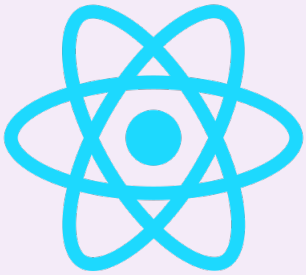


Composition

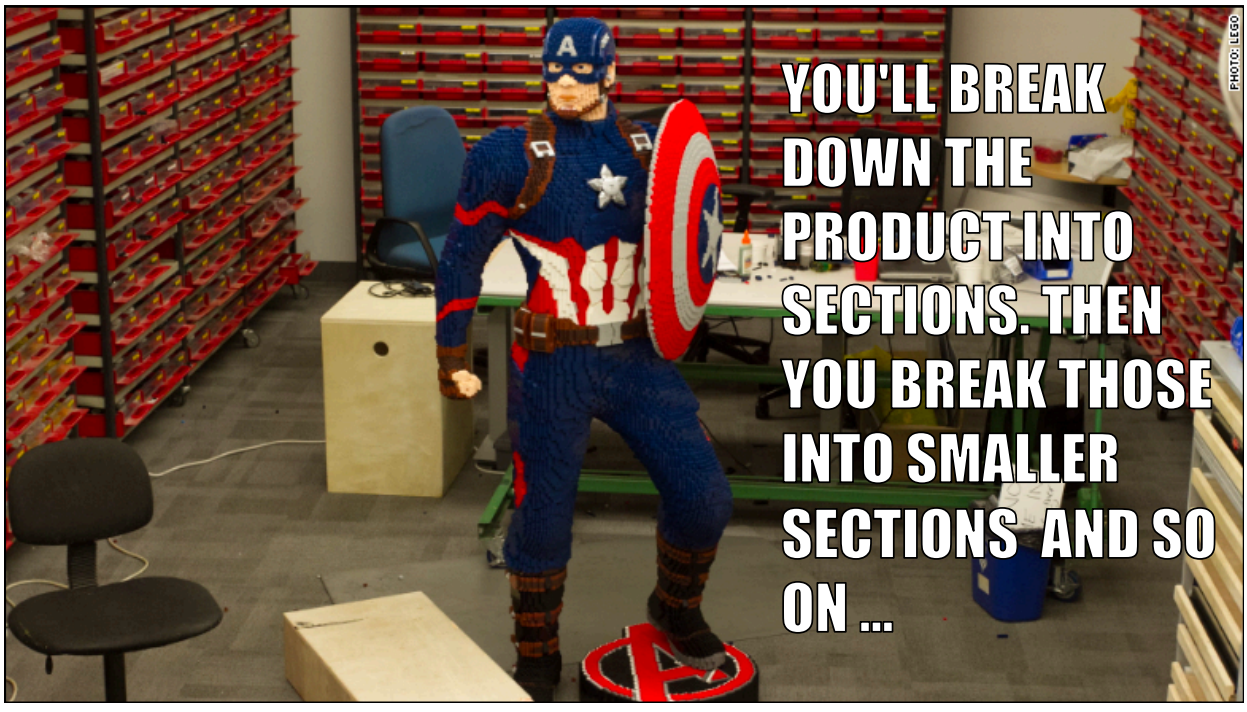
tl;dr

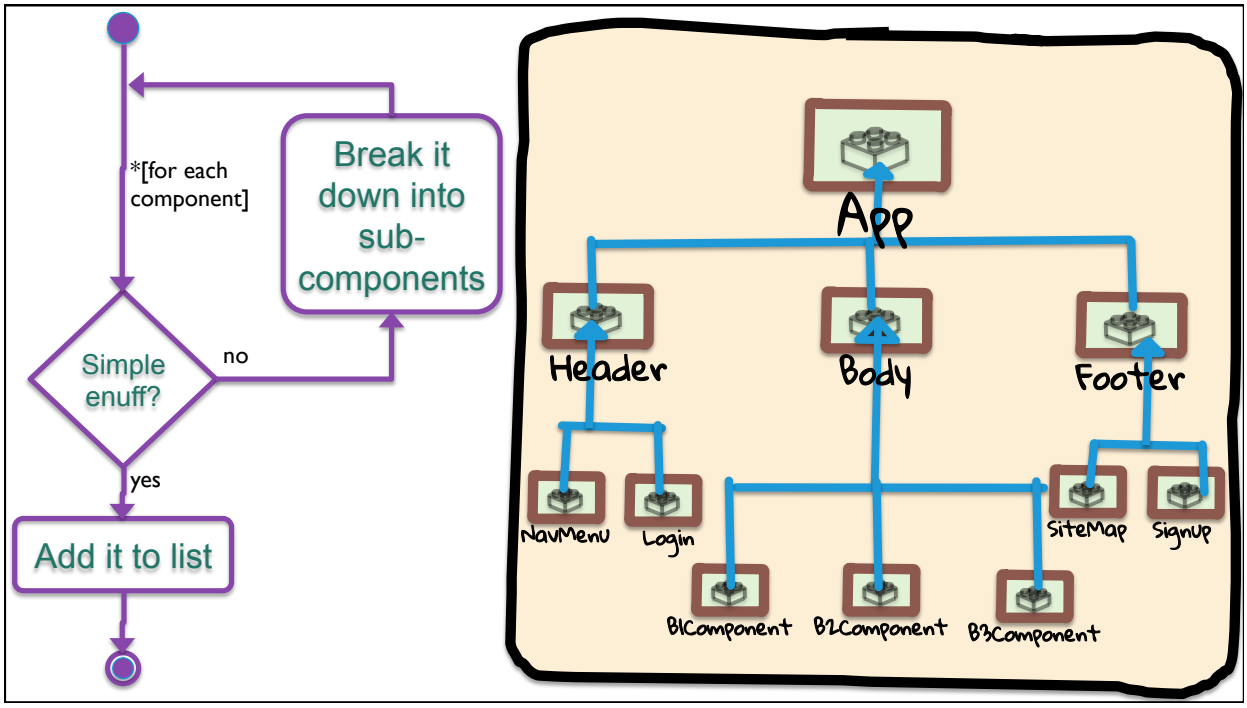
- React is centered around component composition
- To compose, we put the inner component's tag in the outer component's JSX
- To pass data down, write the values as attributes in the host and read them in *props* in the inner where they're immutable
- We cannot pass data back up but we can pass a function from the host to the inner where we pass params into that function
- To talk between other components, you pass data down from the lowest common ancestor and events up through that same ancestor

With React, you'll no longer write pages; you'll write components

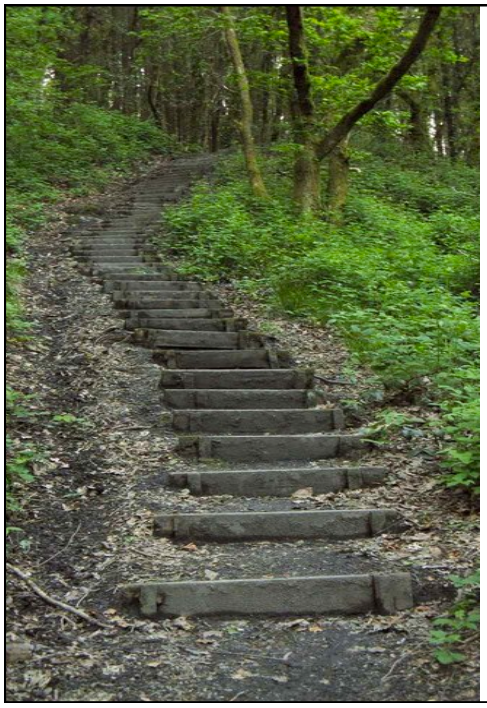
Each component is self-contained and encapsulated

- Even styles are local; CSS no longer cascades through components





How to compose



How to compose

1. Put a <tag> in the host component's JSX
2. If you want data to flow to the inner, use props
3. If you want data to flow up, bind an event

Remember: inner components must be imported into the host component

1. Add a tag

CompanyDirectory.js

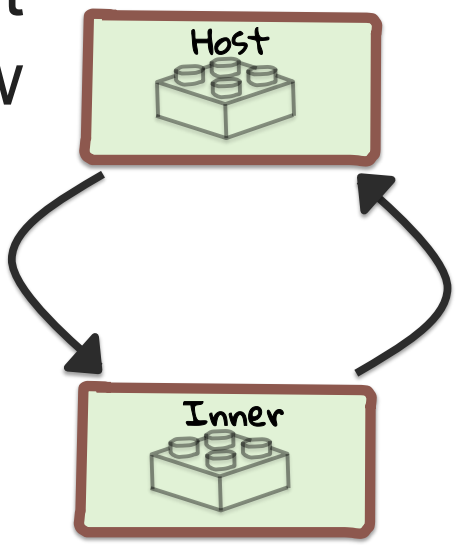
```
import {Person} from './Person';
function CompanyDirectory() {
return <div>
  <Person />
  </div>
}
```

Person.js

```
function Person() {
 return <section>
  <p>I am a person.</p>
 </section>
}
```

A word about data flow

We may want data to flow from host component to inner component and back up again.



props

props are basically the input parameters of a component.

How to pass data from host to inner

CompanyDirectory.js

```
function CompanyDirectory() {
 const [p1,p2] = get2Users();
 return <div>
  <Person hairColor="red" eyeColor="blue"
          first={p1.first} last={p1.last}
          imgSrc={p1.imgSrc} />
  <Person hairColor="brunette"
          eyeColor="brown" imgSrc="noImage.jpg"
          first={p2.first} last={p2.last} />
  </div>;
}
```

To read data in inner from host

Person.js

```
function Person(props) {
 const { first, last, hairColor, eyeColor } = props;

 return <section>
  <p>{first} has {hairColor} hair and {eyeColor} eyes.</p>
  <img src={props.imgSrc} alt={first + " " + last} />
  </section>
}
```

You always read the data with *props*.

Props are immutable (kind of)

- To add something to the props object is an error

```
TypeError: Cannot add property foo, object is
not extensible
```

- To reassign the value of a prop is an error. They're read-only.

```
TypeError: Cannot assign to read only property
'first' of object '#<Object>'
```

Full disclosure: if the prop is an object, the properties of that object can be changed, just not the prop itself

But what if I have data that needs to change? Like based on user input or Ajax calls or something?

- Well, that is not props. It's called state
- Coming soon to a lecture near you!

Passing data back up

From child to parent

3. Pass data up from inner to host

- For performance reasons, data cannot flow up.
- But we can pass a function to an inner to be invoked.
- If this function receives parameters, they will be seen in the host.

3 steps to get data to a host from an inner

In the host (aka parent)	In the inner (aka child)

1. Define a function
• Parameter(s) receive the data needed from the inner
`function funcName(p1, p2) {`
`  // Do stuff with p1 & p2`
`}`
2. Pass that function to the child as a prop
`<inner foo={funcName} />`
	3. Call the function, passing in the data
	`<someElement`
	`onClick={()=>props.foo(a,b)}>`

For example ...

- You have a task list component which is composed of a bunch of task components.

Q: Where is the list?

A: TaskList.js

Q: So where must the "delete" function live?

A: TaskList.js

Q: How will delete know which one to delete?

A: We can give it a task id.

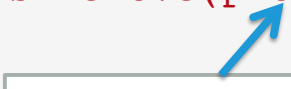
Q: Where is the button to delete a task?

A: Task.js

So how does Task.js tell TaskList,js (the host) what is the task id?

Task.js

```
export function Task(props) {
 return (
  <li>
   <button
    onClick={()=>props.remove(props.id)}
   >✓</button>
   {props.text}
  </li>
 );
}
```



...And emit the event with the value you want to send up.

To send data from inner to host

TaskList.js

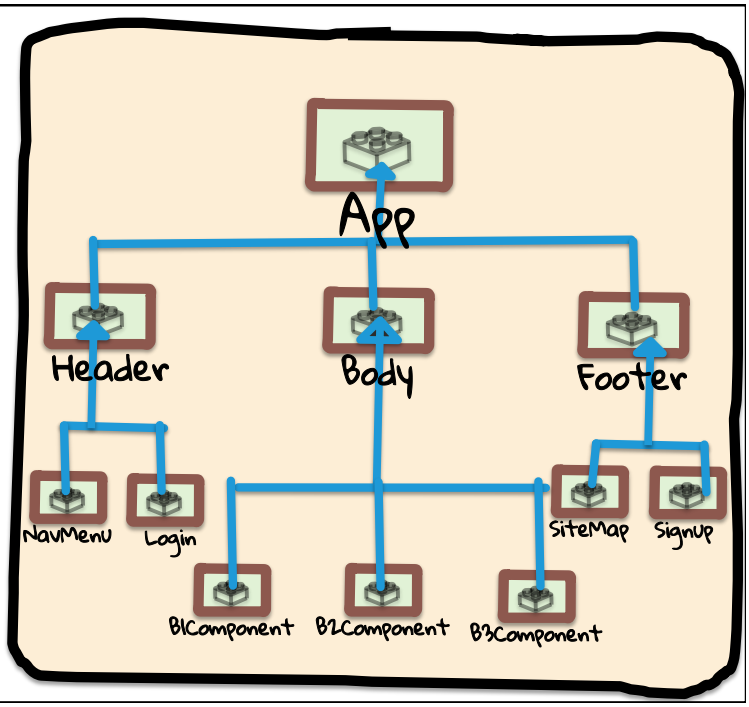
```
function TaskList {
 function delete(taskId){
  tasks=tasks.filter(t=>t.id!==taskId);
 };
 return (
  <div>
   <Task text="buy food" id=10
         remove={delete} />
  </div>
 );
}
```

When the remove event fires, run the delete() method

To read data in host from inner

Between siblings, cousins, and other distant relatives

- Just combine the parents and children techniques.
- Push up using events until you reach the lowest common ancestor, then push down using props



There <u>are</u> other ways to communicate

Global variables	Context (pass *this* down into a child)	refs (not a clean way. Almost non-used and confusing)	The cleanest way is to use a state container like Redux!

tl;dr

- React is centered around component composition
- To compose, we put the inner component's tag in the outer component's JSX
- To pass data down, write the values as attributes in the host and read them in *props* in the inner where they're immutable
- We cannot pass data back up but we can pass a function from the host to the inner where we pass params into that function
- To talk between other components, you pass data down from the lowest common ancestor and events up through that same ancestor