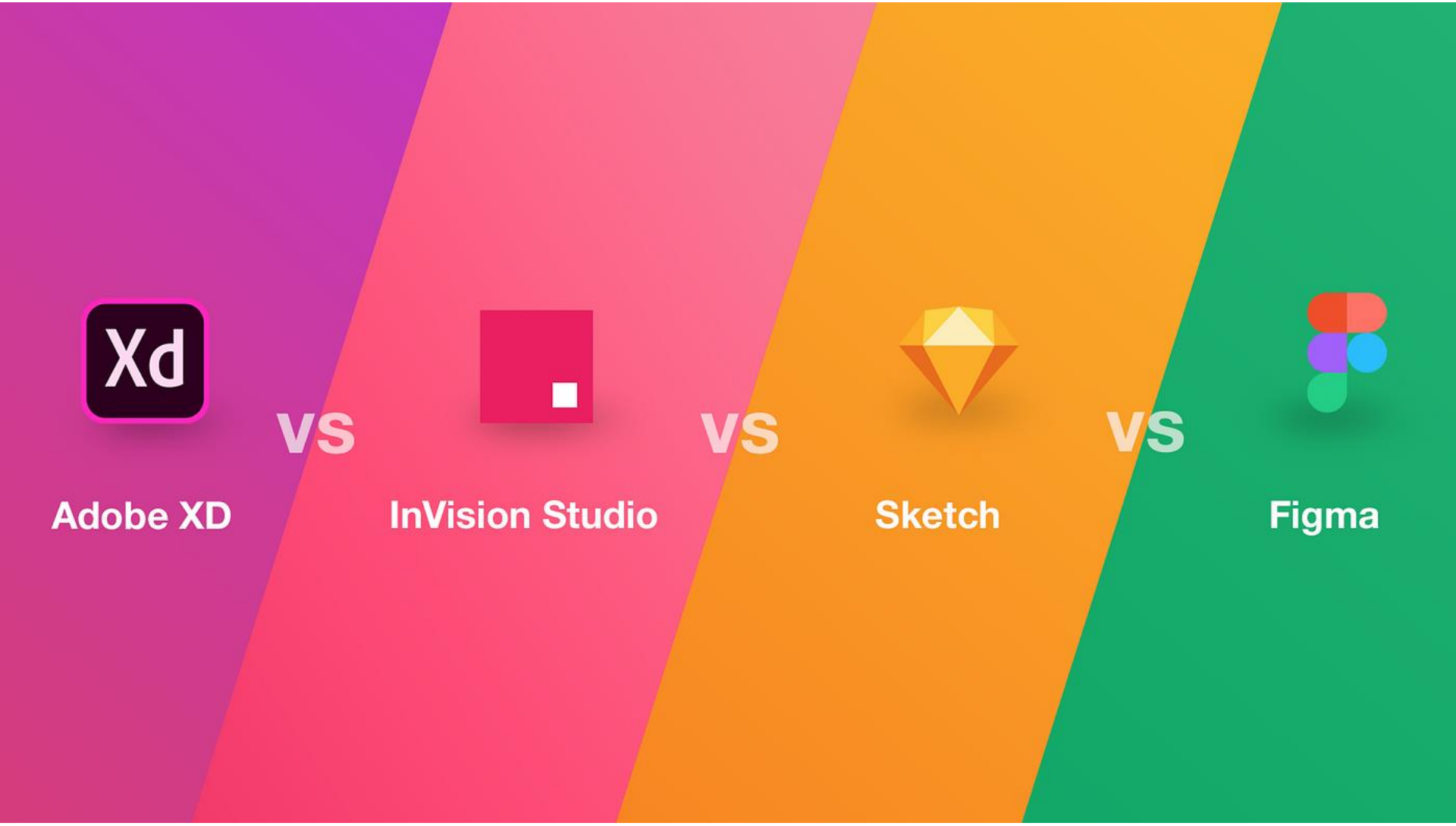
TEXT

SIDE BLOCKS

Library/Framework



UI/UX Design Tools



Figma

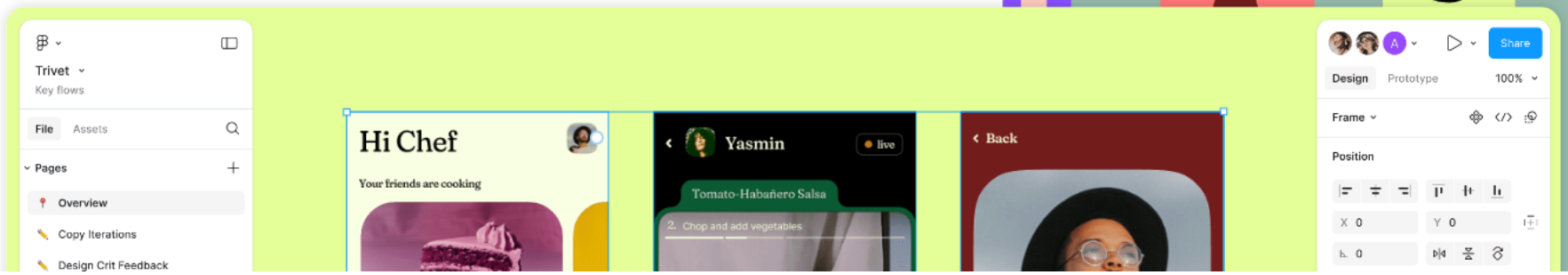
Products Solutions Community Resources Pricing

Contact sales | Log in | Get started for free

Think bigger. Build faster.

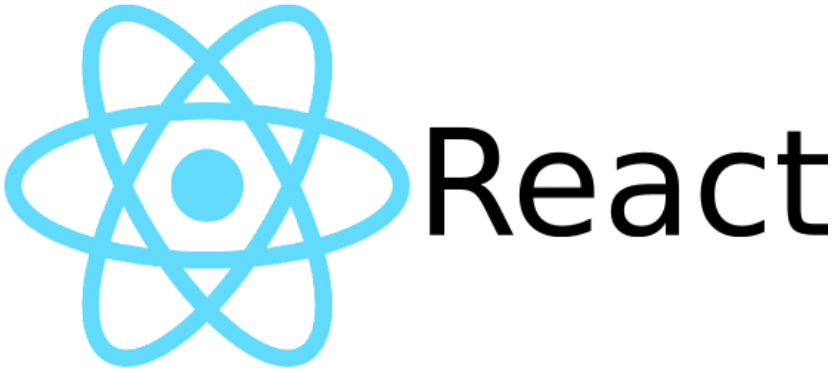
Figma helps design and development teams build great products, together.

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

```jsx
import React from 'react';

class HelloWorld extends React.Component {
  constructor(props) {
    super(props);
  }

  sayHi(event) {
    alert(`Hi ${this.props.name}`);
  }

  render() {
    return (
      <div>
        <a
          href="#"
          onClick={this.sayHi.bind(this)}>Say Hi</a>
      </div>
    );
  }
}

HelloWorld.propTypes = {
  name: React.PropTypes.string.isRequired
};

export default HelloWorld;
```

```jsx
import React from 'react';

const HelloWorld = ({name}) => {
  const sayHi = (event) => {
    alert(`Hi ${name}`);
  };

  return (
    <div>
      <a
        href="#"
        onClick={sayHi}>Say Hi</a>
    </div>
  );
};

HelloWorld.propTypes = {
  name: React.PropTypes.string.isRequired
};

export default HelloWorld;
```

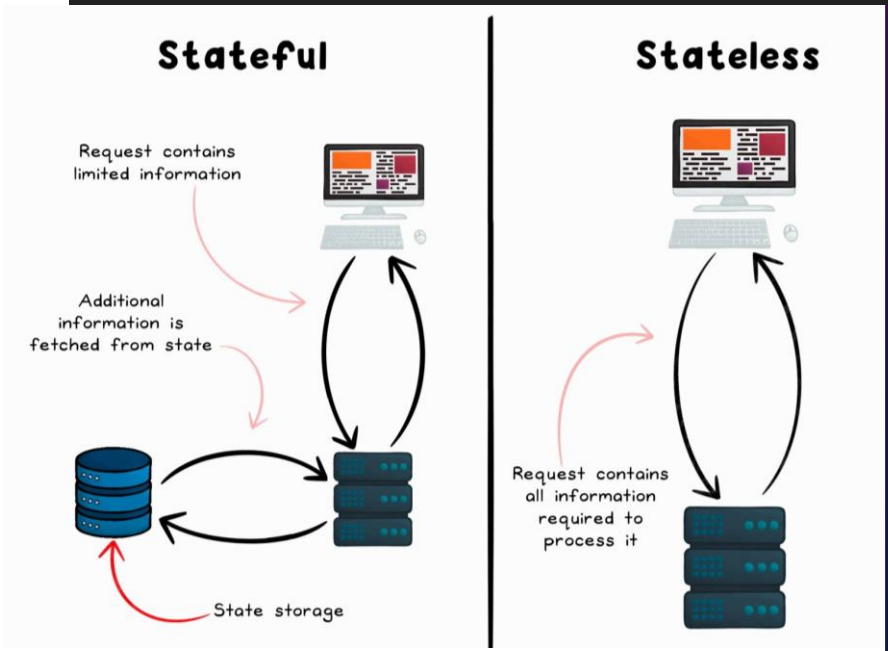
React

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



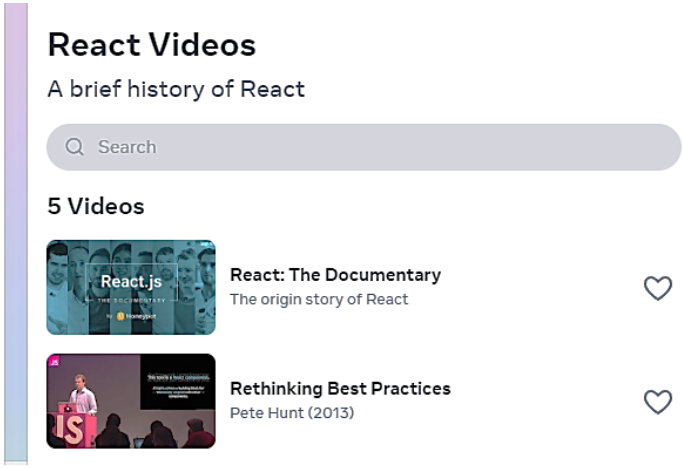
Stateless	Stateful
Does not require the server to retain information about the state.	Requires a server to save information about a session.
Server design, implementation and architecture is simple.	Server design, implementation and architecture is complicated.
Handles crashes well, as we can fail over to a completely new server. Servers are regarded as cheap commodity machines.	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 architecture is easy.	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}”`} />
    </>
  );
}
```



TypeScript

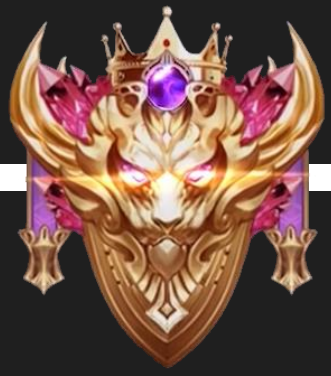
TypeScript is JavaScript with syntax for types.

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

TypeScript Style

```tsx
'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>
  );
}
```

```tsx
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;
```



```tsx
"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

```tsx
"use client";

import React from "react";

type Result = "pass" | "fail";
type underTen = 1 | 2 | 5 | 8;
type numberArray = Array<number>;

const numA: numberArray = [10, 5, -7];
// const numA: numberArray = [10, 5, '12']; //wrong type

interface Person {
  firstName: string;
  lastName: string;
  single: boolean;
}

const person: Person = {
  firstName: "Win",
  lastName: "Vong",
  single: true,
};
```

```tsx
interface Props {
  title: string;
  id?: number;
  test: (item: string) => void; //function
  Item: {
    //object
    name: string;
    price: number;
  };
  person: Person; //prop
  car: string[]; //array
}

const MyButton: React.FC<Props> = ({ title, id }) => {
  const verifyFn = (result: Result) => {
    if (result === "fail") console.log("failed");
  };

  const cal = (a: number, b: number): number => {
    return a + b;
  };

  const getLength = (obj: string | string[]): number => {
    return obj.length;
  };

  const handleClick = () => {
    console.log("clicked");
    console.log(id);
    console.log(cal(5, 7));
  };

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

MyButton.defaultProps = {
  title: "Win",
  id: 15,
};

export default MyButton;
```

Event

onClick

```jsx
import React from "react";

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

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

onScroll

```jsx
import React from "react";

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

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

Event

onChange

```tsx
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

```tsx
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

```tsx
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

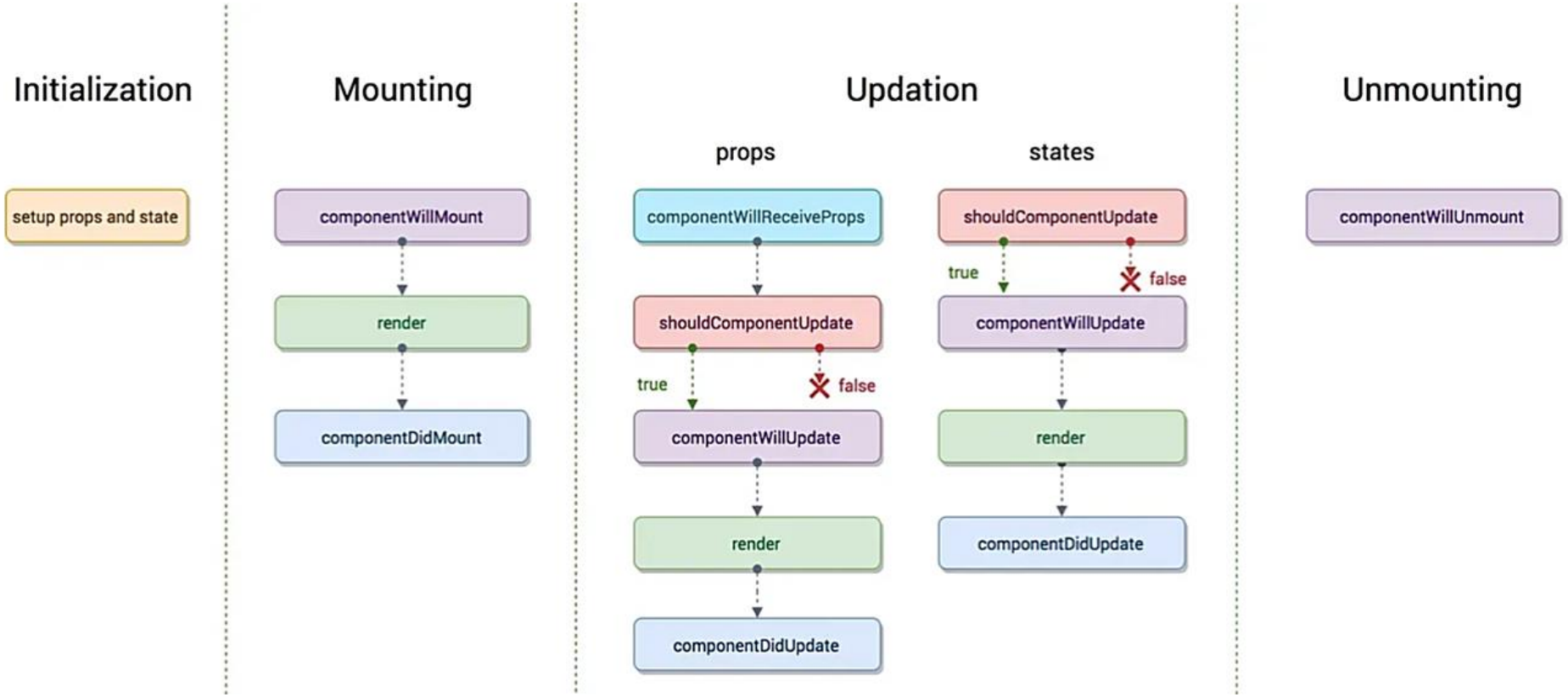
```tsx
import React from "react";

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

```tsx
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>
  );
};
```

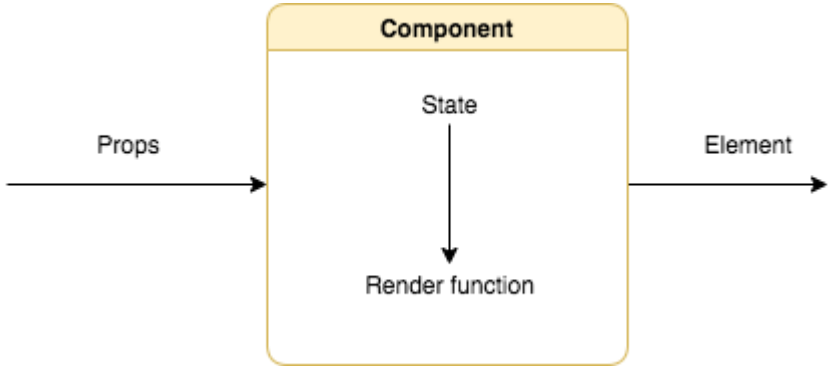
React component lifecycle



https://projects.wojtekmaj.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.

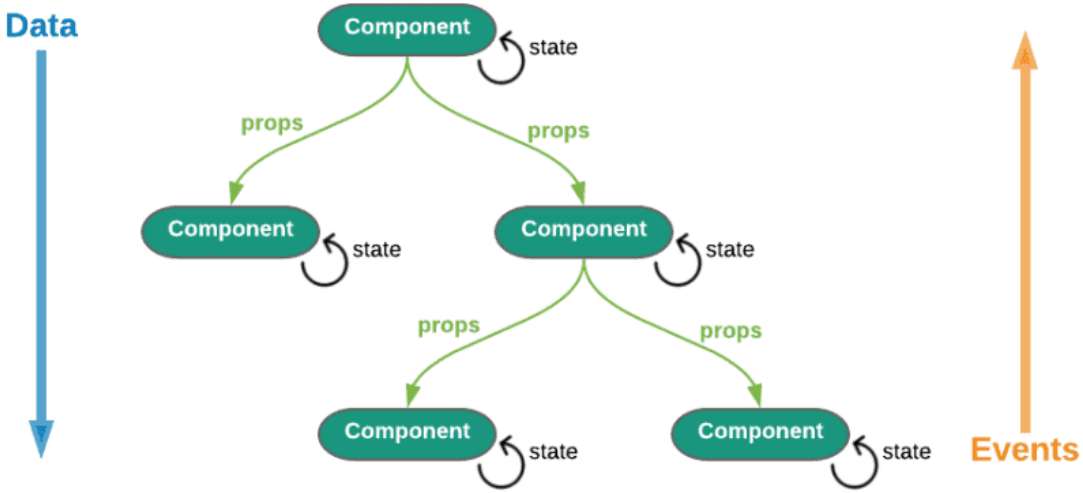


React Data flow

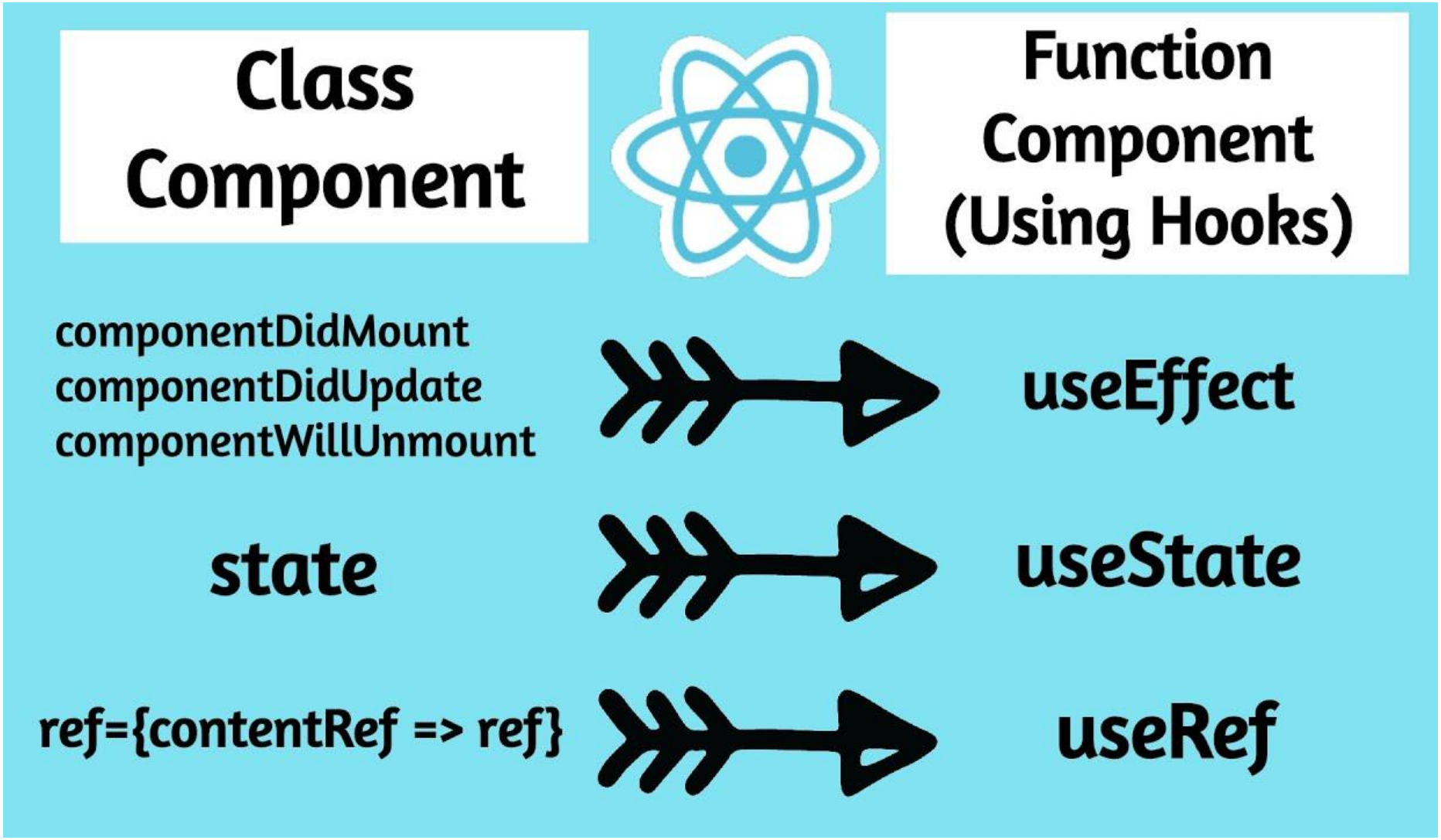
There are two types of data in react

state (private data available in component only)

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



React 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

```tsx
"use client";

import { FC, useState, useEffect } from "react";

interface Num {
  num: number;
}

const Counter: FC<Num> = ({ num }) => {
  const [cnt, setCnt] = useState<number>(0);
  const [cntEff, setCntEff] = useState<number>(0);

  useEffect(() => {
    document.title = `You clicked ${cnt} times`;
    setCntEff(cnt);

    return () => {
      console.log("clean up");
    };
  }, [cnt]);

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

  return (
    <>
      <div>{cnt}</div>
      <div>Effect : {cntEff}</div>
      <button onClick={() => setCnt(inc(cnt))}>Click Me</button>
    </>
  );
};
```

dependency array
If set to [ ]
useEffect will run
only one time

React Hook Recommendation style

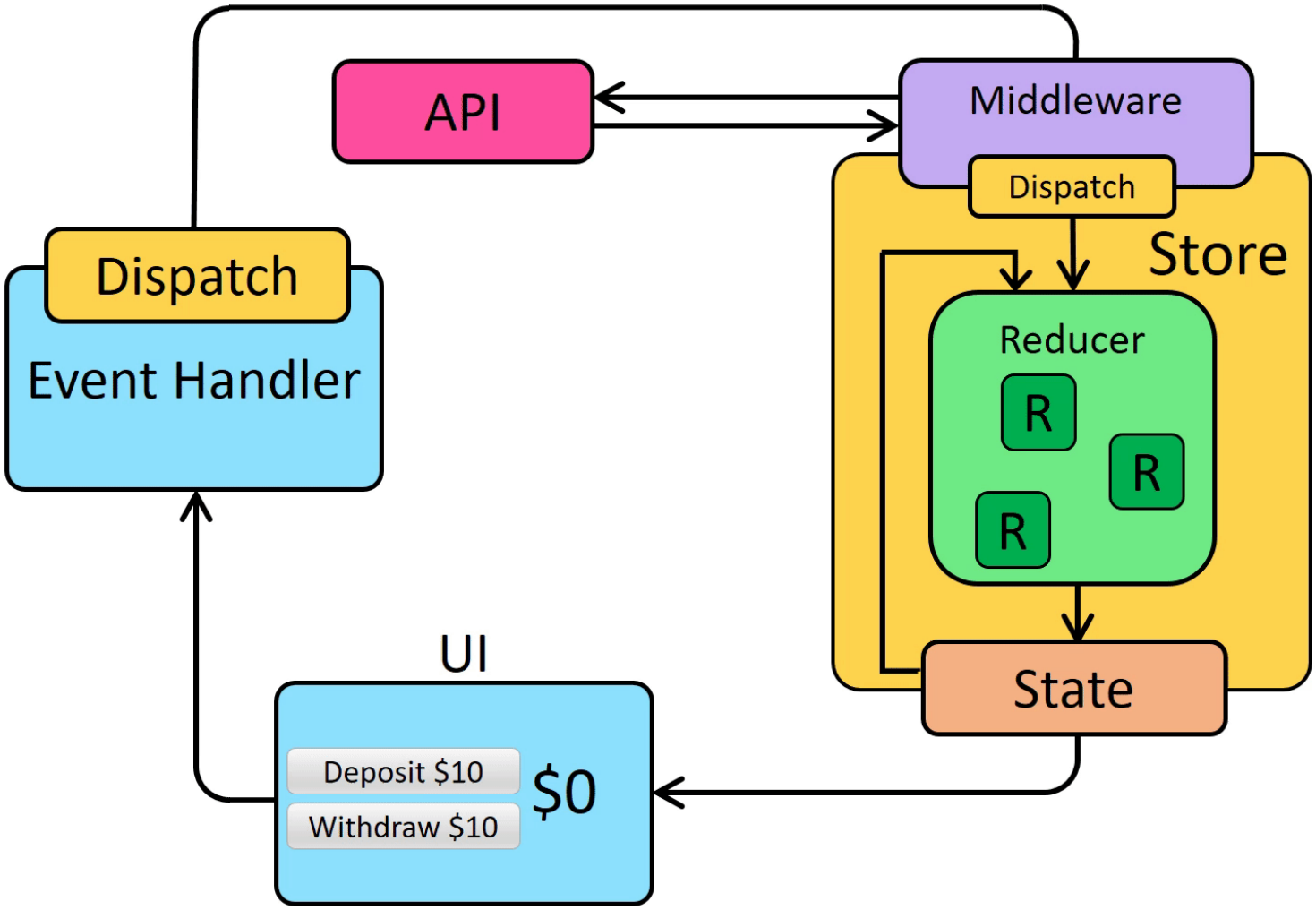
```tsx
const Hook: FC<Props> = ({ initHook }) => {
  const [count, setCount] = useState<number>(initHook);
  const [data, setData] = useState({});

  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=มะพร้าว";

    // 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

Guideline

Assemble components into a web page

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

Q & A