REACT

THE GOAL FOR THIS SESSION

1. *Understand the problems React solves*
2. *Understand how to use React to build reusable UI Components*

AGENDA

1. Why React
2. The new & old docs
3. Introduction to the plugins you installed
4. Installing react
5. functional components #1
6. Nested components
    - the React extension

WHY REACT

HOW REACT WORKS

1. you display data on a webpage
2. a user interacts with it
3. the data changes
4. React updates the UI

1. THE USE OF REUSABLE, COMPOSABLE, AND STATEFUL COMPONENTS

Working with the DOM API is hard. React basically gives developers the ability to work with a virtual browser that is more friendly than the real browser. React's virtual browser acts like an agent between the developer

2. THE NATURE OF REACTIVE UPDATES

React enables developers to declaratively describe their User Interfaces and model the state of those interfaces. This means instead of coming up with steps to describe transactions on interfaces, developers

3. THE VIRTUAL REPRESENTATION OF VIEWS IN MEMORY

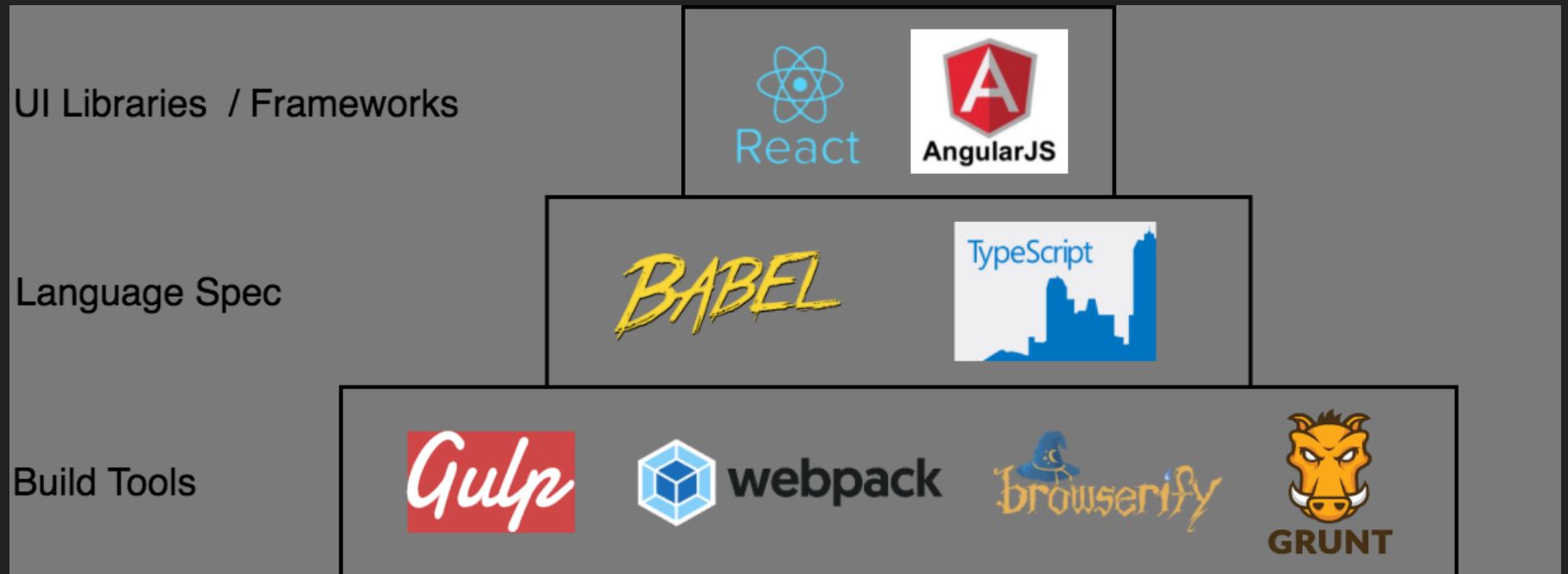
React is just JavaScript, there is a very small API to learn, just a few functions and how to use them. After that, your JavaScript skills are what make you a better React developer.

There are no barriers to entry. A JavaScript developer can become a productive React developer in a few hours.

REALLY?

REACT IS "EASY"

But the tooling around it is HARD



`create-react-app`

which is why we use CRA

https://github.com/facebook/create-react-app

TODO: Vite? https://medium.com/codex/you-should-choose-vite-over-cra-for-react-apps-heres-why-47e2e7381d13

WARNINGS

- React does not use the DOM as such
- It handles all DOM manipulation for us
- Stuff like `querySelector`, `classList`, `createElement` etc are usually a no-go
- Even `setInterval` becomes hard
- We have to do it the React-way

THE NEW & OLD DOCS

The React docs are currently being re-written

A lot of the examples in the old documentation uses the "old school" syntax (classes)

The new docs fix this, but they are not done

- https://reactjs.org/
- https://beta.reactjs.org/

Editors choice: use the new (beta) one

PLUGINS

YOU INSTALLED

1. ESLint: a linting-tool for JS. Warns about common syntax issues (like == vs ===)
2. Import Cost: every time you import some code, it tells you how much code is imported in KB

3. React Developer Tools: Adds react inspecting/debugging in Chrome/FireFox, priceless
4. ES7 React/Redux/GraphQL/React-Native snippets: Adds small Emmet-snippets to VSCode, so you can type e.g `rfce` to get a full React component
5. get-svg-icons: Just a quick little way of getting

INSTALLING & RUNNING REACT

STARTING A NEW PROJECT

Used to be really hard, now? less so

Two easy options

- https://github.com/facebook/create-react-app
- https://vitejs.dev/guide/#scaffolding-your-first-vite-project

Let's go for Vite, since we know it

Disclaimer: I haven't tried Vite/React yet, let's see if it comes back to bite us

1. `npm init vite@latest`
2. Pick React
3. Pick React
4. `cd` to your folder
5. `npm install`
6. `npm run dev`

And then as usual, it's just `npm run dev` to start you live server

and `npm run build` to generate the static version

Which can be deployed to e.g. Netlify

FUNCTIONAL COMPONENTS, #1

JSX

JSX is a preprocessor step that adds XML syntax to JavaScript. You can definitely use React without JSX but JSX makes React a lot more elegant.

Basically, it allows us to write HTML in our JS files

We can invent our own tags!

All tags must be closed

```
1 <MyNav />
2 //or
3 <MyNav></MyNav>
```

UP AND RUNNING, FUNCTIONAL COMPONENTS

Using JSX we can create re-usable components

```
1 function Greeting() {
2   return (
3     <article>
4       <h1>Hi mom</h1>
5     </article>
6   );
7 }
```

The component must return a single element, wrap in a parent element or `<>...</>` if nothing else makes sense

```
1 //BAD CODE (WON'T WORK)
2 function Test(){
3     return (
4         <p>Hello</p>//no parent node / element
5         <p>Monday</p>
6     )
7 }
```

```
1 //BETTER CODE
2 function Test() {
3   return (
4     <div>
5       <p>Hello</p>
6       <p>Monday</p>
7     </div>
8   );
9 }
```

```
1 //BEST CODE?
2 function Test() {
3   return (
4     <>
5       <p>Hello</p>
6       <p>Monday</p>
7     </>
8   );
9 }
```

EVERYTHING IS A COMPONENT

That's all we do, we write components

Small, encapsulated, reusable parts

We can also get other peoples components

The React extension

NESTED COMPONENTS

AND WE CAN NEST COMPONENTS!

```
1  function Container() {
2    return (
3      <article>
4        <Greeting />
5        <Greeting />
6        <Greeting />
7      </article>
8    );
9  }
10 function Greeting() {
11   return <h1>Hi mom</h1>;
12 }
```

Which means that we can now take the first steps into building WebApps / Sites with small, encapsulated modules

The React extension

STYLING, #1

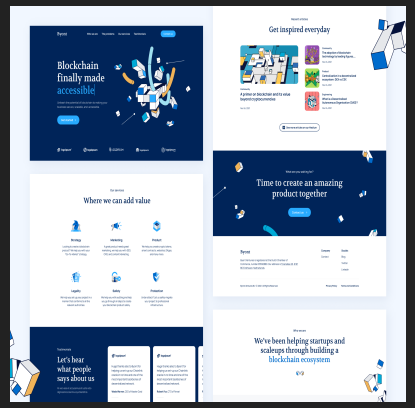
Lots of options exists (as it's just JSX turned into JS turned into HTML

We'll deal with some options later

For now, all you need to know is that:

`class=""` is `className=""` in React

+3/4



Recreate some of this layout using only functional components

At this point we do NOT care about the content

So if you have created a component that does something, but the content in the assignment is different, it doesn't matter

Image borrowed from Dribbble.com