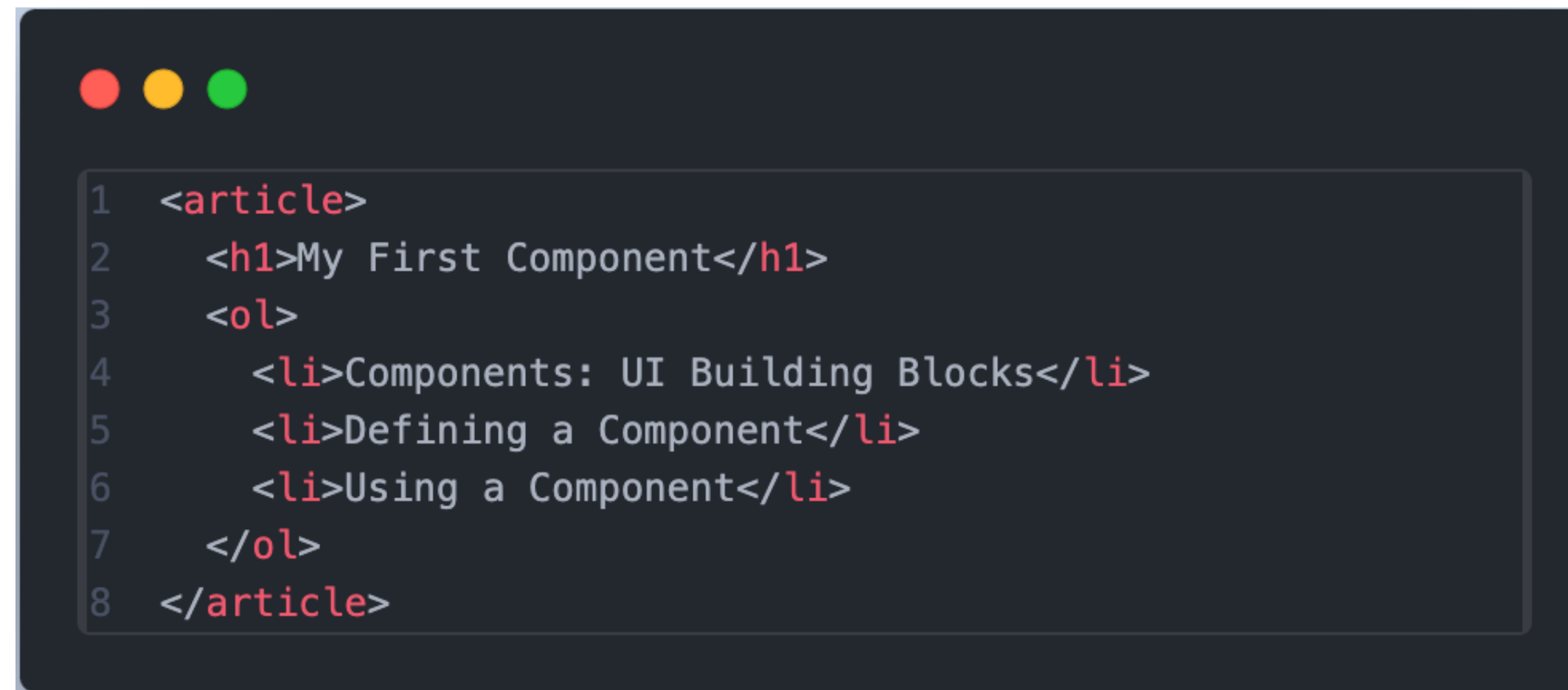


[2] COMPONENTI

REACT for DUMMIES

Istituto Tecnico Tecnologico 'Blaise Pascal' @ Cesena

Nicholas Magi - <nicholas.magi24[@]gmail.com>



```
1 <article>
2   <h1>My First Component</h1>
3   <ol>
4     <li>Components: UI Building Blocks</li>
5     <li>Defining a Component</li>
6     <li>Using a Component</li>
7   </ol>
8 </article>
```

Content **WITHOUT** React

Insieme di codici markup che possono essere ‘cosparsi di *interattività*’

Content **WITH** React

Insieme di funzionalità che possono essere ‘cosparse di *markup*’

Web Development **tradizionale**

```
<div>  
  <p></p>  
  <form>  
  </form>  
</div>
```

HTML

```
isLoggedIn() {...}  
onClick() {...}  
onSubmit() {...}
```

JavaScript

Web Development **w/React**

```
Sidebar() {  
  if (isLoggedIn()) {  
  
    <p>Welcome</p>  
  
  } else {  
  
    <Form />  
  
  }  
}
```

Sidebar.jsx

JSX = JS + HTML



Naming convention dei componenti: **PascalCase**

Senza **lettera maiuscola** iniziale non funzionano!

Markup in JSX

Regole per la scrittura di **markup** in JSX

#1 Restituire sempre e solo **1 tag root**

Utilizzo dei **tag HTML**

```
1  return <div>
2    <h1>Hedy Lamarr's Todos</h1>
3    
8    <ul>
9      ...
10   </ul>
11 </div>
```

Utilizzo dei **Frammenti**

```
1  return <>
2    <h1>Hedy Lamarr's Todos</h1>
3    
8    <ul>
9      ...
10   </ul>
11 </>
```

Regole per la scrittura di **markup** in JSX

#2 Chiudi sempre **qualsiasi** tag HTML

**** → ****

**** → ****

Regole per la scrittura di **markup** in JSX

#3 Usa il **camelCase** *quasi* sempre!

```
<iframe  
  class="..."  
  width="..."  
  height="..."  
  src="..."  
  title="..."  
  frameborder="..."  
  allow="..."  
  referrerpolicy="..."  
  allowfullscreen>  
</iframe>
```



```
<iframe  
  className="..."  
  width="..."  
  height="..."  
  src="..."  
  title="..."  
  frameBorder="..."  
  allow="..."  
  referrerPolicy="..."  
  allowFullScreen=""  
/>
```

Regole per la scrittura di **markup** in JSX

#3 Usa il **camelCase** *quasi* sempre!

Pitfall

For historical reasons, `aria-*` and `data-*` attributes are written as in HTML with dashes.

Perchè?

JavaScript Object

```

  $$typeof: Symbol(react.element)
  key: null
  ▼ props:
    ▼ children: Array(3)
      ► 0: {$$typeof: Symbol(react.element), type: 'h1', key: null, ref: null, props: {...},
      ► 1: {$$typeof: Symbol(react.element), type: 'img', key: null, ref: null, props: {...},
      ► 2: {$$typeof: Symbol(react.element), type: 'p', key: null, ref: null, props: {...}, ...
        length: 3
      ► [[Prototype]]: Array(0)
      ► [[Prototype]]: Object
    ref: null
    type: Symbol(react.fragment)
  ► _owner: FiberNode {tag: 0, key: null, stateNode: null, elementType: f, type: f, ...}
  ► _store: {validated: false}
  _self: undefined

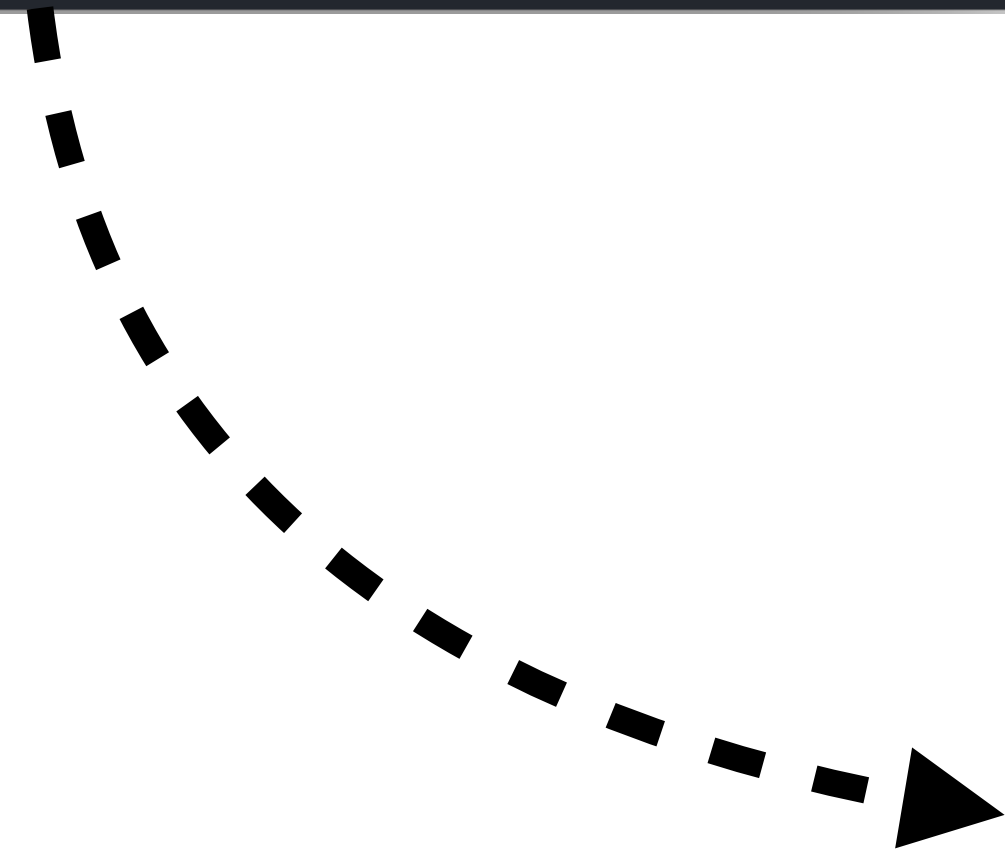
```

```

1  export default function ToDoList() {
2    // [...]
3    return <>
4      <h1>Hedy Lamarr's Todos</h1>
5      
10     <ul>
11       ...
12     </ul>
13   </>
14 }

```

JSX



JavaScript in JSX



```
1  export default function ContactCard() {  
2    let contact = {  
3      name: "Nicholas",  
4      surname: "Magi",  
5      year_of_birth: 2004,  
6      phone: {  
7        prefix: "+39",  
8        number: "4527184048"  
9      },  
10     email: "nicholas.magi24@gmail.com"  
11   }  
12  
13   // return [...]  
14 }
```

```
1 export default function ContactCard() {
2   let contact = {
3     // [...]
4   }
5
6   return <>
7     <h1 style={{ color: "red", backgroundColor: "gray", fontSize: "30px"}}>
8       {contact.name.toUpperCase()}'s information
9     </h1>
10    <ul>
11      {/* - Nicholas Magi */}
12      <li style={{ fontWeight: "bold"}}>{contact.name} {contact.surname}</li>
13
14      {/* - 20 years old */}
15      <li>{calculateAge(contact.year_of_birth)} years old</li>
16
17      {/* - Mobile: +39-4527184048 */}
18      <li>Mobile: {contact.phone.prefix}-{contact.phone.number}</li>
19
20      {/* - Email: nicholas.magi24@gmail.com */}
21      <li>Email: {contact.email}</li>
22    </ul>
23  </>
24 }
```


**Passaggio di props
ai componenti**

#1 - Definire direttamente un **nome** per le **props**

```
1 export function ContactCard({ contact }) {
2
3   return <>
4     <h1 style={{ color: "red", backgroundColor: "gray", fontSize: "30px"}}>
5       {contact.name.toUpperCase()}'s information
6     </h1>
7     <ul>
8       {/* - Nicholas Magi */}
9       <li style={{ fontWeight: "bold"}}>{contact.name} {contact.surname}</li>
10
11      {/* - 20 years old */}
12      <li>{calculateAge(contact.year_of_birth)} years old</li>
13
14      {/* - Mobile: +39-4527184048 */}
15      <li>Mobile: {contact.phone.prefix}-{contact.phone.number}</li>
16
17      <li>Email: {contact.email}</li>
18    </ul>
19  </>
20 }
```

```
1 export function AboutUs() {
2   let contacts =
3     [
4       name: "Nicholas",
5       surname: "Magi",
6       year_of_birth: 2004,
7       phone: {
8         prefix: "+39",
9         number: "4527184048"
10      },
11       email: "nicholas.magi24@gmail.com"
12     ],
13     {
14       name: "David",
15       surname: "Veneti",
16       year_of_birth: undefined,
17       phone: {
18         prefix: "+39",
19         number: "7858368280"
20      },
21       email: "david.veneti@ispascalcomadini.it"
22     },
23   ]
24
25   return <>
26     <ContactCard contact={contacts[0]}/>
27     <hr/>
28     <ContactCard contact={contacts[1]}/>
29   </>
30 }
```


#2 - Utilizzare un nome generico - 'props'



```
1  function Avatar(props) {  
2    let person = props.person;  
3    let size = props.size;  
4    // ...  
5  }
```

N.B.

Disabilita l'IntelliSense

<https://react.dev/learn/thinking-in-react>

Thinking in React

☐ Only show products in stock**Name****Price****Fruits**

Apple	\$1
-------	-----

Dragonfruit	\$1
-------------	-----

Passionfruit	\$2
--------------	-----

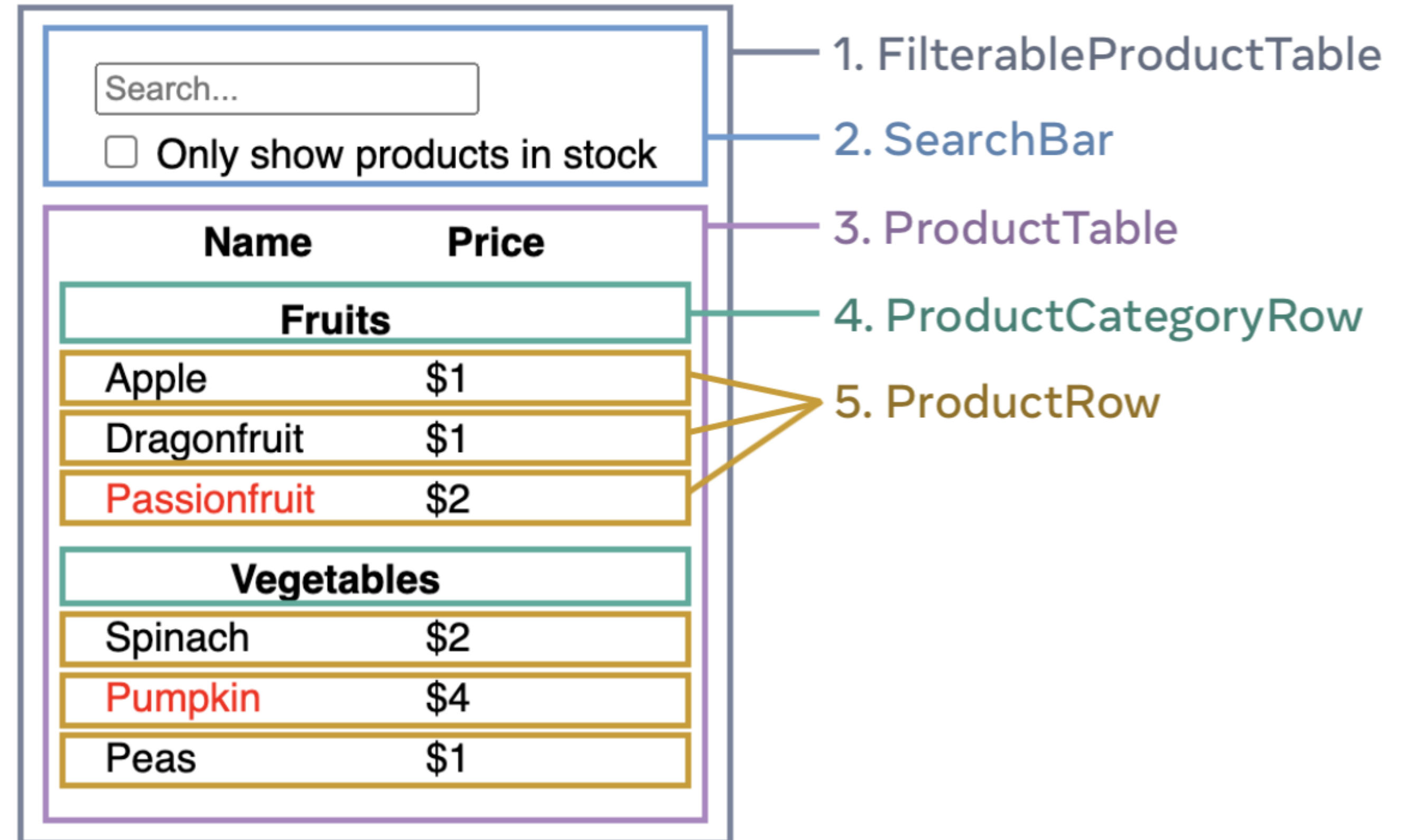
Vegetables

Spinach	\$2
---------	-----

Pumpkin	\$4
---------	-----

Peas	\$1
------	-----

#1 Break the UI into a component hierarchy



App.js Download Reset Fork

```
1 function ProductCategoryRow({ category }) {
2   return (
3     <tr>
4       <th colspan="2">
5         {category}
6       </th>
7     </tr>
8   );
9 }
10
11 function ProductRow({ product }) {
12   const name = product.stocked ? product.name :
13     <span style={{ color: 'red' }}>
14       {product.name}
15     </span>;
16 }
```

Search...

☐ Only show products in stock

Name	Price
Fruits	
Apple	\$1
Dragonfruit	\$1
Passionfruit	\$2
Vegetables	
Spinach	\$2
Pumpkin	\$4
Peas	\$1

▼ Show more

#2

Build a static version in React

#3

**Identify where
your state should live**

#4

Find the **minimal** but **complete**
representation of **UI state**

#5

Add *inverse* data flow