Es6 syntax in react—

Await – The await operator is used to wait for a Promise. It can only be used inside an async function within regular JavaScript code; however, it can be used on its own with JavaScript modules.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/await

Rendering –

Re-Rendering –

**State in React –**

It is an object, used to store information in a component.

Component specific

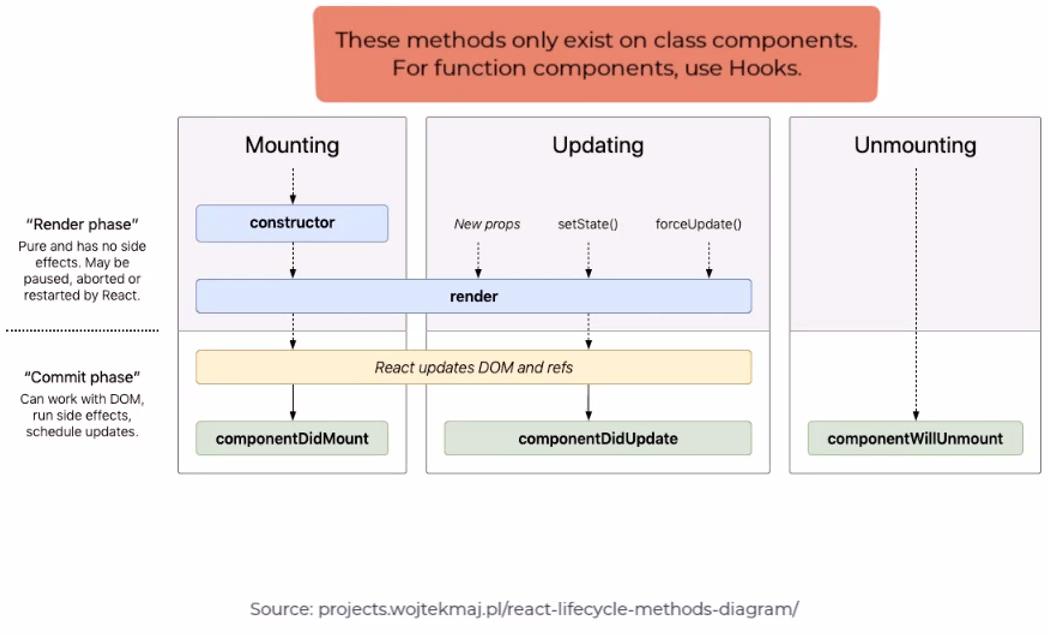
Can be modified

* Class component 🡪 setState()
* Functional component 🡪 useState()

Every time there is a change in the state of the component, the component re-renders

**React life cycle Methods –**

Life cycle methods in class components allow us to run code at particular points in the process



ComponentDidMount :

CmponentDidMount is invoked when a component has been inserted into Dom.

Here are some common use cases for this methods:

* Calling a web service to get some data
* Adding event listers
* Initializing timers

ComponentWillUnmount :

ComponentWillUnmount is invoked just before the component is removed from the Dom.

Here are some common use cases for this methods:

* Removing event listeners
* Canceling active net request
* Removing timers

getDervivedStateFromProps:

getDervivedStateFromProps is invoked every time a component is rendering

it can be used to change state when certain props change

this is a static method in a component class that returns the changed state, or null if there are no changes to the state.

Mounting Phase:

* Constructor()
* getDerivedStateFromProps()
* render()
* ComponentDidMount()

Updating Phase:

* getDerivedStateFromProps()
* shouldComponentUpdate()
* render()
* getSanpshotBeforeUpdate()
* ComponentDidUpdate()

UnMounting Phase:

* ComponentWillUnmount()

Consuming Rest Apis in React –

Representational State Transfer

APIs can contain

* Business logic
* Some validations
* Connectivity to database for curd operations (create, retrieve, update, Delete)

http methods

* + create 🡪 post
  + retrieve 🡪 get
  + update 🡪 put
  + delete 🡪 delete

While Apis respond, http status codes—

200 ok 🡪 Rud (retrieving data, update date, delete data)

201 created 🡪 c (insertion of aanew record data)

404 🡪 page not found

500 🡪 internal not found

401 🡪 unathorized

400 🡪 bad request

Consuming Rest apis in react

* Fetch() 🡪 built in resume - returns a promise object
* Axios() 🡪 third party library, need to install it 🡪 returns a promise object

Cors => cross-origin request

Routing –

In class component –

Import {withRouter} from “react-router-dom”;

Class ClassName extends Component {

onEditPlayerBtn = () => {  
 props.history.push(‘/EditPlayerPage’);

}

Render(){

Return(  
 <button type=”submit” onClick={ onEditPlayerBtn }>button</button>

)

}

}

Export default withRouter(ClassName);

In function Component –

Import {useHistory} from “react-router-dom”;

Function FunctionName(){  
 const history = useHistroy();

onEditPlayerBtn = () => {  
 history.push(‘/EditPlayerPage’);

}

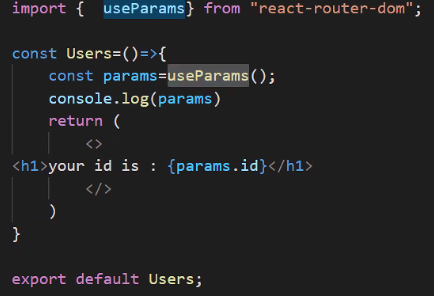
Return(  
 <button type=”submit” onClick={ onEditPlayerBtn }>button</button>

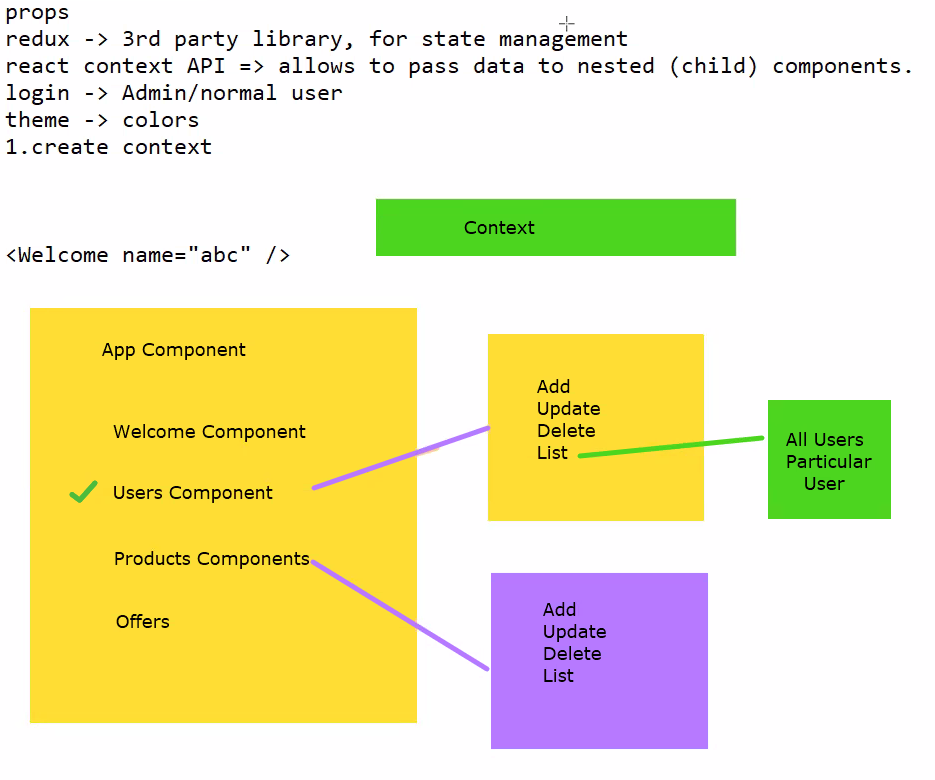
)

}

Export default FunctionName;





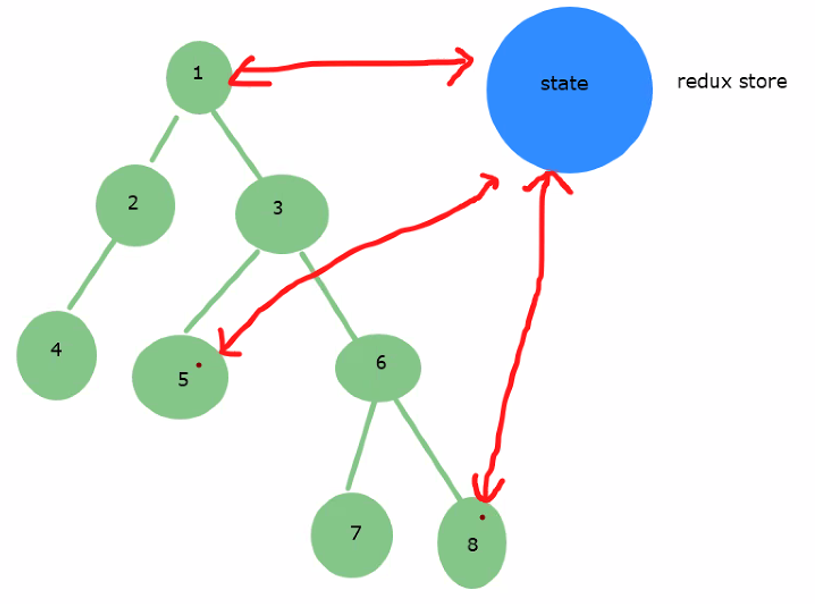


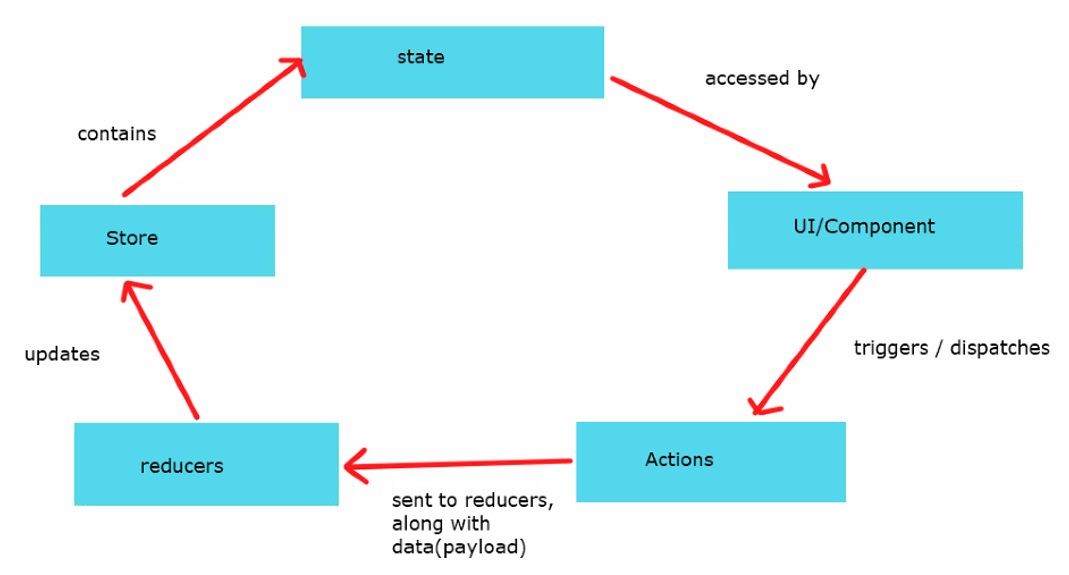
Redux in React –

Redux is an open-source JS library for managing the state.

It maintains a centralized state, in a store. The state in redux cannot be modified directly.

You need to use reducers to update the state. You need to install





Npm install redux react-redux

3-Priniciples of Redux --

Single source of truth (that is redux store)

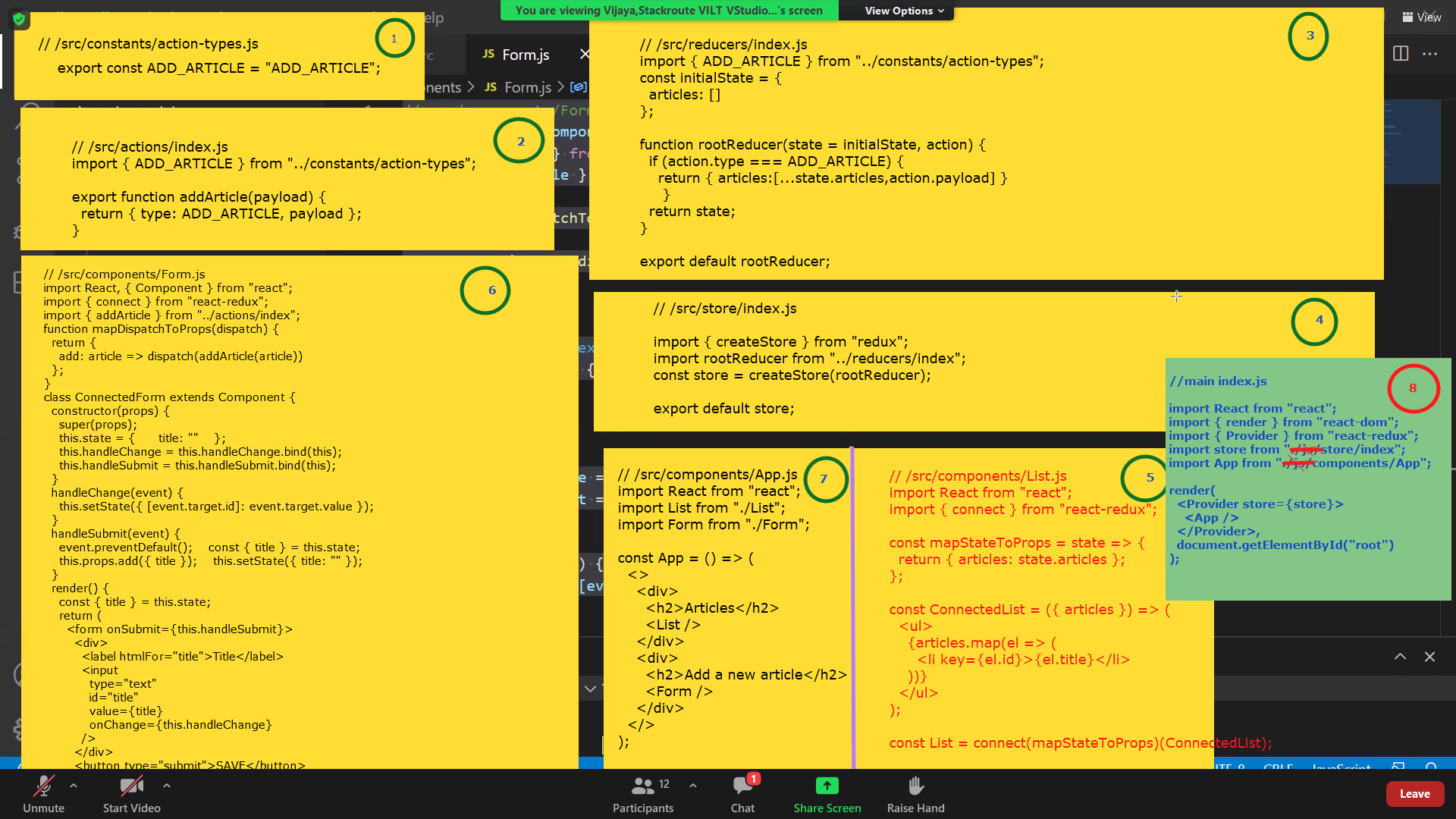
State is read only

Changes are made with pure functions (reducers)

Reducer 🡪 Purpose 🡪 to update the redux store

Reducer functions 🡪 2 parameters, 1st state, 2nd action

Action 🡪 contains type, payload (data)



Testing React Application with Jest and Enzyme –

Jest gets installed automatically when a react project is created using CRA Syntax

Enzyme: open-source testing utility, introduced by Airnb, needs to be installed

Create aa folder called \_\_tests\_\_

* Shallow rendering test => tests only the parent components => shallow()
* Full Dom(tree) rendering test => tests the entire component tree => mount()
* Snapshot test => will snapshot of a component, compare this snapshot with the component

it /test— used to describe the test itself.

describe — creates a block that groups several related tests.

expect — is used every time a value is tested, it is called along with a matcher function to assert a value.

shallow — renders only the component we are testing and not the child component.

mount — renders full DOM including the child component.

Installations related to Enzyme --

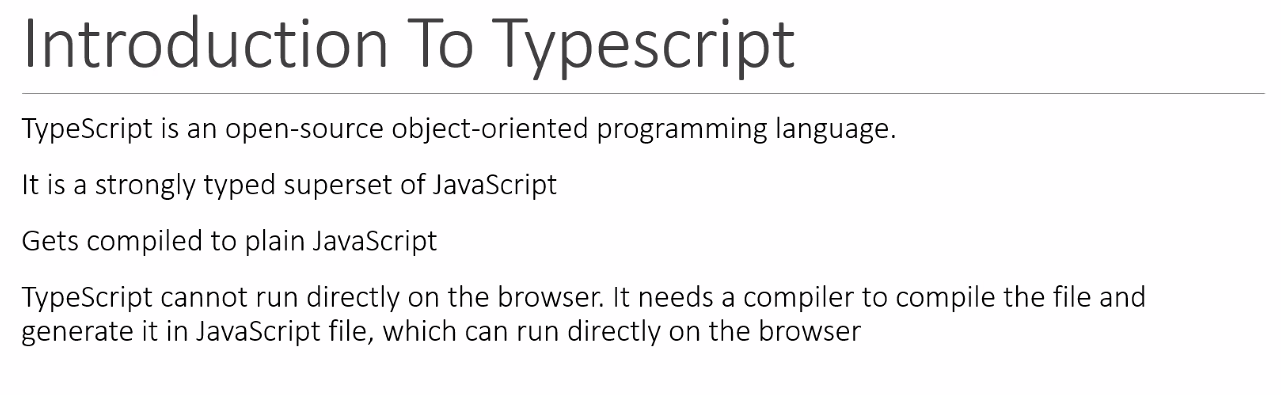
Npm install enzyme

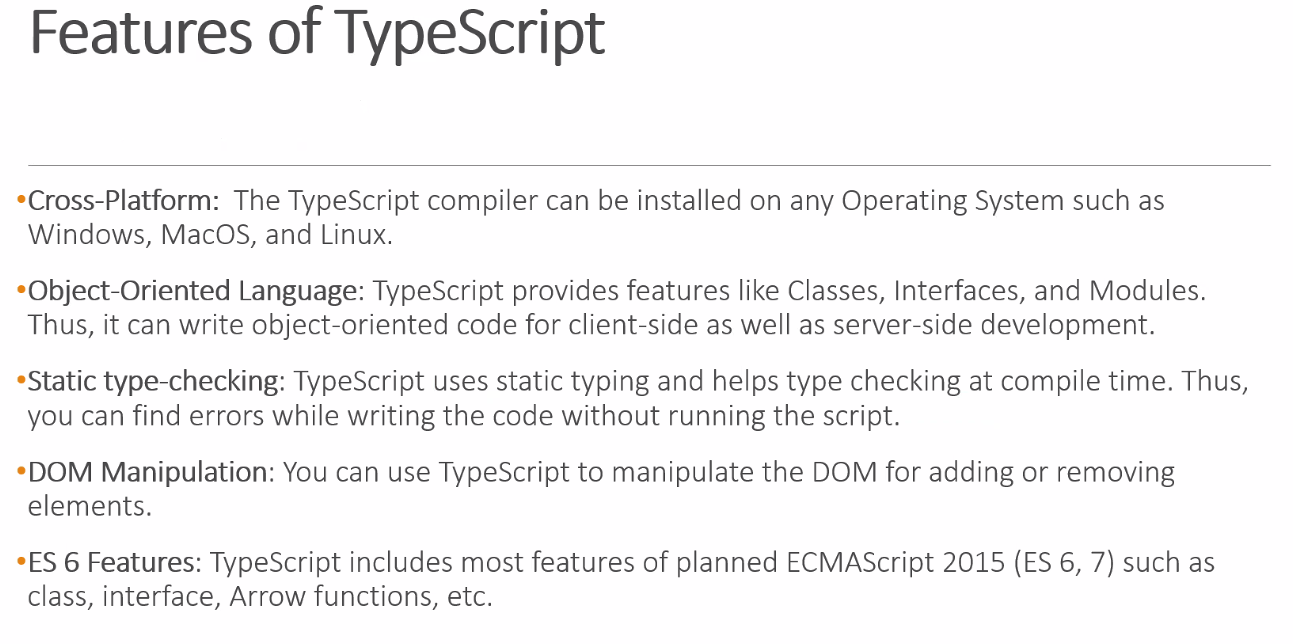
Npm install enzyme-adapter-react-16

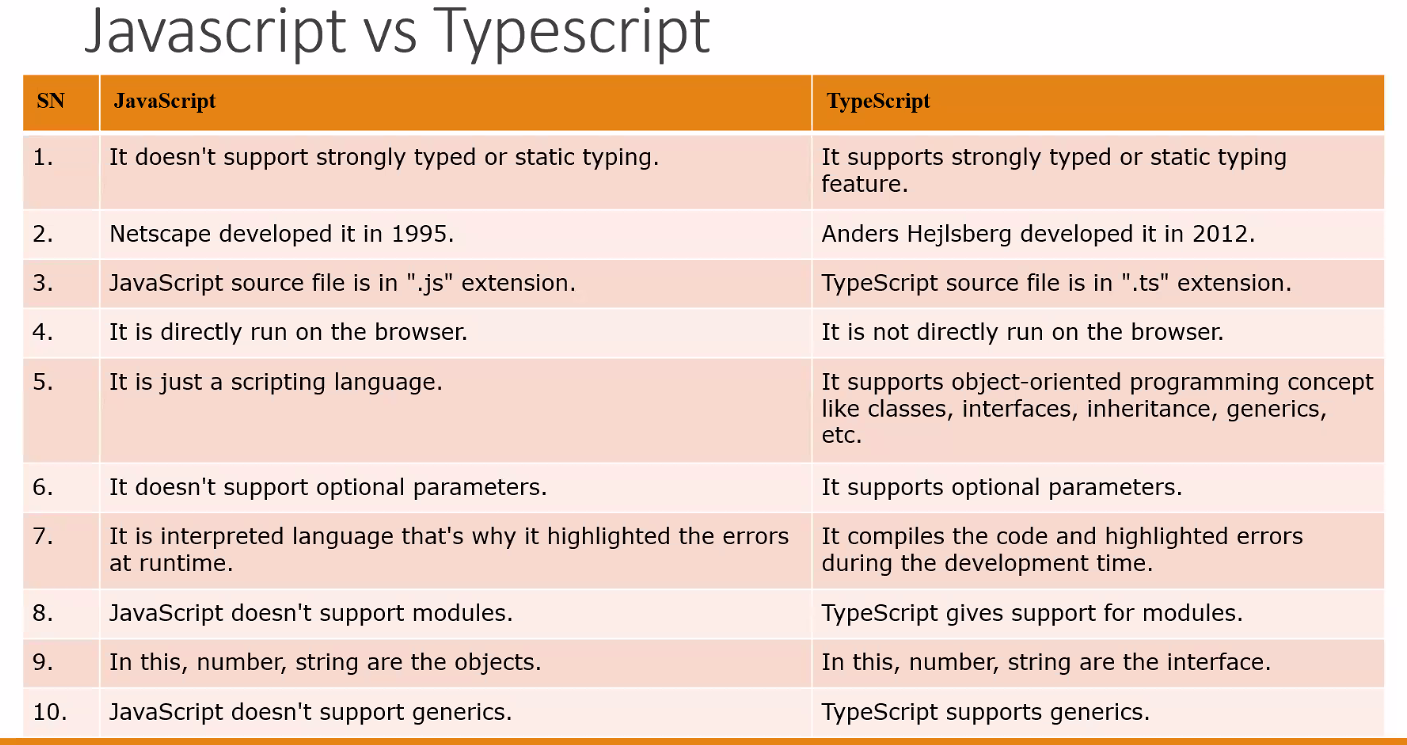
Npm install react-test-renderer

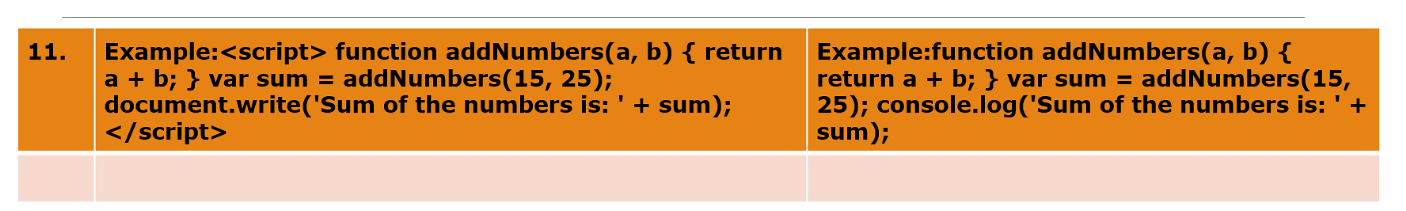
Npm install enzyme-to-json

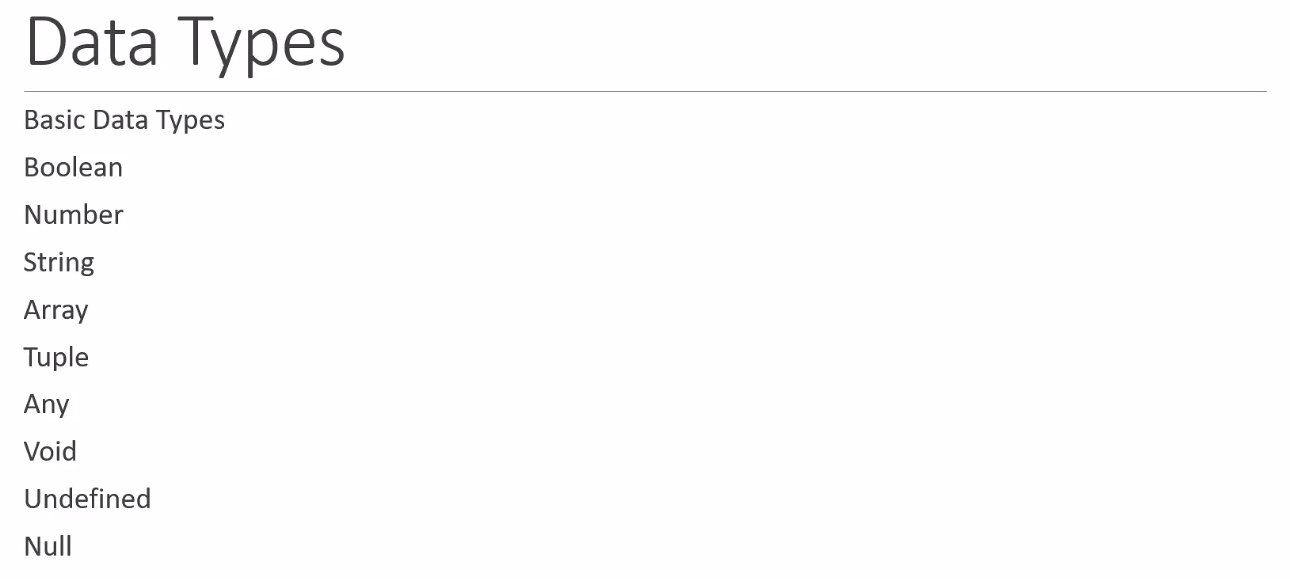
**TypeScript:**











<https://www.kozubek.dev/2019/03/31/react-typescript.html>

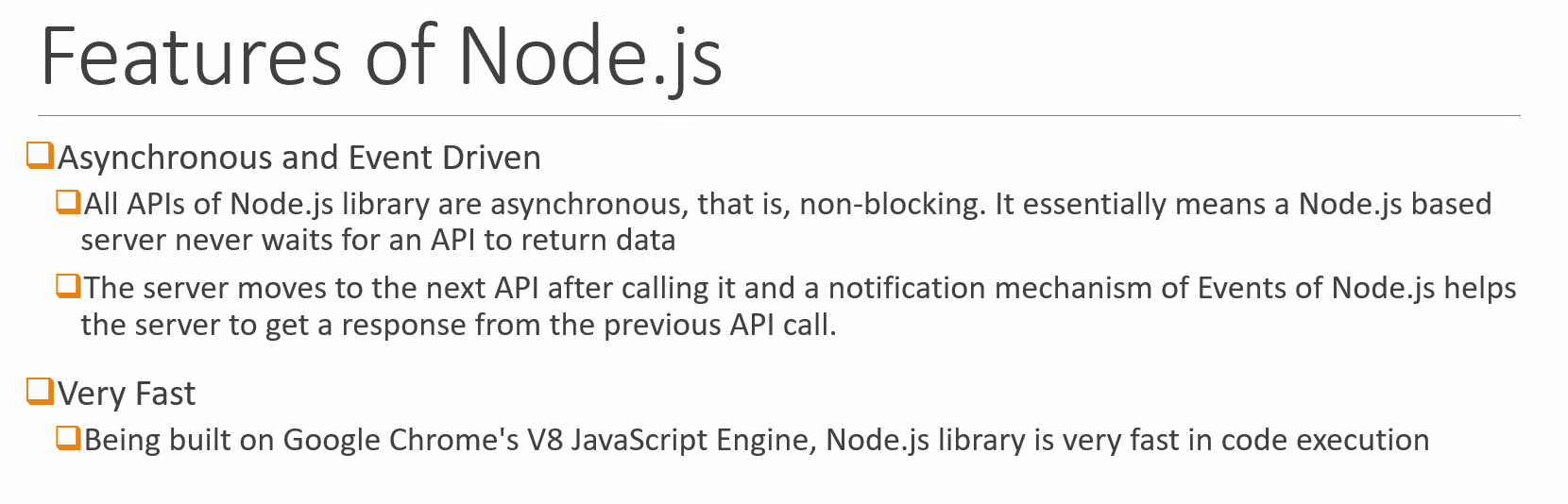
<https://pusher.com/tutorials/react-jest-enzyme/#jest-basics>

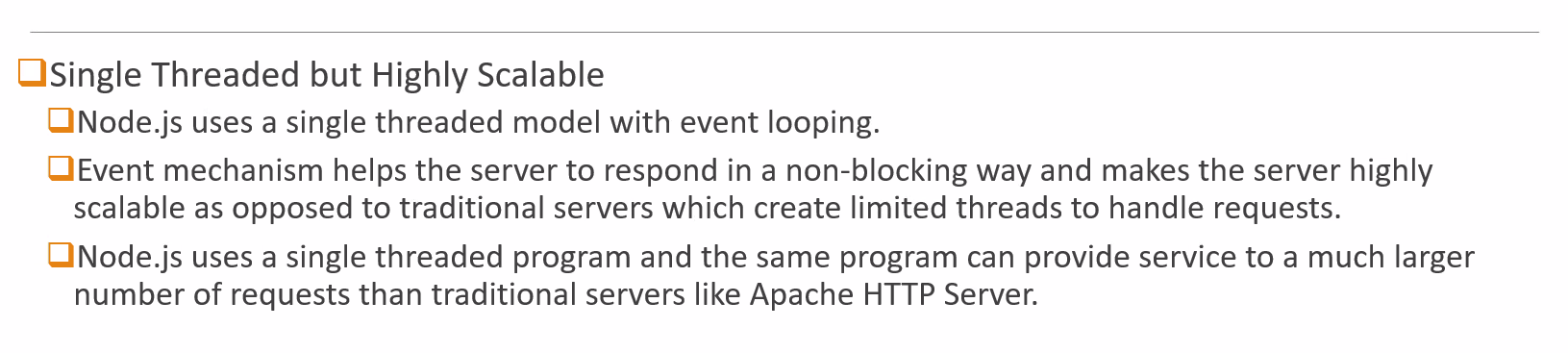
<https://www.typescriptlang.org/download>

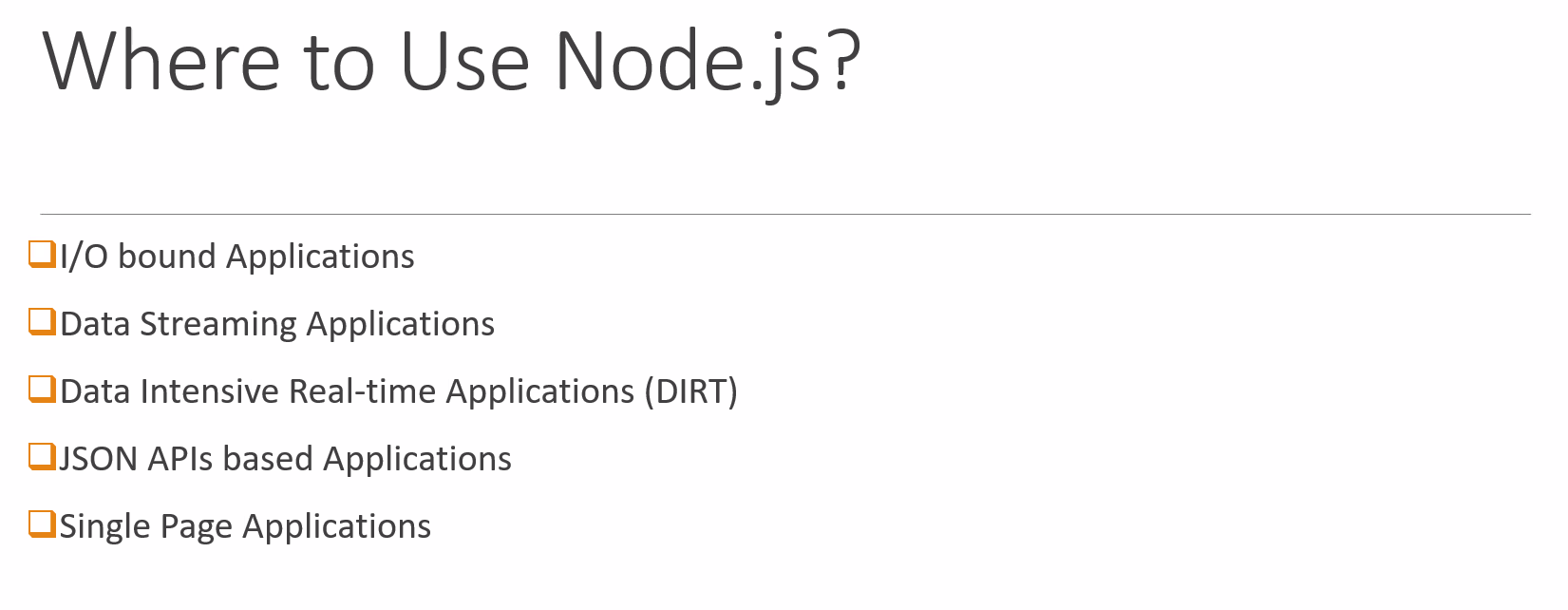
<https://blog.formpl.us/simple-test-in-a-react-application-with-jest-and-enzyme-fb3a5daf7784>

<https://www.robinwieruch.de/react-function-component>

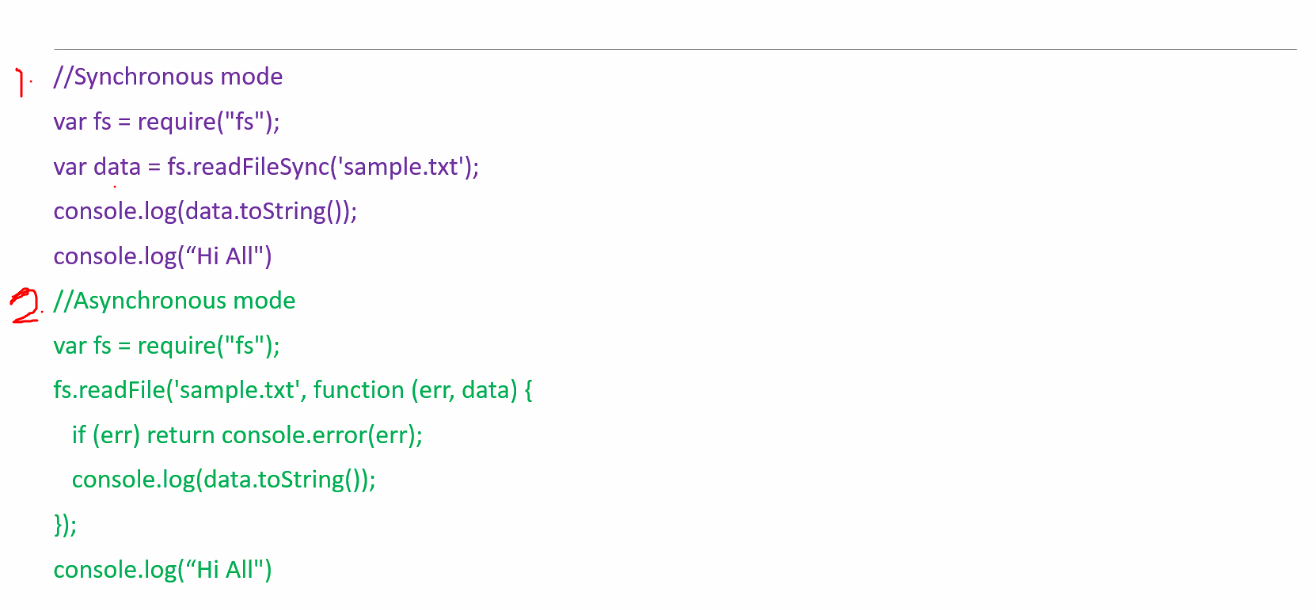
Node –

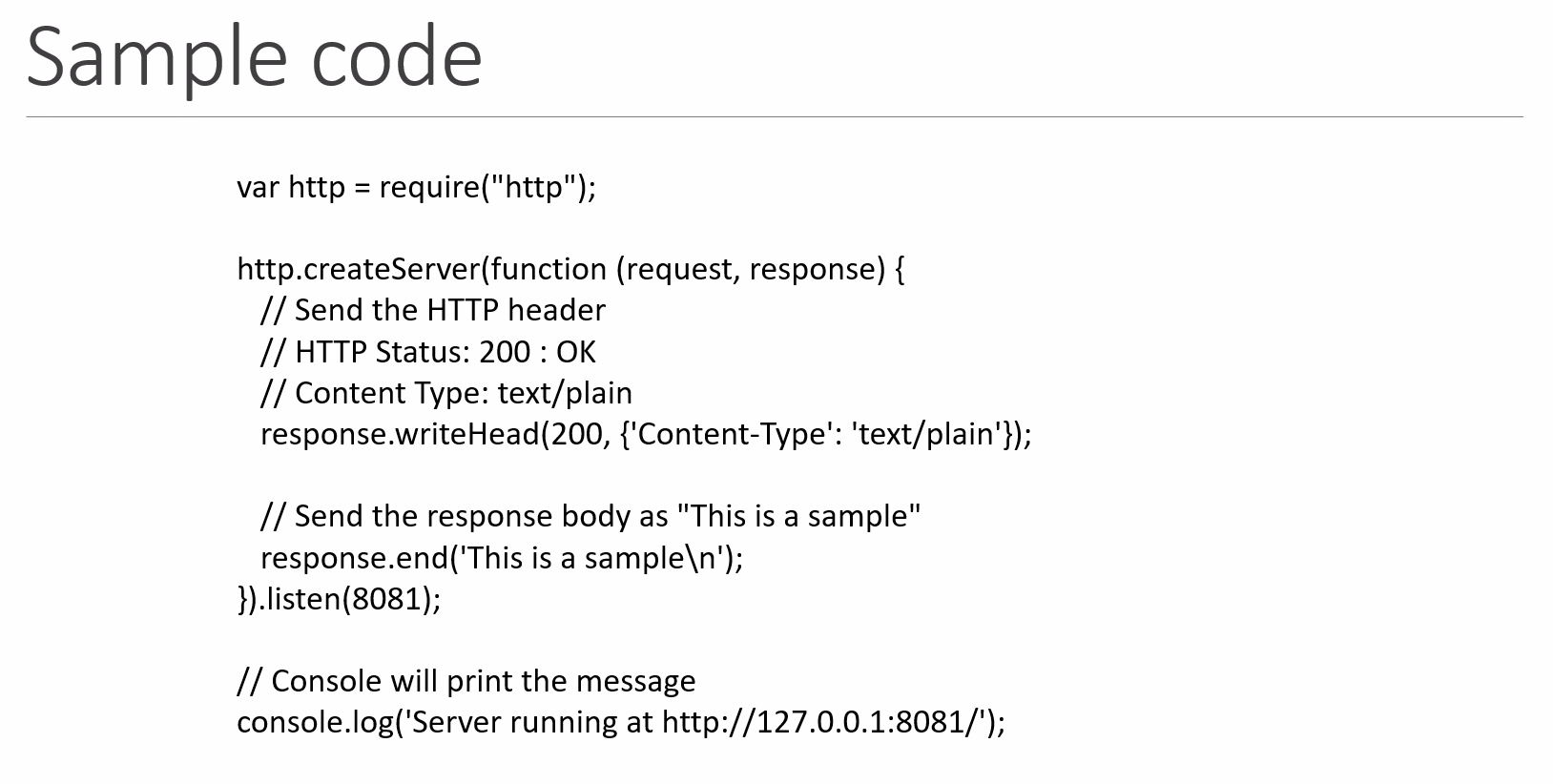


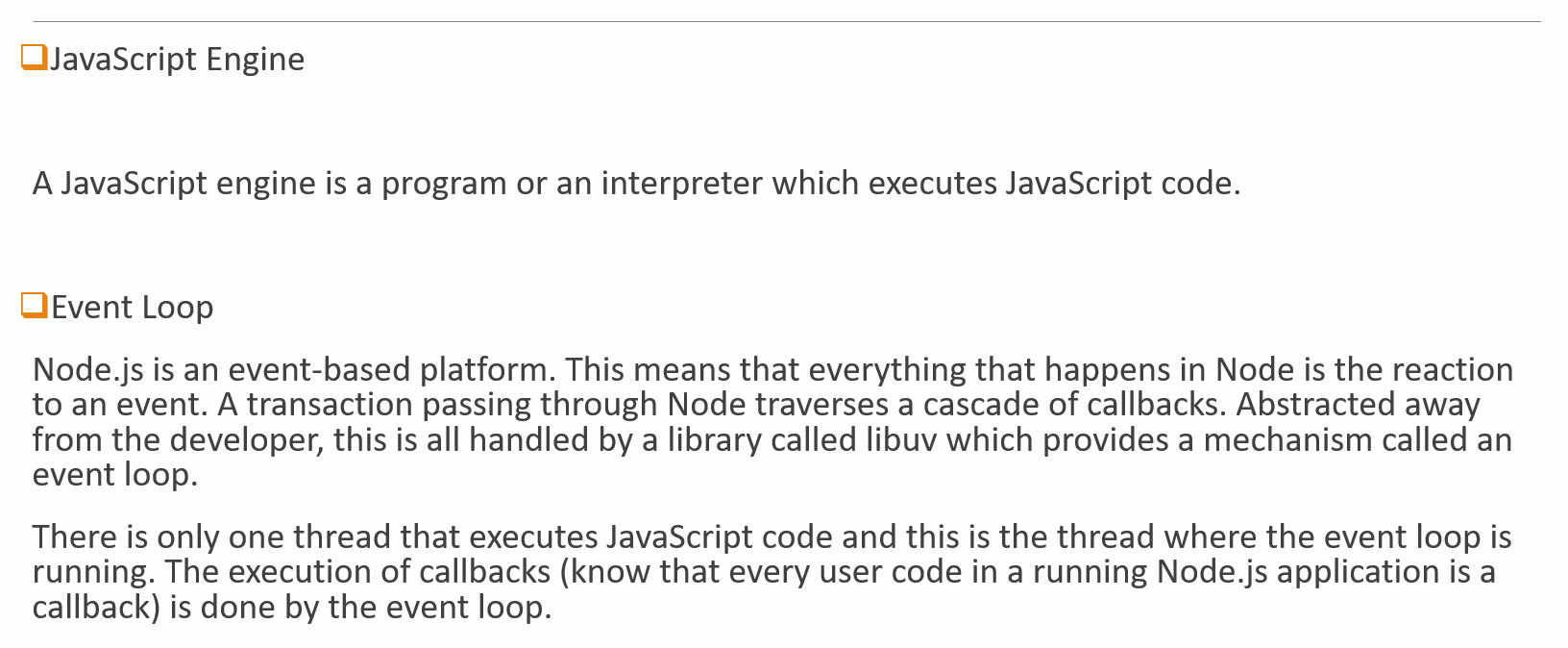


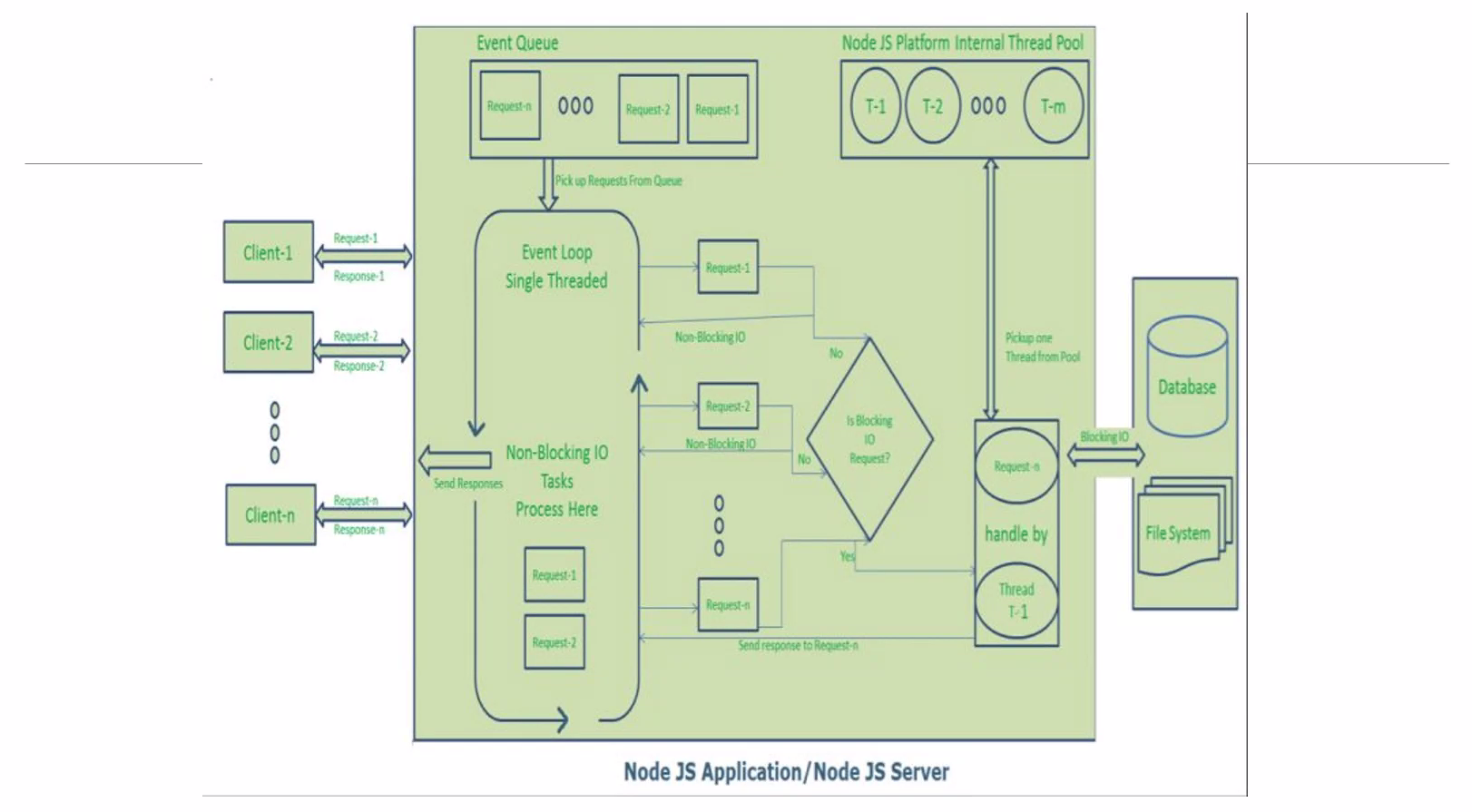


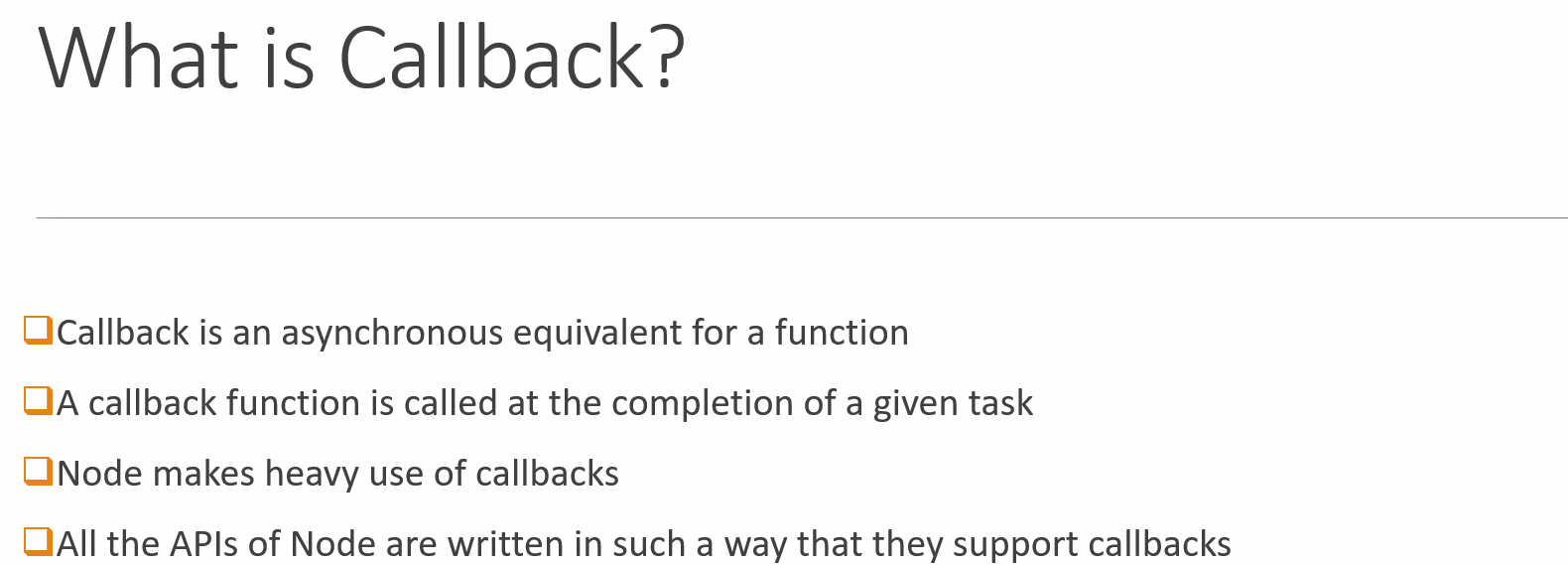








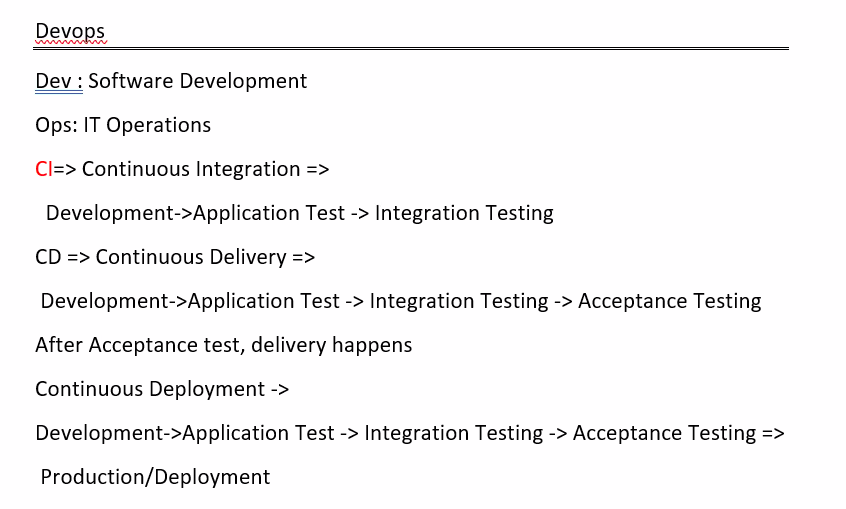




Mocha and Chai—

Mocha is a frame work

Chai is testing library => assert



GitLab Runner –

It is an open source

It is a build instance, used to run jobs over multiple machines and sends/displays the results.

File: .gitlab-ci.yml

Don’t use tabs

Demo\_job\_1:

Tags:

* Ci

Scripts:

* Echo “Hi All”

In your System –

Download gitlab-runner

Copy the downloaded gitlab-runner.exe to c:\Gitlab-Runner

Gitlab-runner.exe install

Gitlab-runner -version