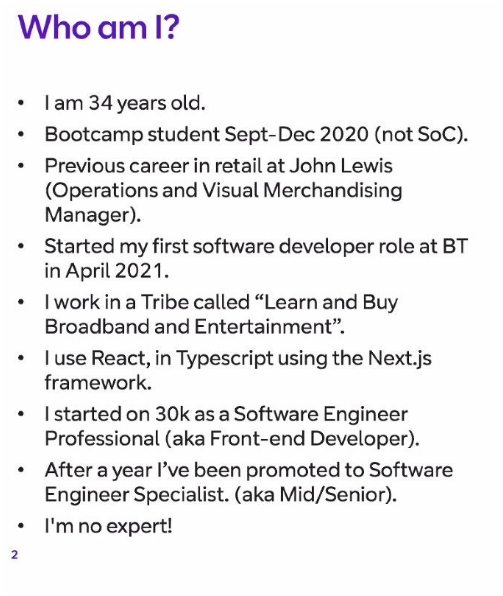
<u>Nick Truby – Software Engineer Specialist (aka Front-end developer) at BT</u>

Last minute presentation!

Use React for his day job and is also a mentor for SOC – will be at Code club on Wednesday.

<u>His background:</u>

Career changer, just like us!



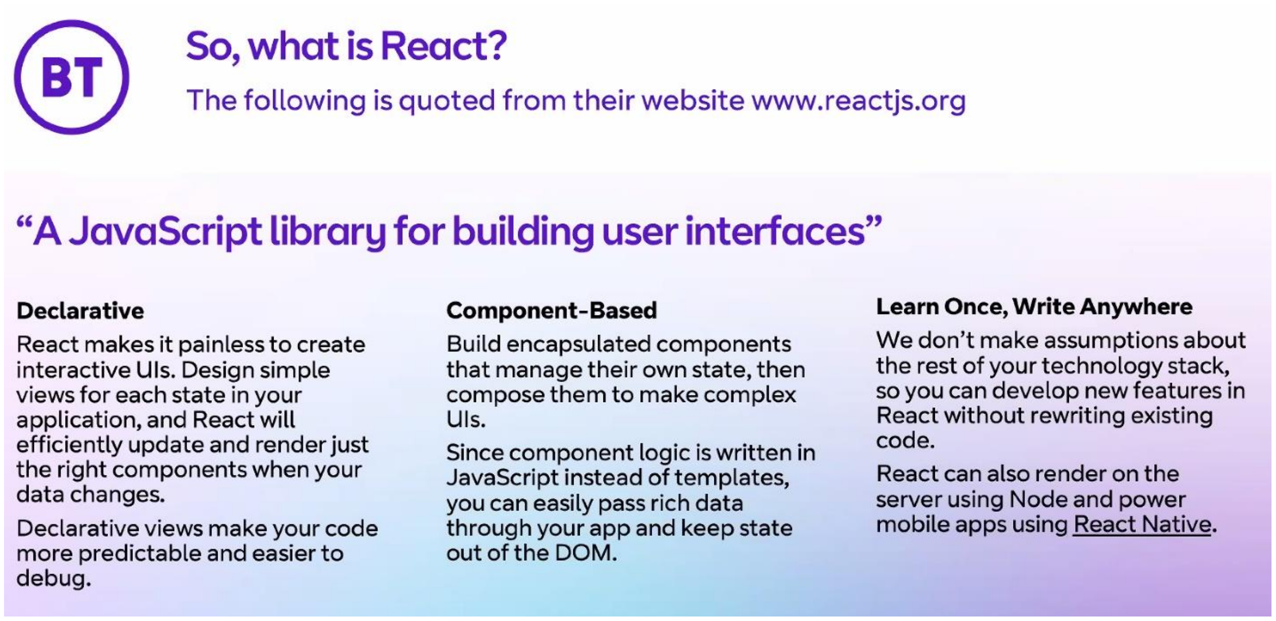
Who am I?

- I am 34 years old.
- Bootcamp student Sept-Dec 2020 (not SoC).
- Previous career in retail at John Lewis (Operations and Visual Merchandising Manager).
- Started my first software developer role at BT in April 2021.
- I work in a Tribe called "Learn and Buy Broadband and Entertainment".
- I use React, in Typescript using the Next.js framework.
- I started on 30k as a Software Engineer Professional (aka Front-end Developer).
- After a year I've been promoted to Software Engineer Specialist. (aka Mid/Senior).
- I'm no expert!

2

Typescript is like a more specific version of JavaScript – you can specify data types and it will make you use the correct data type. Next.js is a framework built on type of React – gives you server-side rendering (for SEO). Payband for "Specialist" starts at £37k.

<u>What is React?</u>



So, what is React?

The following is quoted from their website www.reactjs.org

“A JavaScript library for building user interfaces”

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Based

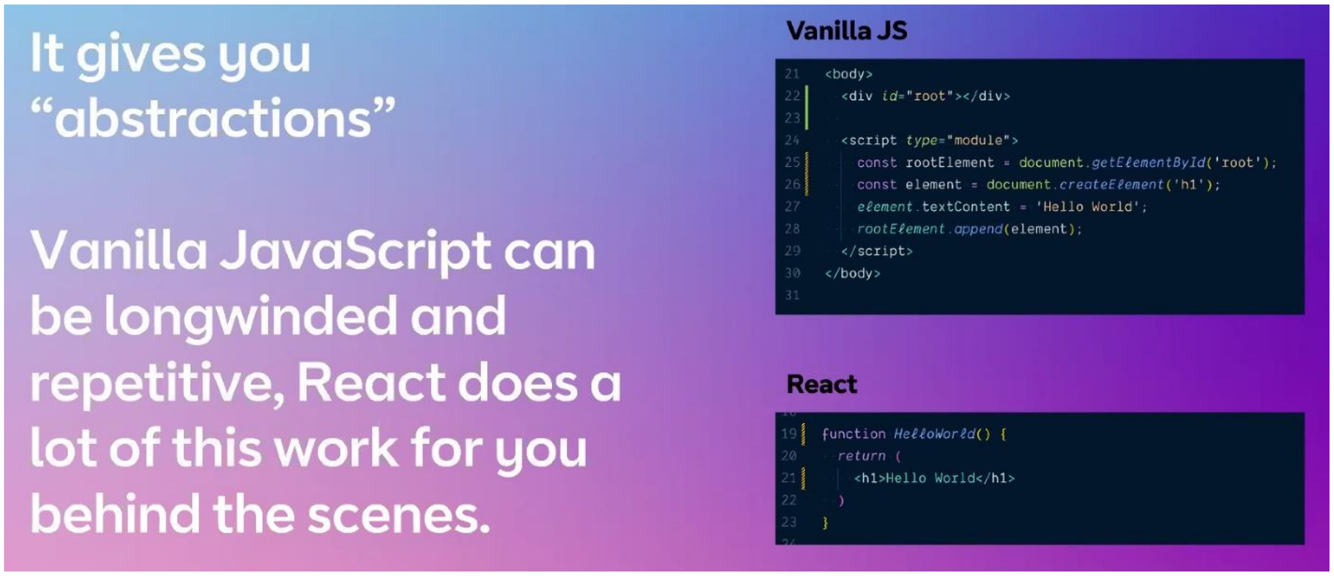
Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

Learn Once, Write Anywhere

We don’t make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using React Native.



It gives you “abstractions”

Vanilla JavaScript can be longwinded and repetitive, React does a lot of this work for you behind the scenes.

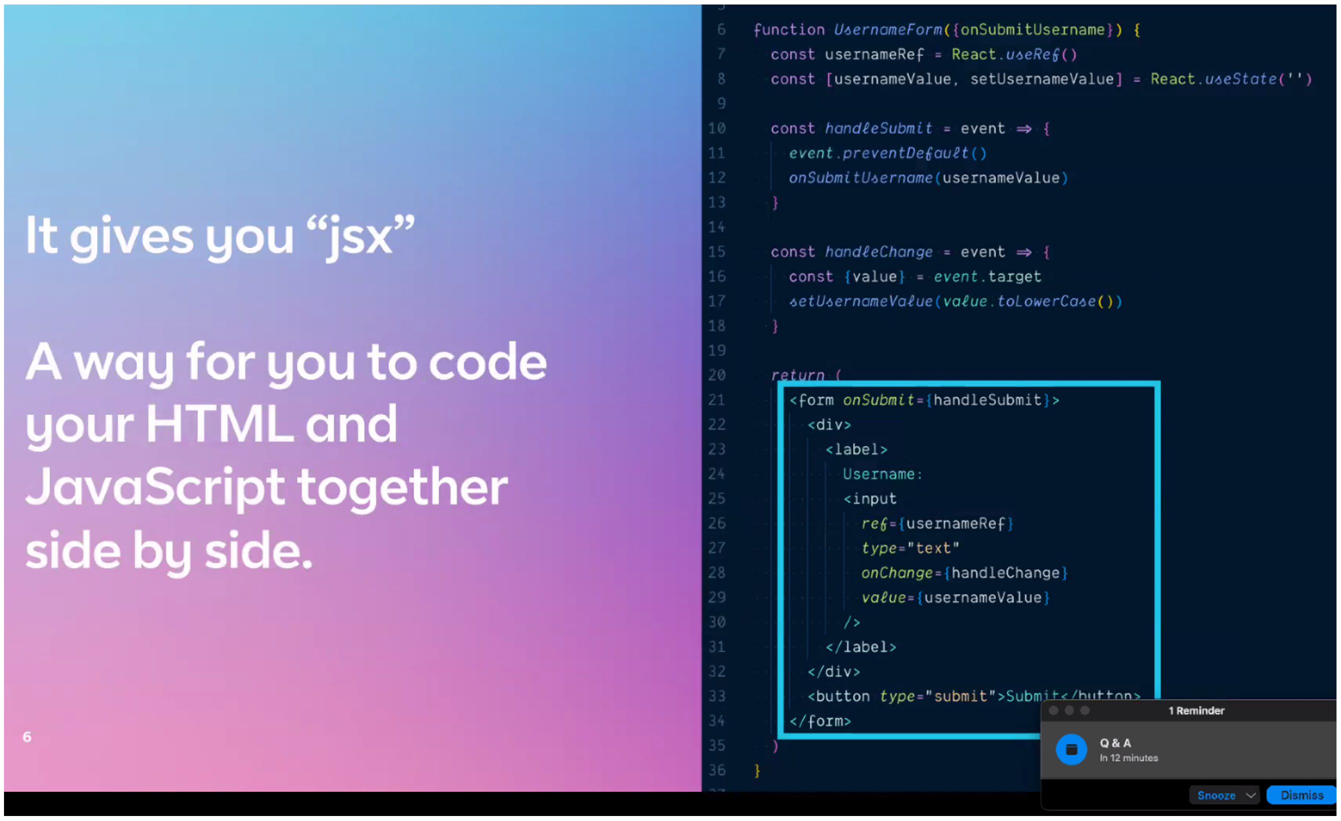
Vanilla JS

```
<body>
  <div id="root"></div>

  <script type="module">
    const rootElement = document.getElementById('root');
    const element = document.createElement('h1');
    element.textContent = 'Hello World';
    rootElement.append(element);
  </script>
</body>
```

React

```
function HelloWorld() {
  return (
    <h1>Hello World</h1>
  )
}
```

Box in blue is JSX – a combination of JS and HTML:

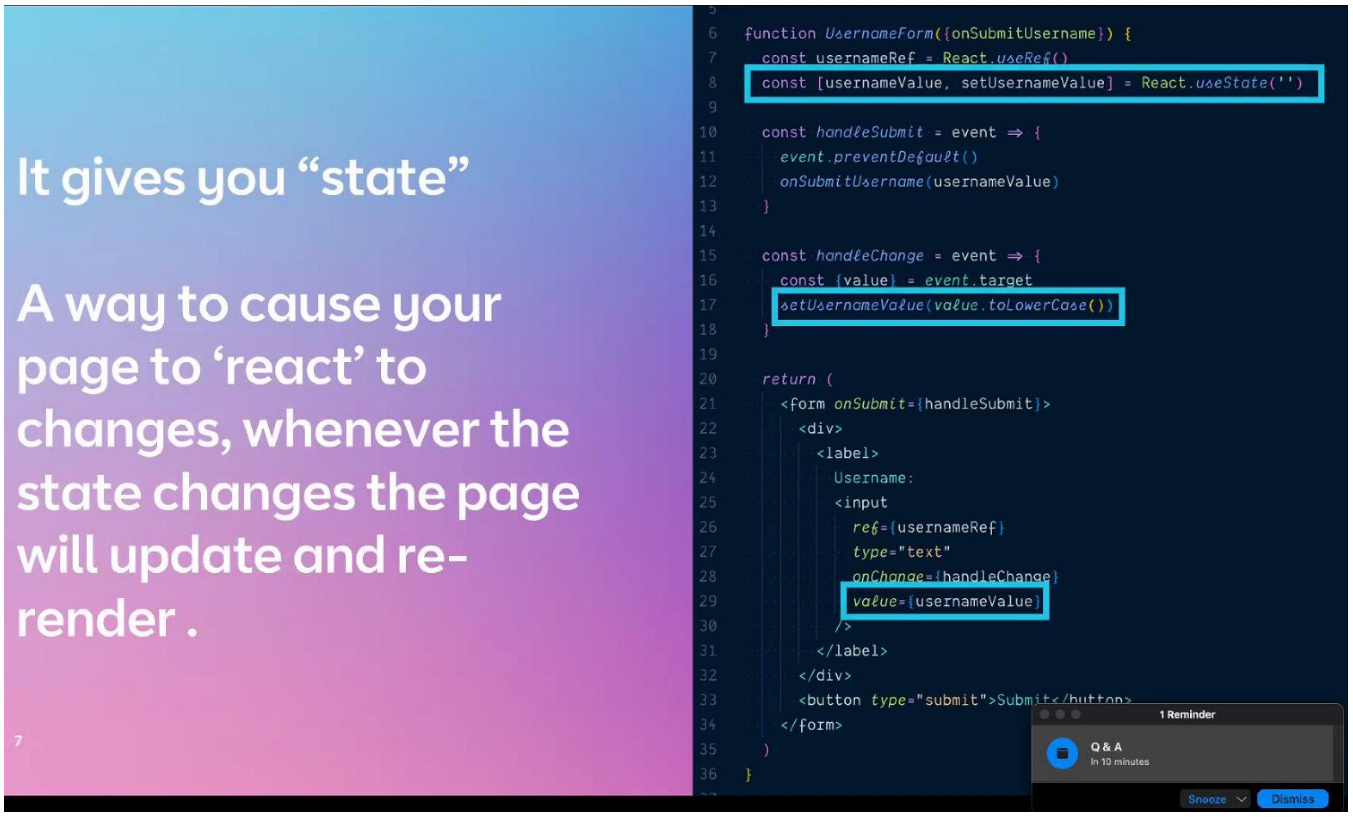


It gives you "jsx"

A way for you to code your HTML and JavaScript together side by side.

6

```
function UsernameForm({onSubmitUsername}) {
  const usernameRef = React.useRef()
  const [usernameValue, setUsernameValue] = React.useState('')

  const handleSubmit = event => {
    event.preventDefault()
    onSubmitUsername(usernameValue)
  }

  const handleChange = event => {
    const {value} = event.target
    setUsernameValue(value.toLowerCase())
  }

  return (
    <form onSubmit={handleSubmit}>
      <div>
        <label>
          Username:
          <input
            ref={usernameRef}
            type="text"
            onChange={handleChange}
            value={usernameValue}
          />
        </label>
      </div>
      <button type="submit">Submit</button>
    </form>
  )
}
```

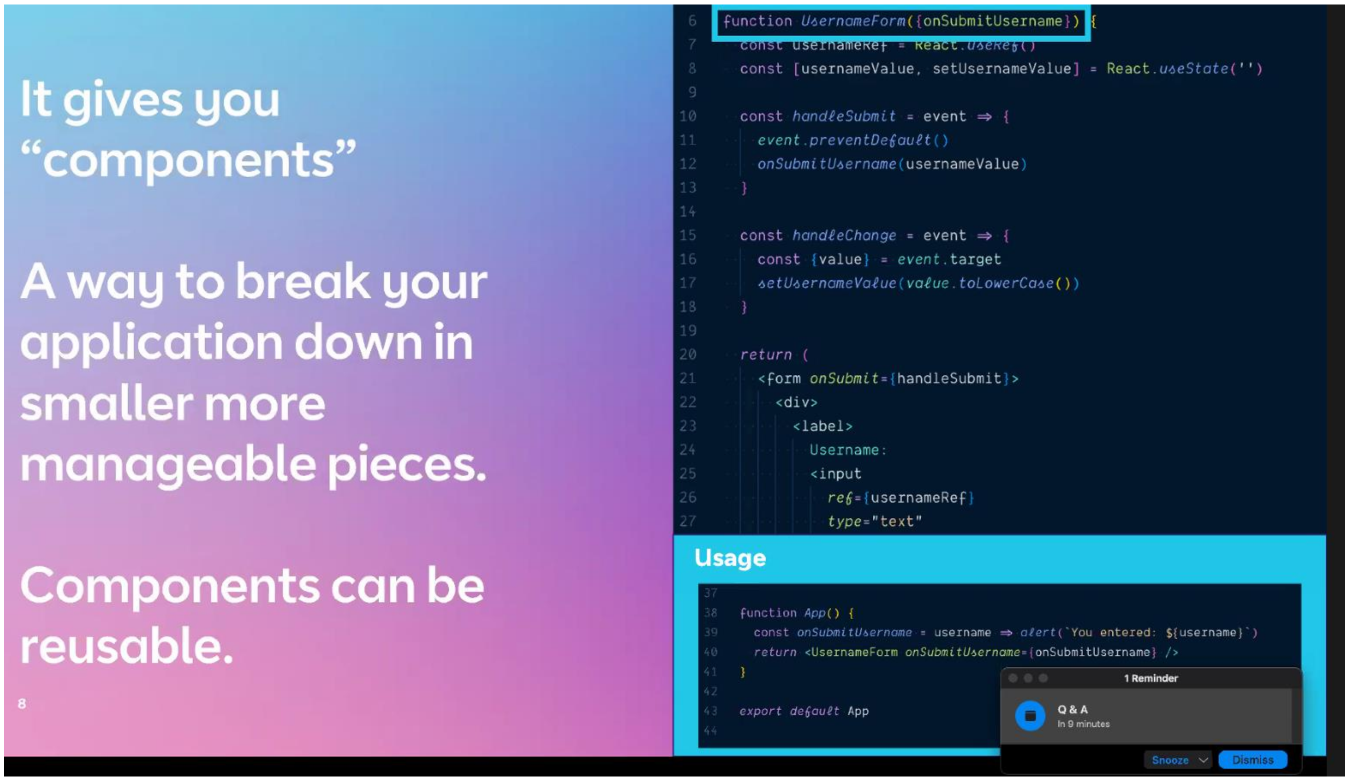


It gives you "state"

A way to cause your page to 'react' to changes, whenever the state changes the page will update and re-render .

7

```
function UsernameForm({onSubmitUsername}) {
  const usernameRef = React.useRef()
  const [usernameValue, setUsernameValue] = React.useState('')

  const handleSubmit = event => {
    event.preventDefault()
    onSubmitUsername(usernameValue)
  }

  const handleChange = event => {
    const {value} = event.target
    setUsernameValue(value.toLowerCase())
  }

  return (
    <form onSubmit={handleSubmit}>
      <div>
        <label>
          Username:
          <input
            ref={usernameRef}
            type="text"
            onChange={handleChange}
            value={usernameValue}
          />
        </label>
      </div>
      <button type="submit">Submit</button>
    </form>
  )
}
```



It gives you “components”

A way to break your application down in smaller more manageable pieces.

Components can be reusable.

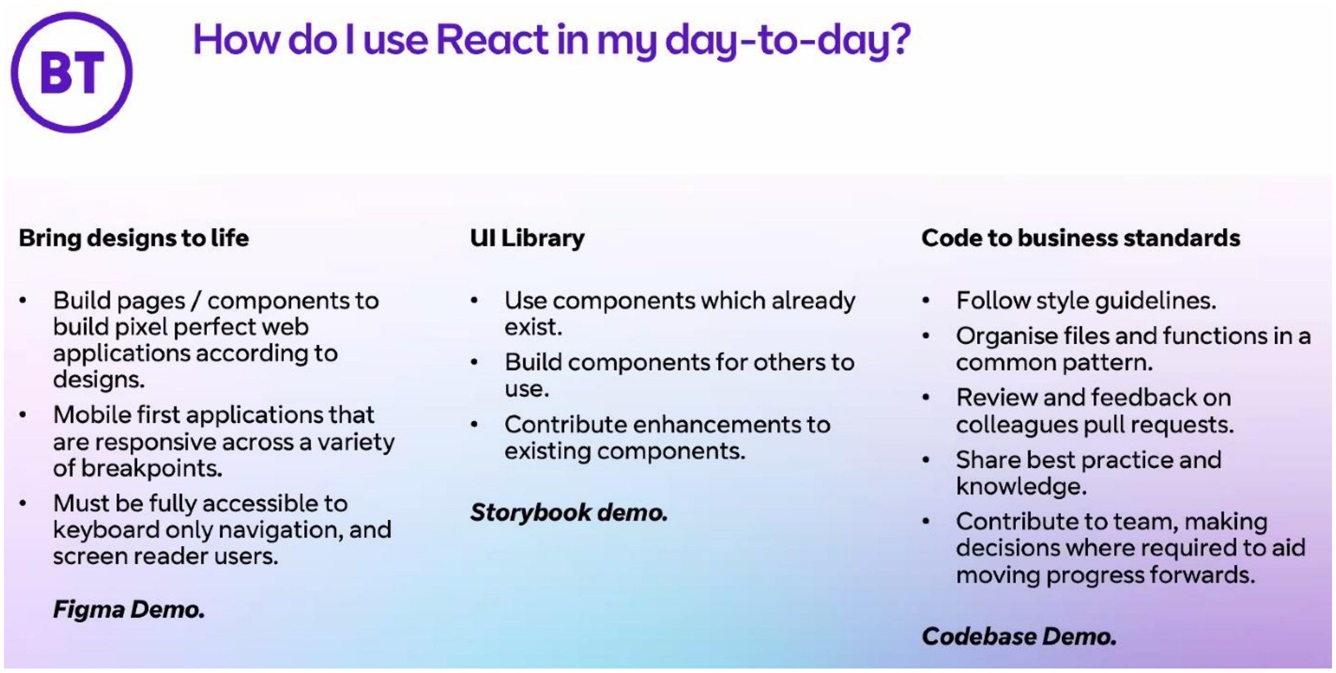
8

```
function UsernameForm({onSubmitUsername}) {
  const usernameRef = React.useRef()
  const [usernameValue, setUsernameValue] = React.useState('')

  const handleSubmit = event => {
    event.preventDefault()
    onSubmitUsername(usernameValue)
  }

  const handleChange = event => {
    const {value} = event.target
    setUsernameValue(value.toLowerCase())
  }

  return (
    <form onSubmit={handleSubmit}>
      <div>
        <label>
          Username:
          <input
            ref={usernameRef}
            type="text"
```

Usage

```
function App() {
  const onSubmitUsername = username => alert(`You entered: ${username}`)
  return <UsernameForm onSubmitUsername={onSubmitUsername} />
}

export default App
```

Starts with mobile first, then scales it up.



How do I use React in my day-to-day?

Bring designs to life

- Build pages / components to build pixel perfect web applications according to designs.
- Mobile first applications that are responsive across a variety of breakpoints.
- Must be fully accessible to keyboard only navigation, and screen reader users.

Figma Demo.

UI Library

- Use components which already exist.
- Build components for others to use.
- Contribute enhancements to existing components.

Storybook demo.

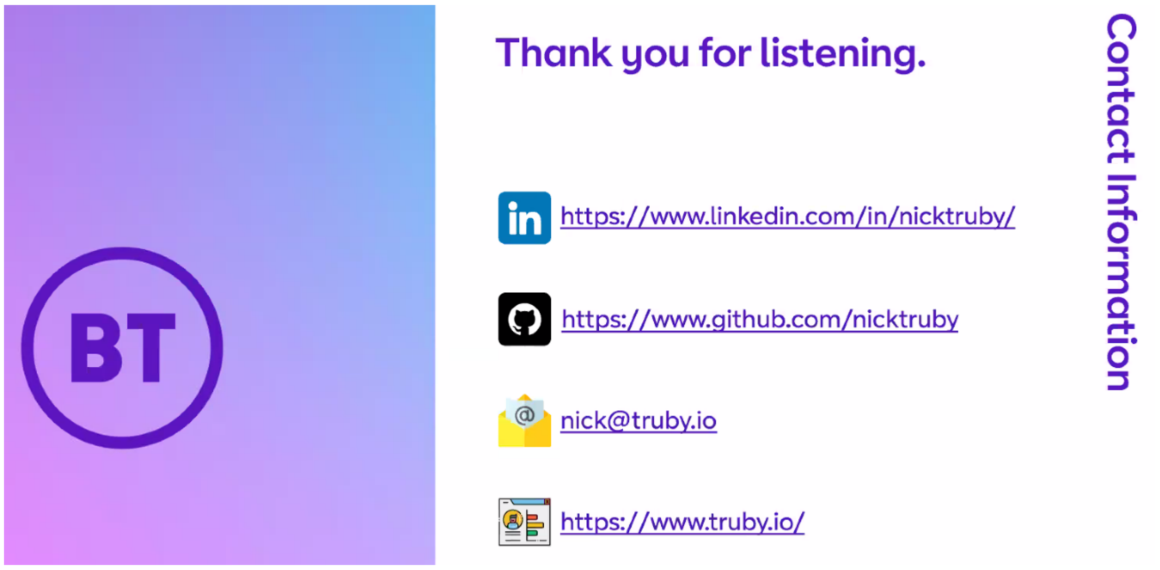
Code to business standards

- Follow style guidelines.
- Organise files and functions in a common pattern.
- Review and feedback on colleagues pull requests.
- Share best practice and knowledge.
- Contribute to team, making decisions where required to aid moving progress forwards.

Codebase Demo.

Showed us some proprietary designs from BT’s library so not screenshotted – these were Figma designs. He talked us through about breaking each wireframe down in to individual components so he can start creating the actual code for it – e.g. h1, a card for broadband with a price on it and list of features, a card for add-ons, various <p>, an accordion (something that expands out). His job is to build components (if not already existing in BT's UI library) and make them work (async functions to fetch data etc).

Also showed us their folder layout (not screenshotted as above) - lots of similar folders to how we lay things out on SOC – particularly showed us a src folder that has subfolders: components, hooks, contexts, screens, pages etc so everything is separated out and imported where needed. Each component has its own folder – containing all files relevant to that component.



Thank you for listening.

Contact Information

https://www.linkedin.com/in/nicktruby/

https://www.github.com/nicktruby

nick@truby.io

https://www.truby.io/

Q & A

Recruitment process/Interview process : Bootcamp he did was similar to SOC so had some recruitment part to it. He could apply for own things immediately as paid for his course. Applied for loads of things, got 4 interviews. 2 rejected, 3rd verbally accepted but never happened and then got BT job. BT was the job he actually wanted but each rejection was tough to take at the time. This is when imposter syndrome is at its highest – so keep an eye on mental health!

Interview process: Computer-based Tech Test – similar to codewars but a bit more technical. Then got interview and was asked to take away a task and then submit it. It was simple and needed to demo a typical day – create a News application, use an API that provides news article by search. Search box that sends API request and renders it to the page. Had to be responsive as was adding news articles each time. This got him to final interview – you don't have to know everything on the job description, but be able to talk about each thing! E.g. research each part of the job description so you get an idea about it, basics of what it might be used for etc. Be honest about what you don't know! Talk about "development areas", not "weakenesses".

Take home tasks were for about a week and he spent at least 2 or 3 days full time on it at least.