React.js

<ODDS /> | <Thaibev />

Goal

- Development
- Maintenance
- Improvement

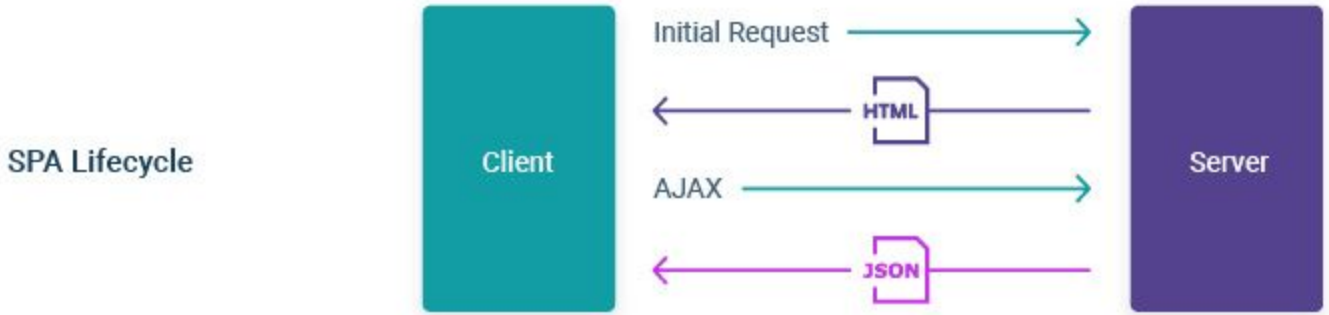
https://www.figma.com/file/ha9MBXL5SeoXbJKaVc3mWJ/SX22-Thai-Rev?node-id=3005%3A7887&t=JffxOMb803DRzLmB-4

Web?

https://www.google.com

Single page applications (SPA) ?
vs
Multi page applications (MPA) ?

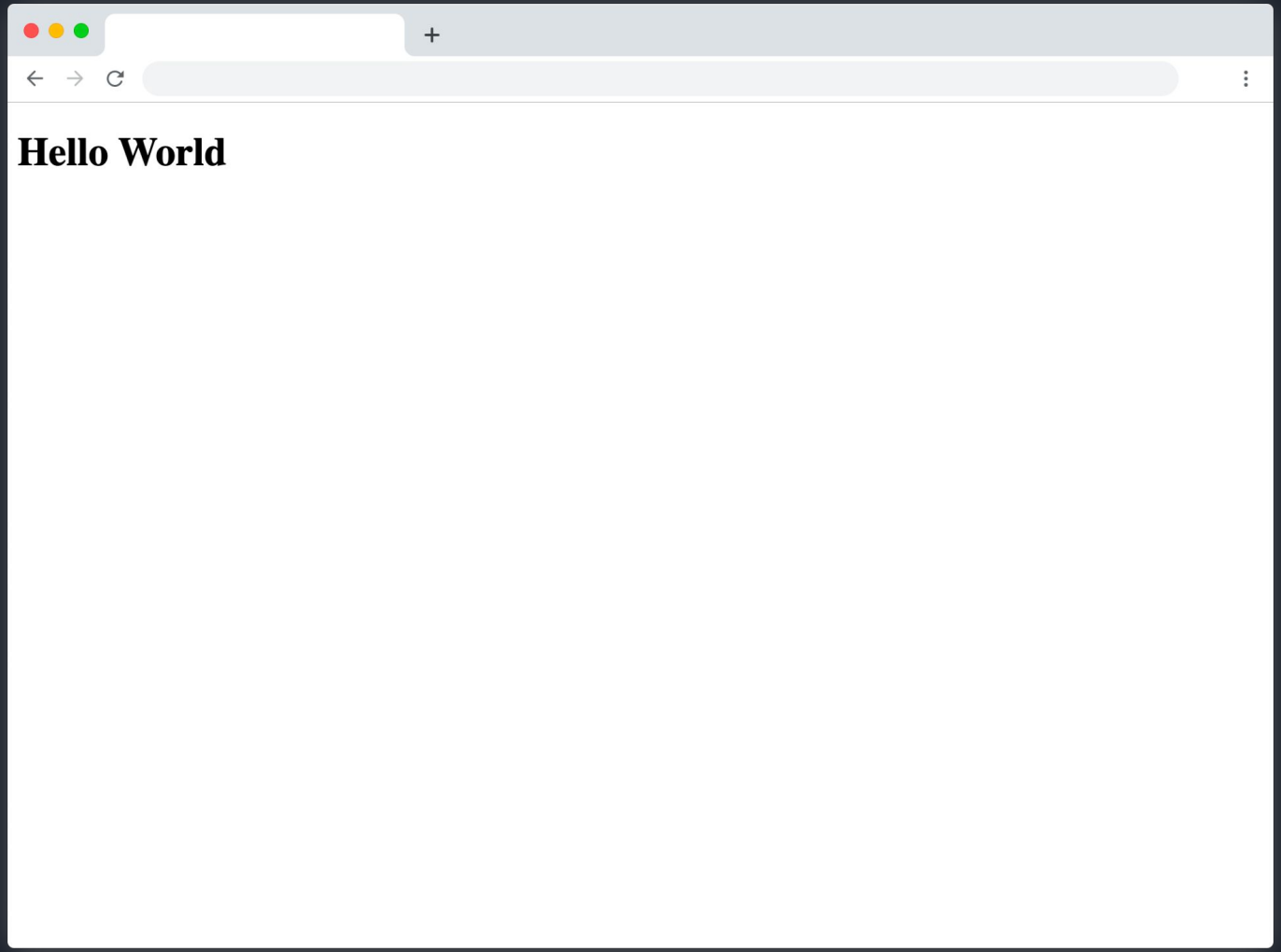

Multi-Page Lifecycle
Client
Initial Request
HTML
Form POST
HTML
Page Reload
Server


SPA Lifecycle
Client
Initial Request
HTML
AJAX
JSON
Server

	SPA	MPA
Sleek UX	✓	
Easy SEO		✓
Security		✓
Less server load	✓	
Offline functionality	✓	
Mobile adaptability	✓	
Application scalability		✓
UI/data separation	✓	
Speed	✓	
Fast launch		✓
Works without JavaScript		✓

HTML - HyperText Markup Language

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Hello World</h1>
</body>
</html>
```

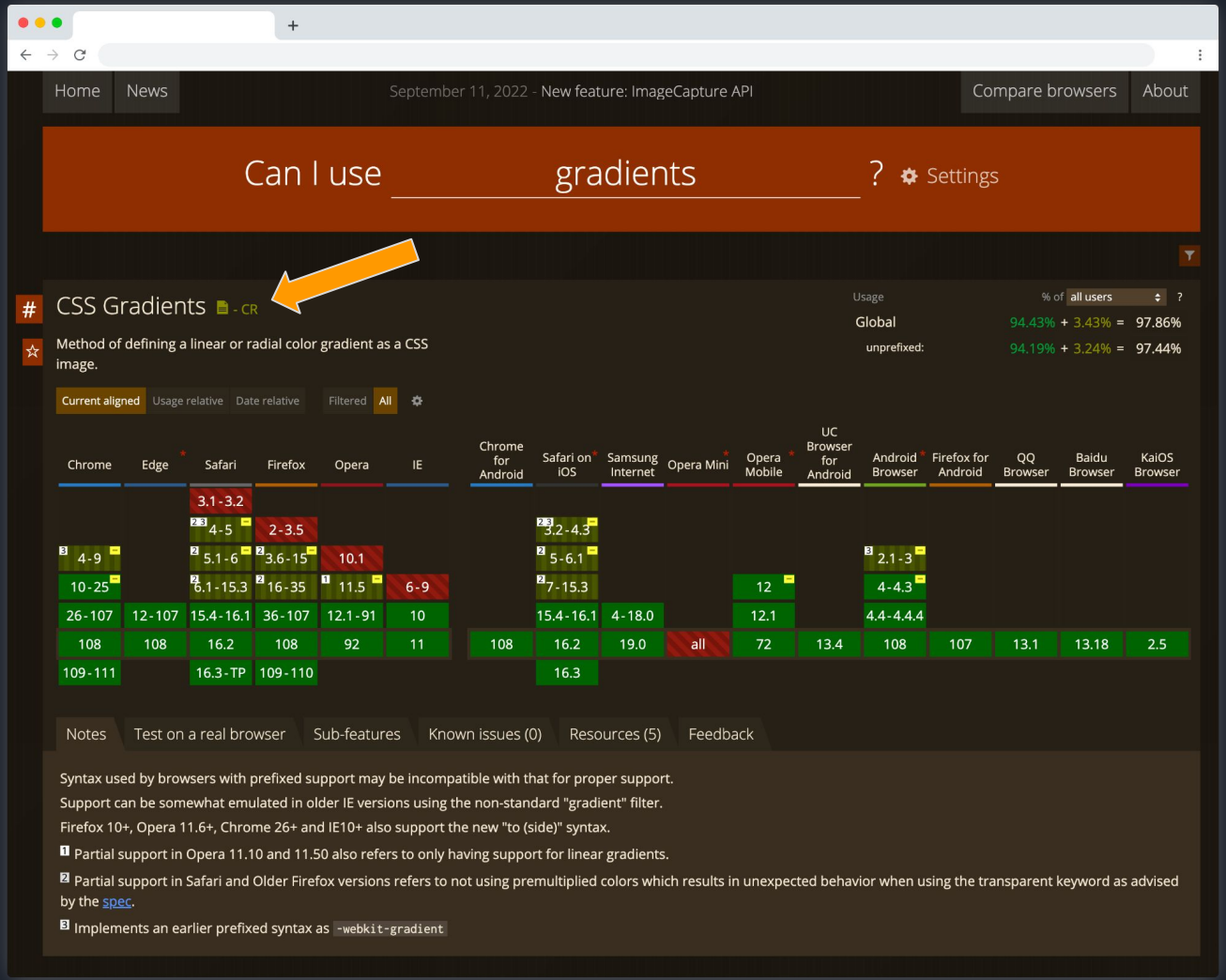

Hello World

CSS

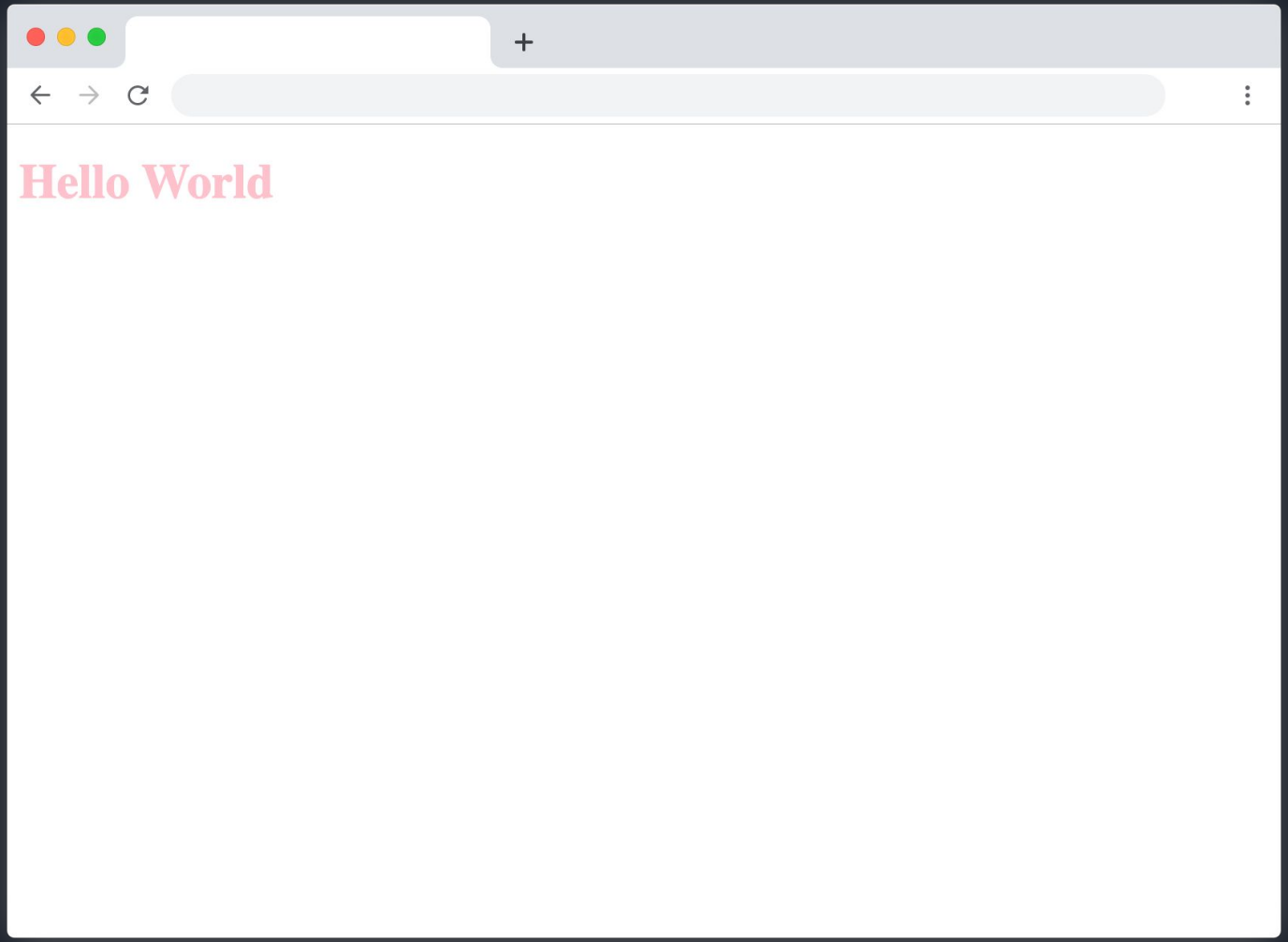
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1 style="color: pink; font-weight: bold;">
        Hello World
    </h1>
</body>
</html>
```

https://developer.mozilla.org/en-US/docs/Web/CSS

https://caniuse.com/

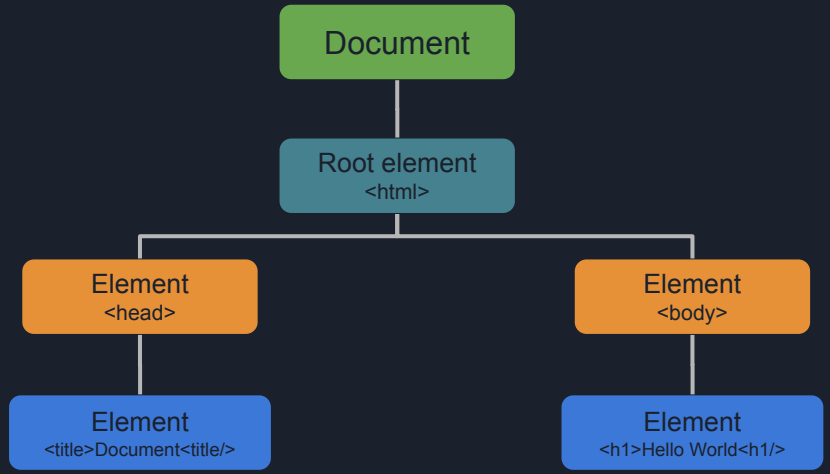

Home
News
September 11, 2022 - New feature: ImageCapture API
Compare browsers
About
Can I use gradients ?
Settings
CSS Gradients - CR
Method of defining a linear or radial color gradient as a CSS image.
Usage
% of all users
Global 94.43% + 3.43% = 97.86%
unprefixed: 94.19% + 3.24% = 97.44%
Current aligned
Usage relative
Date relative
Filtered
All
Chrome
Edge
Safari
Firefox
Opera
IE
Chrome for Android
Safari on iOS
Samsung Internet
Opera Mini
Opera Mobile
UC Browser for Android
Android Browser
Firefox for Android
QQ Browser
Baidu Browser
KaiOS Browser
3.1-3.2
4-5
2-3.5
3.2-4.3
4-9
5.1-6
3.6-15
10.1
5-6.1
2.1-3
10-25
6.1-15.3
16-35
11.5
6-9
7-15.3
12
4-4.3
26-107
12-107
15.4-16.1
36-107
12.1-91
10
15.4-16.1
4-18.0
12.1
4.4-4.4.4
108
108
16.2
108
92
11
108
16.2
19.0
all
72
13.4
108
107
13.1
13.18
2.5
109-111
16.3-TP
109-110
16.3
Notes
Test on a real browser
Sub-features
Known issues (0)
Resources (5)
Feedback
Syntax used by browsers with prefixed support may be incompatible with that for proper support.
Support can be somewhat emulated in older IE versions using the non-standard "gradient" filter.
Firefox 10+, Opera 11.6+, Chrome 26+ and IE10+ also support the new "to (side)" syntax.
1 Partial support in Opera 11.10 and 11.50 also refers to only having support for linear gradients.
2 Partial support in Safari and Older Firefox versions refers to not using premultiplied colors which results in unexpected behavior when using the transparent keyword as advised by the spec.
3 Implements an earlier prefixed syntax as -webkit-gradient

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .title {
            color: pink;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h1 class="title">Hello World</h1>
</body>
</html>
```


Hello World

DOM - Document Object Model

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <h1>Hello World</h1>
</body>
</html>
```

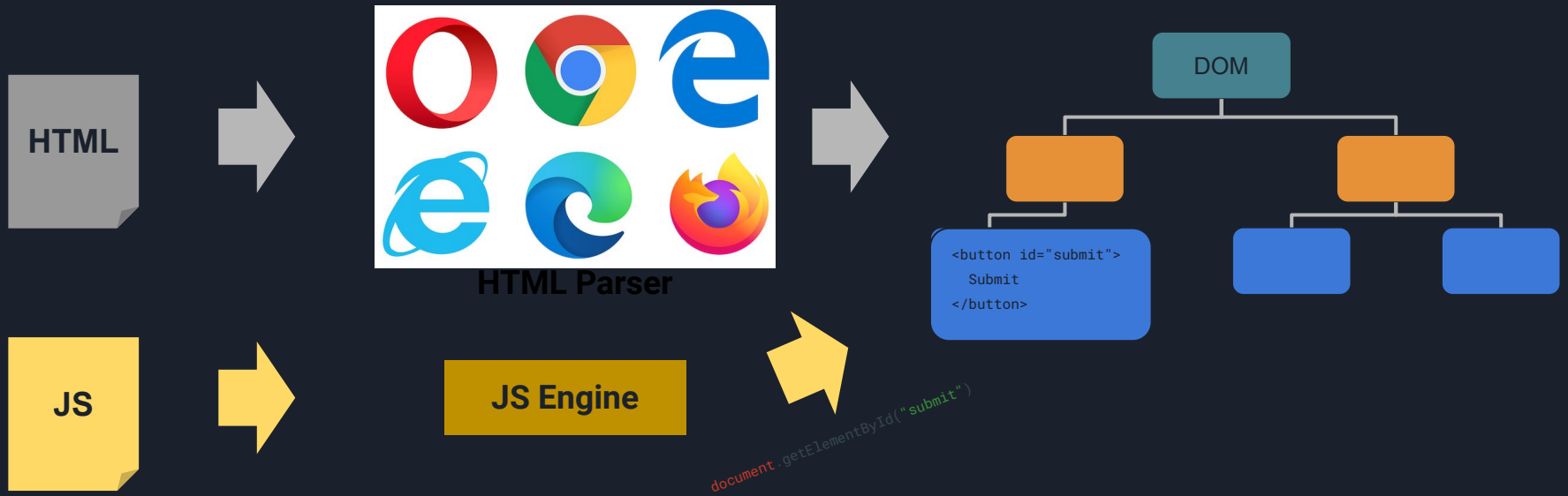



```
document.getElementById(id)
document.getElementsByTagName(name)
document.getElementsByClassName(name)
document.getElementById(id).onclick = function(){code}
```

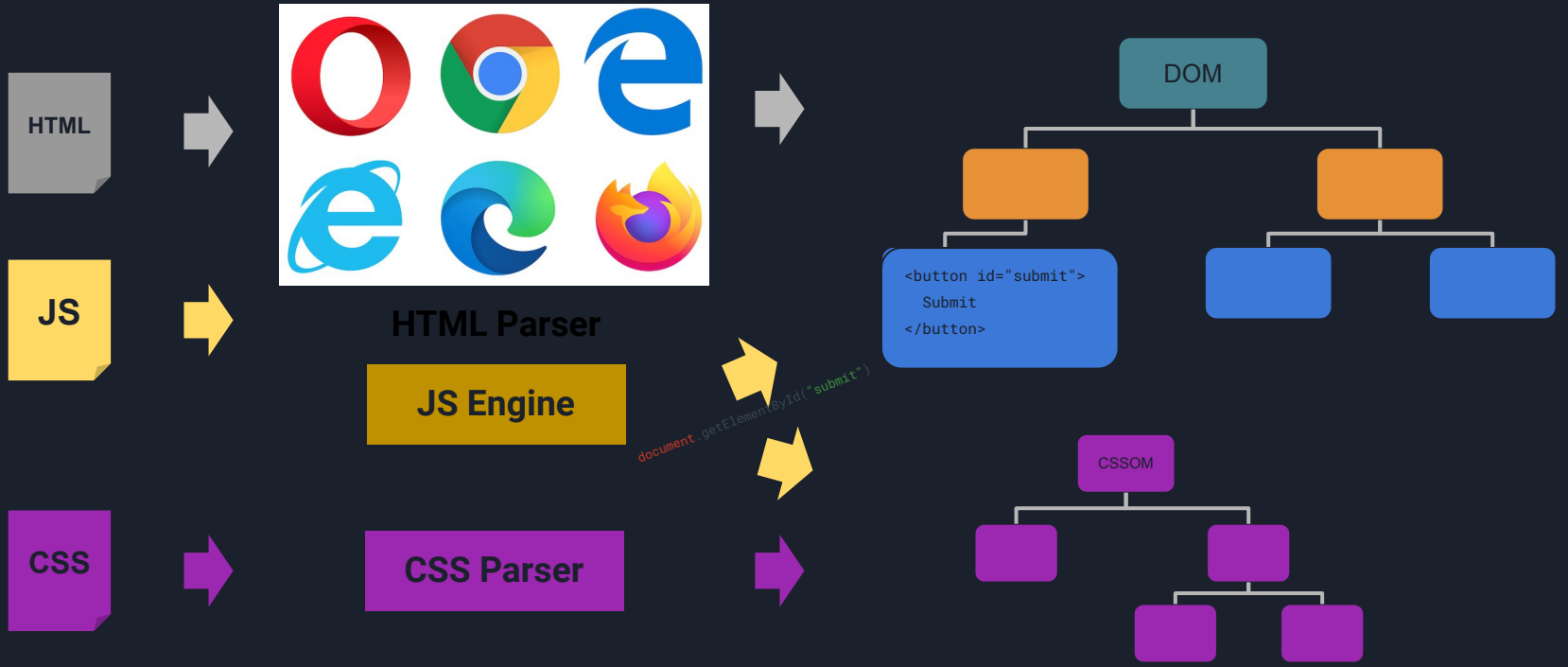
https://dom.spec.whatwg.org/

DOM - Document Object Model

DOM - Document Object Model



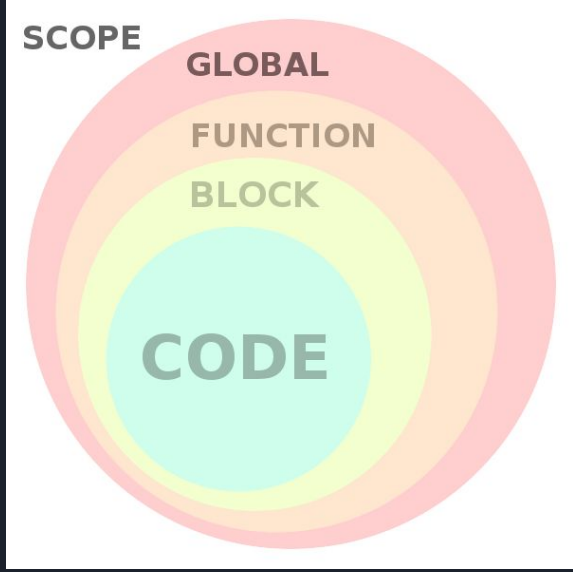
CSSOM - CSS Object Model



Basic Javascript

JavaScript is a programming language that is primarily used to create web pages and applications that are interactive and run in a web browser. It is a client-side language, meaning that it is executed by the user's web browser, rather than on a server.

Variables




```
var z = 'hello'; // globally scoped or function scoped
let x; // block scoped, updated but not re-declared
const y = 42; // block scoped, cannot updated or re-declared
```

https://codesandbox.io/s/01-js-variable-8przpj?file=/src/index.js

JS

Data Type

- Primitive data types
- Non-primitive data types

Primitive data types

Primitive data types are stored directly in memory and are passed by value

`number`: represents numeric values, including integers and floating-point numbers.

`string`: represents a sequence of characters, such as words or phrases.

`boolean`: represents a true or false value.

`null`: represents the absence of a value or a null object reference.

`undefined`: represents the absence of a value or a variable that has not been assigned a value.

`symbol`: represents a unique and immutable value.

Non-primitive data types

`non-primitive data types are stored as references and are passed by reference`

object: Objects can be used to store and manipulate complex data structures, such as arrays and other objects.

array: Arrays are objects and have a length property and a set of methods for manipulating their elements.

function: Functions are objects and have properties and methods, such as the name property and the apply() method.

Data Type

```
let x = 42;
let y = 'Hello, world!';
let z = true;
let a = null;
let b;
let c = {name: 'John', age: 30};
let d = [1, 2, 3, 4, 5];
```

https://codesandbox.io/s/02-data-type-dq4z14?file=/src/index.js

```
"11" - 1 = ?
```

Arithmetic operators

arithmetic operators are used to perform mathematical operations on numbers

```
+: the addition operator, used to add two numbers or concatenate two
strings.
-: the subtraction operator, used to subtract one number from another.
*: the multiplication operator, used to multiply two numbers.
/: the division operator, used to divide one number by another.
%: the modulo operator, used to find the remainder of a division
operation.
++: the increment operator, used to increase a number by 1.
--: the decrement operator, used to decrease a number by 1.
```

"2" + "2" - "2" = ?

JS

Comparison operators

==: the equal operator, used to check if two values are equal.

!=: the not equal operator, used to check if two values are not equal.

===: the strict equal operator, used to check if two values are equal and of the same type.

!==: the strict not equal operator, used to check if two values are not equal or are not of the same type.

>: the greater than operator, used to check if one value is greater than another.

<: the less than operator, used to check if one value is less than another.

>=: the greater than or equal to operator, used to check if one value is greater than or equal to another.

<=: the less than or equal to operator, used to check if one value is less than or equal to another.

```
let x = 4;
let y = 2;
Let z = “4”;
console.log(x == y); // false
console.log(x != y); // true
console.log(x === y); // false
console.log(x !== y); // true
console.log(x > y); // true
console.log(x < y); // false
console.log(x >= y); // true
console.log(x <= y); // false
console.log(y == z); // true
```

```
0 == "0" - true
0 == [] - true
"0" == [] - false
```

Logical operators

logical operators are used to perform logical operations on boolean values and return a boolean result

&&: the logical AND operator, used to check if both operands are true.
||: the logical OR operator, used to check if at least one operand is true.
!: the logical NOT operator, used to negate a boolean value.

Conditional statement

```
if (boolean expression) {
  // code to be executed if the boolean expression is true
} else {
  // code to be executed if the boolean expression is false
}
```

Exercises

- `sum` receive two number, expect return sum of two numbers
- `greeting` receive name, expect return “Hi, my name is {name}”
- `isOdd` receive number, expect return true when number is odd
- `calculateGrade` receive point, expect return grade (A,B,C,D, F)
- `greaterNum` receive two number; expect return max value

https://codesandbox.io/s/exercise-1-6gi5lx

Switch Statement

```
switch (value) {
  case value1:
    // code to be executed if value === value1
    break;
  case value2:
    // code to be executed if value === value2
    break;
  ...
  default:
    // code to be executed if no cases match
}
```

Exercises with switch statement

Write a function that takes in a string as an argument and returns a corresponding message based on the string value. Use a switch statement to handle the following cases:

If the string is 'red', return 'Red is the color of love.'
If the string is 'orange', return 'Orange is the color of oranges.'
If the string is 'yellow', return 'Yellow is the color of the sun.'
If the string is 'green', return 'Green is the color of nature.'
If the string is 'blue', return 'Blue is the color of the sky.'
If the string is 'indigo', return 'Indigo is a deep blue color.'
If the string is 'violet', return 'Violet is a purple color.'

Looping constructs

for loop: used to execute a block of code a specific number of times.

while loop: used to execute a block of code as long as a boolean expression is true.

do-while loop: similar to the while loop, but the code block is executed at least once before the boolean expression is checked.

for-in loop: used to iterate over the **properties of an object**.

for-of loop: used to iterate over the **values of an iterable object**, such as an array.

```for
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

```do-while
let i = 0;
do {
  console.log(i);
  i++;
} while (i < 5);
```

```while
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

```for-of
let arr = [1, 2, 3, 4, 5];
for (let val of arr) {
  console.log(val);
}
```

```for-in
let obj = { key1: 'value1', key2: 'value2', key3: 'value3' };
for (let key in obj) {
  console.log(key + ': ' + obj[key]);
}
```

Exercises

- Write a for loop that iterates over an array of numbers and logs the square of each number to the console.
- Write a while loop that counts down from 10 to 1 and logs the count to the console.
- Write a for loop that iterates over an array of strings and logs the number of characters in each string to the console.
- Write a for loop that iterates over an array of objects and logs a specific property of each object to the console.
- Counting word in array of strings
- Write the function check the number is prime number
- Write the function check the string is palindrome

https://codesandbox.io/s/exercise-2-ycnghy?file=/src/index.js

Async Control Flow

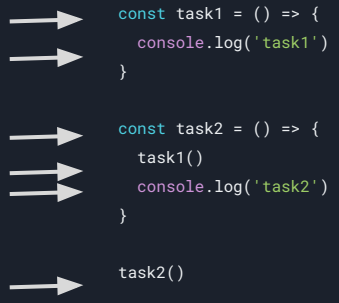
JavaScript engines are built around a single main **event loop**, which is a queue of functions. When a program starts, the engine evaluates all JavaScript code at the top-level scope and then goes "idle" until new "events" (functions) are added to the event loop.

New events may be inserted into the event loop at any time, either by our code or the JavaScript environment (usually in response to I/O), which are then executed in sequence.

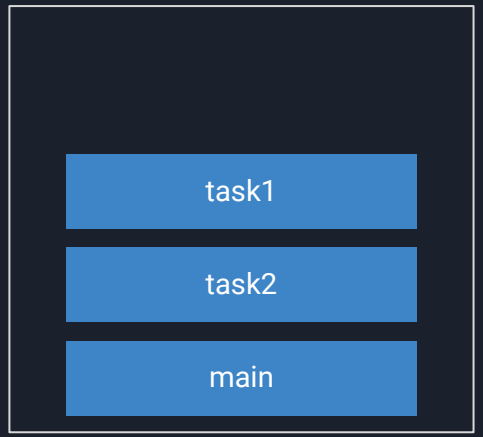
We use **callback functions** and **Promises** to work with the event loop.

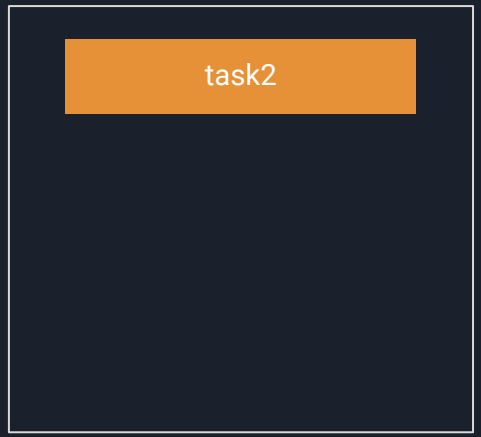
Event Loop

Credit - https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif



```
const task1 = () => {
  console.log('task1')
}

const task2 = () => {
  task1()
  console.log('task2')
}

task2()
```

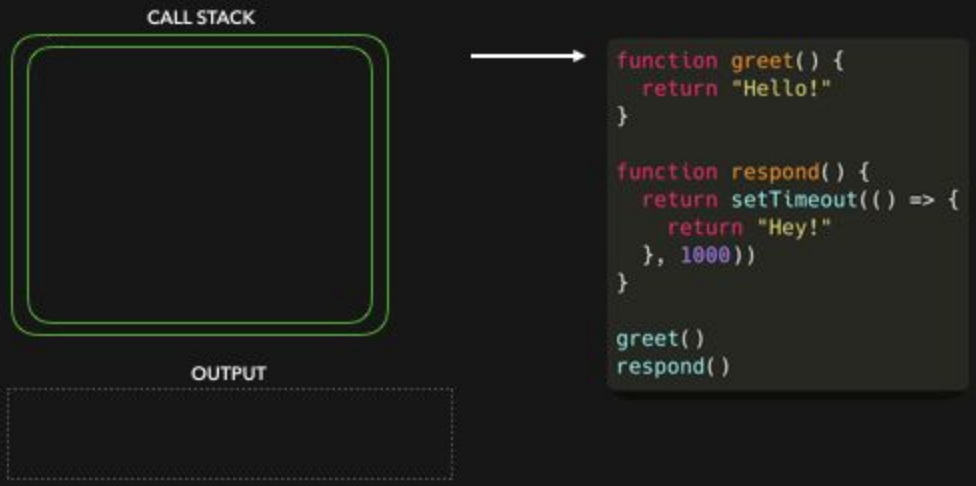




```
function greet() {
    return "Hello!"
}

function respond() {
    return setTimeout(() => {
        return "Hey!"
    }, 1000)
}

greet()
respond()
```

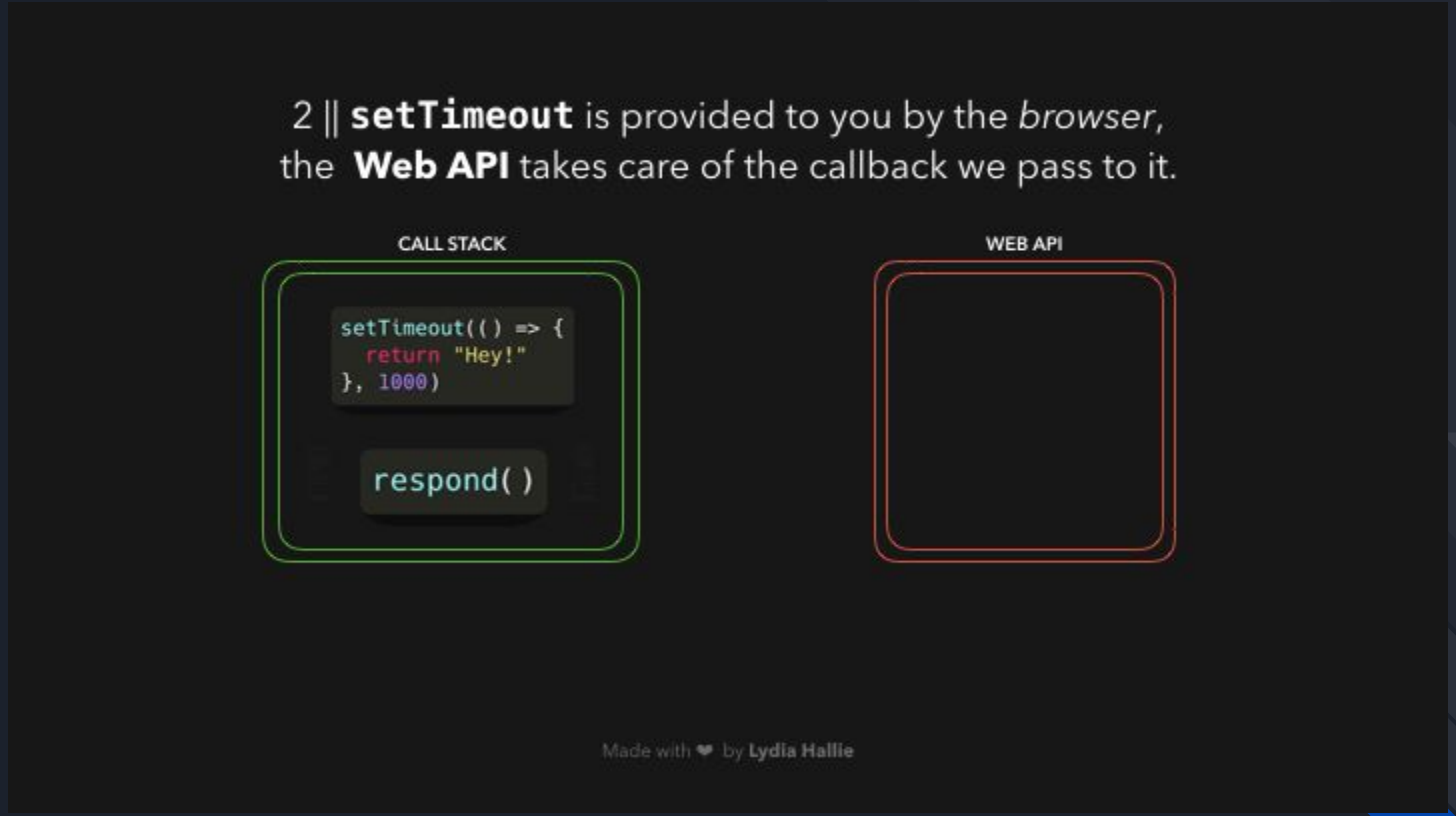
Credit - https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif

1 || Functions get **pushed to** the call stack when they're **invoked** and **popped off** when they **return a value**



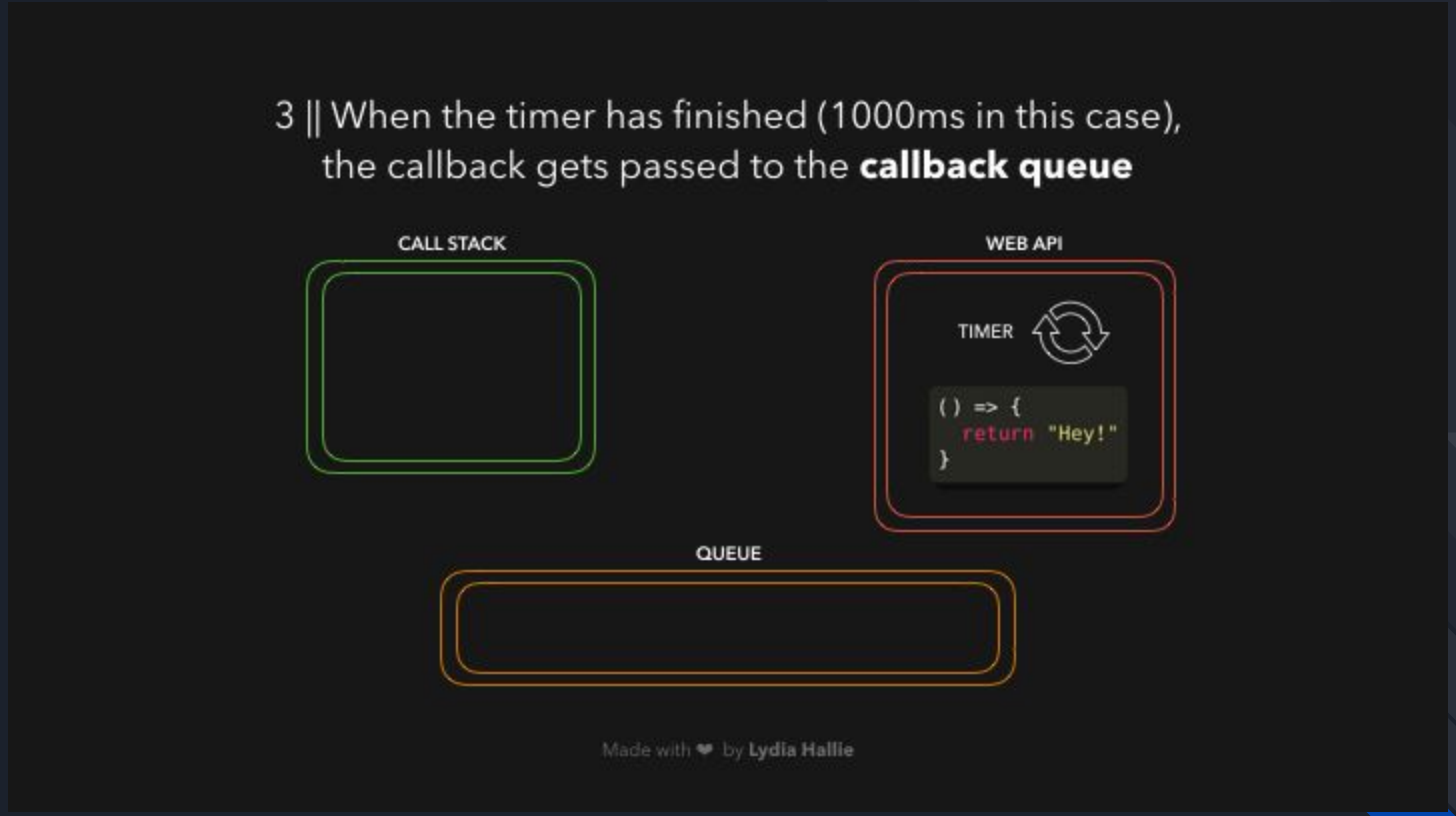
Credit - https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif

JS



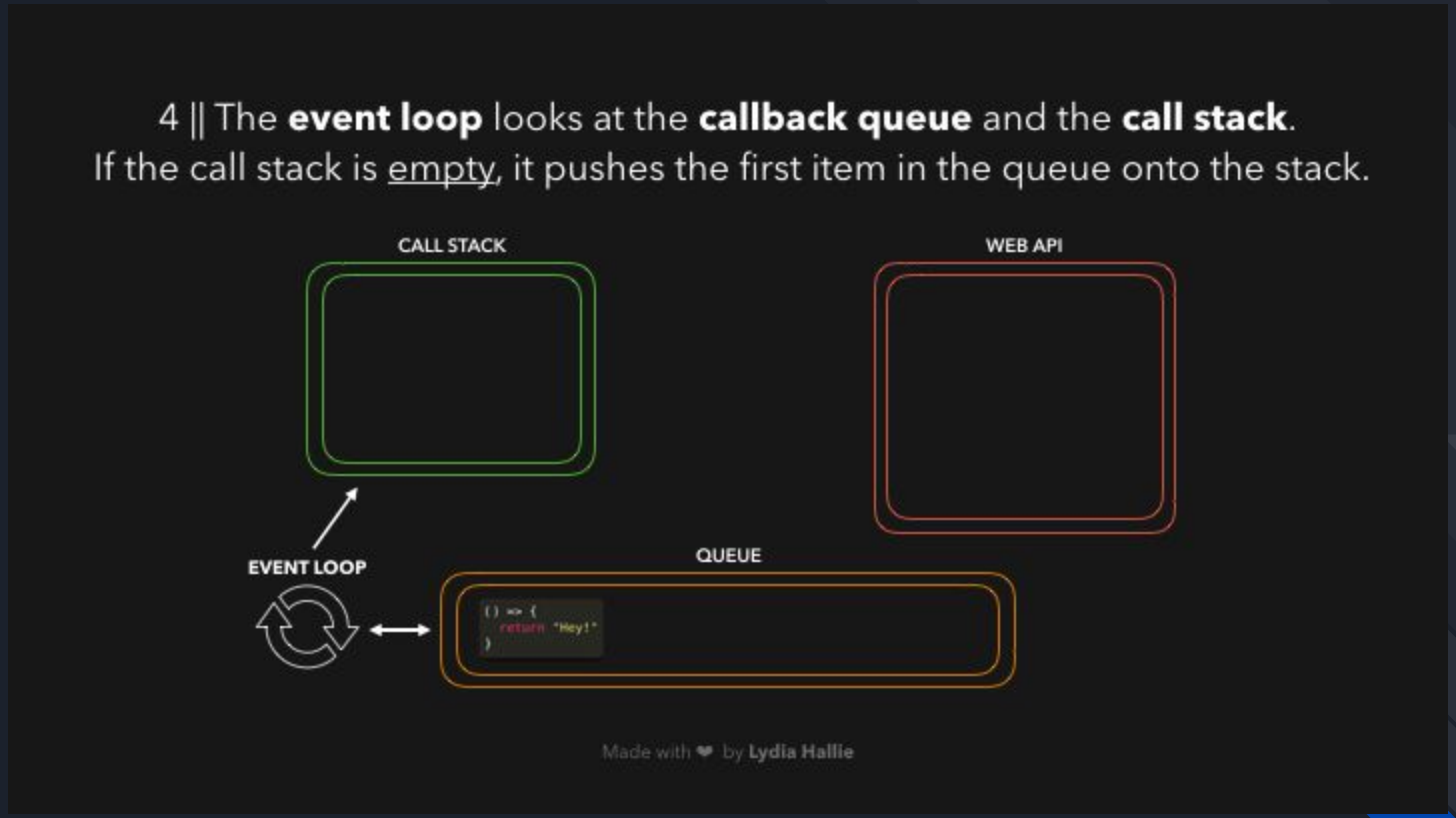
Credit - https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif

JS



Credit - https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif

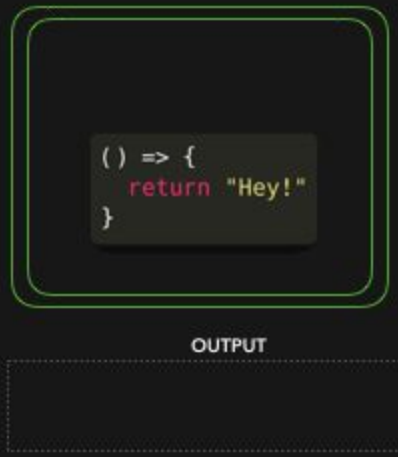
JS



Credit - https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif

JS

5 || The callback is added to the call stack and executed.
Once it returned a value, it gets popped off the call stack.




```
function greet() {
  return "Hello!"
}

function respond() {
  return setTimeout(() => {
    return "Hey!"
  }, 1000))
}

greet()
respond()
```

Credit - https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif

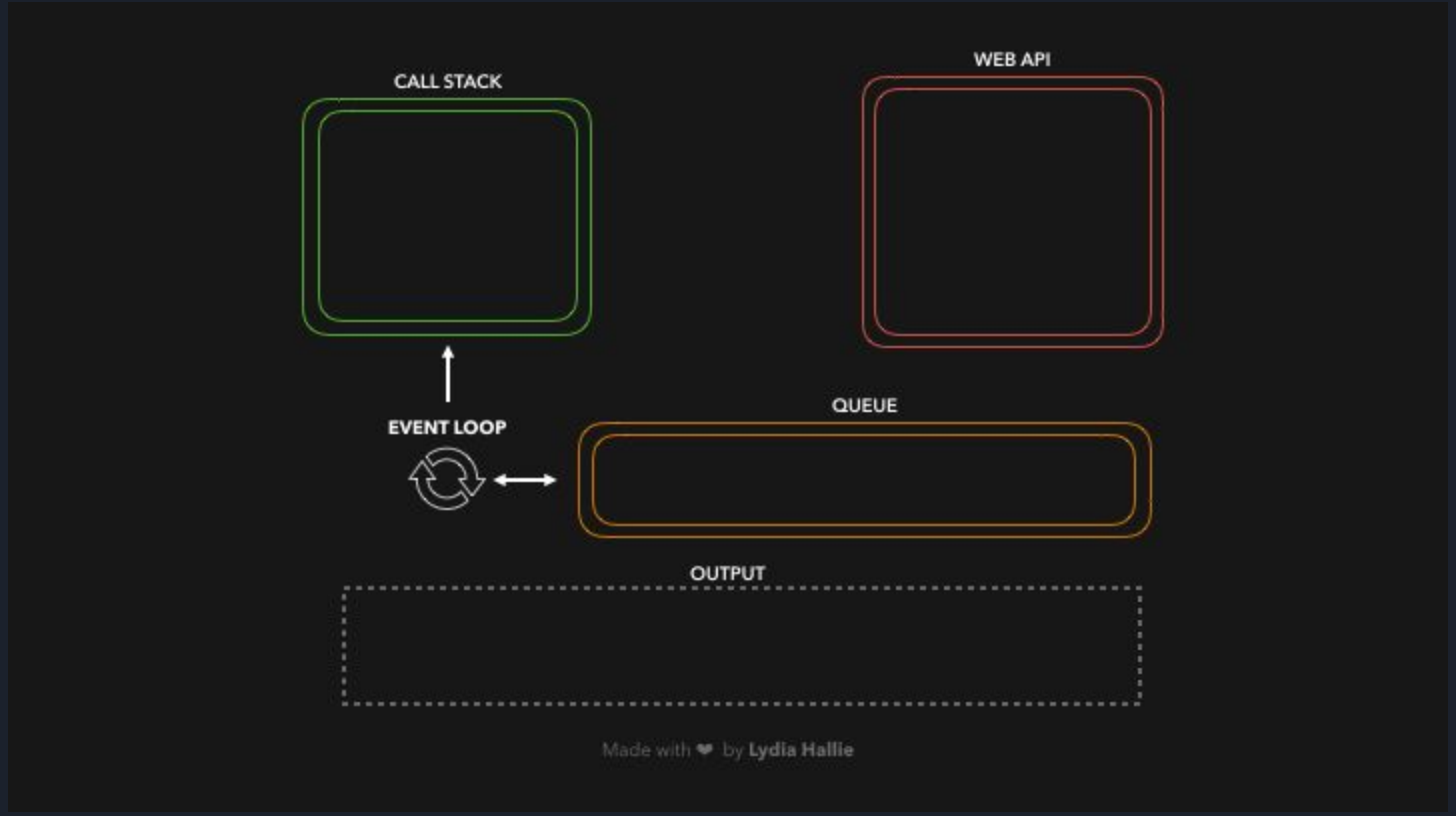
JS

Exercises

```
const foo = () => console.log("First");
const bar = () => setTimeout(() => console.log("Second"), 500);
const baz = () => console.log("Third");

bar();
foo();
baz();
```

https://url.odds.team/X9OrrE

Credit - https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif



Credit - https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif

JS

```
const sayHello = () => {
  setTimeout(() => {
    console.log("Hello");
  }, 2000);
};

const sayWorld = () => {
  sayHello();

  setTimeout(() => {
    console.log("World");
  }, 1000);
};

const say = () => {
  sayWorld();
};

say();
```

https://url.odds.team/prwr9r

https://url.odds.team/xZSi6C

Promises

Promises are an abstraction for asynchronous control flow — they're a proxy object for a value that may not exist yet. A Promise is always in 1 of 3 states internally:

- ✅ fulfilled: The promise has been resolved. Everything went fine, no errors occurred within the promise 🥳
- ❌ rejected : The promise has been rejected. Argh, something went wrong..
- ⏳ pending: The promise has neither resolved nor rejected (yet), the promise is still pending.

A resolved or rejected promise optionally contains a value.

Every promise starts in the pending state, and can only transition once (i.e. a resolved promise never becomes rejected later).

JS

Creating promises

```
const resolvedPromise = Promise.resolve(42)

resolvedPromise.then(value => {
  console.log(value)
})
```

```
const rejectedPromise = Promise.reject('Some problem')

rejectedPromise.catch(value => {
  console.log(value)
})
```

```
const functionPromise = new Promise((resolve, reject) => {
  resolve('Hello')
})

functionPromise
  .then(value => {
    console.log('Resolved', value)
  })
  .catch(value => {
    console.log('Rejected', value)
  })
```

Chaining promises

```
Promise.resolve(42)
  .then(value => value * 2)
  .then(value => Promise.resolve(value + 1000))
  .then(value => {
    console.log(value)
  })
```

```
Promise.resolve(42)
  .then(value => value * 2)
  .then(value => Promise.reject(value))
  .then(value => Promise.resolve(value + 1000))
  .catch(value => {
    console.log('Caught!', value)
    return value
  })
  .then(value => value * 2)
  .then(value => {
    console.log('OK', value)
  })
```

Handling Single and Multiple promises

```
function sleep(delay, value) {
  return new Promise(resolve => {
    setTimeout(() => resolve(value), delay)
  })
}

sleep(3000, 'Devin').then(value => {
  console.log(`Hello ${value}`)
})
```

```
const promiseA = sleep(3000, 'a')
const promiseB = sleep(1000, 'b')
const promiseC = sleep(2000, 'c')

Promise.all([
   promiseA, promiseB, promiseC
]).then(values => {
  console.log(values)
})

Promise.race([
   promiseA, promiseB, promiseC
]).then(value => {
  console.log(value)
})
```

Exercises

- Ex1 - https://url.odds.team/yfEgQk

Async and Await

```
function sleep(delay, value) {
  return new Promise((resolve) => {
    setTimeout(() => resolve(value), delay)
  })
}

async function run() {
  const value = await sleep(2000, 'hello')

  console.log(value)

  return value
}

run()
```

https://url.odds.team/Qgmr8u

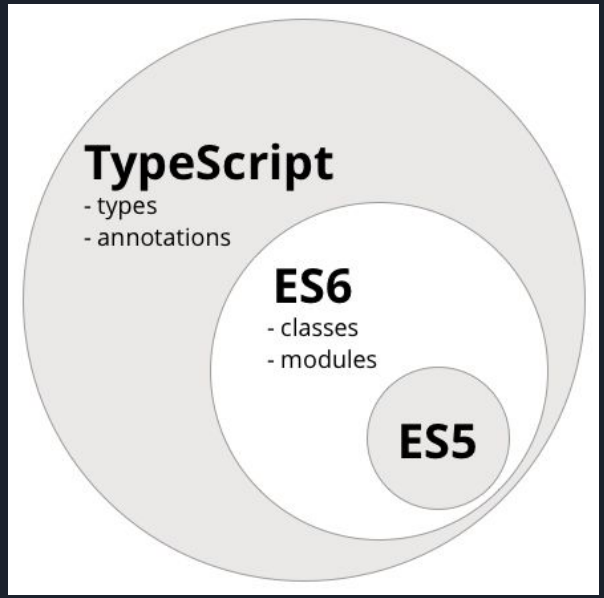
JS

Exercises

- Ex1 - https://url.odds.team/kxSSSA

Typescript

TypeScript is a programming language that is a superset of JavaScript. It adds optional **static typing** to JavaScript, which can help to **catch errors and bugs at compile-time** rather than runtime. TypeScript code is transpiled (converted) into JavaScript code, so it can run in any JavaScript environment.



TS

Transpile Ts to Js

```typescript
// TypeScript sample.ts
function greet(name: string): string {
  return `Hello, ${name}!`;
}

console.log(greet('John')); // 'Hello, John!'
```

npm install -g typescript

tsc sample.ts --outFile ./sample.js

Try it!

```
function add(x: number, y: number): number {
  return x + y;
}

console.log(add(10, 20)); // prints 30
```

https://url.odds.team/LevdrX

Module

a module is a way to organize and reuse code across your application

https://url.odds.team/5gcEkn

Module (Export as single value/object)

```ts
// math.ts

export default function add(x: number, y: number): number {

  return x + y;

}

// main.ts

import add from './math';

console.log(add(10, 20)); // prints 30
```

Importing from modules (libs)

```
import moment from 'moment'

console.log(moment().format('MMMM Do YYYY, h:mm:ss a'))
```

Introduction to React

- React is a JavaScript **Library** for building user interfaces
- Developed by **Facebook** and now maintained by a community of developers
- Declarative approach to building UI: describe the desired state, React takes care of the rest
- **Virtual DOM**: lightweight representation of the actual DOM, helps improve performance

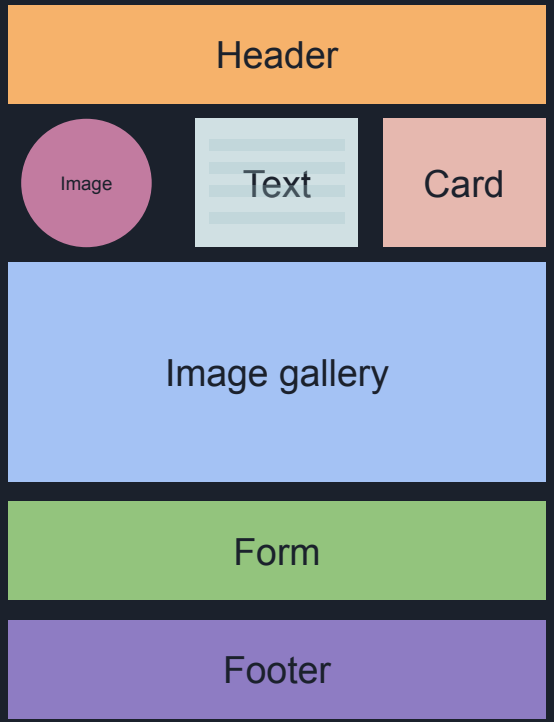
React

Facebook
Open Source

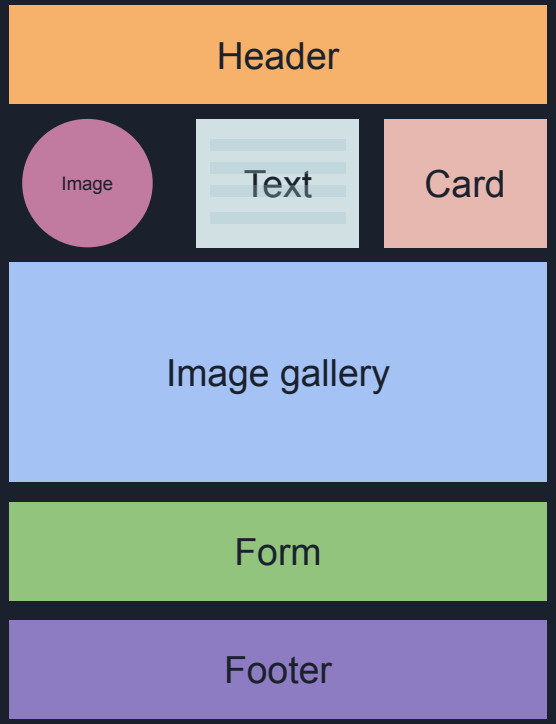
React Concepts

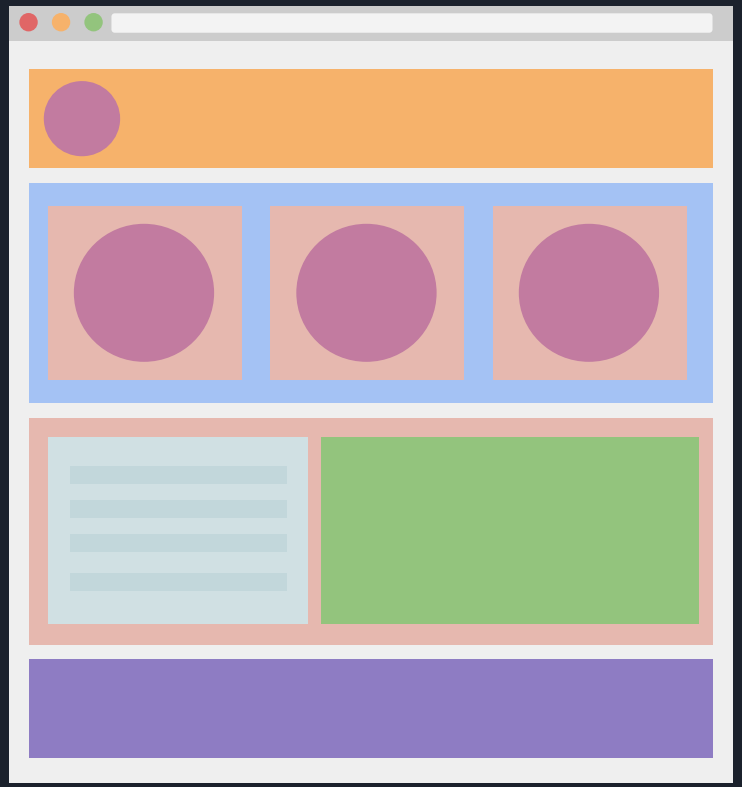
- Components, Props, State
- Virtual DOM
- JSX

Component

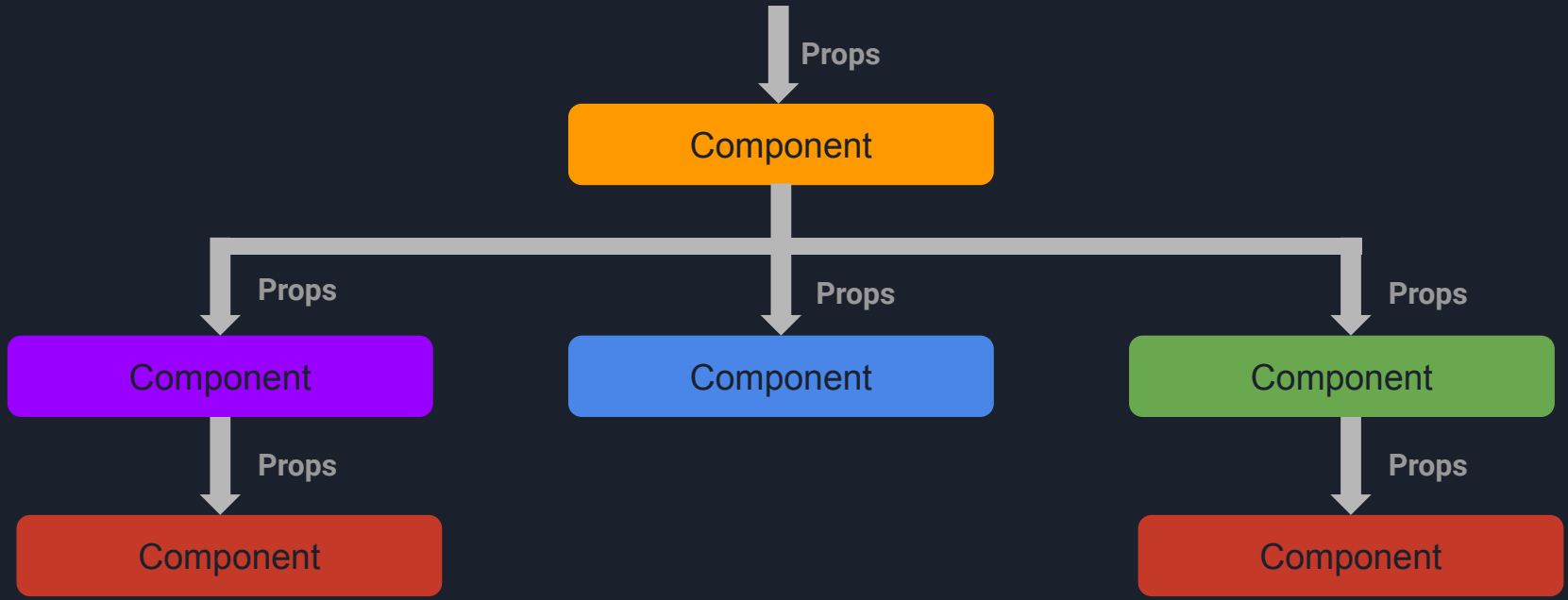


Component

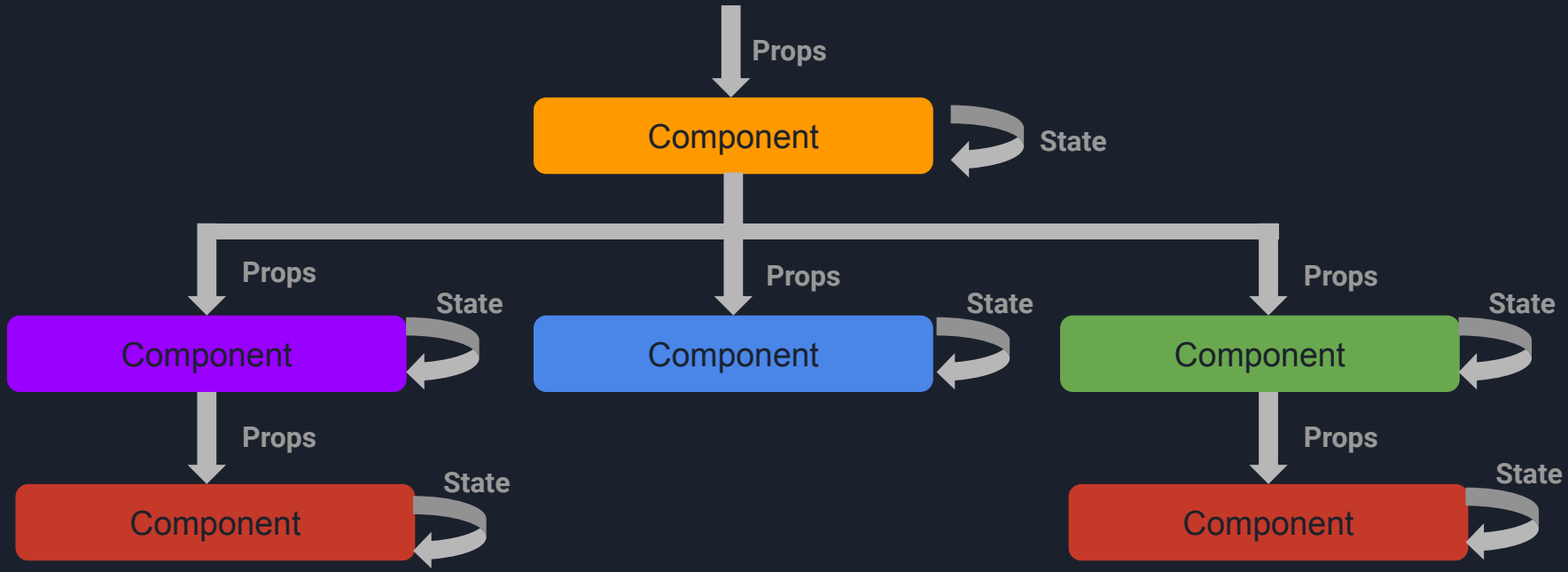




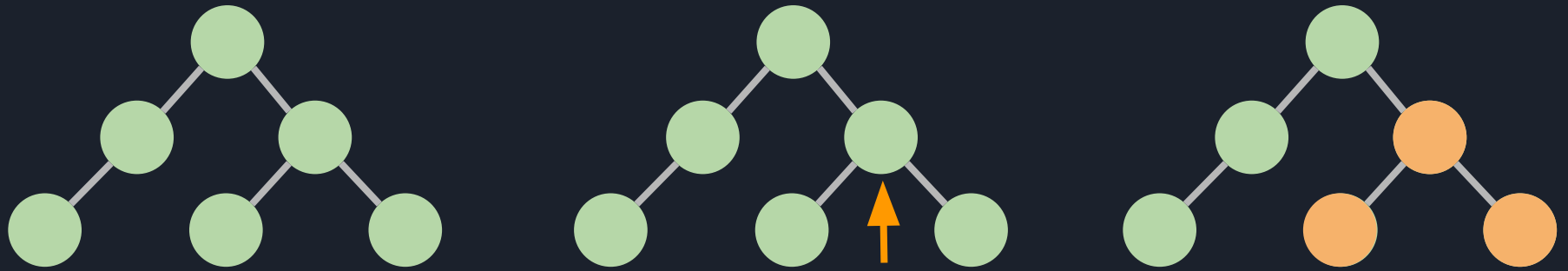
Props


Props
Component
Props
Component
Props
Component
Props
Component
Props
Component
Props
Component

State


Props
Component
State
Props
Component
State
Props
Component
State
Props
Component
State
Props
Component
State
Props
Component
State

Virtual DOM



JSX

- Syntax extension for JavaScript that allows you to write **HTML-like** code in your React **components**
- Transformed into vanilla JavaScript by Babel
- Used to create React elements and inject dynamic data

```
<div>Hello, World!</div>
<h1 className="title">Hello, World!</h1>
<p>Hello, World!</p>
<a href="/">Home</a>
<img src="logo.png" alt="Logo" />
<button onClick={handleClick}>Click me</button>
```

```jsx
const element = <div />
```

```jsx
const element = React.createElement("div", null);
```

```jsx
const element = (
  <div>
    <button />
    <span />
  </div>
)
```

```jsx
const element = React.createElement("div", null,
    React.createElement("button", null),
    React.createElement("span", null)
);
```

Setting up...

NodeJS

Windows/Linux/MacOS
Download: https://nodejs.org/en/

Editor (vs code)

Windows/Linux/MacOS
Download: https://code.visualstudio.com/

MacOS

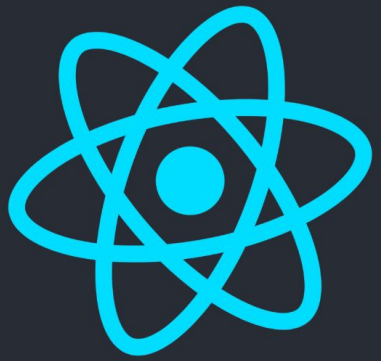
```
$ brew install node
```

Init app

```
npm install -g create-react-app

create-react-app my-app --template typescript
```

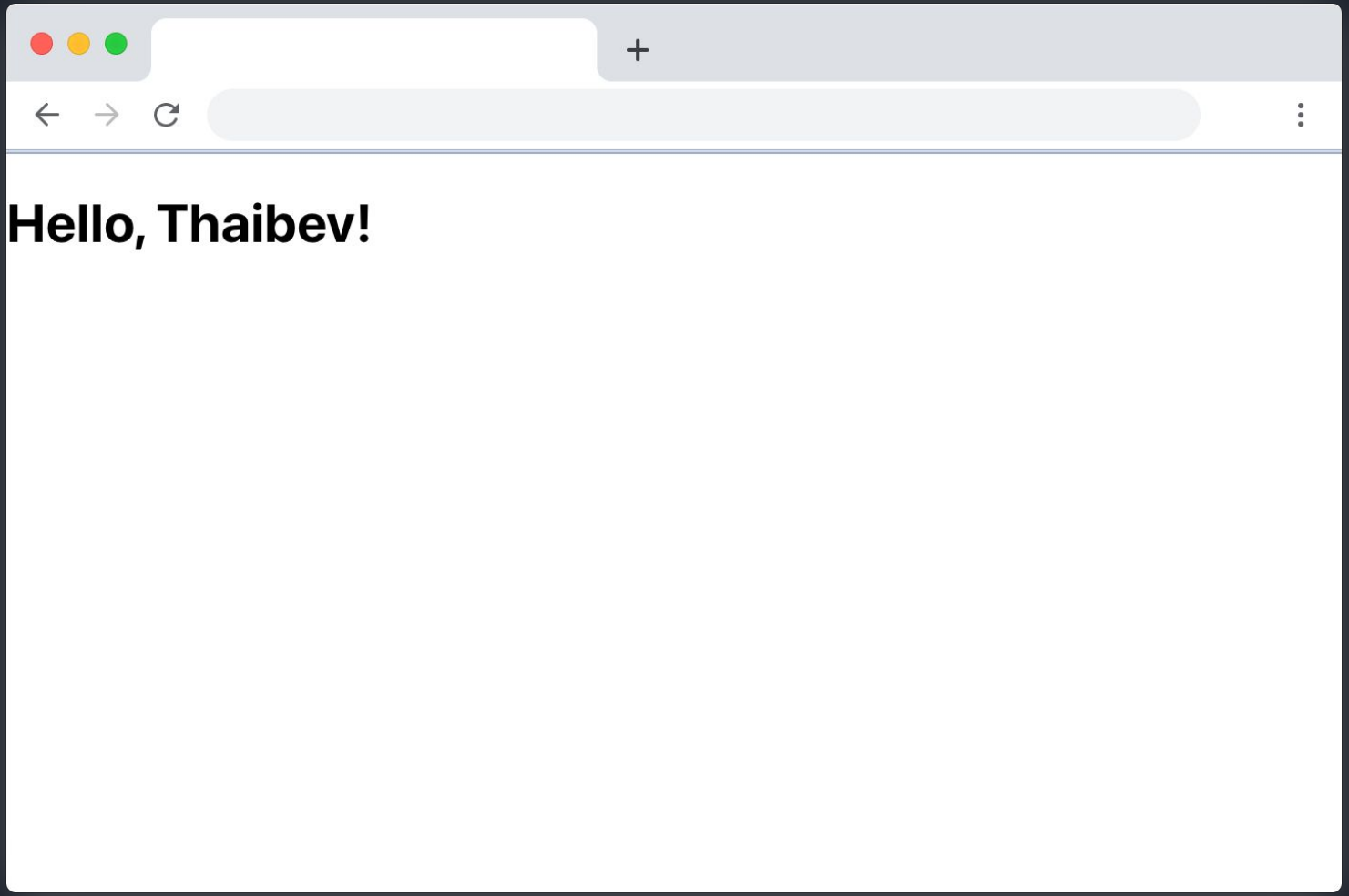
```
cd my-app
npm run start
```

http://localhost:3000



Edit `src/App.js` and save to reload.

Learn React


Hello, Thaibev!

```tsx
import React from 'react';

function HelloWorld() {
  const name = 'John';
  const age = 30;

  return (
    <div>
      <h1>Hello, {name}!</h1>
      <p>You are {age} years old.</p>
    </div>
  );
}

export default HelloWorld;
```

https://codesandbox.io/s/crazy-lena-ewknsp?file=/src/HelloWorld.tsx

```php
<?php
$name = "John Doe";
$age = 30;
      echo "<h1>Hello, {$name}!</h1><br/>";
      echo "<p>You are {$age} years old.</p>";
?>
```

Display Data

```
import React from 'react';

function App() {
  const name = 'Bob';
  const age = 32;

  return (
    <div>
        <span>Hello, my name is {name} and I am {age} years old.</span>
    </div>
  );
}

export default App;
```

https://codesandbox.io/s/funny-gates-tnlztl?file=/src/App.tsx

Conditional Rendering

```
import React from 'react';

function App() {
  const isLoading = true;

  return (
    <div>
      {isLoading ? <p>Loading...</p> : <p>Loaded</p>}
    </div>
  );
}

export default App;
```

Conditional Rendering (II)

```
import React from 'react';

function App() {
  const isLoading = true;

  return (
    <div>
      {isLoading && <p>Loading...</p>}
      {!isLoading && <p>Loaded</p>}
    </div>
  );
}

export default App;
```

List rendering

```
interface Student {
  id: number;
  name: string;
  age: number;
}
...
students.map((student) =>
  <li key={student.id}>
     {student.name}, Age: {student.age}
  </li>
);
```

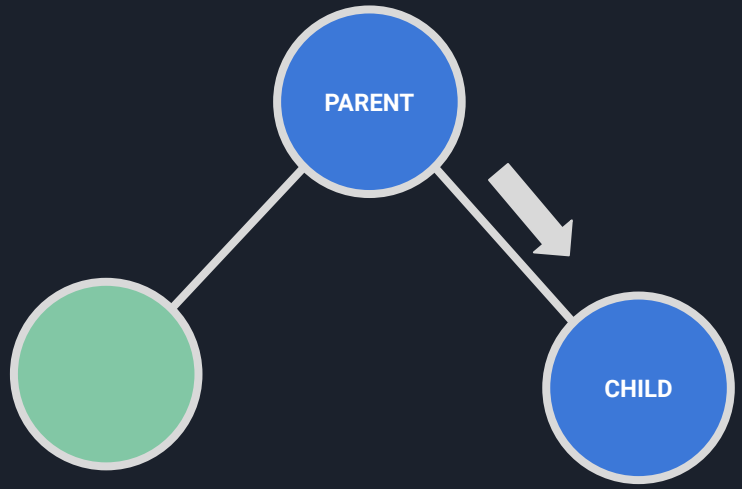
```
const students: Student[] = [
  {id: 1, name: 'Alice', age: 22},
  {id: 2, name: 'Bob', age: 23},
  {id: 3, name: 'Charlie', age: 24},
  {id: 4, name: 'David', age: 25},
  {id: 5, name: 'Eve', age: 26},
];
```

Component

```
function HelloWorld() {

  return (   <div>      <h1>Hello world</h1>   </div>   )

}

....

<HelloWorld />

{HelloWorld()}
```

Props

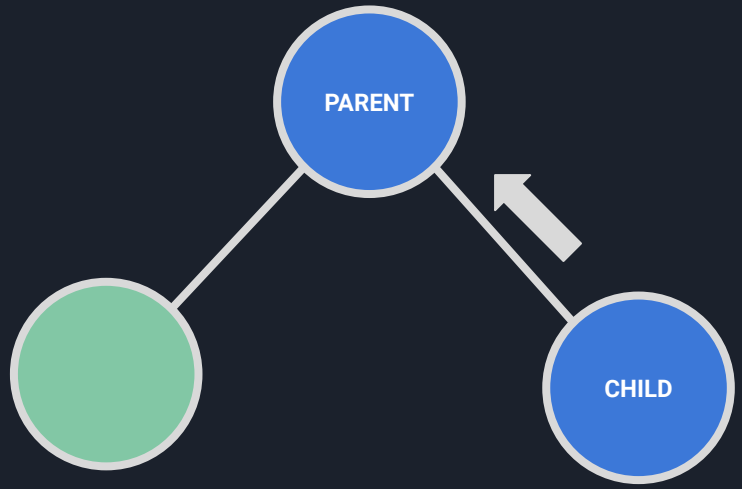
- Way to pass data from a parent component to a child component in React
- Read-only, should not be modified by the child component
- Reusable components, more flexible and configurable


PARENT
CHILD

```
import React from 'react';

function Profile(props: Student) {
  return (
    <div>
      <h1>Hello, {props.name}!</h1>
      <p>You are {props.age} years old.</p>
    </div>
  );
}

export default Profile;
```

Use component with props

```
import React from 'react';

Function Home() {
  return (
    <div>
      <Profile name="John" age={20} />
      <Profile name="Jenny" age={17} />
    </div>
  );
}

export default Home;
```


PARENT
CHILD

Props as callback

```
import React from 'react';


function MyComponent() {
  const handleClick = () => {
    console.log('Button was clicked');
  };
  return <Button onClick={handleClick} />;
}
```

```
interface Props {
  onClick: () => void;
}
const Button = (props: Props) => {
  return <button onClick={props.onClick}>Click me</button>;
}
```

You select {}

image

{name}

Managing Data

React Hook

- useState

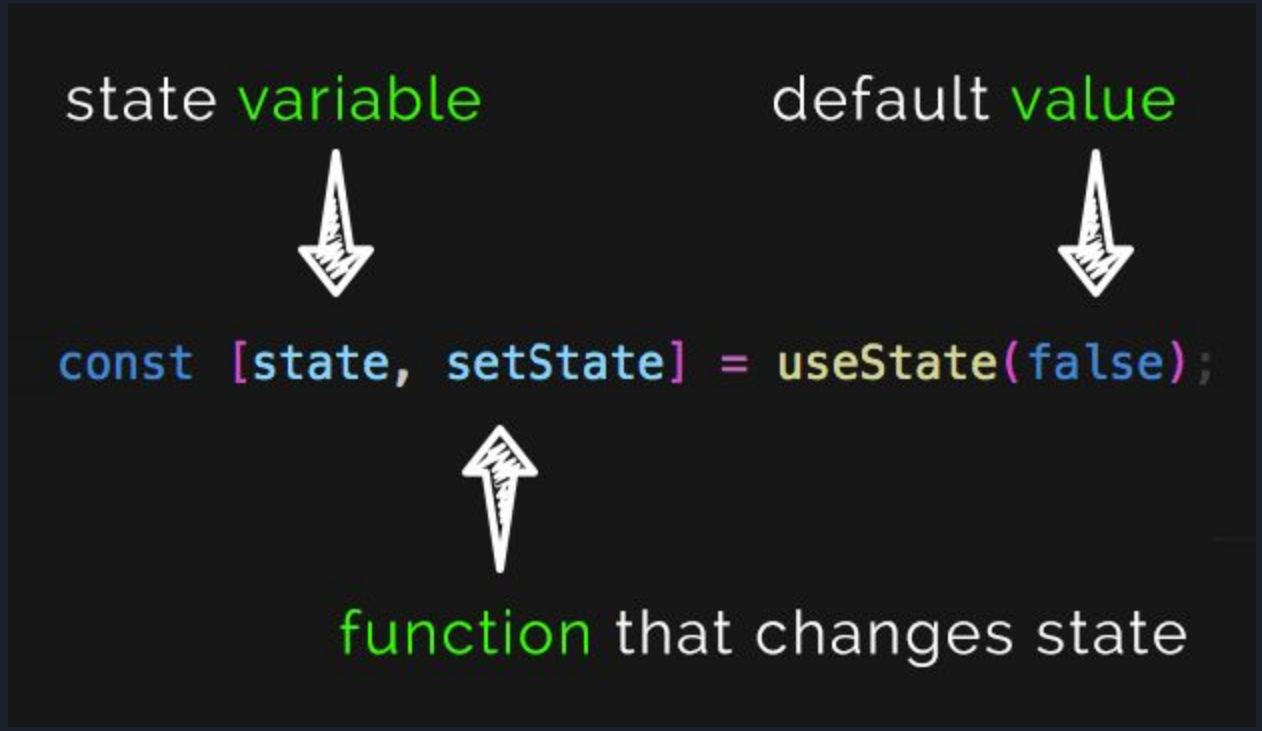
- useEffect
- useContext
- useReducer
- useCallback
- useMemo
- useRef
- customHook

LocalStorage

Create todo using react

React Hook - useState

is a hook in React that allows you to **add state** to **functional components**.
It returns an array with two elements: the current state value and a function to update it.


state variable
default value
const [state, setState] = useState(false);
function that changes state

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

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

useEffect

The useEffect hook takes two arguments: a function to run when the effect is performed, and an array of values that, when changed, will cause the effect to be re-run. In this case, the **Array is empty**, which means that the effect will only be **run once**, when the **component is initially** rendered.

Function to run when the **effect performed**

```
useEffect(sideEffectFunction, [stateToTrack]);
```

An array of **values** that **when changed**

```
useEffect(() => {
    console.log('all the time');
});
```

first render AND update

```
useEffect(() => {
    console.log('only once');
}, []);
```

first render ONLY

```
useEffect(() => {
    console.log(`on ${variable} update`);
}, [variable]);
```

update ONLY

```jsx
import { useEffect } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  // Similar to componentDidMount and componentDidUpdate:
  useEffect(() => {
    // Update the document title using the browser API
    document.title = `You clicked ${count} times`;
  });

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Local Storage

Local storage is a web storage feature that allows web applications to store data locally within the user's browser.

Local storage uses key-value pairs to store data.

The data is stored in the browser, and is accessible only by the web application that created it.

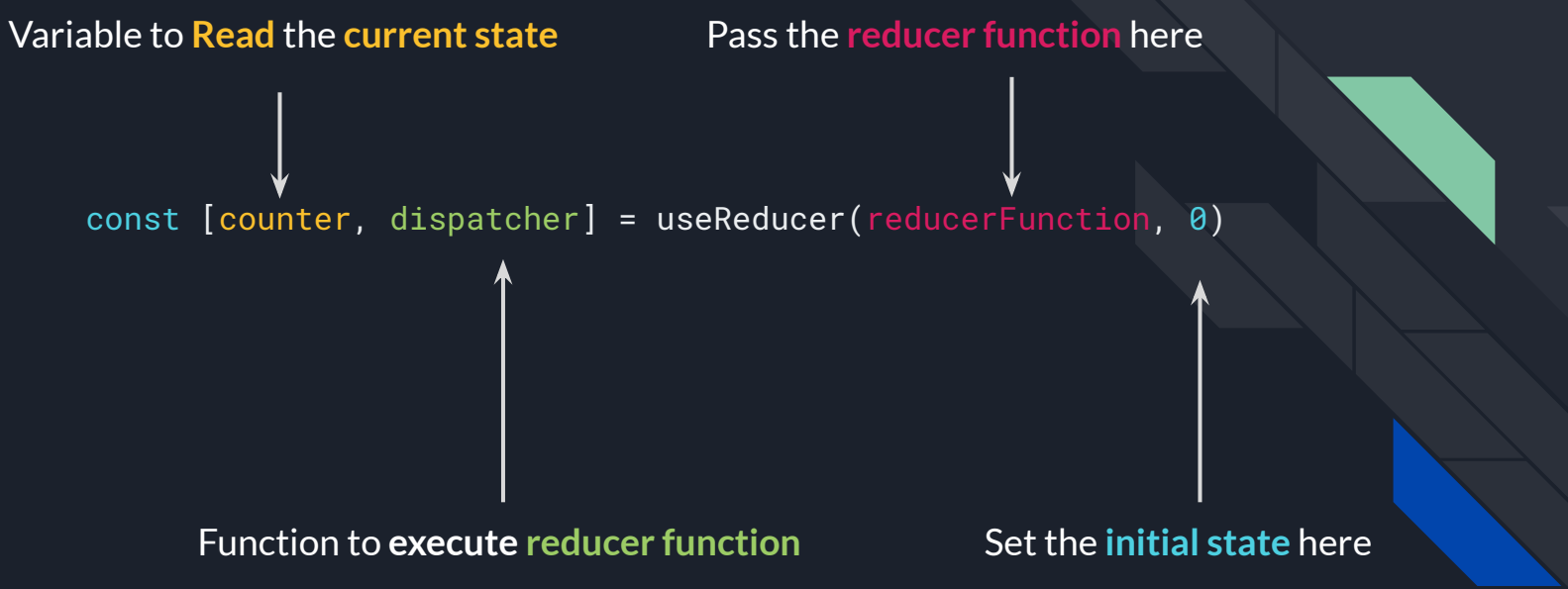
```
// Storing data in local storage
localStorage.setItem("username", "johndoe");

// Retrieving data from local storage
let username = localStorage.getItem("username");
console.log(username); // "johndoe"

// Removing data from local storage
localStorage.removeItem("username");
```

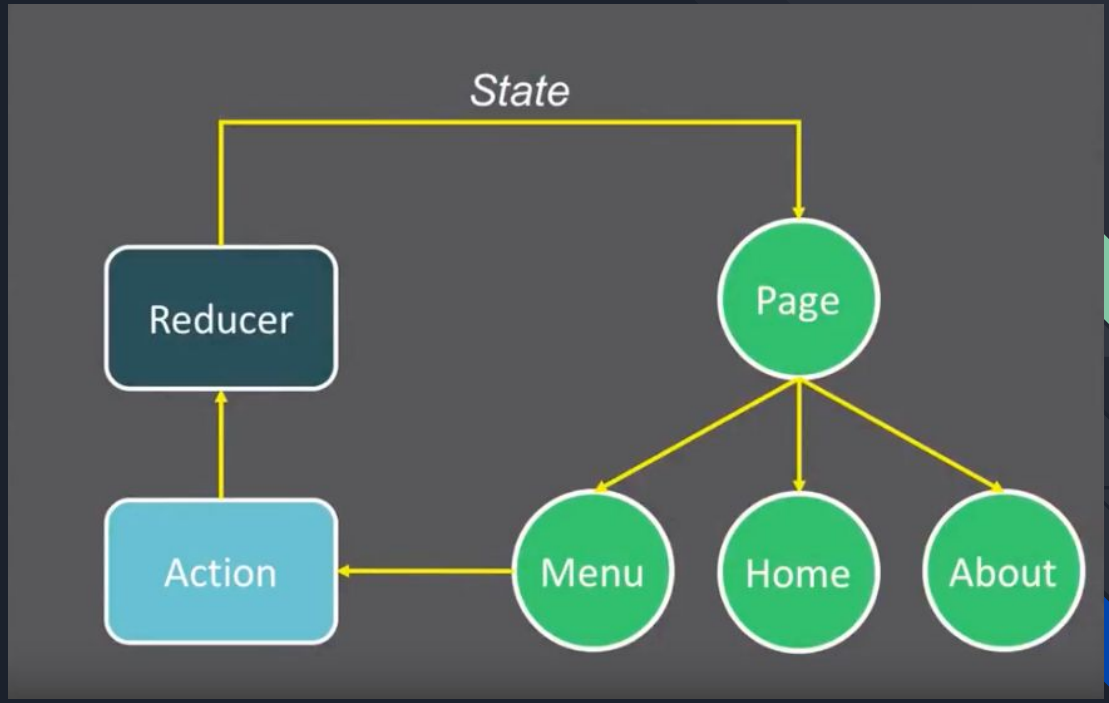
useCallback

useCallback is a hook in React that allows you to optimize the performance of a component by only **re-creating a callback function when specific props or state have changed.**

```
const TodoItem = (props: TodoItemProps) => {
  const printTodo = () => {
    console.log('Todo: ' + props.todo.text)
  }
  // const printTodo = useCallback(() => {
  //   console.log('Todo: ' + props.todo.text)
  // }, [props.todo.text])

  return <li onClick={printTodo}>{props.todo.text}</li>
}
```

useReducer


Variable to Read the current state
Pass the reducer function here
const [counter, dispatcher] = useReducer(reducerFunction, 0)
Function to execute reducer function
Set the initial state here


State
Reducer
Page
Action
Menu
Home
About

```jsx
import { useReducer } from 'react';

function reducer(state, action) {
  switch (action.type) {
    case 'increment':
      return { count: state.count + 1 };
    case 'decrement':
      return { count: state.count - 1 };
    default:
      throw new Error();
  }
}

function Counter() {
  const [state, dispatch] = useReducer(reducer, { count: 0 });
  return (
    <>
      Count: {state.count}
      <button onClick={() => dispatch({ type: 'increment' })}>+</button>
      <button onClick={() => dispatch({ type: 'decrement' })}>-</button>
    </>
  );
}
```

useMemo

```
const filteredTodos = useMemo(() => {
  switch (filter) {
    case 'all':
      return todos;
    case 'active':
      return todos.filter((todo) => !todo.completed);
    case 'completed':
      return todos.filter((todo) => todo.completed);
    default:
      return todos;
  }
}, [todos, filter]);
```

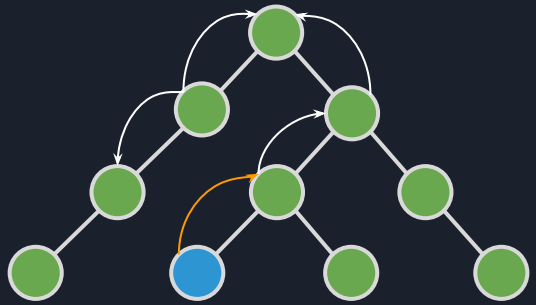
useRef

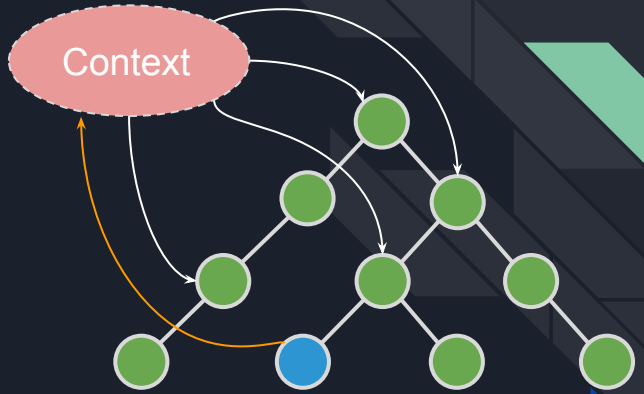
```jsx
import { useRef } from 'react';

function TextInputWithFocusButton() {
  const inputEl = useRef(null);
  const onButtonClick = () => {
    // `current` points to the mounted text input element
    inputEl.current.focus();
  };
  return (
    <>
      <input ref={inputEl} type="text" />
      <button onClick={onButtonClick}>Focus the input</button>
    </>
  );
}
```

Custom Hooks

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

function useCounter(initialCount: number) {
  const [count, setCount] = useState(initialCount);
  const increment = () => setCount(count + 1);
  const decrement = () => setCount(count - 1);
  return { count, increment, decrement };
}

function Counter() {
  const { count, increment, decrement } = useCounter(0);
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
}
```

useContext

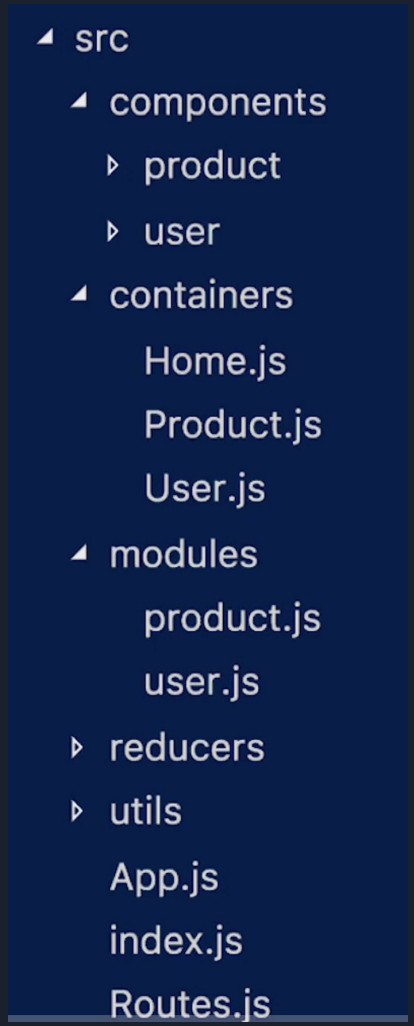



Context

```jsx
import { useContext, useState } from 'react';
// Create a context for the current theme (with "dark" as the default)
const ThemeContext = React.createContext('dark');
function ThemedButton() {
  // Use the context to get the current theme
  const theme = useContext(ThemeContext);
  return <button style={{ background: theme }}>I am styled by the current theme</button>;
}

function App() {
  // Declare a state variable for the current theme
  const [theme, setTheme] = useState('dark');
  return (
    <ThemeContext.Provider value={theme}>
      <ThemedButton />
      <button onClick={() => setTheme(theme === 'dark' ? 'light' : 'dark')}>
        Toggle theme
      </button>
    </ThemeContext.Provider>
  );
}
```

Project Structures


src
actions
index.js
product.js
user.js
components
product
user
constants
containers
Home.js
Product.js
User.js
reducers
utils
App.js
index.js
Routes.js


src
components
product
user
containers
Home.js
Product.js
User.js
modules
product.js
user.js
reducers
utils
App.js
index.js
Routes.js


src
common
styles
colors.js
Button.js
Input.js
features
product
Detail.js
List.js
redux.js
user
utils
App.js
index.js
Routes.js

How to choosing the best structure

1. Readability
2. Distance
3. Reusability
4. Low Coupling
5. Maintainability

Styling in React

Use traditional CSS stylesheets

App.css

```css
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

App.tsx

```tsx
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.tsx</code> and save to reload.
        </p>
        <a className="App-link" href="https://reactjs.org">
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

Use inline styles

```
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div style={{ textAlign: 'center' }}>
      <header style={{ height: '40vmin', pointerEvents: 'none' }}>
        ...
      </header>
    </div>
  );
}

export default App;
```

Use a CSS-in-JS library like (Styled Components)

```
import styled from 'styled-components';

const StyledHeader = styled.header`
    height: 40vmin;
    pointer-events: none;
`;

function App() {
  return (
    <div style={{ textAlign: 'center' }}>
      <StyledHeader>
        ...
      </StyledHeader>
    </div>
  );
}

export default App;
```

UI Components

https://trends.google.com/trends/explore?date=all&q=%2Fg%2F11jnqr1mj0,PrimeReact,Material%20UI%20react,Semantic%20UI%20react,React%20Admin

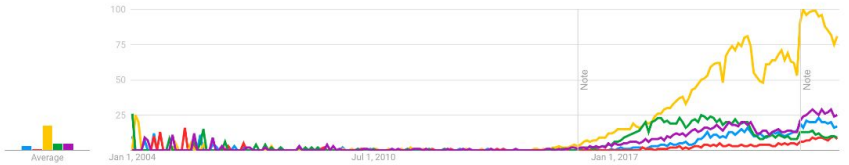
Google Trends | Compare

Ant Design	PrimeReact	Material UI react	Semantic UI react	React Admin
Topic	Search term	Search term	Search term	Search term

Worldwide | 2004 - present | All categories | Web Search

Note: This comparison contains both Search terms and Topics, which are measured differently. LEARN MORE

Interest over time



Compared breakdown by region

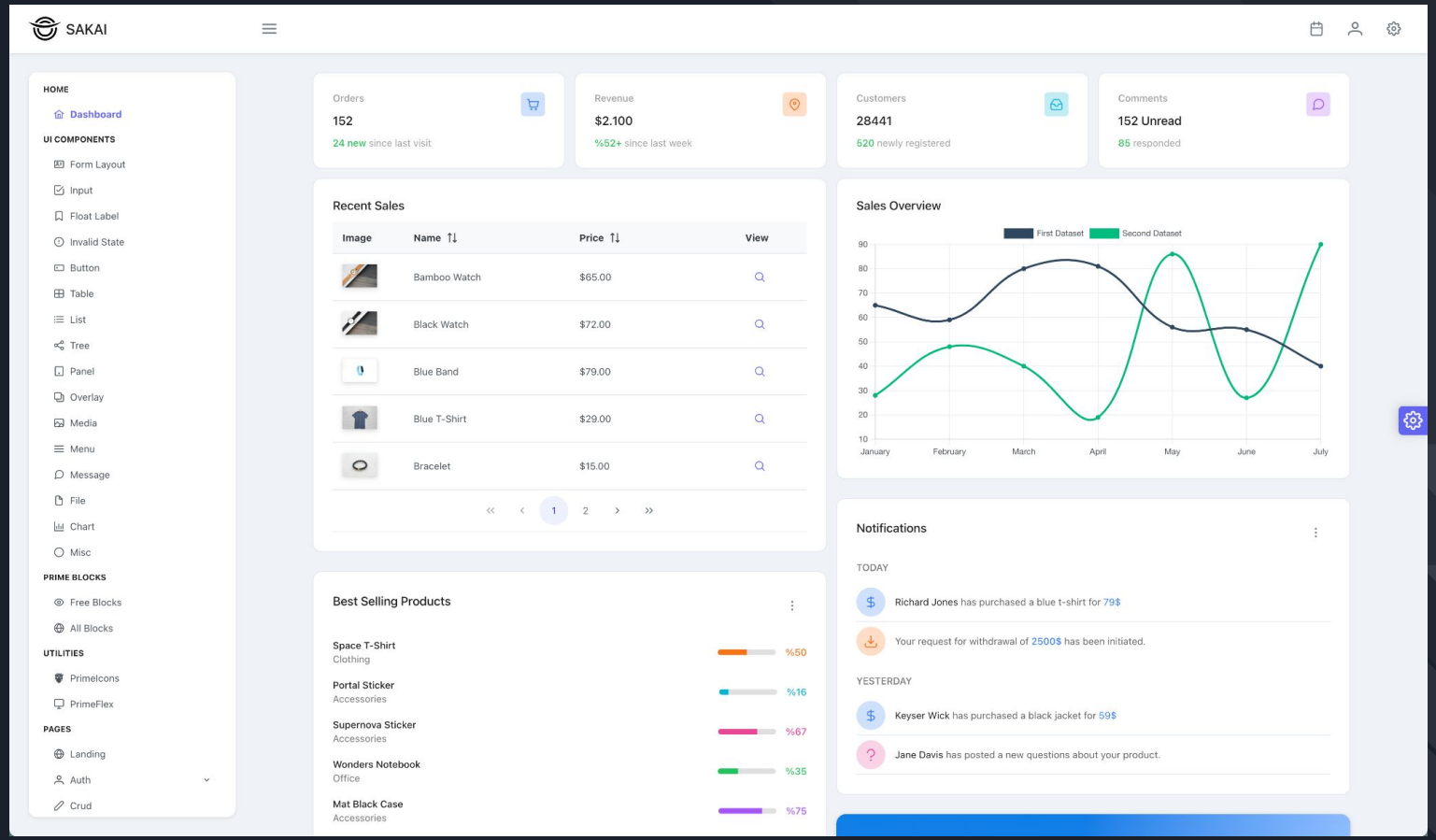
Region

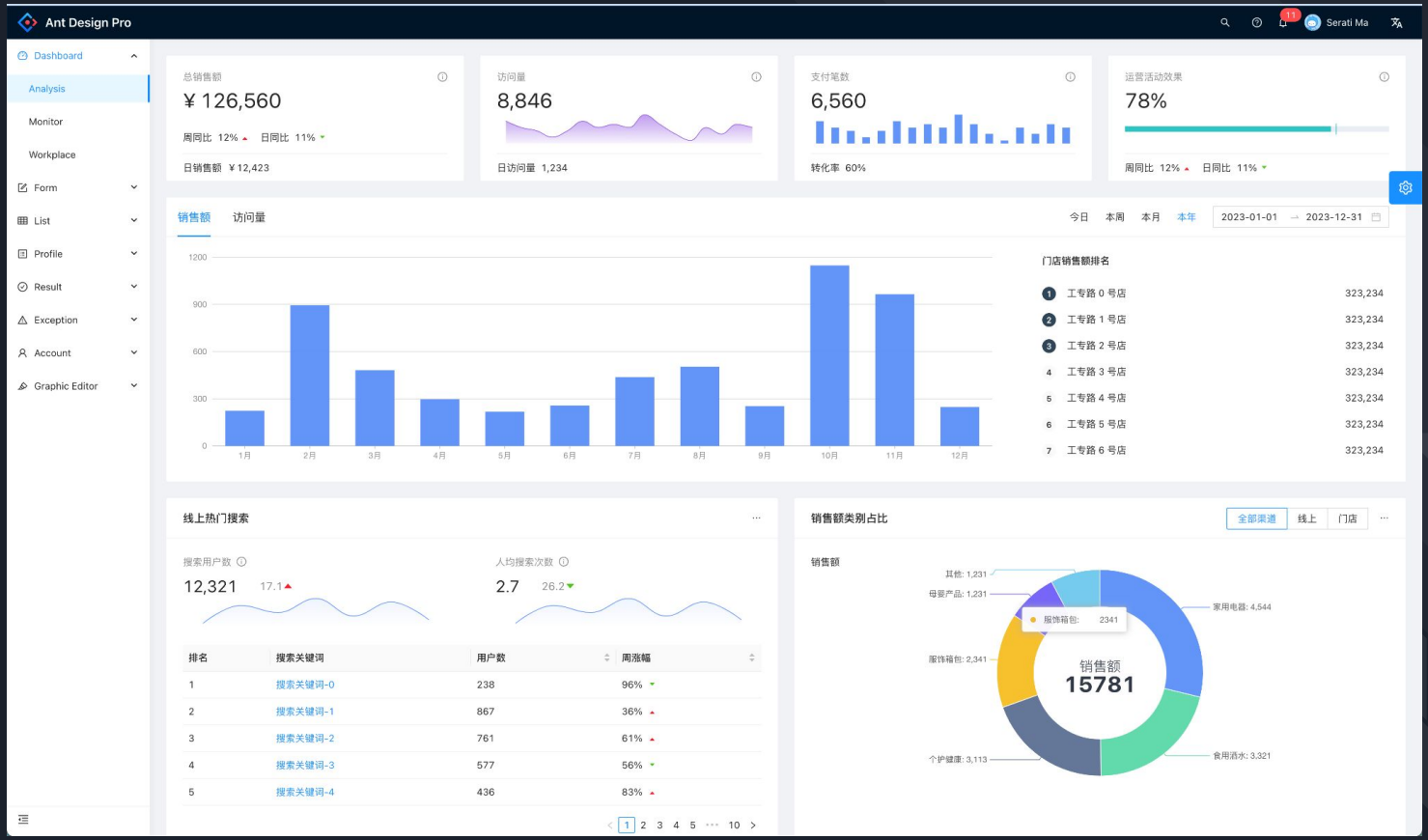
Ant Design PrimeReact Material UI react Semantic UI react React Admin

Color intensity represents percentage of searches LEARN MORE

Sort: Interest for Ant Design

1	China	
2	Uzbekistan	
3	Taiwan	
4	Kazakhstan	
5	Vietnam	

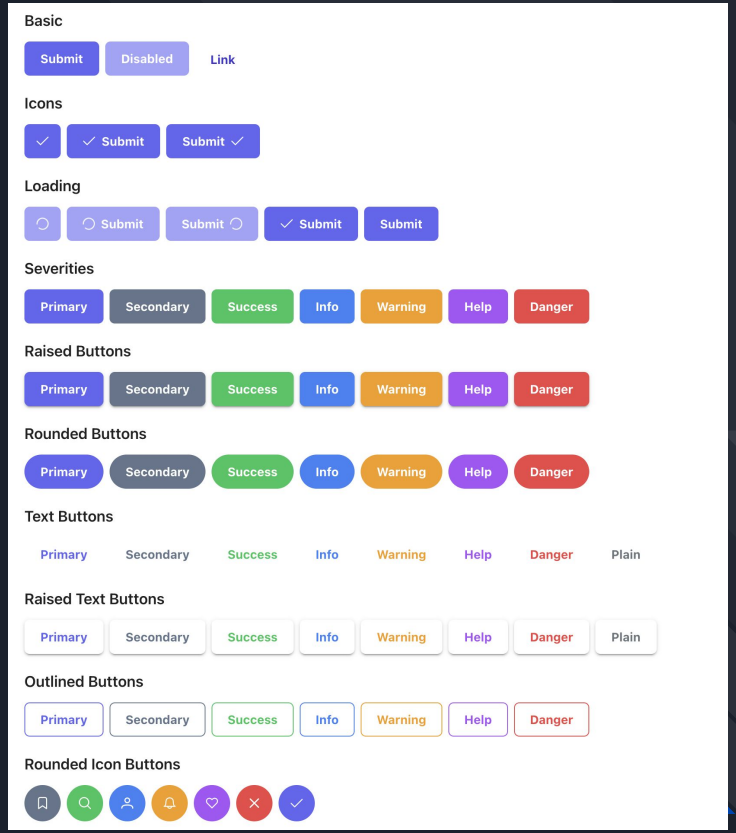

SAKAI
HOME
Dashboard
UI COMPONENTS
Form Layout
Input
Float Label
Invalid State
Button
Table
List
Tree
Panel
Overlay
Media
Menu
Message
File
Chart
Misc
PRIME BLOCKS
Free Blocks
All Blocks
UTILITIES
PrimeIcons
PrimeFlex
PAGES
Landing
Auth
Crud
Orders
152
24 new since last visit
Revenue
$2.100
%52+ since last week
Customers
28441
520 newly registered
Comments
152 Unread
85 responded
Recent Sales
Image
Name
Price
View
Bamboo Watch
$65.00
Black Watch
$72.00
Blue Band
$79.00
Blue T-Shirt
$29.00
Bracelet
$15.00
1
2
Sales Overview
First Dataset
Second Dataset
90
80
70
60
50
40
30
20
10
January
February
March
April
May
June
July
Notifications
TODAY
Richard Jones has purchased a blue t-shirt for 79$
Your request for withdrawal of 2500$ has been initiated.
YESTERDAY
Keyser Wick has purchased a black jacket for 59$
Jane Davis has posted a new questions about your product.
Best Selling Products
Space T-Shirt
Clothing
%50
Portal Sticker
Accessories
%16
Supernova Sticker
Accessories
%67
Wonders Notebook
Office
%35
Mat Black Case
Accessories
%75


Ant Design Pro
Serati Ma
Dashboard
Analysis
Monitor
Workplace
Form
List
Profile
Result
Exception
Account
Graphic Editor
总销售额
¥ 126,560
周同比 12%
日同比 11%
日销售额 ¥ 12,423
访问量
8,846
日访问量 1,234
支付笔数
6,560
转化率 60%
运营活动效果
78%
周同比 12%
日同比 11%
销售额
访问量
今日
本周
本月
本年
2023-01-01
2023-12-31
1200
900
600
300
0
1月
2月
3月
4月
5月
6月
7月
8月
9月
10月
11月
12月
门店销售额排名
1 工专路 0 号店 323,234
2 工专路 1 号店 323,234
3 工专路 2 号店 323,234
4 工专路 3 号店 323,234
5 工专路 4 号店 323,234
6 工专路 5 号店 323,234
7 工专路 6 号店 323,234
线上热门搜索
搜索用户数
12,321
17.1
人均搜索次数
2.7
26.2
排名
搜索关键词
用户数
周涨幅
1 搜索关键词-0 238 96%
2 搜索关键词-1 867 36%
3 搜索关键词-2 761 61%
4 搜索关键词-3 577 56%
5 搜索关键词-4 436 83%
1 2 3 4 5 ··· 10
销售额类别占比
全部渠道
线上
门店
销售额
其他: 1,231
母婴产品: 1,231
服饰箱包: 2,341
个护健康: 3,113
家用电器: 4,544
食用酒水: 3,321
服饰箱包: 2341
销售额
15781

```
npm install primereact # core ui primereact
npm install primeicons # icons primereact
npm install primeflex  # flex layouts
```

```
import "primereact/resources/themes/lara-light-indigo/theme.css";
import "primereact/resources/primereact.min.css";
import "primeicons/primeicons.css";
import "primeflex/primeflex.css";
```

Try use button...



React Router (v6)

React Router is a library for React that allows you to handle **client-side routing in your application**. It provides a way to map URLs to specific components, so that when a user navigates to a specific URL, the corresponding component is rendered.

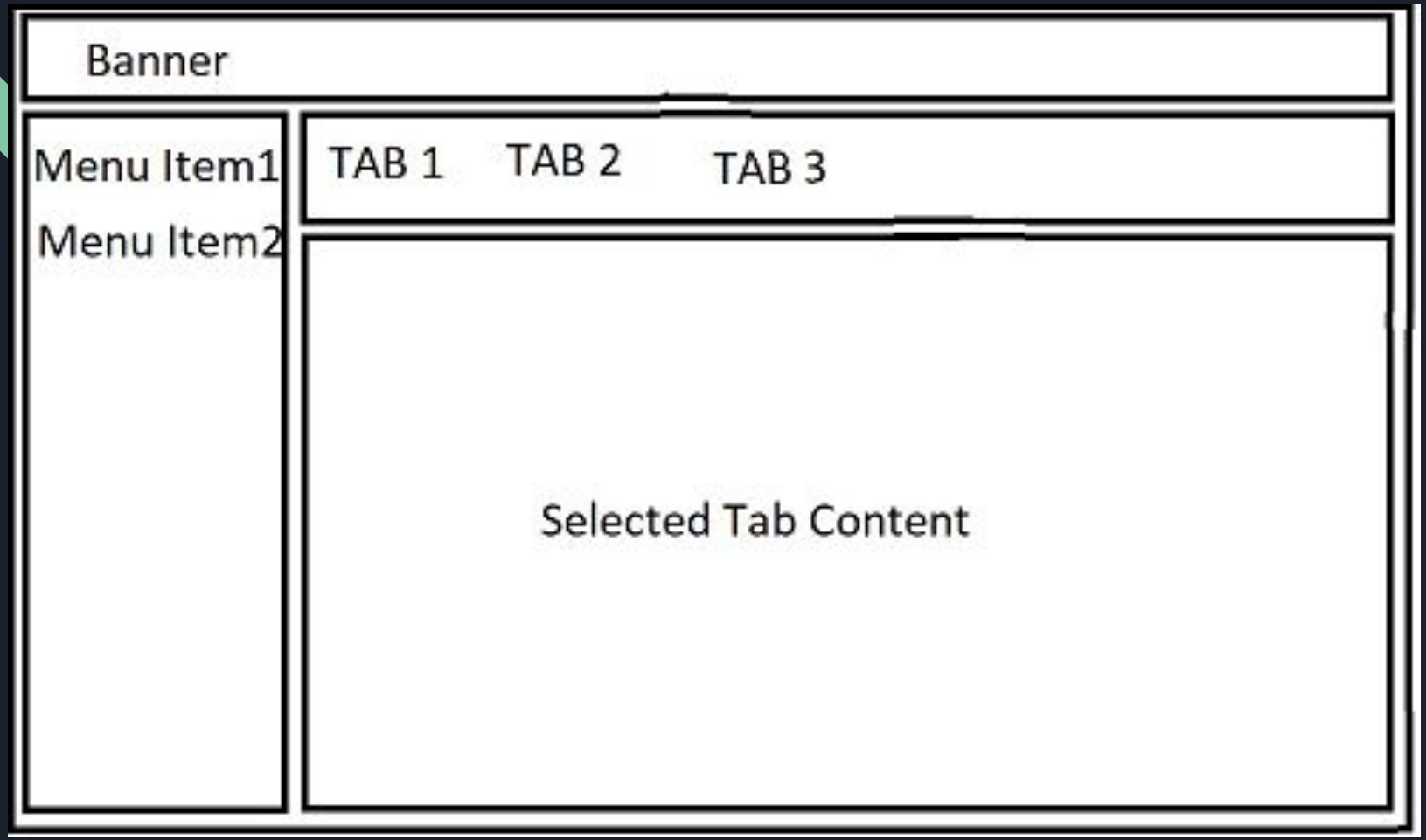
$ npm install react-router-dom

Benefits of using React Router

- **Dynamic routing**: React Router allows you to define dynamic routes, which can change based on the data in your application. This allows you to create more flexible and dynamic user interfaces.

- **Programmatic navigation**: React Router also provides a way to navigate programmatically, which means you can use JavaScript to change the URL and render a different component without requiring the user to click on a link.

- **Server-side rendering**: React Router can also be used for server-side rendering, which allows you to pre-render your application on the server before it is sent to the browser. This can improve the performance and SEO of your application.


Banner
Menu Item1
Menu Item2
TAB 1
TAB 2
TAB 3
Selected Tab Content

```
npm install react-router-dom
```

Call API in React

Using the Fetch API

```
fetch("https://api.example.com/items")
    .then(res => res.json())
    .then((result) => {
        // code ...
    })
    .catch((error) => {
        // handle error
    })
```

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Using the Axios

$ npm install axios

```
axios.get("https://api.example.com/items")
    .then((result) => {
        // code ...
    })
    .catch((error) => {
        // handle error
    })
```

Auth Guard with useContext

Create List Page

Show Data Table from API

Create Service

Build static file

npm run build

Build Production

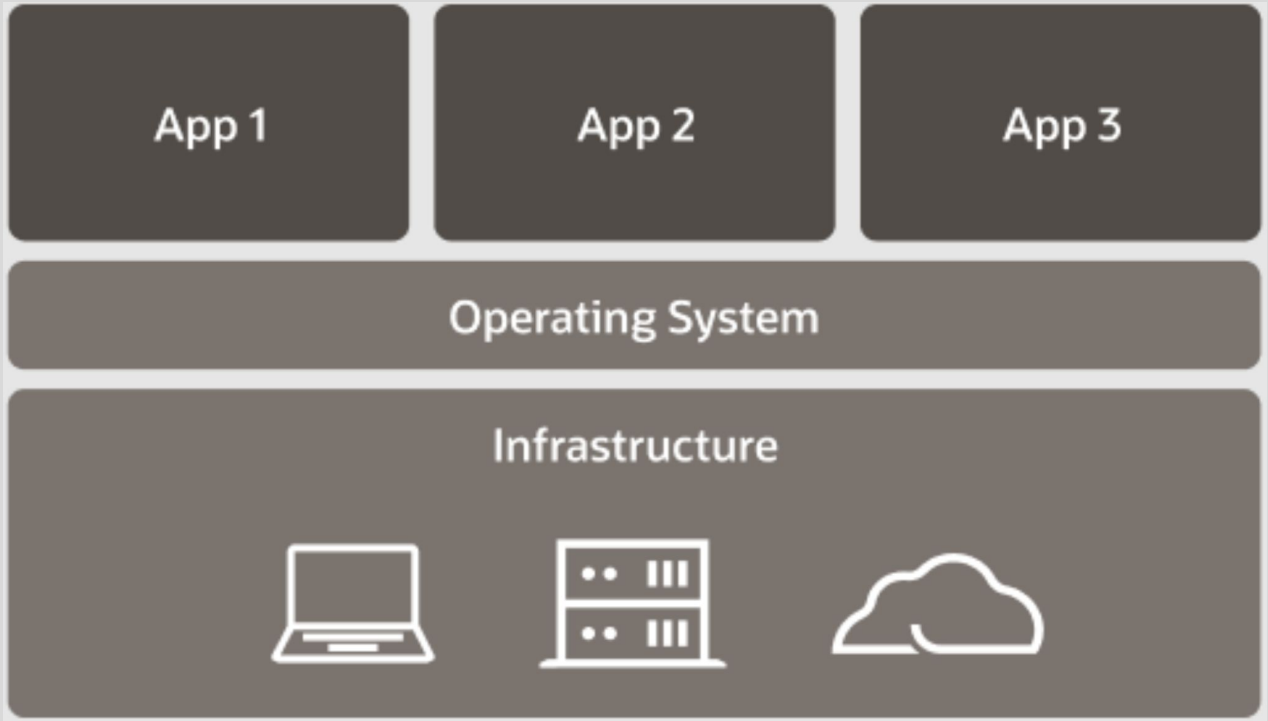
npm run build:prod

package.json

```
{
  "scripts": {
    "build:prod": "env-cmd -f .env.prod npm run build"
  }
}
```

1. Copy

Dockerizing a React app


App 1
App 2
App 3
Operating System
Infrastructure

Problem when not using container

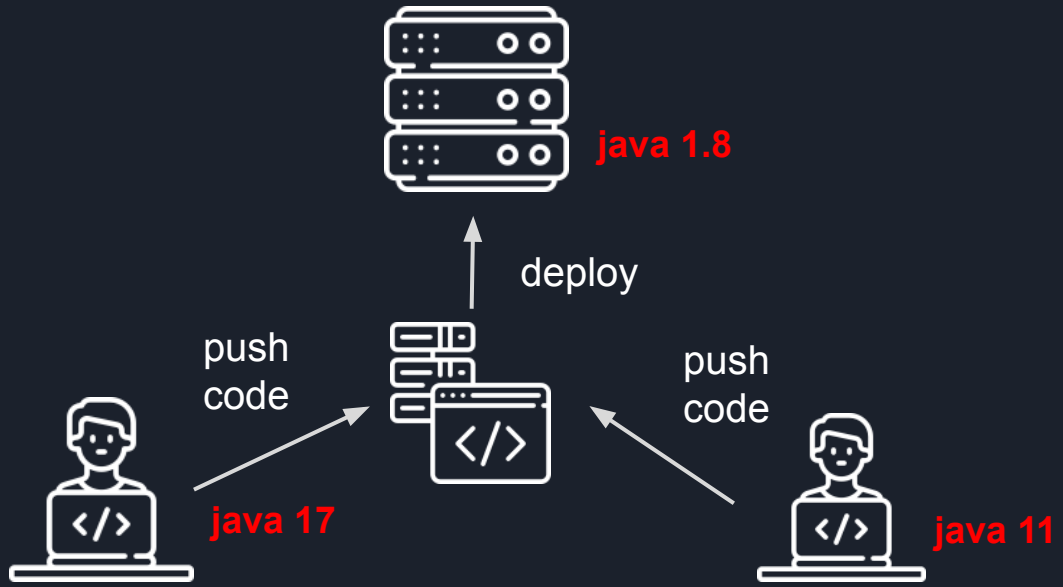
Configuration !?

Virtual Machine

Version !?

Problem when not using container

Version



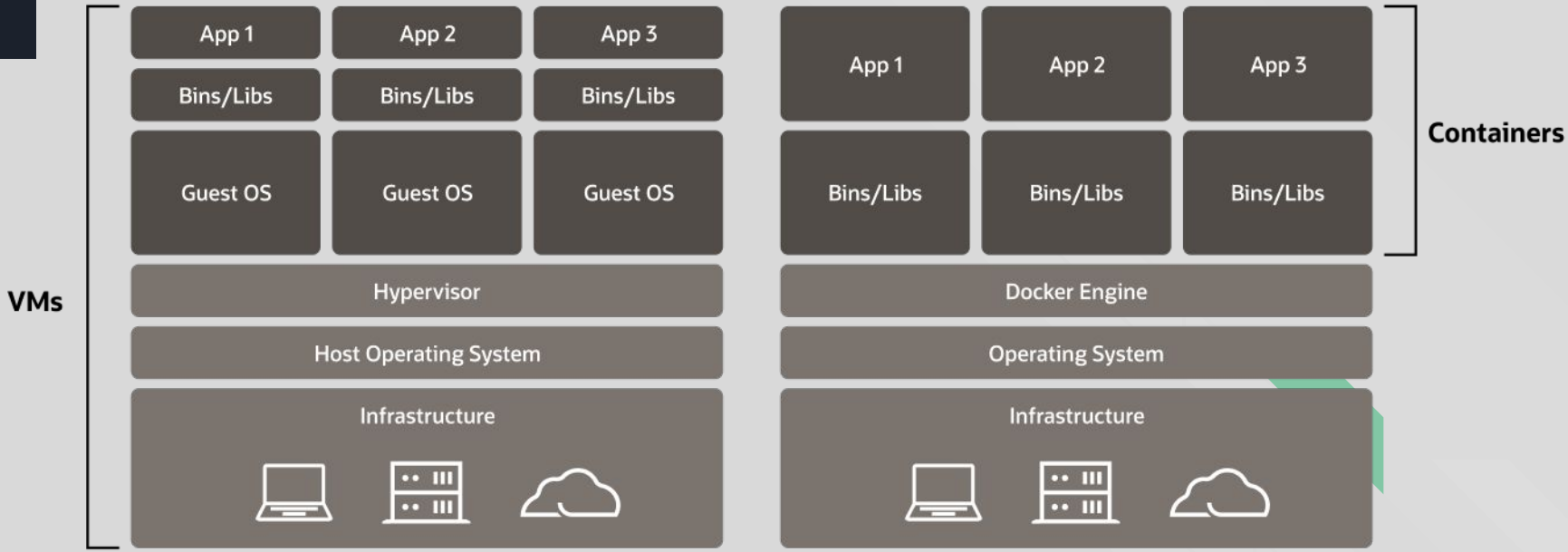
Problem when not using container

Configuration

manual config

- faster!
- less error

build ship run



Virtual Machines

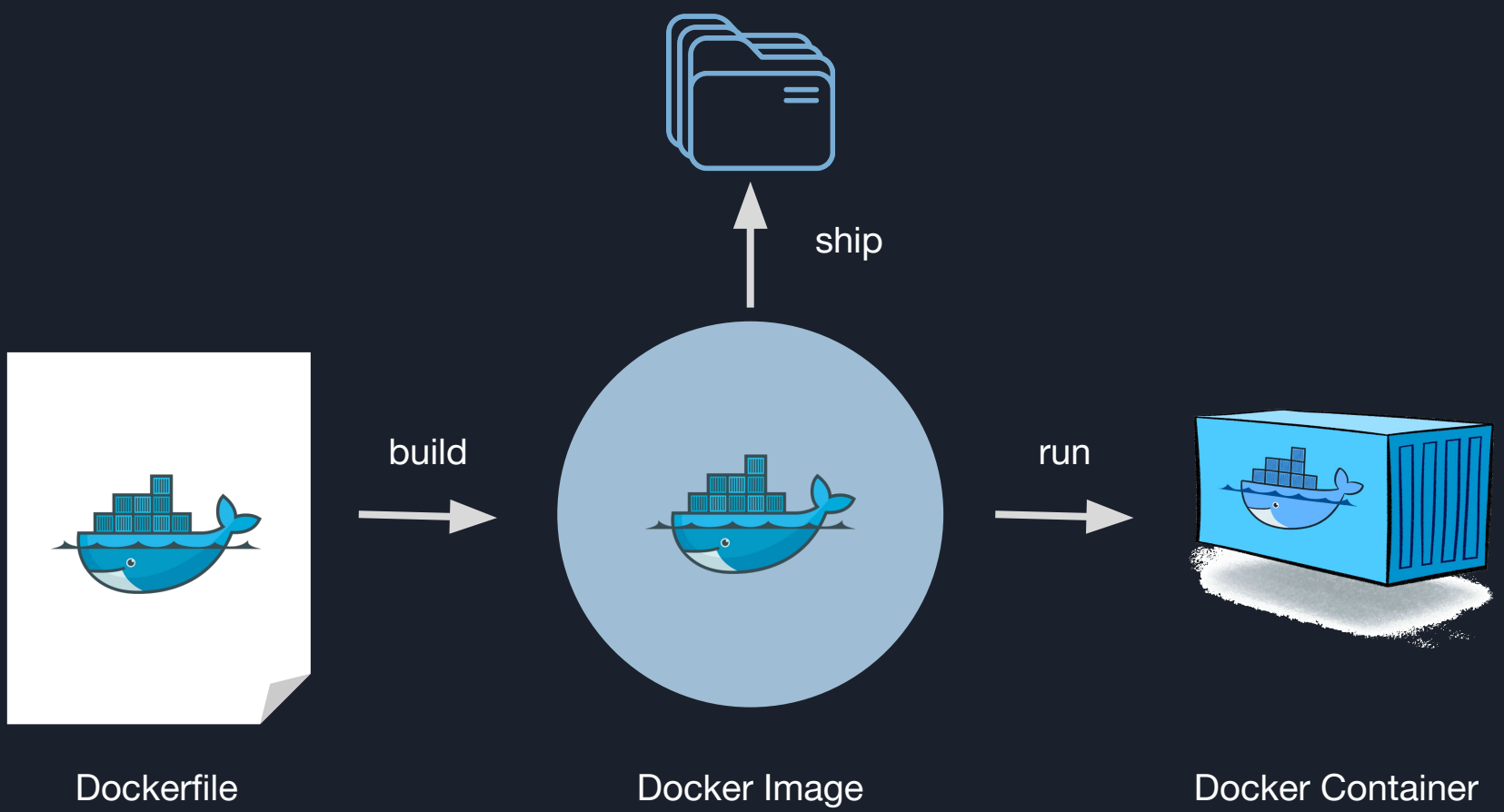
• Each virtual machine (VM) includes the app, the necessary binaries and libraries and an **entire guest operating system**

Containers

• Containers include the app and all of its dependencies, but **share the kernel** with other containers.

• Run as an isolated process in userspace on the host operating system.

• Not tied to any specific infrastructure - containers run on any computer, on any infrastructure and in any cloud.


ship
build
run
Dockerfile
Docker Image
Docker Container

Dockerfile

(Dockerfile is a instructions to build Docker images)

```
FROM node:18

COPY . /react-app

WORKDIR /react-app

RUN npm install

ENV VERSION=1.0

ENV NODE_ENV=production

EXPOSE 3000

CMD ["npm", "run", "start"]
```

Dockerfile

(Dockerfile is a instructions to build Docker images)

FROM node:18

COPY . /react-app

WORKDIR /react-app

RUN npm install

ENV VERSION=1.0

ENV NODE_ENV=production

EXPOSE 3000

CMD ["npm", "run", "start"]

OS + System packages

Source code

Configuration

Library dependencies

Configuration

Deployment