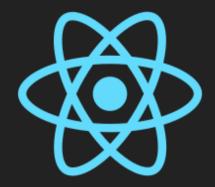
REACT



THE GOAL FOR THIS SESSION

- 1. Understand what props are and how they are used
- 2. Experience how to pass down props to a component, several layers deep
- 3. Learn how to break a UI in to multiple files using

import/export default

AGENDA

- 1. props
- 2. import / export default
- 3. prop drilling
- 4. mapping, #1

props

A FIRST LOOK AT props

Our code is not yet re-usable

Our greeting component can only say hi to our moms

In React, we can invent our own properties that the component can read

props are immutable, you cannot/should not change them

We'll get back to this when it makes sense, but...

WHENEVER A PROP CHANGE, THE COMPONENT IS UPDATED!!

- In the dev tools we can inspect React components as well
- We can manipulate state and props
- Let's take a look

MY MENTAL MODEL FOR UNDERSTANDING PROPS

- 1. All props/attributes we pass to a component will be
- 2. turned in to an object
- 3. passed unto the component
- 4. who receives it as a parameter called props

OR

Let's imagine a component called <MyComponent />

```
1 //We pass props to the component
2 <MyComponent
3 header="Hi Mom"
4 type="primary"
5 number={3}
6 data={[1, 2, 3]}
7 status={{
8 loading: false,
9 message: "not there yet",
10 }}
11 />;
```

```
1 //React turns it into an object
2 {
3   header: "Hi Mom",
4   type: "primary",
5   number: 3,
6   data: [1, 2, 3],
7   status: {
8    loading: false,
9   message: "not there yet",
10  },
11 };
```

```
1 //which is passed to the function
2 MyComponent({
3   header: "Hi Mom",
4   type: "primary",
5   number: 3,
6   data: [1, 2, 3],
7   status: {
8    loading: false,
9   message: "not there yet",
10  },
11 });
```

```
2 export default function MyComponent(props) {
     console.log(props); /* =>
10
12
13
14
```

import /

export default

SPLITTING / CLEANING UP OUR CODE

This is probably my favourite part, splitting up our code

We'll put each component in a separate file, and

- Increase readability
- Improve maintainablility
- Improve re-usability
- Make it easier to work together

The process is simple

- 1. Put each component in a separate file
- 2. Name the file the same as your component and add .js or .jsx
- 3. Add export default in front of the function definition
- 4. Import it in the "parent" component file

EXAMPLE

```
2 export default function Navigation() {
  return (
  <a href="#">Home</a>
  );
 2 import Navigation from "./Navigation";
                                                           export default function App() {
    return (
      <div id="wrapper">
      <main>...</main>
10
     );
11 }
```

PROP DRILLING

OL

PASSING DOWN PROPS

- We've seen props, and we can pass them from parent to child
- But very often we need to pass our data down several levels
- If so, just repeat the process

EXAMPLE, MANUAL

```
1 function App() {
                                                             const user = {
      name: "Jonas",
    email: "jofh@kea.dk",
    kids: 3,
    };
    return (
    <div id="app">
         <Navigation name={user.name} email={user.email} />
12 );
13 }
14
15 function Navigation(props) {
     raturn (
```

App => Navigation => Profile

EXAMPLE, DESTRUCTURING

(just forward everything)

```
1 function App() {
                                                            const user = {
   name: "Jonas",
       email: "jofh@kea.dk",
   kids: 3,
    };
    return (
    <div id="app">
12 );
13 }
15 function Navigation(props) {
     return (
```

- If This was all React could do I would already be kind of happy
- Components are an amazing way of building UI's
- But we need interactivity
- And for that we have state and events

MAPPING, #1

map'ING RENDERING ARRAYS

- We have an array of data
- We want to render components based on that data
- React already knows what to do with an array of components / DOM elements

BABY STEP #1

BABY STEP #2

We can ++ it, and use components

- But mostly, we have a "simple" array with data
- .map to the rescue
- Remember, map works on an array, and returns a new array for us

BABY STEP #3

We can .map it, and use components

```
1 export default function Test() {
2   const kids = ["Storm", "Birk", "Liv"];
3   const mapped = kids.map((kid) => <Kid name={kid} />);
4
5   return {mapped};
6 }
7 function Kid(props) {
8   return {props.name};
9 }
```

+3/4

Let's see what you can figure out Take what you built earlier and

- 1. fix the text's by passing props
- 2. See any repeated components? create an array of data and map through it to display the components

When to stop is up to you, keep hacking until you get