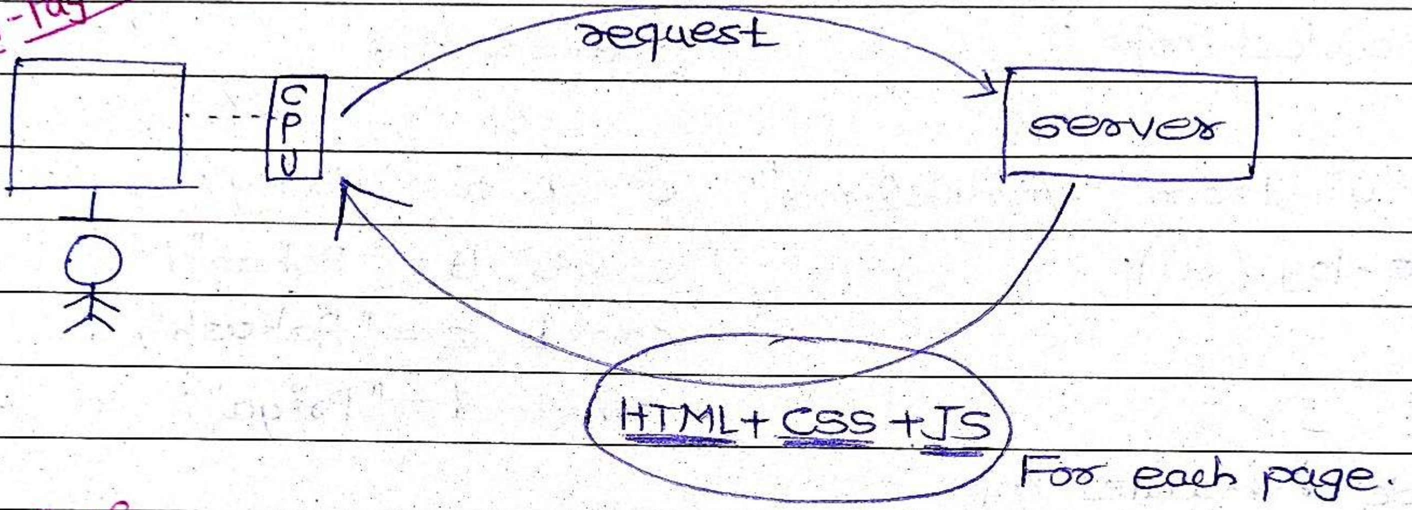
⑤ Multi Page Website / Single Page Website =

Multi-Page



Single-Page

Once HTML, CSS, JS is downloaded through response, the same page will be populated using Javascript and API's.

* Node_modules has all dependencies to create react_app. When we give code to someone, we delete this. we can add this folder using npm install. Hence we add folder in .gitignore.

* Bootstrap = add in public/index.html

CSS → above title
&
Java → below ~~</body>~~ above body

11:00 min watch.

getbootstrap.com

* To deploy the Project :

Use > npm run build.
Optimized Version.

* Don't add secrets in public folder.

⑥ Props & its types in react =

Commit code until now.

```
> git add .
> git commit -m "Video5 Completed"
```

* See ES7 extension shortcuts.

- no need to import react, As we r using function-based components.

- named export / default export =

module1.mjs	module2.mjs

import ui from './module2.mjs'
console.log (ui);
	const a = "Harry";
	const b = "Rohan";
	const c = "Aakash";
	const d = "Priya";

> node .\module1.mjs Aakash	export default c;

- ~~default~~ ~~exp~~ named export =

import dz, {a, c, d} from
'./module2.mjs'
 	
console.log(c);
console.log(a);
	const a = " ";
	const b = " ";
	const c = " ";
	const d = " ";
	export {a};
	export {c};
	export {d};

- 09:37 to 10:40
- 12:07 = props
  = to change title dynamically.
  = 2 changes in 2 files.

rfc
react
function
based
component

◎ Navbar.js → <title> {props.title} </title>
◎ App.js → return (
      <>
      <Navbar title = "TextUtils" />
      </>
    );

- prop-types =
- default proptypes =

} use both of them together.

* Props & PropTypes are used for passing readOnly attributes between React Component.

* PropTypes ensure that passed value is of correct datatype.
This makes sure that we don't receive an error at the very end of app, by the console, which might not be easy to deal with.

(7) State & handling events in React =

- components / TextForm.js
- copy paste form code on bootstrap.

Check errors in console in every execution

* State belong to one component.
* 1 import React, {useState} from 'react'
2 const [text, setText] =
      useState('Enter text here');

React hooks ⟶ ↑

useState is 1 hook

/ State Variable

11:00

Page No.
Date

* <u>to update state of text</u> =

return (

- <h1> {props.heading} - ~~bea~~ {text} </h1>
- <textarea className="form-control"
  value = {text}
  id = "myBox">
  </textarea>
- setText("new text"); // correct way.

19:00 23:00

☑ how to handle event

☑ how to set state

(8) 01:10
~~05:30~~

(11)(13)
(9) Exercise add other features text analyzer. (1)

(12)(13)
(10) 01:00 component/About.js

* <u>Enable Dark Mode" Button</u> using useState Hook =

(2)

05:24

08:20

10:40

13:00

15:32

(11) 03:40

(13) Add auto-dismissing alert msg =

- components/Alert.js

(14) Adding custom color theme

Exercise = (1) add themes

(2) Use JS to, not to count empty string.

Page No.
Date

(15) Change title Dynamically =

(16) React Router
- Navigate smoothly from one page to another without reloading whole site/page.

> npm install react-router-dom
> import 4 Lines

06:00

✓ - use <Route exact - - - - >
<link to=" " />

(17) - Download Git
- —/— Github
- Deployment of Create-react-app (search)
  └→ Github Pages.io
> npm run build (static application) (run build folder)
-
- install gh-pages using npm

✓ * React router doesn't work well with github pages.

* Remove About page.

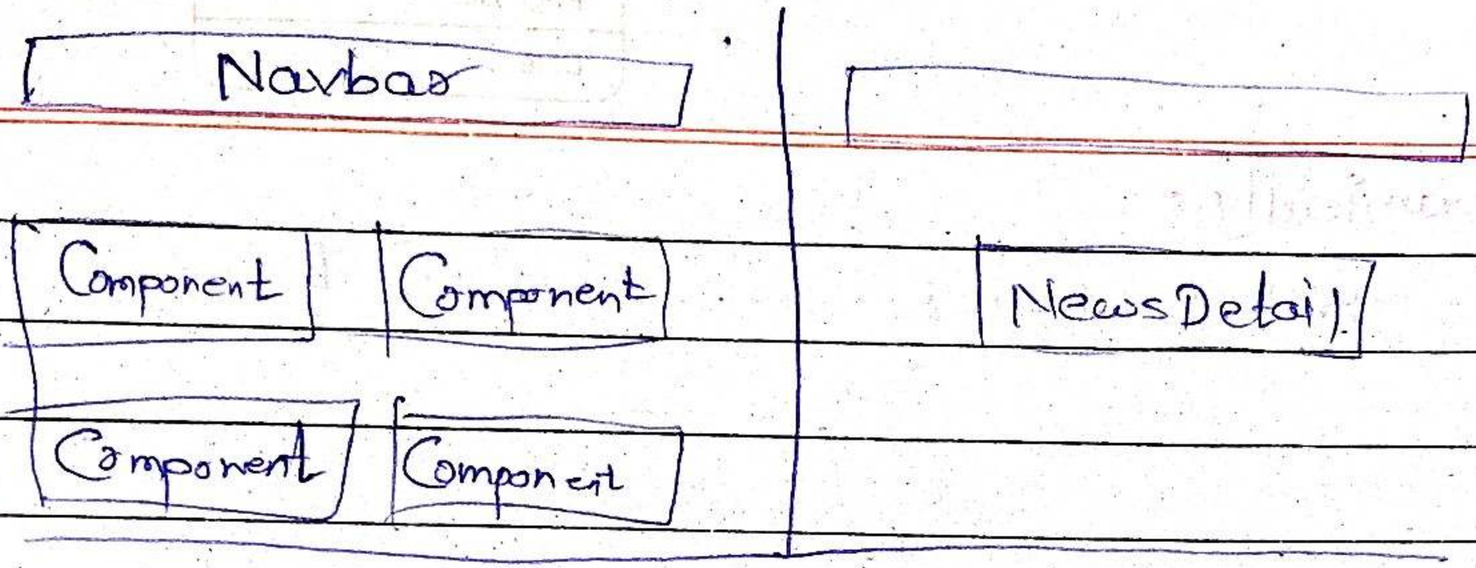
Deploy Website.

(18) Purchase domain + host TextUtils on VPS.

> npm run build.

*

$\frac{20}{22}$ Class based components = (News App)



= rcep with proptypes.

- Props = we can't change
- State = we can change, dynamically, without reloading page.

Page No.
Date

- Dependency
  npm i -f --save axios @material-ui/core.

* React - router =
① npm install react-router-dom
② import BrowseRouter
   Switch
   Route
   Link in App.js
③ Close all tags like <Navbar>
   <News>
   in <Router >
   —
   —
   </Router> tag
④ Copy <Switch> tag above closing <Router> as it is.
⑤ Replace <a href = ... with >
   <Link href = "/. . . ">
⑥ Replace href with *to =

React Component LifeCycle =

- Mounting = Birth of Component
- Update = Growth of Component
- Unmount the Component = Death of Component

Methods =
① render ( ) =
- used to render html of component in react
- required for a class based component to render DOM
- It runs during mounting & updating of component
- render ( ) method should be pure i.e
  we cannot modify state inside it.

Page No.
Date

② ComponentDidMount()
- runs after component output has been rendered to DOM.

③ ComponentDidUpdate()
- invoked as soon as updating happens.
- updates DOM in response to prop or state changes.

④ componentWillUnmount ()
- called just before component is unmounted and destroyed.
- used to perform cleanups.

* Adding infinite scrolling to newsapp =
  - Search = react-infinite-scroll.

  npm install --save react-infinite-scroll-component.

▪ infinite scroll $\xrightarrow{}$ News.js = Only code added from react site.
▪ React top loading bar =
- > npm install react-top-loading-bar
- Used in 2 ways : ① ref
  ② state ✓
  → App.js

□ Converting class based components app to function based =
- use react hooks.

- hooks = Features of class based components in function based components.
- It allows you to use state and other react features without writing class.
- hooks are functions which "hook into" React State and lifecycle features from function components.

Page No.
Date

Commonly used React hooks =

① useState = (text, setText)

② useEffect =
↳ can do work of componentDidUpdate.

③ useContext =
↳ when many components are there, complex structure.
↳ makes .. globally available.

④ useRef
↳ holder for DOM element

* We can have both components in single app (class and function)

```
> gid add .
> git commit -m "Commit msg"
> git remote add origin Link_http
> git remote
> git remote -v
> git push -u origin master
> git remote rm origin      (remove origin).
```

Page No.
Date

- react detector = add to chrome
- react developer tools = add to chrome.
- download NodeJS for react
- Component in ReactJS
- Create react component =

```
    ReactDOM.render(
        React.createElement('ul', null,
            React.createElement('li', null, 'Item1'),
            React.createElement('li', null, 'Item2')
        ),
        document.getElementById('react-container')
    )
  </script>
```

- JSX in React JS =

Copy and paste bable.min.js <script/> line in head and use directly.

```
    ReactDOM.render(
            <ul>
                <li> --- </>
    )
```

- Class based Component =

```
export class News extends Component {
    static defaultProps = {

    }
    static propTypes = {

    }
    constructor (props) {
        super (props);
        this.state = {

        }
    }
```

Page No.
Date

```
export class News extends Component{
  async updateNews(){
        this.setState({        })
  }
  render(){
        return(
              <>


              </>
        )
    }
}
```

- Stateless Component in react =

```
      Const myComponent = () => {
            return <div> hello
                        </div>
      }
```

- Props in react =

```
                                   "business"
    <News    category = ~~{category}~~ />
        render() { return <>
        <h1> {this.props.category} </h1>
            </>
        }

    <News  category= "general" />
```

- events in reactJS =

```
      ~~et~~ class News extends Component{
            editing(){
            }
            render(){
                  return(
                        <div>
                              <button onClick=
                                    {"this.editing}/>
```

Page No.
Date

- State in ReactJS =

Refs in React JS =
→ to get value of textarea

```
<textarea ref="newText"/>

save=() => {
    var val = this.refs.newText.value;
    console.log(val);
  }
```

JSX = Javascript XML
- JSX allows us to write html in React
- It is a syntax extension to Javascript based in ES6, "newest version of Javascript"
- JSX allows to write html in React, by converting HTML into React components.

ES6 = (2015)
- ECMA Script 6th version.
- created to standardize Javascript

Babel in React =
- toolchain used to convert ECMAScript 2015+ code into a backwards compatible version of Javascript in current and older browsers or environments.

- helps developers use latest Javascript features, without worrying about support for said features in older browsers.