REACT

THE GOAL FOR THIS SESSION

1. *Undertsand how we communicate between components*
2. *Pass callbacks from parents to children*
3. *Be able to use and modify complex state in React*

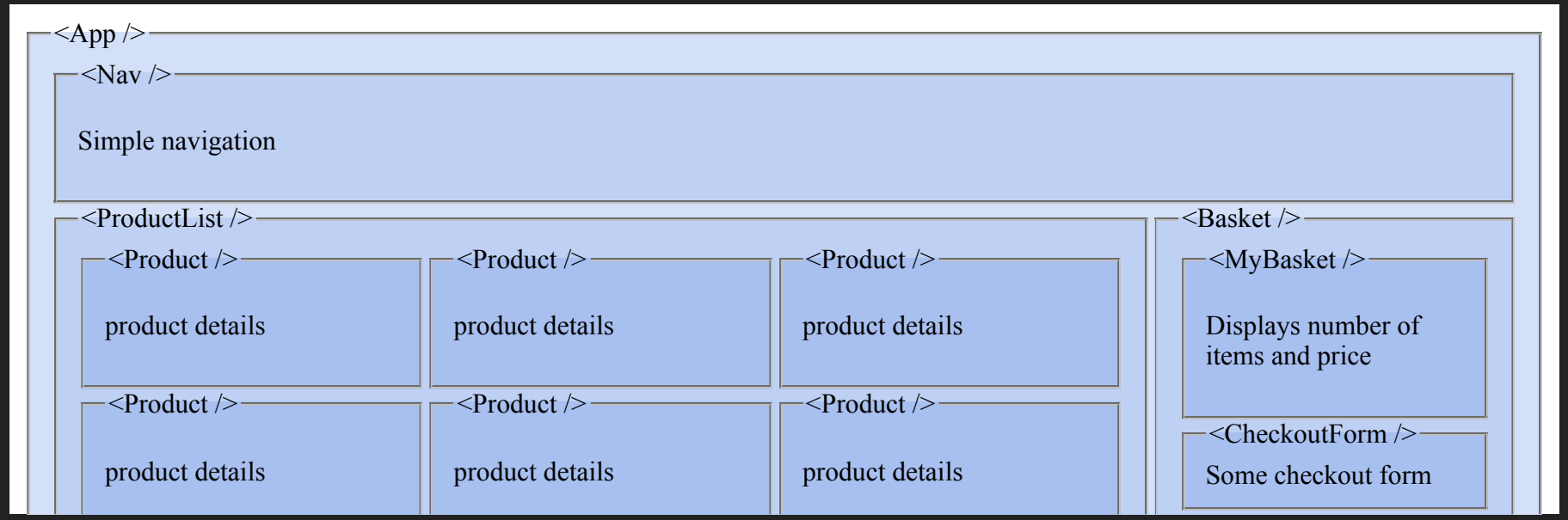
AGENDA

1. uni-directional data-flow
2. passing callbacks
3. state, #2
    - Add to a state-array
    - Remove from a state array
    - update a state array
4. Solving the ToDo list

UNI-DIRECTIONAL DATA-FLOW

UNI-DIRECTIONAL DATA FLOW

- **Data is sent down the tree**, as props (Parent => Child)
- **Data can be sent back up** (Child => Parent)
- **Data can NOT be sent sideways** ~~(Sibling => Sibling)~~



- Where are the siblings?

PLACING STATE

1. Each piece of "state" should be owned by one component
2. A single component above all the components that need the state in the hierarchy

So who would need to know if the Nav is open or closed?

Where do you put state then?

PASSING CALLBACKS

We've seen how to pass `props` from parent to child

If we want to pass events/data from a child to a parent, we need to use `callbacks`

It's not that different from vanilla JS

```
1  //this is a callback
2  setInterval(callback, 2000);
3
4  //so is this
5  elems.forEach(callback);
6
7  //and so is this
8  async function getData(url, callback) {
9    const response = await fetch(urls);
10   const json = await response.json();
11   callback(json);
12 }
13 getData("https://....", console.log);
```

```jsx
function Parent(props) {
  const myCallback = (e) => {
    console.log("a child sent this up", e);
  };
  return <Child childClickHandler={myCallback} />;

}

function Child(props) {
  return <button onClick={props.childClickHandler}>Click Me</b
}
```

Sometimes we have to pass them really far down the hierarchy

STATE, #2

useState

WITH ARRAYS

- Breaking a design into components is easy enough
- Adding +1 to a counter is easy enough
- But our state has been pretty simple so far, that's not always the case
- https://reactjs.org/docs/hooks-reference.html#usestate

Repeat after me

I MUST NEVER MODIFY STATE DIRECTLY

I MUST NEVER MODIFY STATE DIRECTLY

I MUST NEVER MODIFY STATE DIRECTLY

I must never modify state directly

Always use the "updater" function

So when it comes to modifying an array, it's all about

making a copy, modifying that, and then settings that to state

are these Good or Bad?

- `.push` is BAD, it modifies the original
- `.concat` is fine, it creates a copy
- `[...orig, newThing]` is fine, it creates a copy
- `{...orig, newThing}` is fine, it creates a copy
- `.splice` is BAD, it modifies the original
- `.slice` is fine, it creates a copy

SAFELY, REMOVING FROM AN ARRAY, VANILLA

```
let persons = [
  { name: "Jonas", id: 1 },
  { name: "Klaus", id: 2 },
  { name: "Peter", id: 3 },
];
function remove(id) {
  return persons.filter((person) => person.id !== id);
}
```

Removing an item from a state-array

```
1  import { useState } from "react";
2  export default function StateArray() {
3    const [persons, setPersons] = useState([
4      { name: "Jonas", id: 1 },
5      { name: "Klaus", id: 2 },
6      { name: "Peter", id: 3 },
7    ]);
8    const removePerson = (id) => {
9      setPersons((prevState) => prevState.filter((person) => person.id !=
10   };
11   return (
12     <section>
13       <ul>
14         {persons.map((item) => {
15           return (
```

SAFELY, ADDING TO AN ARRAY, VANILLA

```
let persons = [
  { name: "Jonas", id: 1 },
  { name: "Klaus", id: 2 },
  { name: "Peter", id: 3 },
];
function add(newPerson) {
  return persons.concat(newPerson);
}
```

Adding an item to a state-array

```jsx
import { useState } from "react";
export default function StateArray() {
  const [persons, setPersons] = useState([
    { name: "Jonas", id: 1 },
    { name: "Klaus", id: 2 },
    { name: "Peter", id: 3 },

  ]);
  const addPerson = () => {
    setPersons((prevState) =>
      prevState.concat({
        name: "Dannie",
        id: 4,
      })
    );
  };
```

SAFELY, CHANGING AN ARRAY, VANILLA

```
let persons = [
  { name: "Jonas", id: 1 },
  { name: "Klaus", id: 2 },
  { name: "Peter", id: 3 },
];
function modify(id, newName) {
  const newPersons = persons.map((person) => {
    if (person.id === id) {
      person.name = newName;
    }
    return person;
  });
  return newPersons;
}
const nextPersons = modify(1, "Dannie");
```

Modifying an item from a state-array

```
1  import { useState } from "react";
2  export default function StateArray() {
3    const [persons, setPersons] = useState([
4      { name: "Jonas", id: 1 },
5      { name: "Klaus", id: 2 },
6      { name: "Peter", id: 3 },
7    ]);
8    const modifyPerson = (id, newName) => {
9      setPersons((prevState) =>
10       prevState.map((person) => {
11         if (person.id === id) {
12           person.name = newName;
13         }
14         return person;
15       })
```

SOLVING THE TODO LIST

What we know

1. Create the structure
2. Modify states
3. Reading forms in vanilla

What we do not know

1. submit event? `<form onSubmit={} />`
2. Unique id's (UUID) or just `Math.random()`

+3/4

1. Re-create the ToDo list
2. Go back to the animal base, implement adding, removing, modifying aninals