

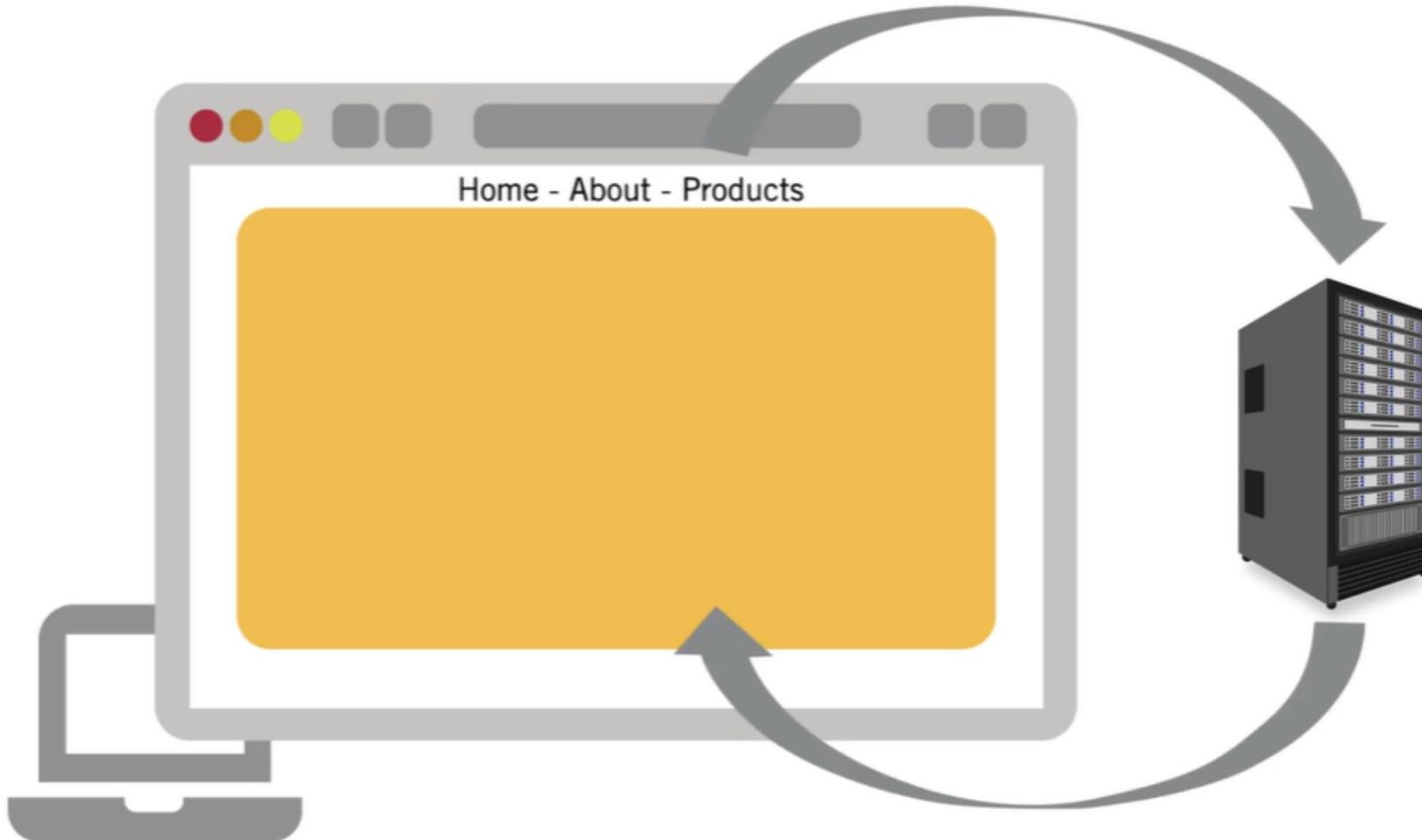


Professional Certificate in Coding: Full Stack Development with MERN: Week 18

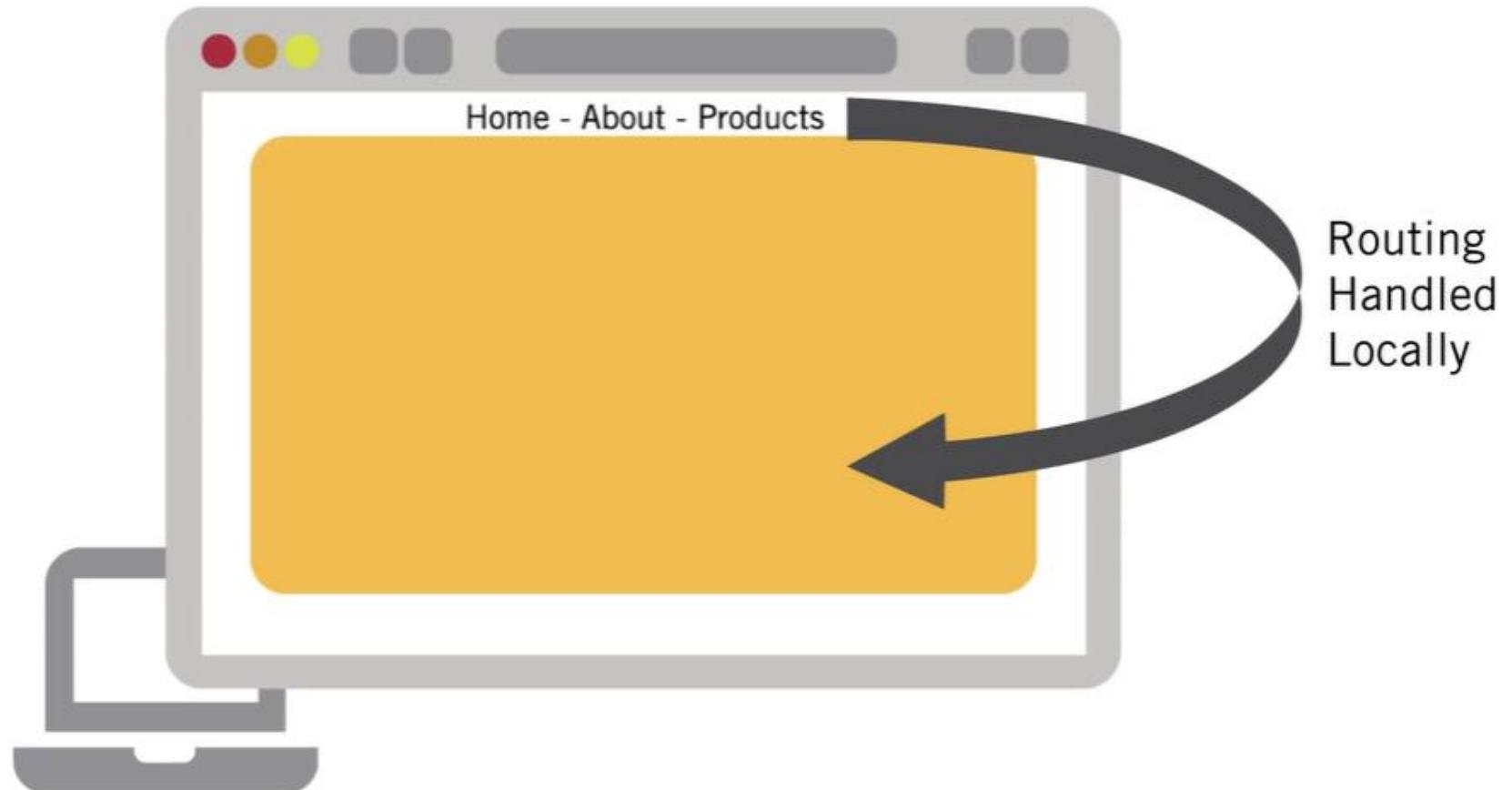
---

**Bad Bank Exercise**

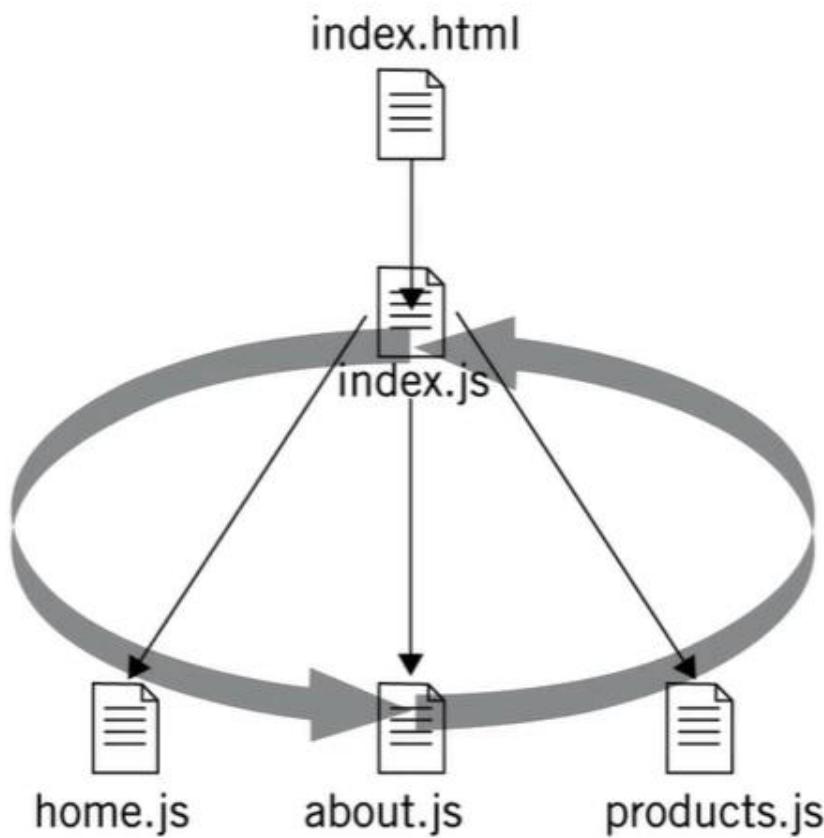
# History – Server Request Per Page



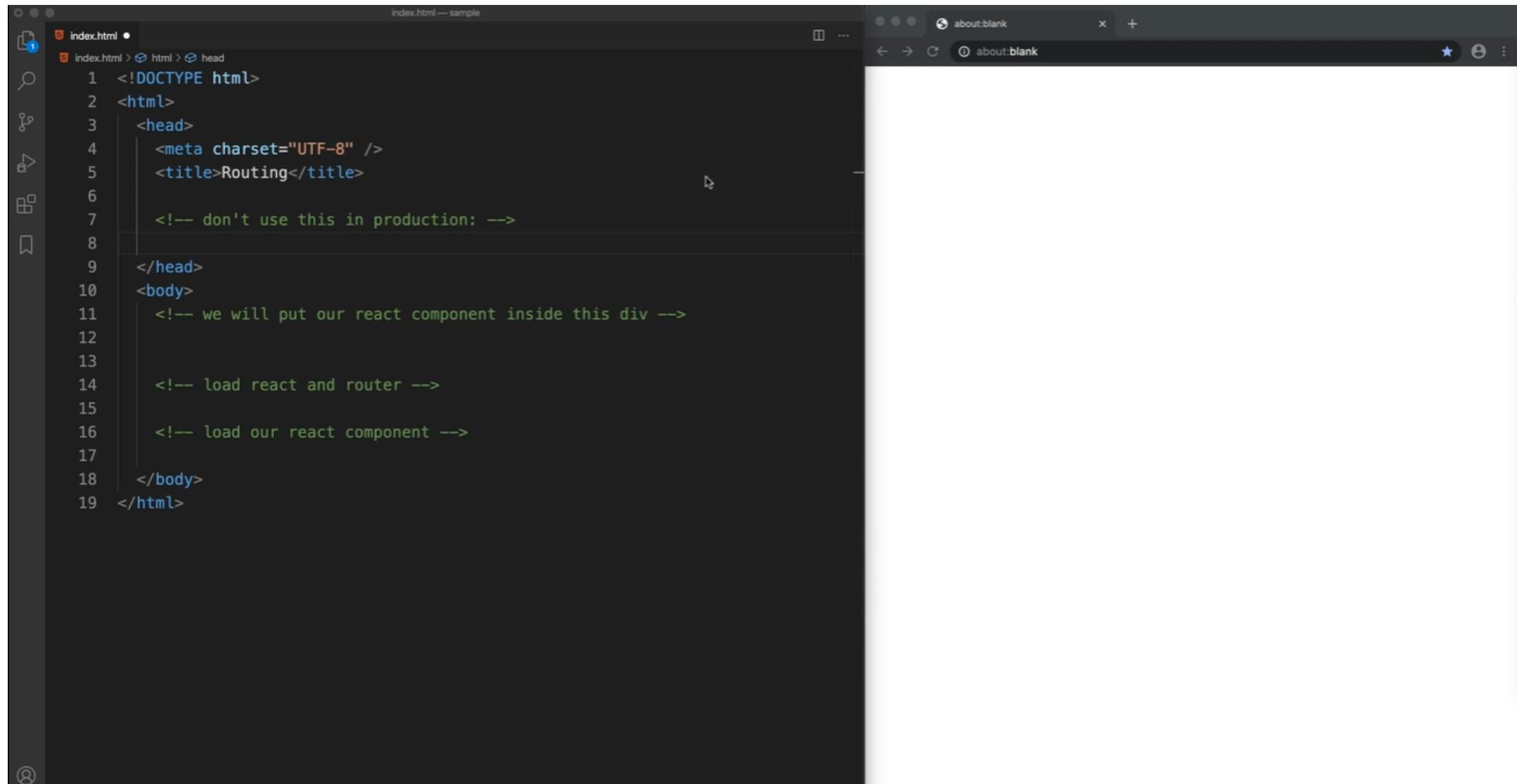
# SPA – Single Page Application – Local Routing



# Routing



# Routing: Demonstration (1/19)

A screenshot of a code editor and a web browser. The code editor on the left shows the file 'index.html' with the following content:

```
index.html — sample
index.html •
index.html > ↗ html > ↗ head
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8" />
5      <title>Routing</title>
6
7      <!-- don't use this in production: -->
8
9    </head>
10   <body>
11     <!-- we will put our react component inside this div -->
12
13
14     <!-- load react and router -->
15
16     <!-- load our react component -->
17
18   </body>
19 </html>
```

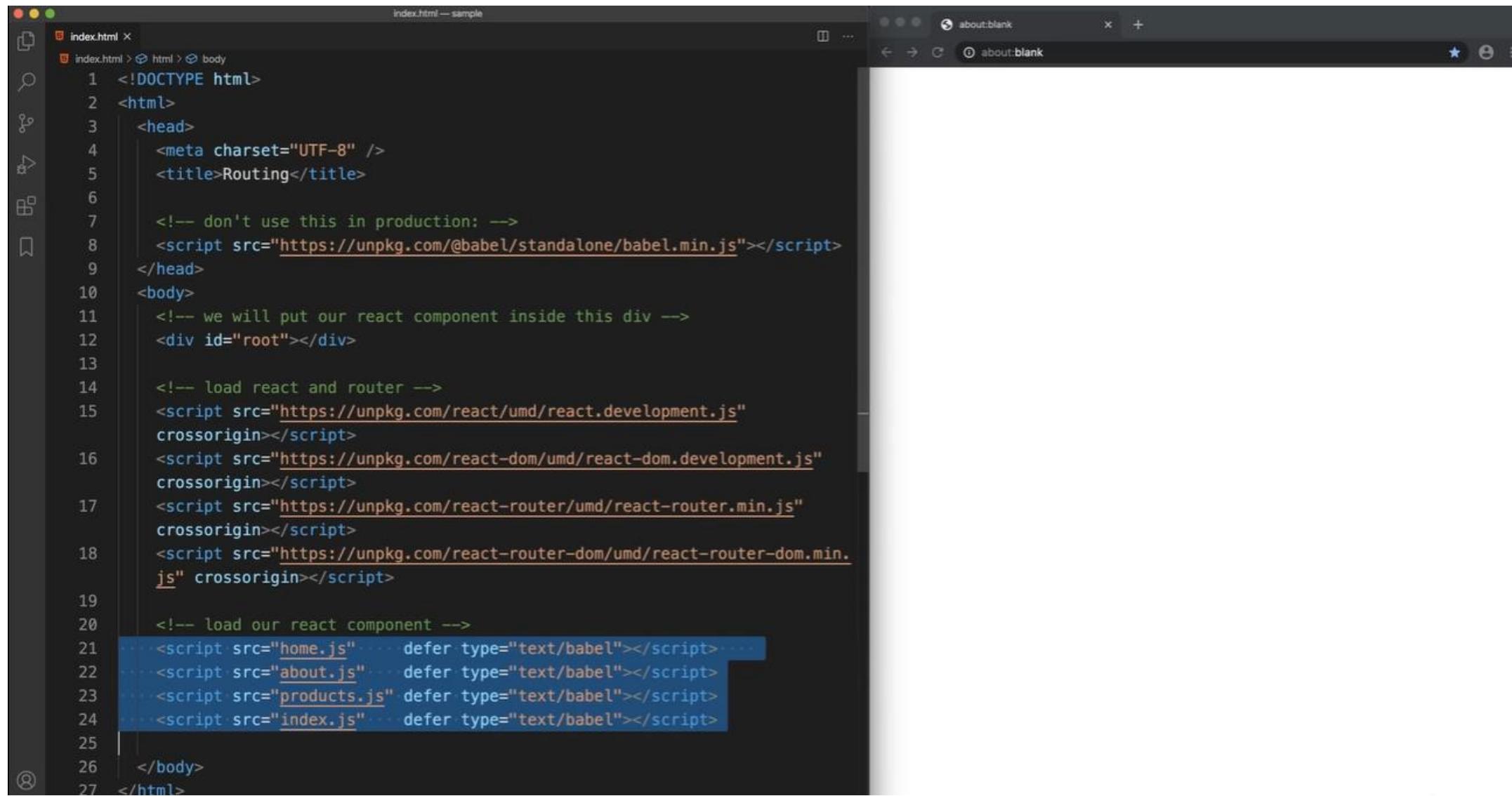
The browser window on the right shows a blank white page with the title 'about:blank'. The browser's address bar also displays 'about:blank'.

The code editor shows the file 'index.html' with the following content:

```
index.html — sample
index.html •
index.html > ↗ html > ↗ head
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8" />
5      <title>Routing</title>
6
7      <!-- don't use this in production: -->
8
9    </head>
10   <body>
11     <!-- we will put our react component inside this div -->
12
13
14     <!-- load react and router -->
15
16     <!-- load our react component -->
17
18   </body>
19 </html>
```

The browser window shows a blank white page with the title 'about:blank'.

# Routing: Demonstration (2/19)

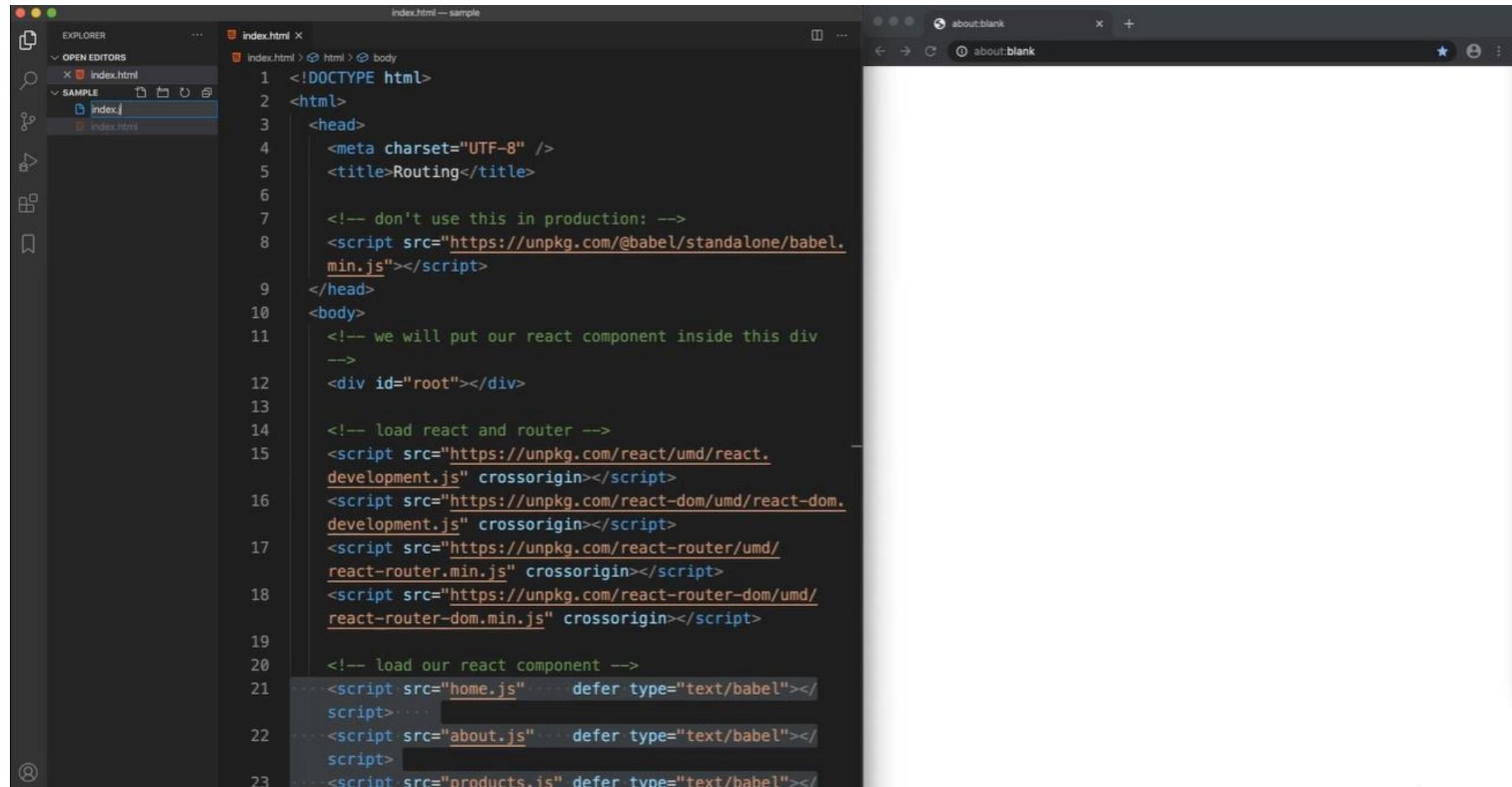


The image shows a code editor on the left and a browser window on the right. The code editor displays the file `index.html` with the following content:

```
index.html — sample
index.html > html > body
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8" />
5      <title>Routing</title>
6
7      <!-- don't use this in production: -->
8      <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
9    </head>
10   <body>
11     <!-- we will put our react component inside this div -->
12     <div id="root"></div>
13
14     <!-- load react and router -->
15     <script src="https://unpkg.com/react/umd/react.development.js"
16           crossorigin></script>
16     <script src="https://unpkg.com/react-dom/umd/react-dom.development.js"
17           crossorigin></script>
17     <script src="https://unpkg.com/react-router/umd/react-router.min.js"
18           crossorigin></script>
18     <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.
19           js" crossorigin></script>
19
20     <!-- load our react component -->
21     <script src="home.js" defer type="text/babel"></script>
21     <script src="about.js" defer type="text/babel"></script>
22     <script src="products.js" defer type="text/babel"></script>
23     <script src="index.js" defer type="text/babel"></script>
24
25   </body>
26
27 </html>
```

The browser window on the right shows a blank page with the title "about:blank".

# Routing: Demonstration (3/19)



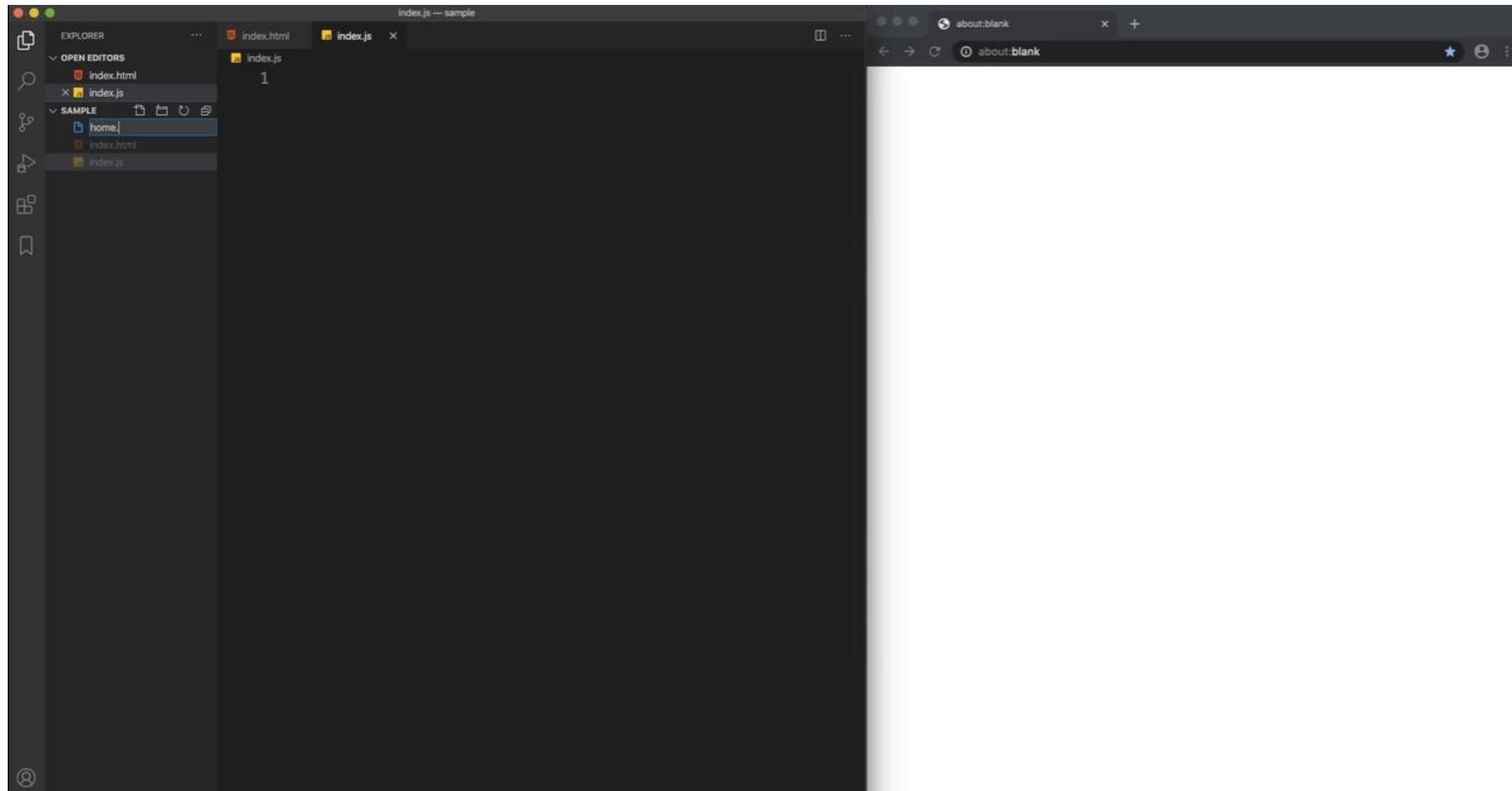
The image shows a code editor on the left and a web browser on the right. The code editor displays the file `index.html` with the following content:

```
index.html — sample
index.html
  index.html
  index.js
  index.html

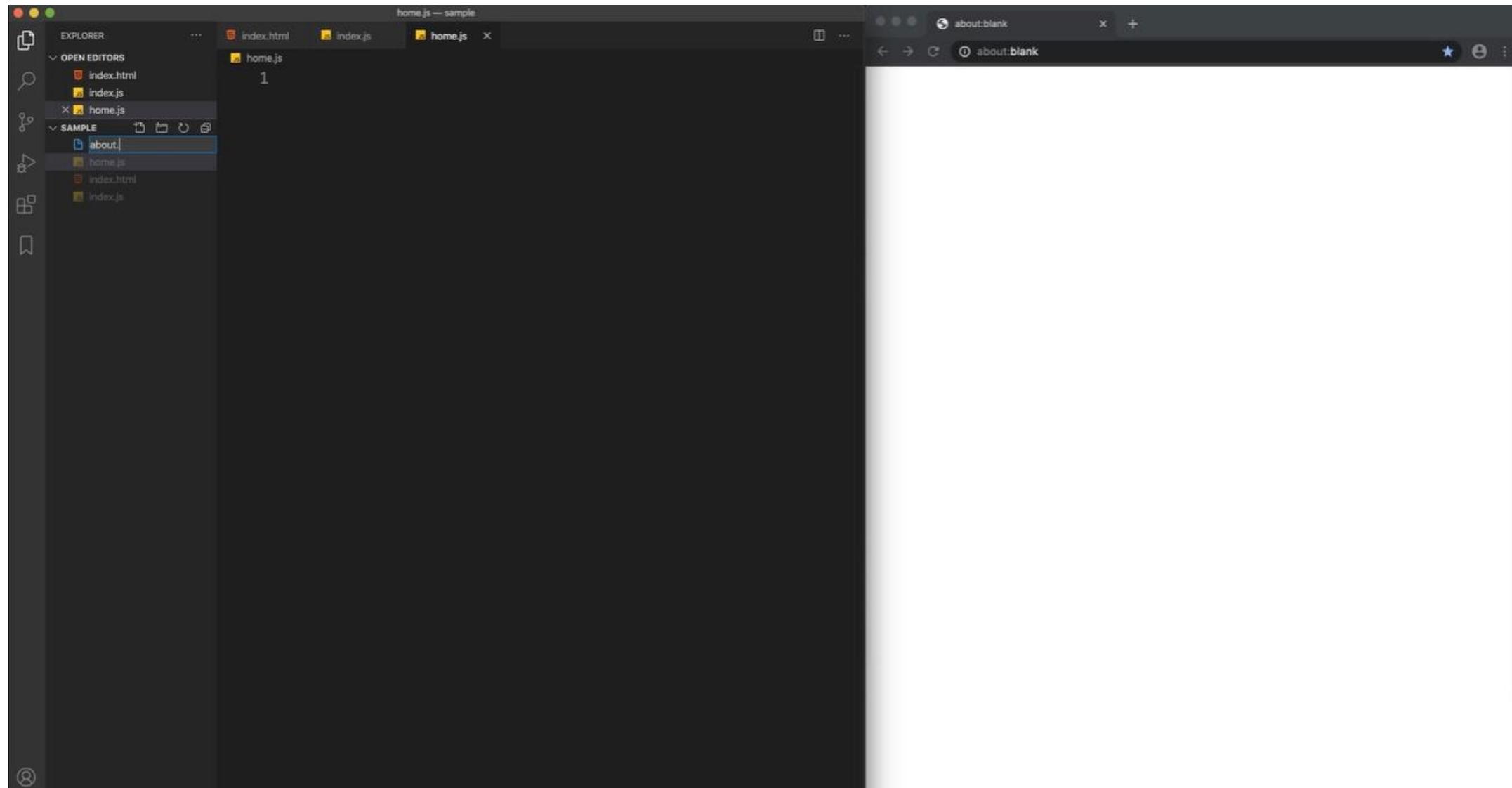
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8" />
5     <title>Routing</title>
6
7     <!-- don't use this in production: -->
8     <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
9   </head>
10  <body>
11    <!-- we will put our react component inside this div
12    -->
13    <div id="root"></div>
14
15    <!-- load react and router -->
16    <script src="https://unpkg.com/react/umd/react.development.js" crossorigin></script>
17    <script src="https://unpkg.com/react-dom/umd/react-dom.development.js" crossorigin></script>
18    <script src="https://unpkg.com/react-router/umd/react-router.min.js" crossorigin></script>
19    <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.js" crossorigin></script>
20
21    <!-- load our react component -->
22    <script src="home.js" defer type="text/babel"></script>
23    <script src="about.js" defer type="text/babel"></script>
24    <script src="products.js" defer type="text/babel"></script>
```

The browser window on the right shows a blank page with the title "about:blank".

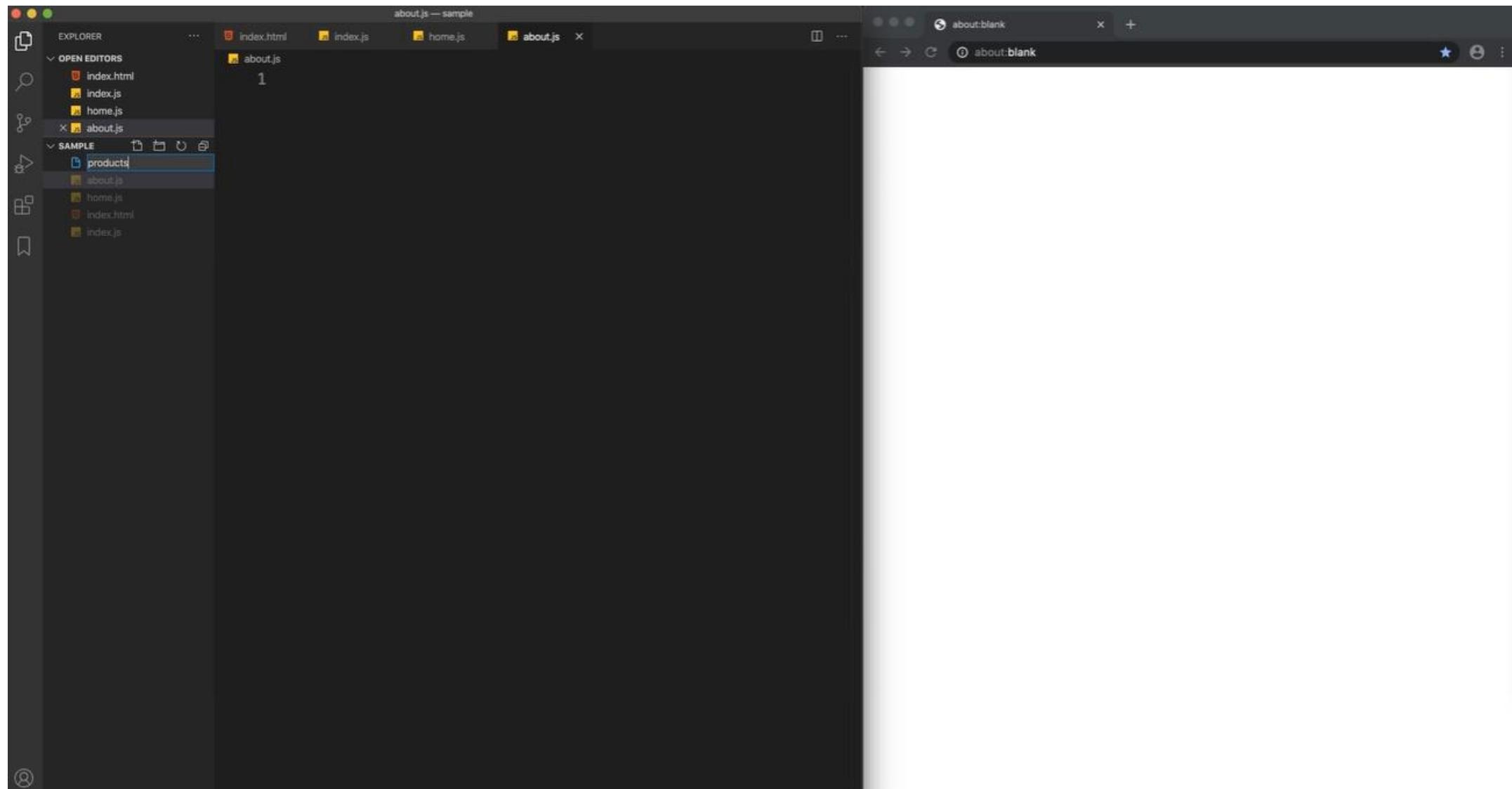
# Routing: Demonstration (4/19)



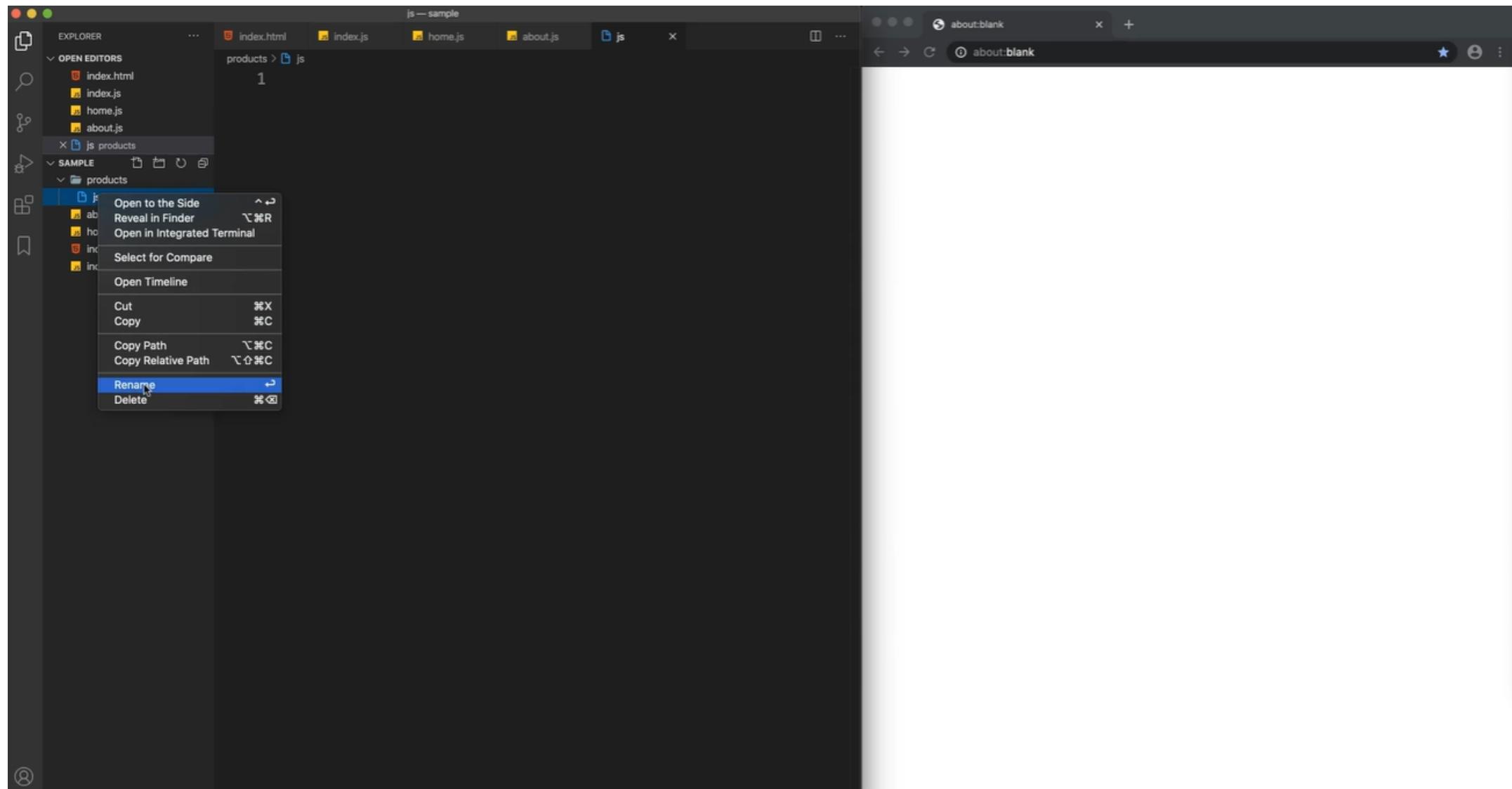
# Routing: Demonstration (5/19)



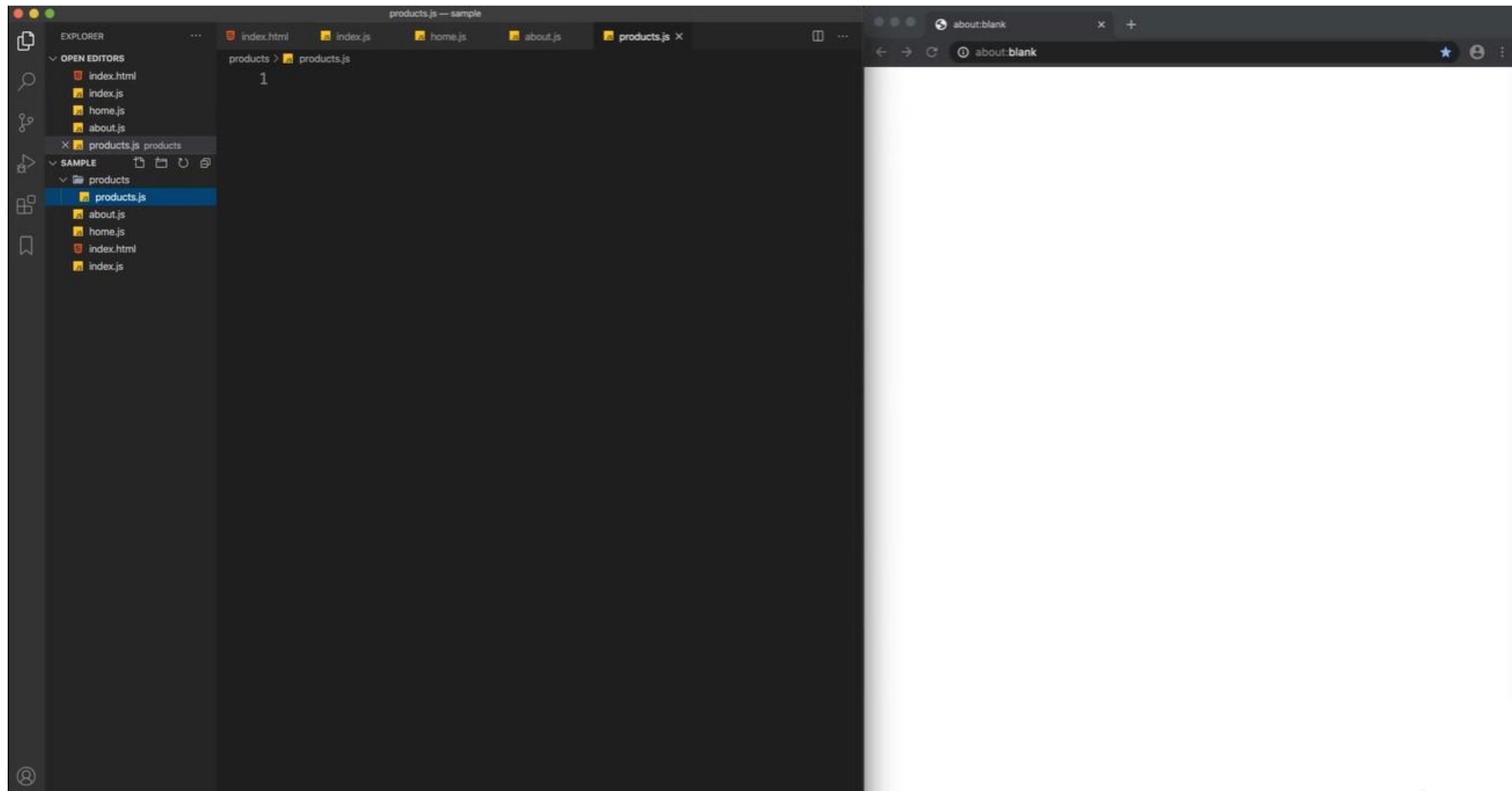
# Routing: Demonstration (6/19)



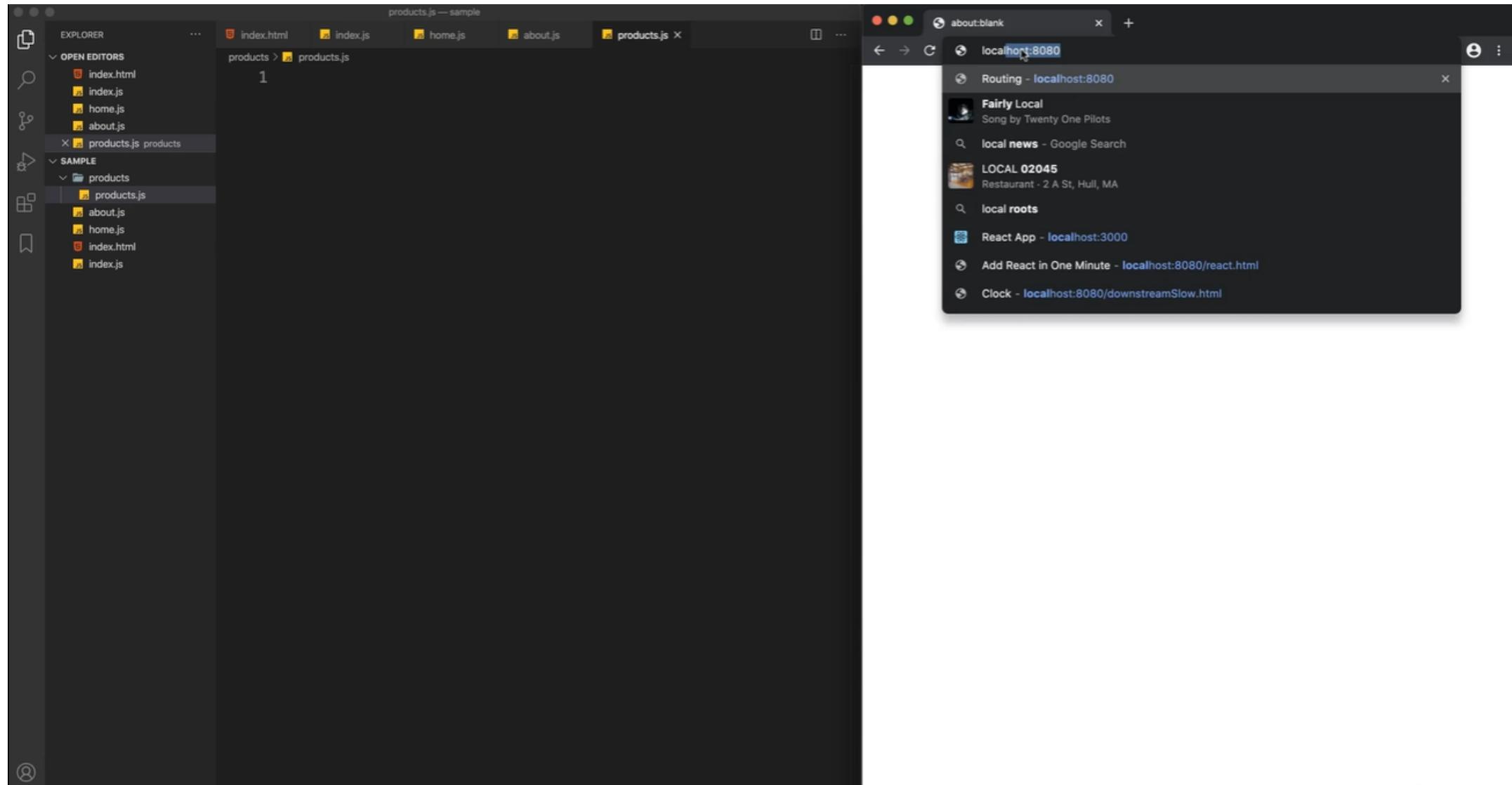
# Routing: Demonstration (7/19)



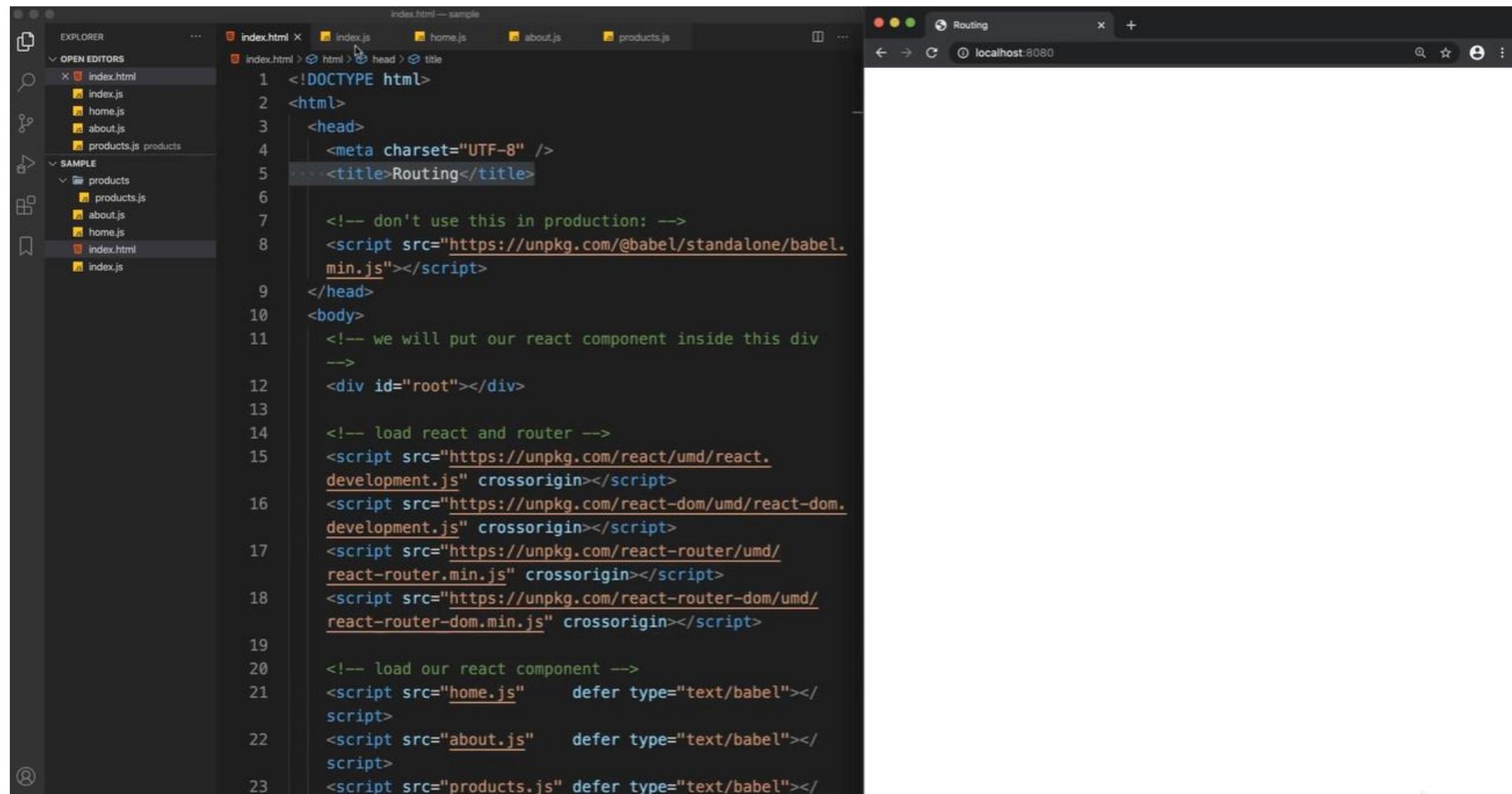
# Routing: Demonstration (8/19)



# Routing: Demonstration (9/19)



# Routing: Demonstration (10/19)



The image shows a code editor interface with two tabs open: 'index.html — sample' and 'index.html'. The left pane displays the file structure of a 'SAMPLE' folder containing 'products', 'about.js', 'home.js', and 'index.html'. The right pane shows the content of 'index.html'.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Routing</title>

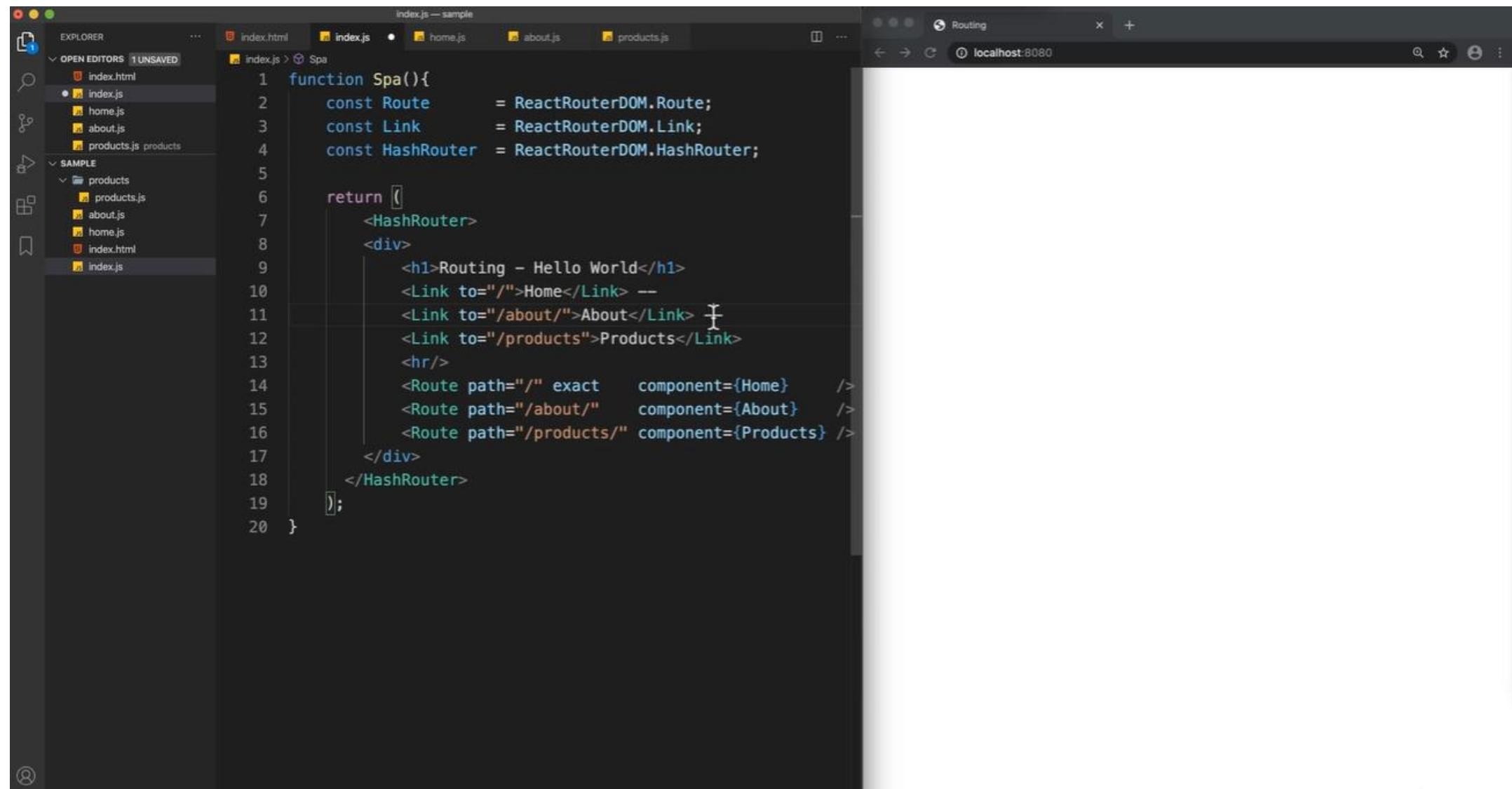
    <!-- don't use this in production: -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>
    <!-- we will put our react component inside this div
    -->
    <div id="root"></div>

    <!-- load react and router -->
    <script src="https://unpkg.com/react/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom/umd/react-dom.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-router/umd/react-router.min.js" crossorigin></script>
    <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.js" crossorigin></script>

    <!-- load our react component -->
    <script src="home.js" defer type="text/babel"></script>
    <script src="about.js" defer type="text/babel"></script>
    <script src="products.js" defer type="text/babel"></script>
```

The browser window titled 'Routing' shows the output at 'localhost:8080', displaying the title 'Routing' and the React component structure.

# Routing: Demonstration (11/19)



The screenshot shows a code editor interface with a dark theme. On the left, the Explorer sidebar lists files: index.html, index.js, home.js, about.js, products.js, and a folder named SAMPLE containing products.js, about.js, home.js, index.html, and index.js. The main editor area displays the following code:

```
index.js — sample
index.js > Spa
1 function Spa(){
2     const Route      = ReactDOM.Route;
3     const Link       = ReactDOM.Link;
4     const HashRouter = ReactDOM.HashRouter;
5
6     return [
7         <HashRouter>
8             <div>
9                 <h1>Routing - Hello World</h1>
10                <Link to="/">Home</Link> --
11                <Link to="/about/">About</Link> -
12                <Link to="/products">Products</Link>
13                <hr/>
14                <Route path="/" exact component={Home} />
15                <Route path="/about/" component={About} />
16                <Route path="/products/" component={Products} />
17            </div>
18        </HashRouter>
19    ];
20}
```

The browser tab at the top right is titled "Routing" and shows the URL "localhost:8080".

# Routing: Demonstration (12/19)

The screenshot displays a code editor interface with a dark theme. On the left, the Explorer sidebar shows a file tree with the following structure:

- OPEN EDITORS: index.html, index.js
- SAMPLE:
  - products:
    - products.js
  - about.js
  - home.js
  - index.html
  - index.js

The main editor area contains the file `home.js` with the following content:

```
function Home(){
  return (
    <div>
      <h3>Home Component</h3>
    </div>
  )
}
```

To the right of the editor is a browser window titled "Routing" with the URL "localhost:8080". The browser displays the text "Home Component" inside a single `<div>` element.

# Routing: Demonstration (13/19)

```
about.js — sample
index.html index.js home.js about.js products.js
about.js > About
1 function About(){
2     return [
3         <div>
4             <h3>About Component</h3>
5         </div>
6     ]
7 }
```

# Routing: Demonstration (14/19)

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists files: index.html, index.js, home.js, about.js, products.js (selected), and a folder named products. The main editor area contains the following code:

```
products.js — sample
products > products.js > Products
1 function Products(){
2     return [
3         <div>
4             <h3>Products Component</h3>
5         </div>
6     ]
7 }
```

To the right of the editor is a browser window titled "Routing" showing the result at "localhost:8080". The browser displays a simple page with the heading "Products Component" centered within a div.

# Routing: Demonstration (15/19)

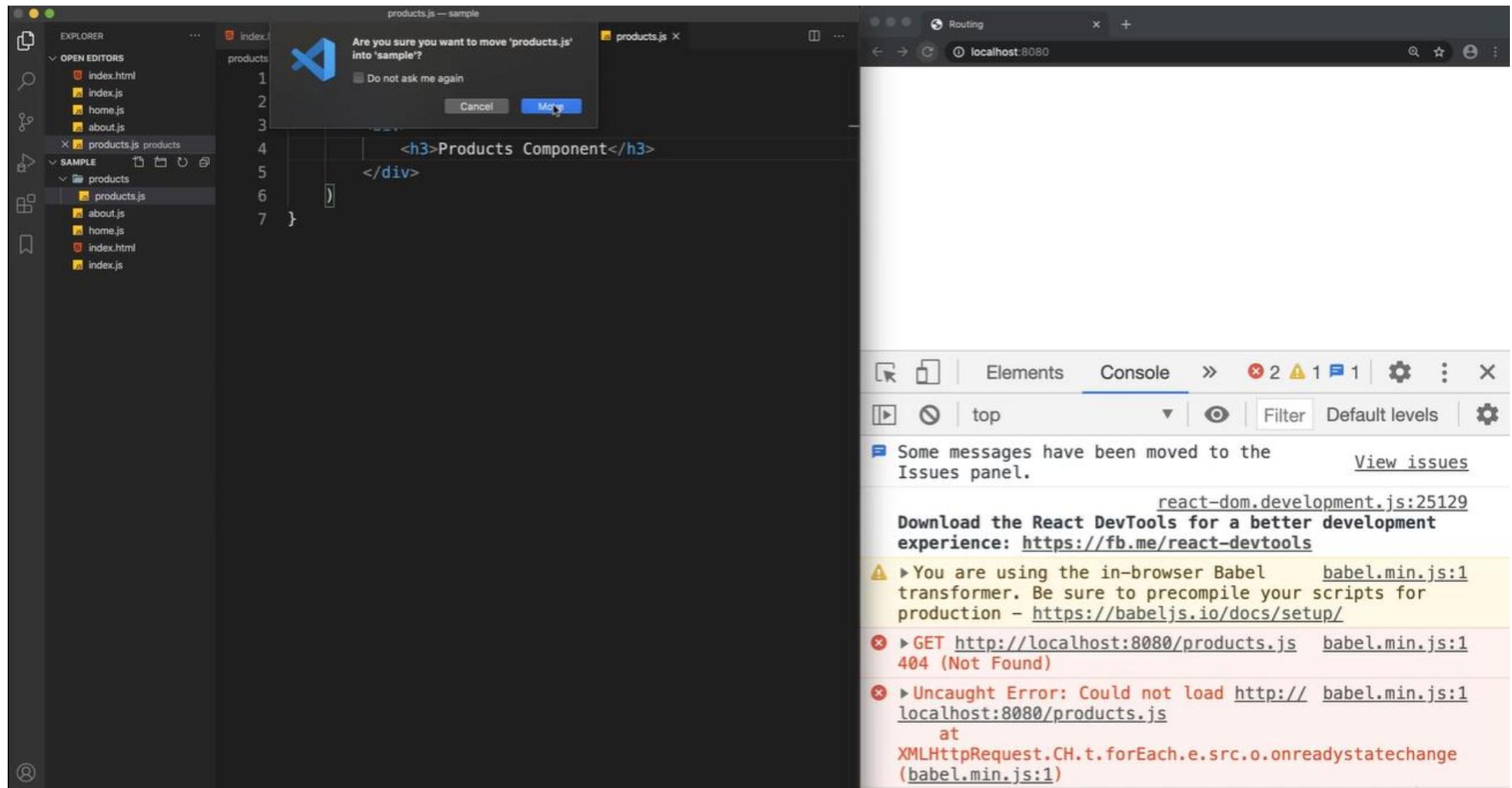
The screenshot shows a development environment with two main panes. On the left is a code editor (VS Code) displaying a file named `home.js` with the following content:

```
function Home(){
  return (
    <div>
      <h3>Home Component</h3>
    </div>
  )
}
```

The right pane shows a web browser window titled "Routing" with the URL `localhost:8080`. The browser displays a blank white page. Below the browser is the browser's developer tools console tab, which shows the following messages:

- A message indicating that some messages have been moved to the Issues panel, with a link to "View issues".
- A warning about using the in-browser Babel transformer, with a link to "babel.min.js:1" and "https://babeljs.io/docs/setup/".
- An error message: "Failed to load resource: the server :8080/products.js:1 responded with a status of 404 (Not Found)".
- An error message: "Uncaught Error: Could not load http://lo babel.min.js:1 calhost:8080/products.js at XMLHttpRequest.CH.t.forEach.e.src.o.onreadystatechange (babel.min.js:1)".

# Routing: Demonstration (16/19)



# Routing: Demonstration (17/19)

The image shows a code editor on the left and a browser window on the right.

**Code Editor (VS Code):**

- Explorer:** Shows files: index.html, index.js, home.js, about.js, products.js.
- Open Editors:** Shows index.js (active), index.html, home.js, about.js, products.js.
- Code:** index.js (sample) contains the following code:

```
index.js — sample
index.html index.js home.js about.js products.js
index.js > ...
1 function Spa(){
2     const Route      = ReactDOM.Route;
3     const Link       = ReactDOM.Link;
4     const HashRouter = ReactDOM.HashRouter;
5
6     return (
7         <HashRouter>
8             <div>
9                 <h1>Routing - Hello World</h1>
10                <Link to="/">Home</Link> --
11                <Link to="/about/">About</Link> --
12                <Link to="/products">Products</Link>
13                <hr/>
14                <Route path="/" exact component={Home} />
15                <Route path="/about/" component={About} />
16                <Route path="/products/" component={Products} />
17            </div>
18        </HashRouter>
19    );
20 }
21
22 ReactDOM.render(
23     <Spa/>,
24     document.getElementById('root')
25 )
```

**Browser:** Routing - Hello World

localhost:8080/#/

Routing - Hello World

Home --About --Products

---

Home Component

# Routing: Demonstration (18/19)

The image shows a development environment with two main panes. The left pane is a code editor displaying a file named `index.js` (under `sample`) which contains the following code:

```
function Spa(){
  const Route      = ReactDOM.Route;
  const Link      = ReactDOM.Link;
  const HashRouter = ReactDOM.HashRouter;

  return (
    <HashRouter>
    <div>
      <h1>Routing - Hello World</h1>
      <Link to="/">Home</Link> --
      <Link to="/about/">About</Link> --
      <Link to="/products">Products</Link>
      <hr/>
      <Route path="/" exact component={Home} />
      <Route path="/about/" component={About} />
      <Route path="/products/" component={Products} />
    </div>
  </HashRouter>
);
ReactDOM.render(
  <Spa/>,
  document.getElementById('root')
)
```

The right pane shows a web browser window titled "Routing" with the URL `localhost:8080/#/about/`. The page content is:

# Routing - Hello World

[Home](#) -- [About](#) -- [Products](#)

---

## About Component

# Routing: Demonstration (19/19)

The image shows a split-screen view. On the left is a code editor (VS Code) displaying a file named `index.js` with the following content:

```
index.js — sample
index.html index.js home.js about.js products.js
index.js > Spa

function Spa(){
  const Route = ReactRouterDOM.Route;
  const Link = ReactRouterDOM.Link;
  const HashRouter = ReactRouterDOM.HashRouter;

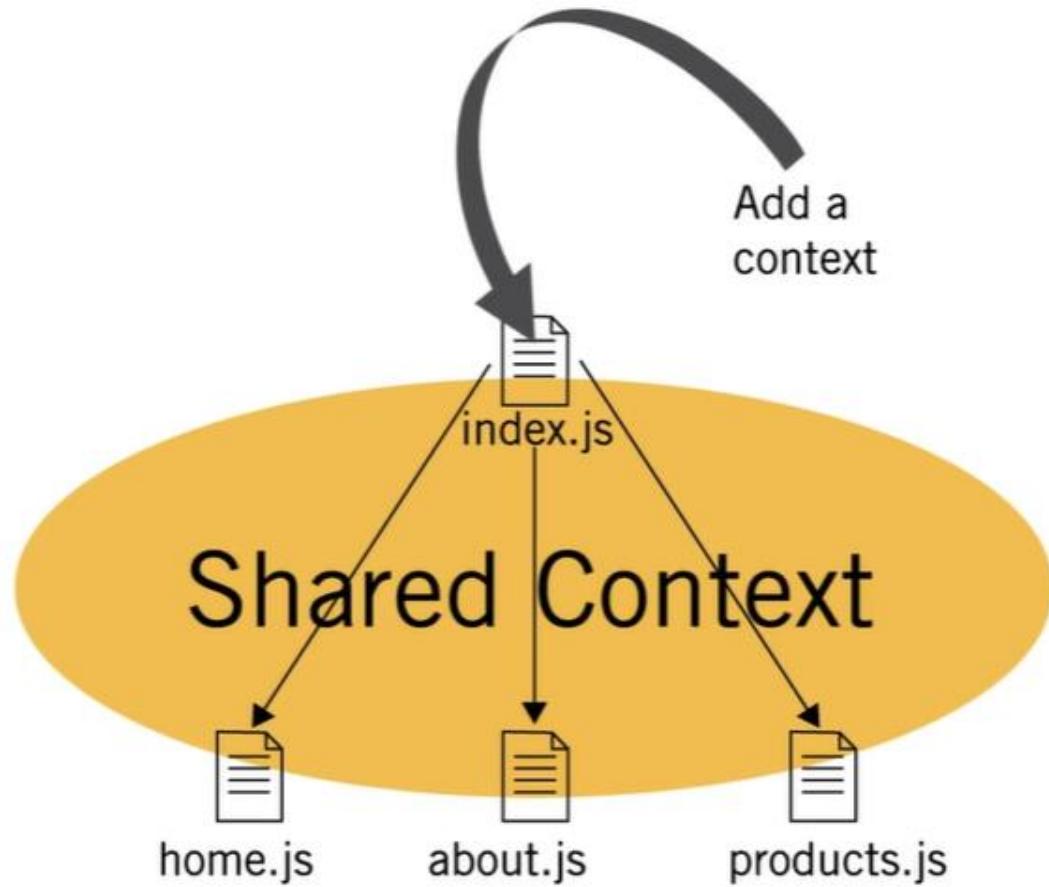
  return (
    <HashRouter>
      <div>
        <h1>Routing - Hello World</h1>
        <Link to="/">Home</Link> --
        <Link to="/about/">About</Link> --
        <Link to="/products">Products</Link>
        <hr/>
        <Route path="/" exact component={Home} />
        <Route path="/about/" component={About} />
        <Route path="/products/" component={Products} />
      </div>
    </HashRouter>
  );
}

ReactDOM.render(
  <Spa/>,
  document.getElementById('root')
)
```

The code defines a `Spa` function that sets up a `HashRouter` component. Inside the router, there's a `div` element containing an `h1` header and three `Link` components for "Home", "About", and "Products". Below the links is a horizontal line separator (`<hr/>`). Three `Route` components are defined to handle the root URL, the "/about/" URL, and the "/products/" URL, each rendering its respective component (`Home`, `About`, or `Products`).

On the right is a browser window titled "Routing" showing the result of running the application at `localhost:8080/#/products`. The page has a blue header with the text "Routing - Hello World". Below the header is a navigation bar with underlined links: "Home" (active), "About", and "Products". A horizontal line follows the links. Below the line is a section titled "Products Component".

# Shared Context



# Shared Context: Demonstration (1/6)

The image shows a code editor on the left and a browser window on the right. The code editor displays `index.js` with the following content:

```
index.js — sample
index.html index.js home.js about.js products.js
index.js > Spa
1 const UserContext = React.createContext(null);
2
3 function Spa(){
4     const Route      = ReactRouterDOM.Route;
5     const Link       = ReactRouterDOM.Link;
6     const HashRouter = ReactRouterDOM.HashRouter;
7
8     return (
9         <HashRouter>
10        <div>
11            <h1>Routing - Hello World</h1>
12            <Link to="/">Home</Link> --
13            <Link to="/about/">About</Link> --
14            <Link to="/products">Products</Link>
15            <hr/>
16            <UserContext.Provider value={{users:['peter']}}>
17                <Route path="/" exact component={Home} />
18                <Route path="/about/" component={About} />
19                <Route path="/products/" component={Products} />
20            </UserContext.Provider>
21        </div>
22    </HashRouter>
23 );
24 }
25
26 ReactDOM.render(
27     <Spa/>,
28     document.getElementById('root')
29 )
```

The browser window shows the application running at `localhost:8080/#/products`. The title is "Routing - Hello World". Below it is a navigation bar with links: Home -- About -- Products. The main content area is titled "Products Component".

## Shared Context: Demonstration (2/6)

The screenshot illustrates a development environment with two main components: a code editor and a web browser.

**Code Editor:** On the left, a code editor displays the file `home.js` (part of a sample project). The code defines a `Home` component that uses the `UserContext` provided by a parent component. It includes a JSON dump of the `users` array from the context.

```
function Home(){
  const ctx = React.useContext(UserContext);

  return (
    <div>
      <h3>Home Component</h3>
      {JSON.stringify(ctx.users)}
    </div>
  )
}
```

**Browser:** On the right, a browser window titled "Routing" shows the output at `localhost:8080/#/`. The page title is "Routing - Hello World". Below it, a navigation bar lists [Home](#), [About](#), and [Products](#). A section titled "Home Component" displays the JSON string `["peter"]`.

# Shared Context: Demonstration (3/6)

The image shows a split-screen development environment. On the left, a code editor displays the file `products.js` with the following content:

```
products.js — sample
products.js > Products
1 function Products(){
2   const ctx = React.useContext(UserContext);
3   ctx.users.push(Math.random().toString(36).substr(2, 5));
4
5   return (
6     <div>
7       <h3>Products Component</h3>
8       {JSON.stringify(ctx.users)}
9     </div>
10  )
11 }
```

On the right, a web browser window titled "Routing" shows the application running at `localhost:8080/#/products`. The page title is "Routing - Hello World". Below it is a navigation bar with links: "Home" -- "About" -- "Products". A section titled "Products Component" contains the JSON string `["peter","5g7fw"]`.

# Shared Context: Demonstration (4/6)

The screenshot displays a development environment with two main components: a code editor on the left and a browser window on the right.

**Code Editor (products.js — sample):**

```
products.js > Products
1 function Products(){
2     const ctx = React.useContext(UserContext);
3     ctx.users.push(Math.random().toString(36).substr(2, 5));
4
5     return (
6         <div>
7             <h3>Products Component</h3>
8             {JSON.stringify(ctx.users)}
9         </div>
10    )
11 }
```

**Browser Window (Routing - Hello World):**

The browser title is "Routing - Hello World". The address bar shows "localhost:8080/#/products". The page content includes:

# Routing - Hello World

[Home](#) -- [About](#) -- [Products](#)

---

## Products Component

```
["peter","5g7fw","m9scv","ap82h","v70ny"]
```

# Shared Context: Demonstration (5/6)

The screenshot displays a development environment with two main components: a code editor and a web browser.

**Code Editor:** On the left, a code editor window titled "products.js — sample" shows the following code:

```
products.js > Products
1 function Products(){
2     const ctx = React.useContext(UserContext);
3     ctx.users.push(Math.random().toString(36).substr(2, 5));
4
5     return (
6         <div>
7             <h3>Products Component</h3>
8             {JSON.stringify(ctx.users)}
9         </div>
10    )
11 }
```

**Browser Output:** On the right, a browser window titled "Routing" shows the output of the application at `localhost:8080/#`. The page title is "Routing - Hello World". Below the title, a navigation bar shows "Home --About --Products". The main content area is titled "Home Component" and displays the JSON string: `["peter","5g7fw","m9scv","ap82h","v70ny"]`.

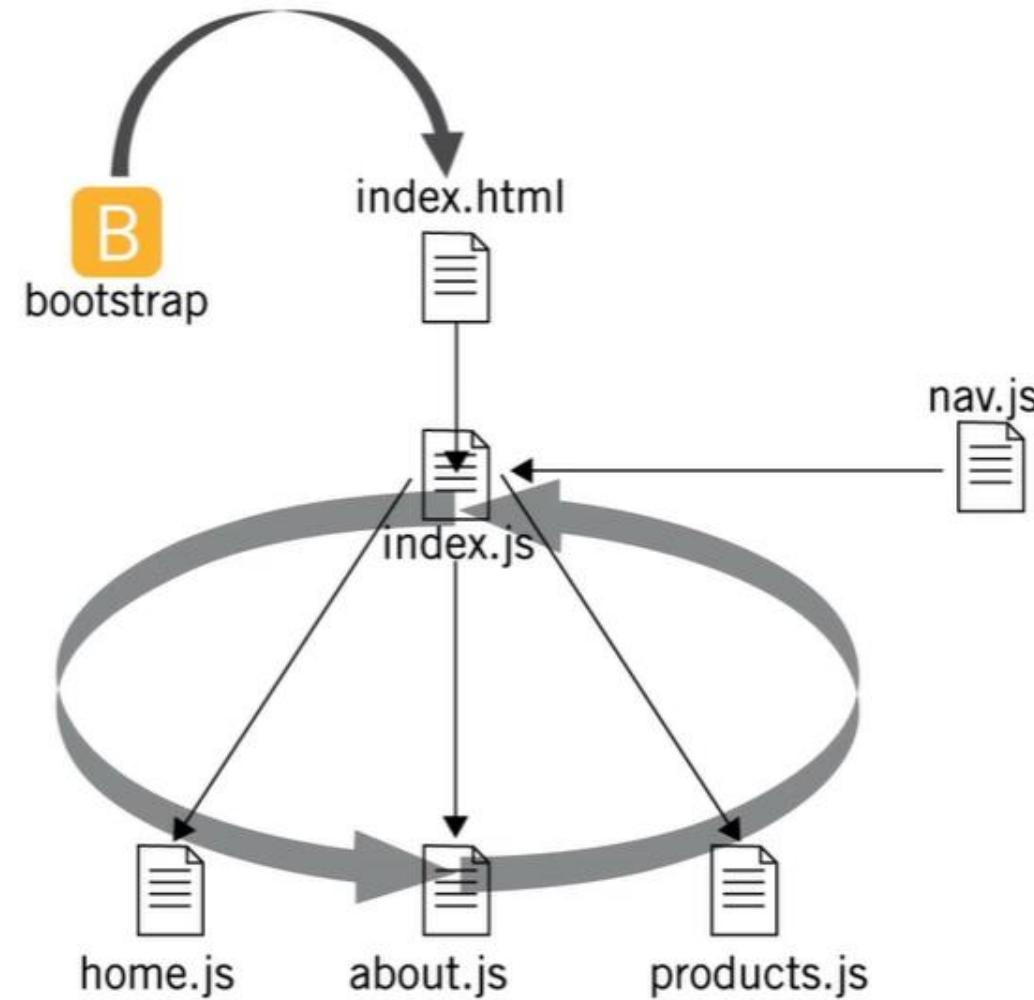
# Shared Context: Demonstration (6/6)

The image shows a split-screen development environment. On the left, a code editor displays the file `products.js` with the following content:

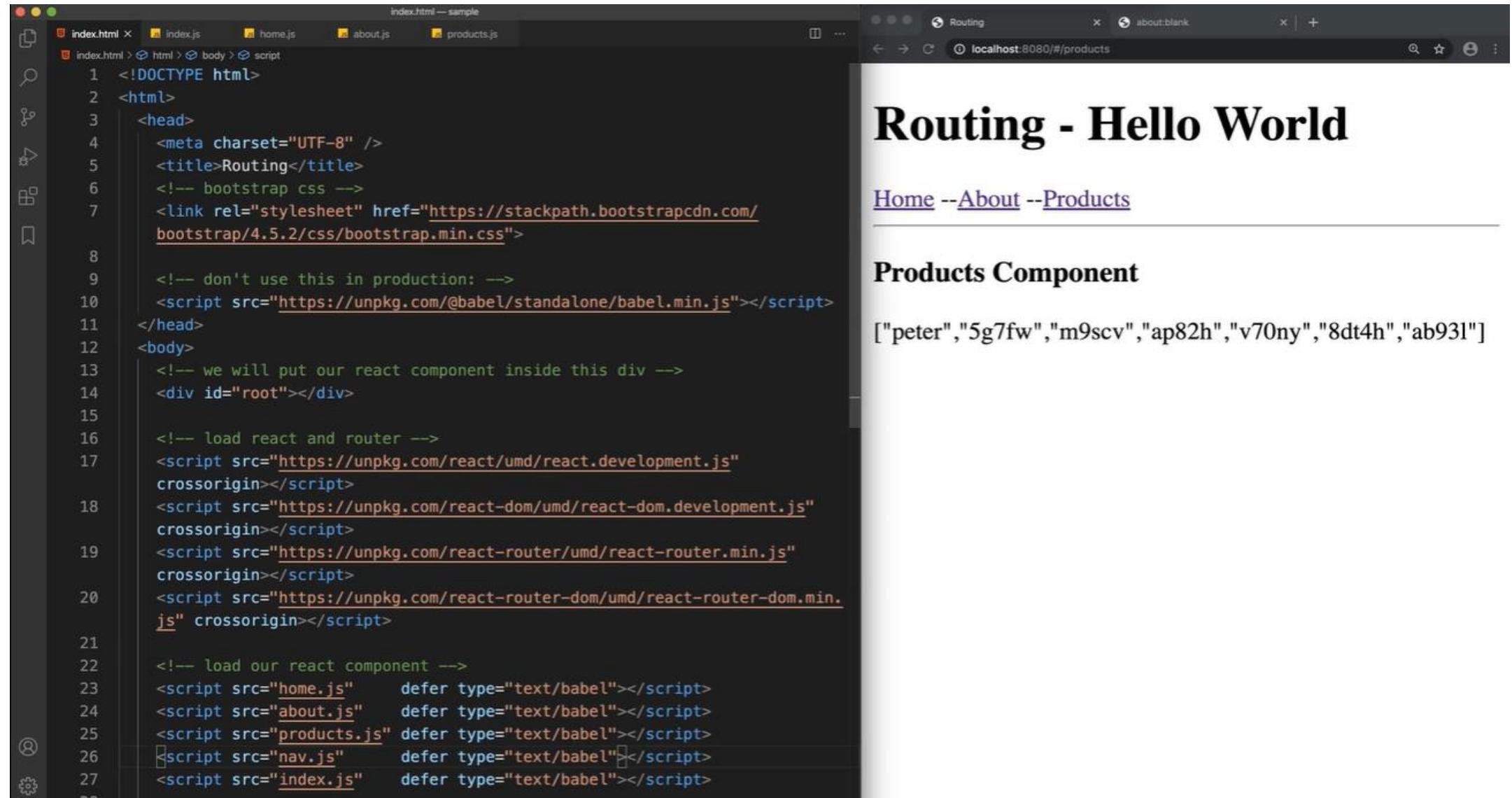
```
products.js — sample
products.js > Products
1 function Products(){
2   const ctx = React.useContext(UserContext);
3   ctx.users.push(Math.random().toString(36).substr(2, 5));
4
5   return (
6     <div>
7       <h3>Products Component</h3>
8       {JSON.stringify(ctx.users)}
9     </div>
10  )
11 }
```

On the right, a web browser window titled "Routing" shows the application running at `localhost:8080/#/products`. The page title is "Routing - Hello World". Below it is a navigation bar with links: "Home" -- "About" -- "Products". The main content area is titled "Products Component" and contains the JSON string: `["peter", "5g7fw", "m9scv", "ap82h", "v70ny", "8dt4h", "ab93l"]`.

# Styles



# Styles: Demonstration (1/9)



The image shows a code editor on the left and a web browser on the right. The code editor displays the file `index.html` with the following content:

```
index.html — sample
index.html x index.js home.js about.js products.js
index.html > html > body > script
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8" />
5      <title>Routing</title>
6      <!-- bootstrap css -->
7      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
bootstrap/4.5.2/css/bootstrap.min.css">
8
9      <!-- don't use this in production: -->
10     <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
11   </head>
12   <body>
13     <!-- we will put our react component inside this div -->
14     <div id="root"></div>
15
16     <!-- load react and router -->
17     <script src="https://unpkg.com/react/umd/react.development.js"
crossorigin></script>
18     <script src="https://unpkg.com/react-dom/umd/react-dom.development.js"
crossorigin></script>
19     <script src="https://unpkg.com/react-router/umd/react-router.min.js"
crossorigin></script>
20     <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.
js" crossorigin></script>
21
22     <!-- load our react component -->
23     <script src="home.js" defer type="text/babel"></script>
24     <script src="about.js" defer type="text/babel"></script>
25     <script src="products.js" defer type="text/babel"></script>
26     <script src="nav.js" defer type="text/babel"></script>
27     <script src="index.js" defer type="text/babel"></script>
```

The browser window on the right shows the URL `localhost:8080/#/products`. The page title is "Routing - Hello World". Below the title, there is a navigation bar with links: "Home" (underlined), "About", and "Products". A horizontal line separates the navigation from the main content area. The main content area contains the heading "Products Component" and a JSON array: `["peter","5g7fw","m9scv","ap82h","v70ny","8dt4h","ab93l"]`.

# Styles: Demonstration (2/9)

The image shows a code editor on the left and a web browser on the right. The code editor displays the file `index.html` with the following content:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8" />
5     <title>Routing</title>
6     <!-- bootstrap css -->
7     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
8
9     <!-- don't use this in production: -->
10    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
11  </head>
12  <body>
13    <!-- we will put our react component inside this div -->
14    <div id="root"></div>
15
16    <!-- load react and router -->
17    <script src="https://unpkg.com/react/umd/react.development.js" crossorigin></script>
18    <script src="https://unpkg.com/react-dom/umd/react-dom.development.js" crossorigin></script>
19    <script src="https://unpkg.com/react-router/umd/react-router.min.js" crossorigin></script>
20    <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.js" crossorigin></script>
21
22    <!-- load our react component -->
23    <script src="home.js" defer type="text/babel"></script>
24    <script src="about.js" defer type="text/babel"></script>
```

The browser window on the right shows the output of the code at `localhost:8080/#/products`. The title is "Routing - Hello World". Below the title, there is a navigation bar with links: "Home" (underlined), "About", and "Products". A horizontal line separates the header from the main content area. The main content area contains the heading "Products Component" and a JSON array: `["peter","5g7fw","m9scv","ap82h","v70ny","8dt4h","ab93l"]`.

# Styles: Demonstration (3/9)

The image shows a split-screen view. On the left, a code editor (VS Code) displays a file structure with several files: index.html, nav.js, index.js, home.js, about.js, and products.js. The nav.js file is currently open in the editor. On the right, a web browser is displaying the Bootstrap documentation at [getbootstrap.com/docs/4.5/getting-started/introduction/](https://getbootstrap.com/docs/4.5/getting-started/introduction/). The page features a purple header with the Bootstrap logo and navigation links for Home, Documentation, Examples, Icons, Themes, Expo, and Blog. A dropdown menu titled "Nav" is open, listing components such as Navbar / Navbar, Navs / Navs, Navs / Base nav, Scrollspy / Example in navbar, and Scrollspy / Example with nested nav. Below the dropdown, there are sections for Quick start, CSS, and JS, along with a sidebar for search, build tools, and accessibility.

# Styles: Demonstration (4/9)

The image shows a split-screen environment. On the left, a code editor (VS Code) displays a file structure with several files: index.html, nav.js, index.js, home.js, about.js, and products.js. The nav.js file is currently open in the editor. On the right, a web browser window is open to the Bootstrap documentation page for the Nav component. The browser title is "Routing" and the URL is "getbootstrap.com/docs/4.5/components/navs/". The page content includes a search bar, a sidebar with navigation links like "Getting started", "Layout", "Content", and a main content area for the Nav component. The main content area shows two examples of Nav markup. The first example uses a standard 

 class="nav"> structure, while the second example uses a  element. Both examples include classes for different states: Active, Link, and Disabled. A note below the examples explains that classes are used throughout, making the markup flexible. The note also mentions that the .nav uses `display: flex`, which affects how the nav links behave compared to standard nav items.

nav.js — sample

index.html nav.js index.js home.js about.js products.js

B Home Documentation Examples Icons Themes Expo Blog v4.5

Search...

Active Link Link Disabled

Getting started

Layout

Content

Components

- Alerts
- Badge
- Breadcrumb
- Buttons
- Button group
- Card
- Carousel
- Collapse
- Dropdowns
- Forms
- Input group
- Jumbotron
- List group
- Media object
- Modal
- Navs
- Navbar
- Pagination
- Popovers
- Progress
- Scrollspy
- Spinners

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
  </li>
</ul>
```

Classes are used throughout, so your markup can be super flexible. Use `<ul>`s like above, `<ol>` if the order of your items is important, or roll your own with a `<nav>` element. Because the `.nav` uses `display: flex`, the nav links behave the same as nav items would, but without the extra markup.

```
<nav class="nav">
  <a class="nav-link active" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
</nav>
```

# Styles: Demonstration (5/9)

The image shows a code editor on the left and a browser window on the right. The code editor displays a file named `nav.js` with the following content:

```
function Nav(){
  return(
    <ul className="nav">
      <li className="nav-item">
        <Link className="nav-link" to="/">Home</Link>
      </li>
      <li className="nav-item">
        <Link className="nav-link" to="/about/">About</Link>
      </li>
      <li className="nav-item">
        <Link className="nav-link" to="/products">Products</Link>
      </li>
    </ul>
  );
}
```

The browser window shows a Bootstrap 4.5 navigation bar with three items: Home, About, and Products. The "Home" link is highlighted with a red background, indicating it is the active page.

On the right side of the browser window, there is a sidebar with a search bar and a list of components under the heading "Components". The "Navs" section is expanded, showing examples for "Active", "Link", and "Disabled" states. The "Active" example includes the following HTML code:

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
</ul>
```

A note below the code states: "Classes are used throughout, so your markup can be super flexible. Use `<ul>`s like above, `<ol>` if the order of your items is important, or roll your own with a `<nav>` element. Because the `.nav` uses `display: flex`, the nav links behave the same as nav items would, but without the extra markup."

At the bottom of the sidebar, there is another example for the "Navs" section:

```
<nav class="nav">
  <a class="nav-link active" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled" href="#" tabindex="-1" aria-di</a>
</nav>
```

# Styles: Demonstration (6/9)

The image shows a split-screen environment. On the left, a code editor displays a file named `index.js` with the following content:

```
index.js — sample
index.html index.js index.js index.js about.js products.js
index.js > Spa
1 const UserContext = React.createContext(null);
2
3 function Spa(){
4     const Route      = ReactDOM.Route;
5     const Link       = ReactDOM.Link;
6     const HashRouter = ReactDOM.HashRouter;
7
8     return (
9         <HashRouter>
10        <div>
11            <h1>Routing - Hello World</h1>
12            <Nav/>
13            <hr/>
14            <UserContext.Provider value={{users:['peter']}}>
15                <Route path="/" exact component={Home} />
16                <Route path="/about/" component={About} />
17                <Route path="/products/" component={Products} />
18            </UserContext.Provider>
19        </div>
20    </HashRouter>
21 );
22 }
23
24 ReactDOM.render(
25     <Spa/>,
26     document.getElementById('root')
27 )
```

On the right, a web browser window displays the Bootstrap documentation for the `Nav` component at [getbootstrap.com/docs/4.5/components/navs/](https://getbootstrap.com/docs/4.5/components/navs/). The page shows examples of different navigation styles, including active, link, and disabled states. Two examples are highlighted with red boxes:

**Active**

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Link</a>
  </li>
</ul>
```

**Copy**

**Classes are used throughout, so your markup can be super flexible. Use `<ul>`s like above, `<ol>` if the order of your items is important, or roll your own with a `<nav>` element. Because the `.nav` uses `display: flex`, the nav links behave the same as nav items would, but without the extra markup.**

**Link**

**Link**

**Disabled**

**Components**

- Alerts
- Badge
- Breadcrumb
- Buttons
- Button group
- Card
- Carousel
- Collapse
- Dropdowns
- Forms
- Input group
- Jumbotron
- List group
- Media object
- Modal
- Navs**
- Navbar
- Pagination
- Popovers
- Progress
- Scrollspy
- Spinners

**Active**

```
<nav class="nav">
  <a class="nav-link active" href="#">Active</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link" href="#">Link</a>
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Link</a>
</nav>
```

**Copy**

# Styles: Demonstration (7/9)

The screenshot displays a development environment with a code editor and a browser window.

**Code Editor:** The left pane shows a code editor with a dark theme. The file being edited is `nav.js`, which contains the following code:

```
function Nav(){
  return(
    <ul className="nav">
      <li className="nav-item">
        <Link className="nav-link" to="/">Home</Link>
      </li>
      <li className="nav-item">
        <Link className="nav-link" to="/about/">About</Link>
      </li>
      <li className="nav-item">
        <Link className="nav-link" to="/products">Products</Link>
      </li>
    </ul>
  );
}
```

**Browser:** The right pane shows a browser window titled "Routing" with the URL `localhost:8080/#/products`. The page content is blank, indicating a rendering issue.

**Developer Tools:** Below the browser window are the developer tools. The "Console" tab is selected, showing the following error message:

```
Uncaught ReferenceError: Link is not defined
  at Nav (<anonymous>:8:39)
  at renderWithHooks (react-dom.development.js:14938)
  at mountIndeterminateComponent (react-dom.development.js:17617)
  at beginWork (react-dom.development.js:18731)
  at HTMLUnknownElement.callCallback (react-dom.development.js:182)
  at Object.invokeGuardedCallbackDev (react-dom.development.js:231)
  at invokeGuardedCallback (react-dom.development.js:286)
  at beginWork$1 (react-dom.development.js:23338)
  at performUnitOfWork (react-dom.development.js:2229)
  at workLoopSync (react-dom.development.js:22265)
```

# Styles: Demonstration (8/9)

The image shows a split-screen development environment. On the left, a code editor displays the file `index.js` with the following content:

```
index.js — sample
index.html  nav.js  index.js  home.js  about.js  products.js
index.js > ...
1 const Route      = ReactRouterDOM.Route;
2 const Link       = ReactRouterDOM.Link;
3 const HashRouter = ReactRouterDOM.HashRouter;
4 const UserContext = React.createContext(null);
5
6 function Spa(){
7     return (
8         <HashRouter>
9             <div>
10                <h1>Routing - Hello World</h1>
11                <Nav/>
12                <hr/>
13                <UserContext.Provider value={{users:['peter']}}>
14                    <Route path="/" exact component={Home} />
15                    <Route path="/about/" component={About} />
16                    <Route path="/products/" component={Products} />
17                </UserContext.Provider>
18            </div>
19        </HashRouter>
20    );
21 }
22
23 ReactDOM.render(
24     <Spa/>,
25     document.getElementById('root')
26 )
```

On the right, a web browser window titled "Routing" is open at the URL `localhost:8080/#/products`. The page title is "Routing - Hello World". It features a navigation bar with links to "Home", "About", and "Products". Below the navigation, the text "Products Component" is displayed, followed by the JSON array `["peter","0o796"]`.

# Styles: Demonstration (9/9)

The image shows a code editor on the left and a web browser on the right. The code editor displays the file `index.js` with the following content:

```
index.js — sample
index.html index.js nav.js index.js x home.js about.js products.js

1 const Route      = ReactRouterDOM.Route;
2 const Link       = ReactRouterDOM.Link;
3 const HashRouter = ReactRouterDOM.HashRouter;
4 const UserContext = React.createContext(null);
5
6 function Spa(){
7     return (
8         <HashRouter>
9             <div>
10                <h1>Routing - Hello World</h1>
11                <Nav/>
12                <hr/>
13                <UserContext.Provider value={{users:['peter']}}>
14                    <Route path="/" exact component={Home} />
15                    <Route path="/about/" component={About} />
16                    <Route path="/products/" component={Products} />
17                </UserContext.Provider>
18            </div>
19        </HashRouter>
20    );
21 }
22
23 ReactDOM.render(
24     <Spa/>,
25     document.getElementById('root')
26 )
```

The browser window on the right shows the application running at `localhost:8080/#`. The title is "Routing - Hello World". The navigation bar includes links for "Home", "About", and "Products". The main content area displays "Home Component" and the user list `["peter","Oo796","endry"]`.

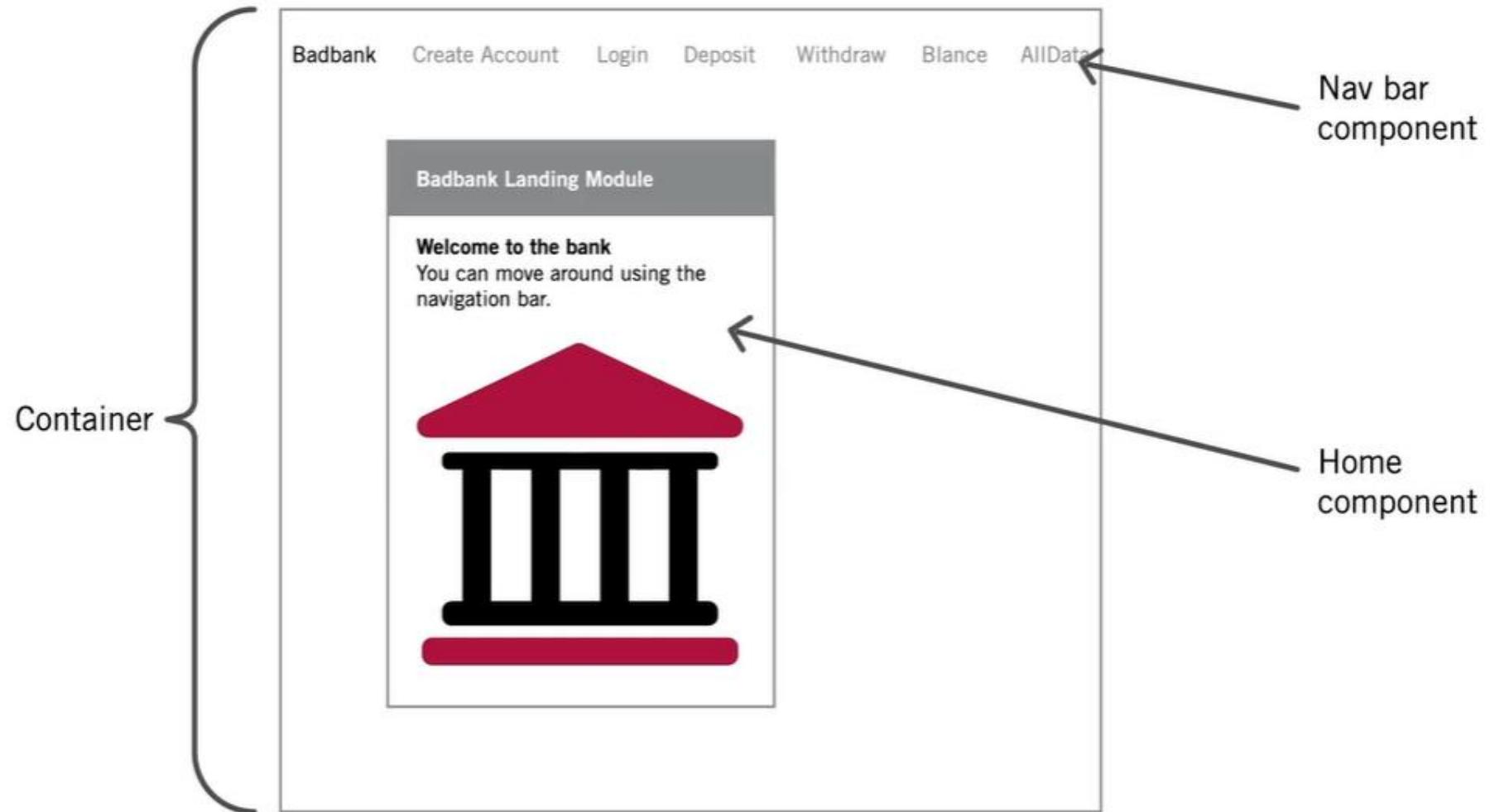
# Bad Bank – No Security



Bad Bank

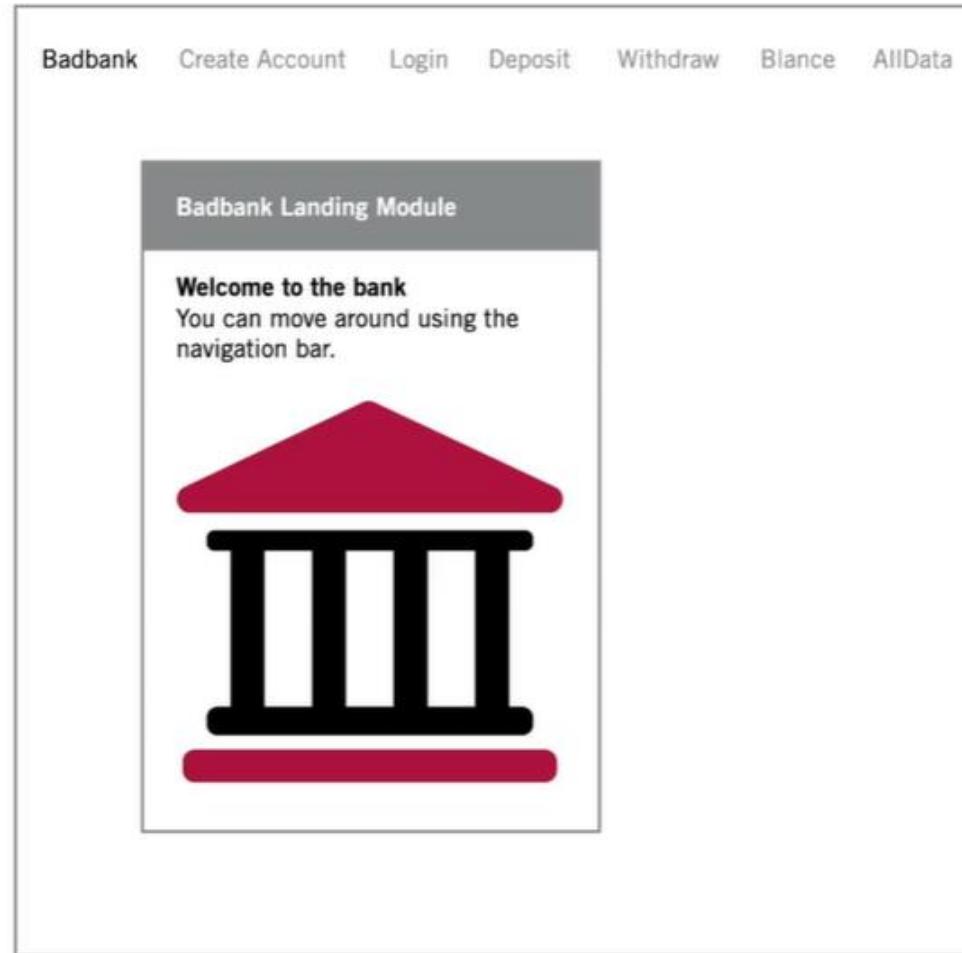


# Design



- Routing
- Context
- Styles
- Components
- Parent-Child components
- Decisions on where to keep state
- Forms
- Handling events

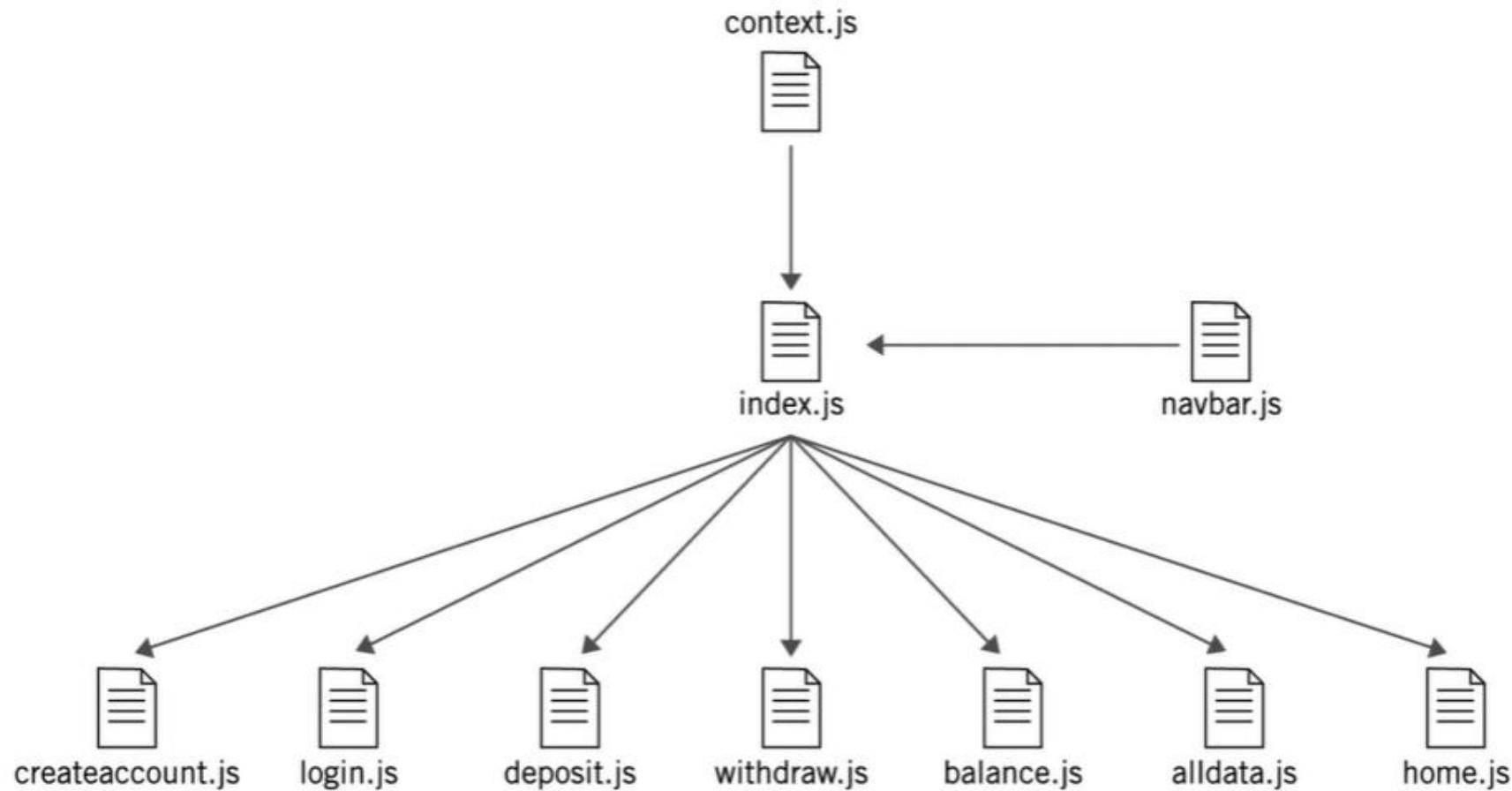
# Design – Components Loaded Based On Path



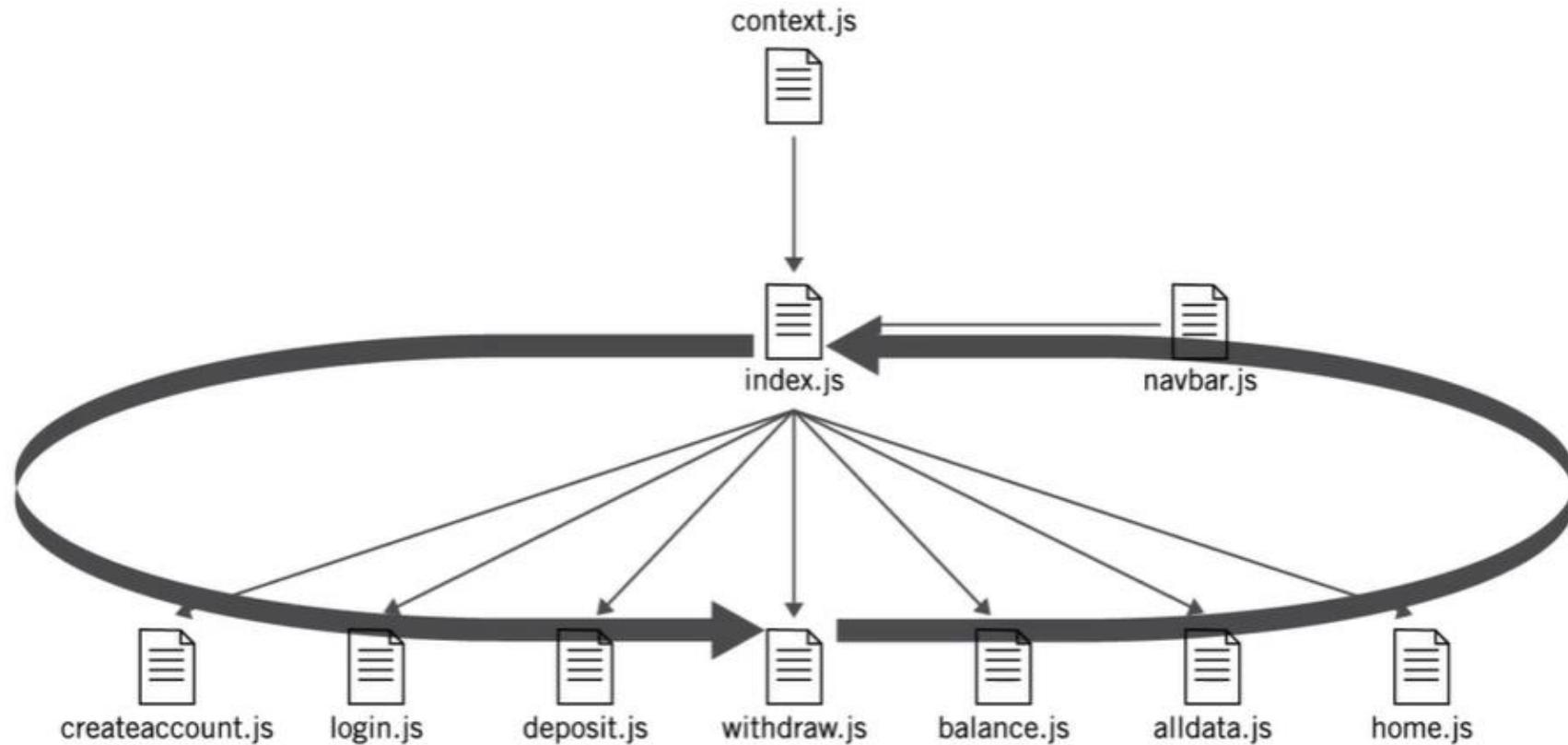
## Components:

- Home
- Create account
- Login
- Deposit
- Withdraw
- Balance
- All data

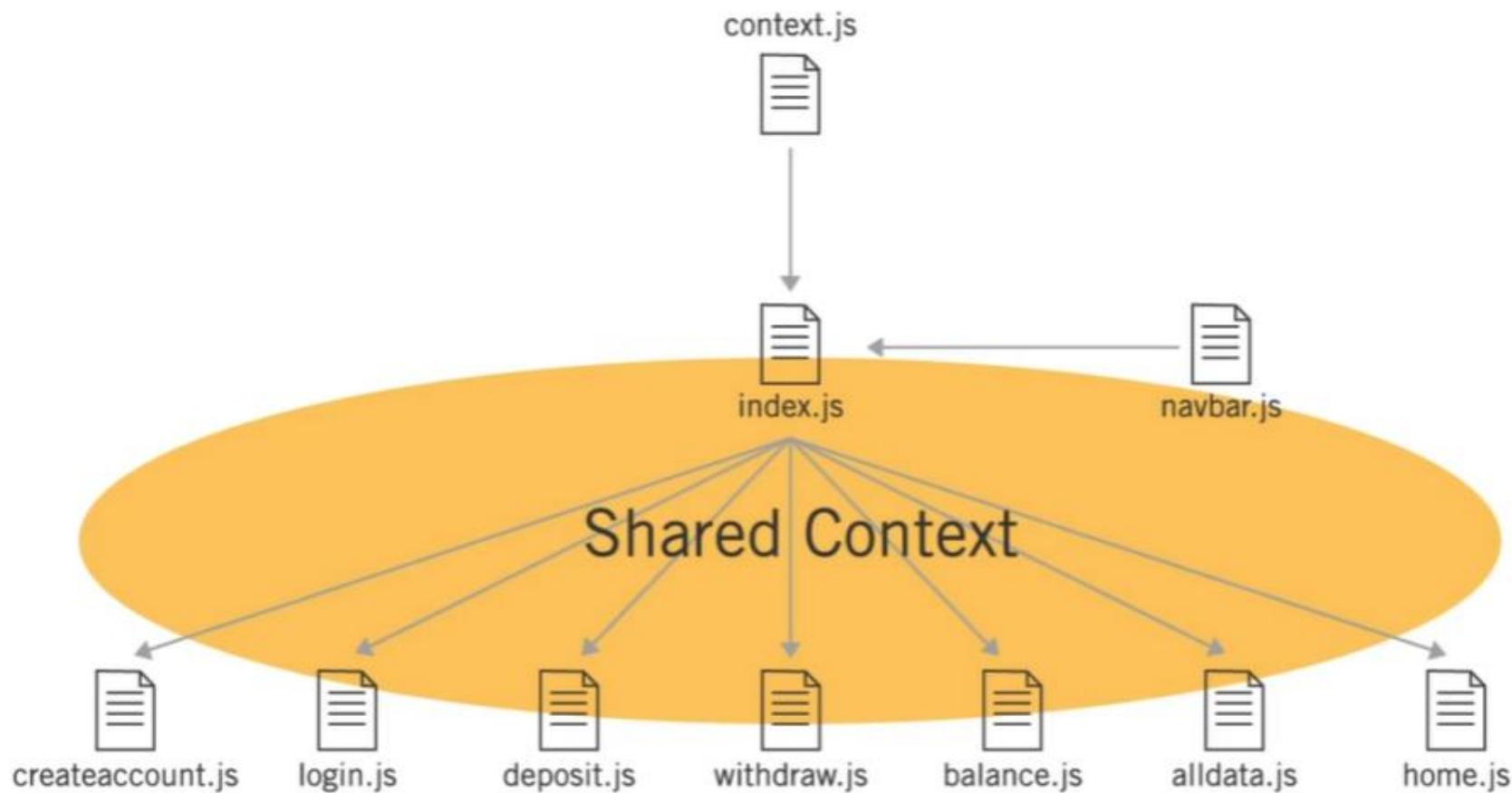
# Architecture



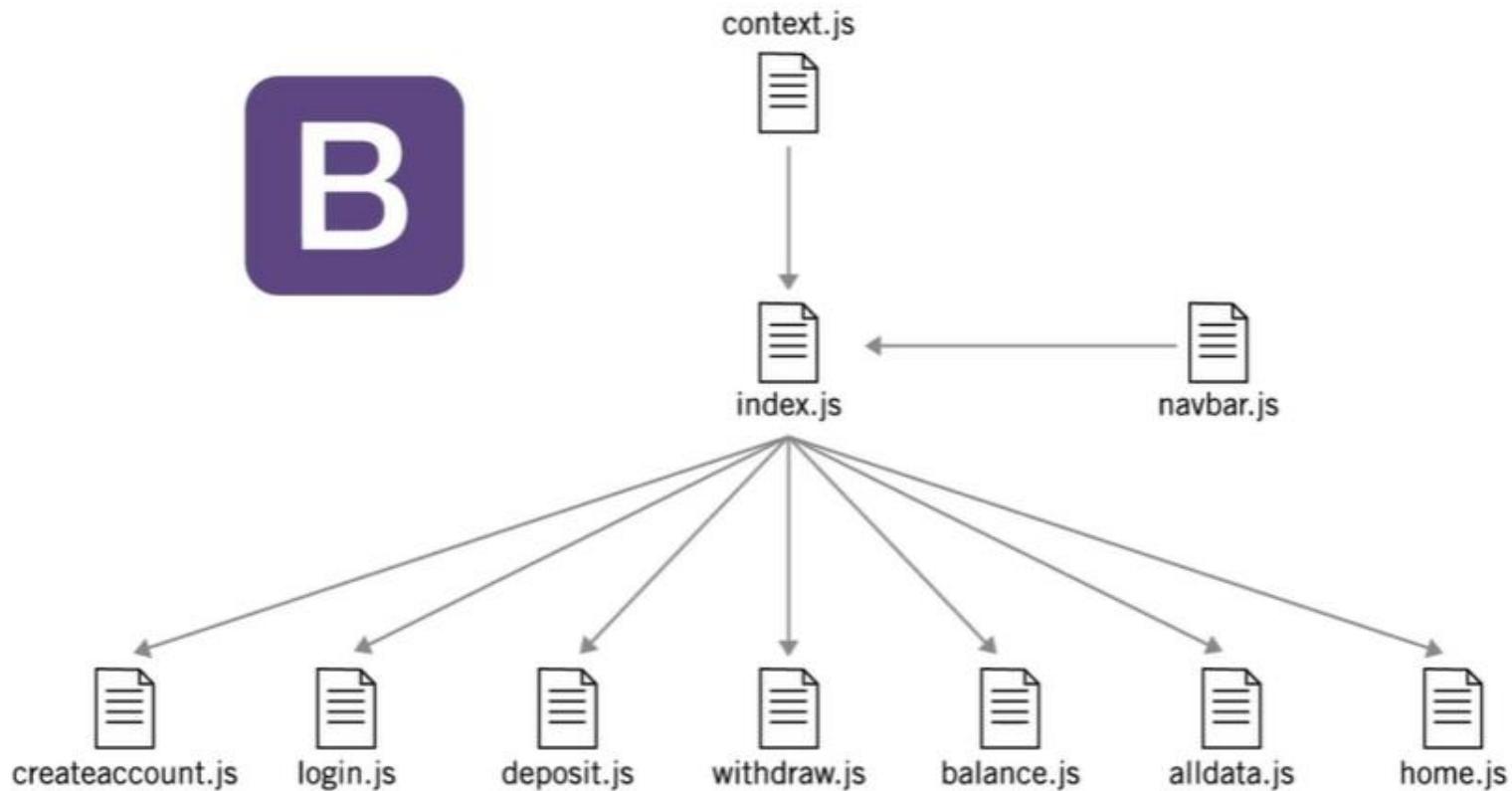
# Routing



# Shared Context



# Styles – Bootstrap

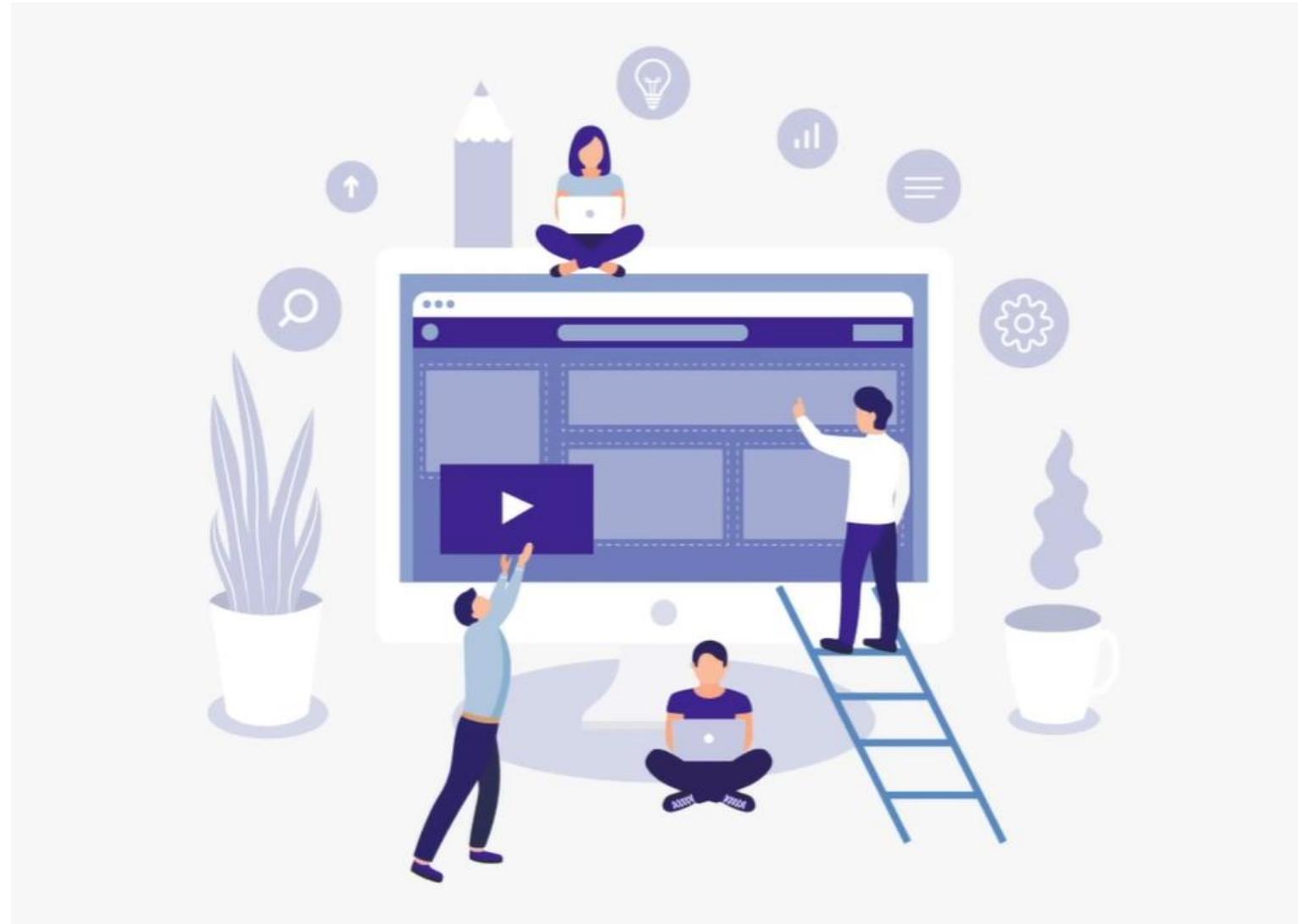


# Create Bad Bank Application Files (1/2)

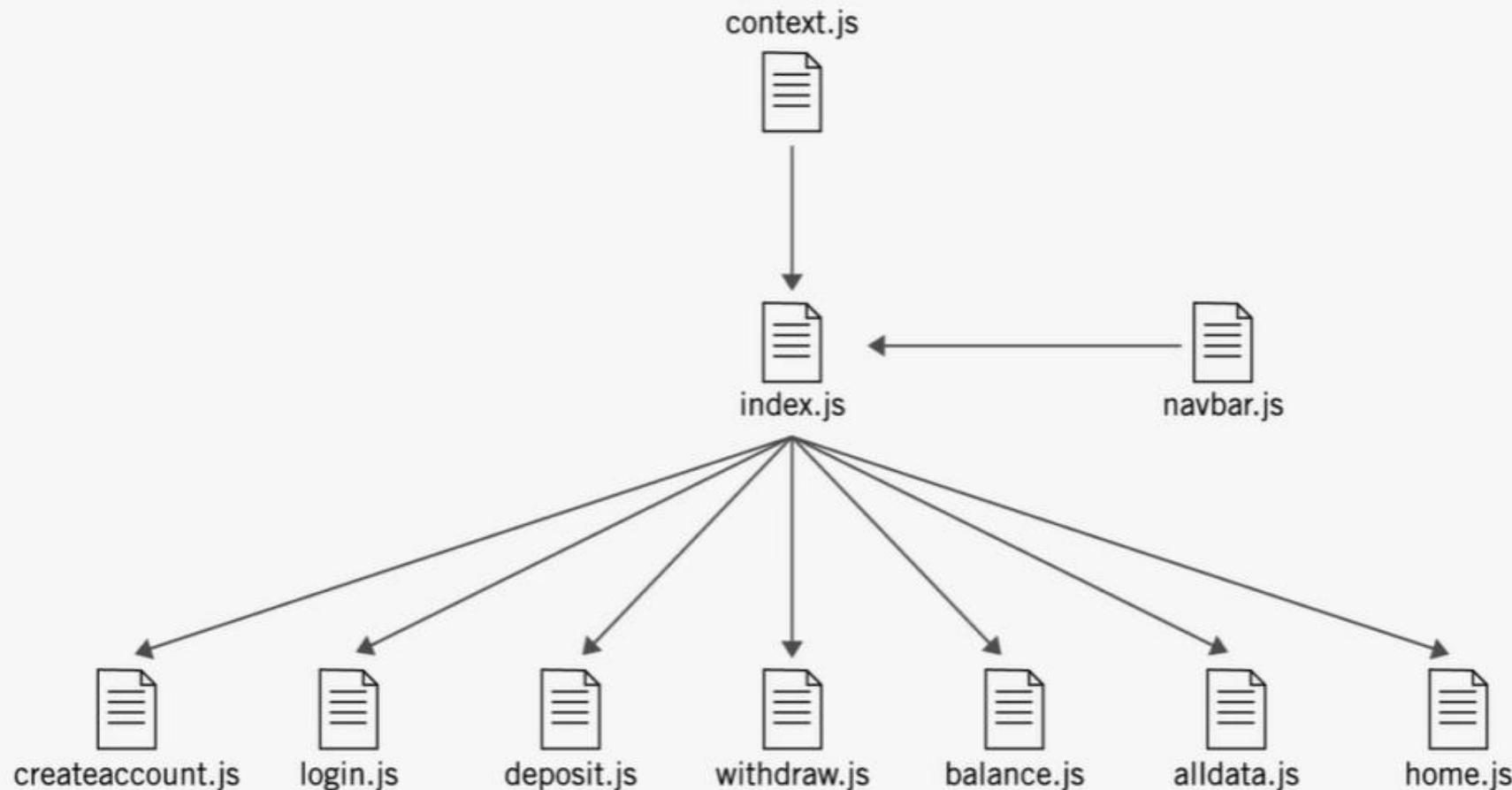
```
}), $(document).on("click.filter_log_click", ".trg_fl_item", function() {
    var a = $(this),
        b = a.data().meta,
        c = b.key,
        d = b.value,
        f = function.state.filters[c],
        g = function.state.options[c].form_el,
        j = $(function.settings.filters_Form_container_Selector);
    if ("select" == g) {
        if ("range_slider" == function.state.options[c].type) return delete function.state.filters[c], void function.make_query();
        var k = f.indexOf(d);
        return -1 < k && f.splice(k, 1), f.length ? function.state.filters[c] = f : delete function.state.filters[c], void function.make_query();
    }
    return "checkbox" == g ? (delete function.state.filters[c], void function.make_query()) : void 0
}), $(function.settings.clear_filters_toggle).click(function() {
    $(function.settings.filters_log_container_selector).find(".filter-option-log").remove(), $.each(function.state.filters, function() {
        $(".ub_filter_name_" + a).ub_filter("ResetUIOnly")
    }), function.state.filters = {}, function.make_query()
}), $(document).on("click", ".trg_clear_all", function() {
    function.state.filters = [], function.make_query()
}), function.isMobile && $(document).on("click", ".trg_mobi_close_filter_window", function() {
    setTimeout(function() {
        read_filters_and_make_query()
    }, 22)
}), $(".trg_tb_search").click(function() {
    $(".tb-search-outer").toggleClass("search-active"), $(".tb-search-input").focus()
}), $(".tb-search-input,#query_tb_mobi").on("blur", function() {
    var a = $(this);
    is_blank(a.val()) && (a.val(""), $(".tb-search-outer").toggleClass("search-active"))
}).on("keyup", function() {
    var b = $(this),
        c = b.val();
    if (!is_blank(c)) {
        b.parent().addClass("has-input");
        var d = event.keyCode || event.which;
        "13" == d && (function.state.query = c, function.make_query(), $("#ub-ac-outer").addClass("hidden"))
    } else $(".tb-search-outer").removeClass("has-input")
}), $(".tb-search-clear-button").click(function() {
    $(this).parent().removeClass("has-input").find("input").val("").focus()
}), $(document).on("click", ".query-search-clear-button", function() {
    function.state.query = "", function.make_query()
}), $(".trg_mobile_search").click(function() {
    $("#search-page-mobile").toggleClass("hidden"), setTimeout(function() {

```

## Create Bad Bank Application Files (2/2)



# Architecture

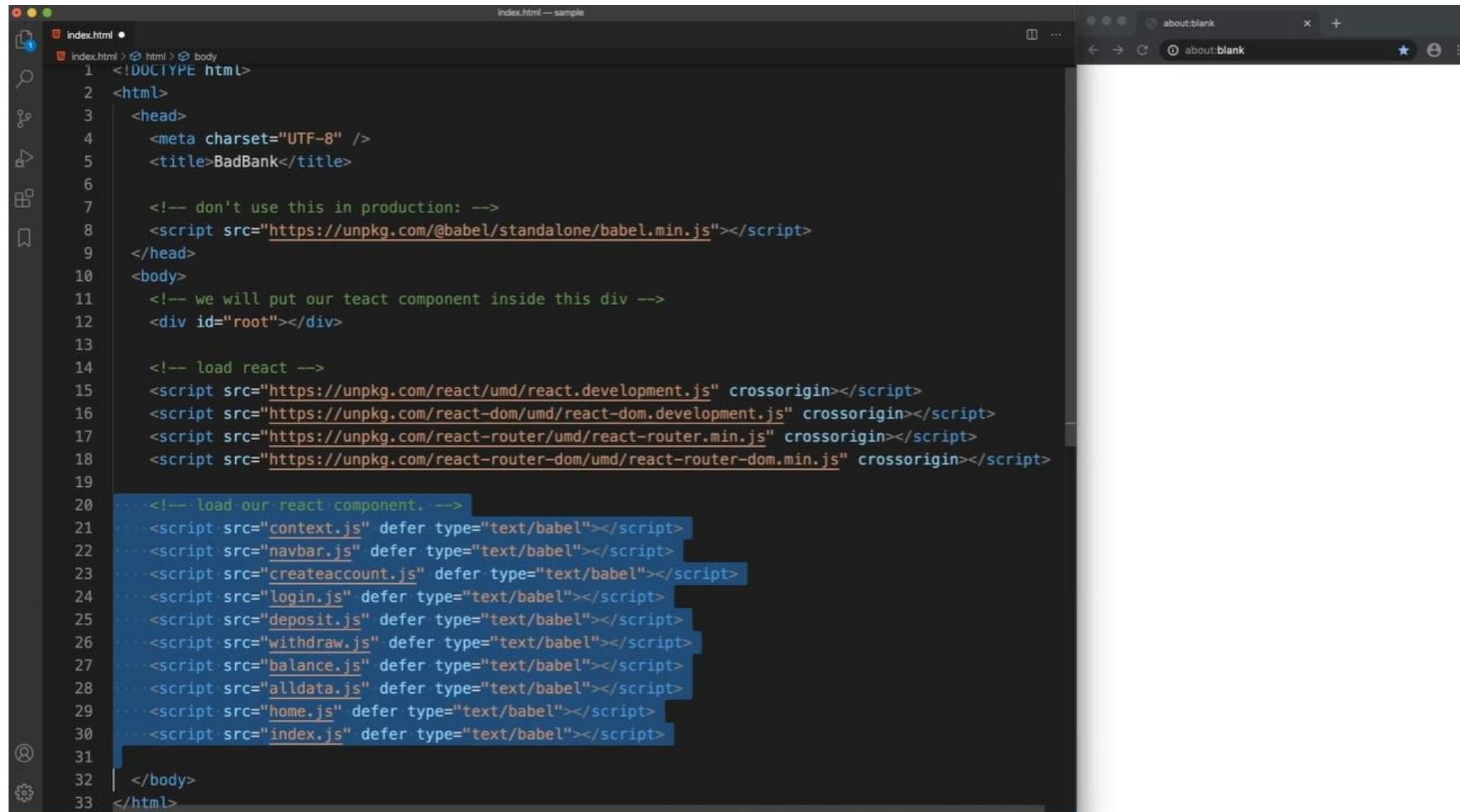


## File Loading – Done In index.html

components  
loaded in  
index.html

```
<script src = "context.js" defer type = "text/label" ></script>
<script src = "navbar.js" defer type = "text/label" ></script>
<script src = "createaccount.js" defer type = "text/label" ></script>
<script src = "login.js" defer type = "text/label" ></script>
<script src = "deposit.js" defer type = "text/label" ></script>
<script src = "withdraw.js" defer type = "text/label" ></script>
<script src = "balance.js" defer type = "text/label" ></script>
<script src = "alldata.js" defer type = "text/label" ></script>
<script src = "home.js" defer type = "text/label" ></script>
<script src = "index.js" defer type = "text/label" ></script>
```

# Components Loaded: Demonstration (1/6)



The image shows a code editor on the left and a browser window on the right. The code editor displays the file `index.html` with the following content:

```
index.html — sample
index.html •
index.html > html > body
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8" />
5      <title>BadBank</title>
6
7      <!-- don't use this in production: -->
8      <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
9    </head>
10   <body>
11     <!-- we will put our react component inside this div -->
12     <div id="root"></div>
13
14     <!-- load react -->
15     <script src="https://unpkg.com/react/umd/react.development.js" crossorigin></script>
16     <script src="https://unpkg.com/react-dom/umd/react-dom.development.js" crossorigin></script>
17     <script src="https://unpkg.com/react-router/umd/react-router.min.js" crossorigin></script>
18     <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.js" crossorigin></script>
19
20     <!-- load our react component. -->
21     <script src="context.js" defer type="text/babel"></script>
22     <script src="navbar.js" defer type="text/babel"></script>
23     <script src="createaccount.js" defer type="text/babel"></script>
24     <script src="login.js" defer type="text/babel"></script>
25     <script src="deposit.js" defer type="text/babel"></script>
26     <script src="withdraw.js" defer type="text/babel"></script>
27     <script src="balance.js" defer type="text/babel"></script>
28     <script src="alldata.js" defer type="text/babel"></script>
29     <script src="home.js" defer type="text/babel"></script>
30     <script src="index.js" defer type="text/babel"></script>
31
32   </body>
33 </html>
```

The browser window on the right shows a blank page titled "about:blank". The code editor has syntax highlighting for HTML and JavaScript.

## Components Loaded: Demonstration (2/6)

The screenshot shows a code editor and a terminal window side-by-side.

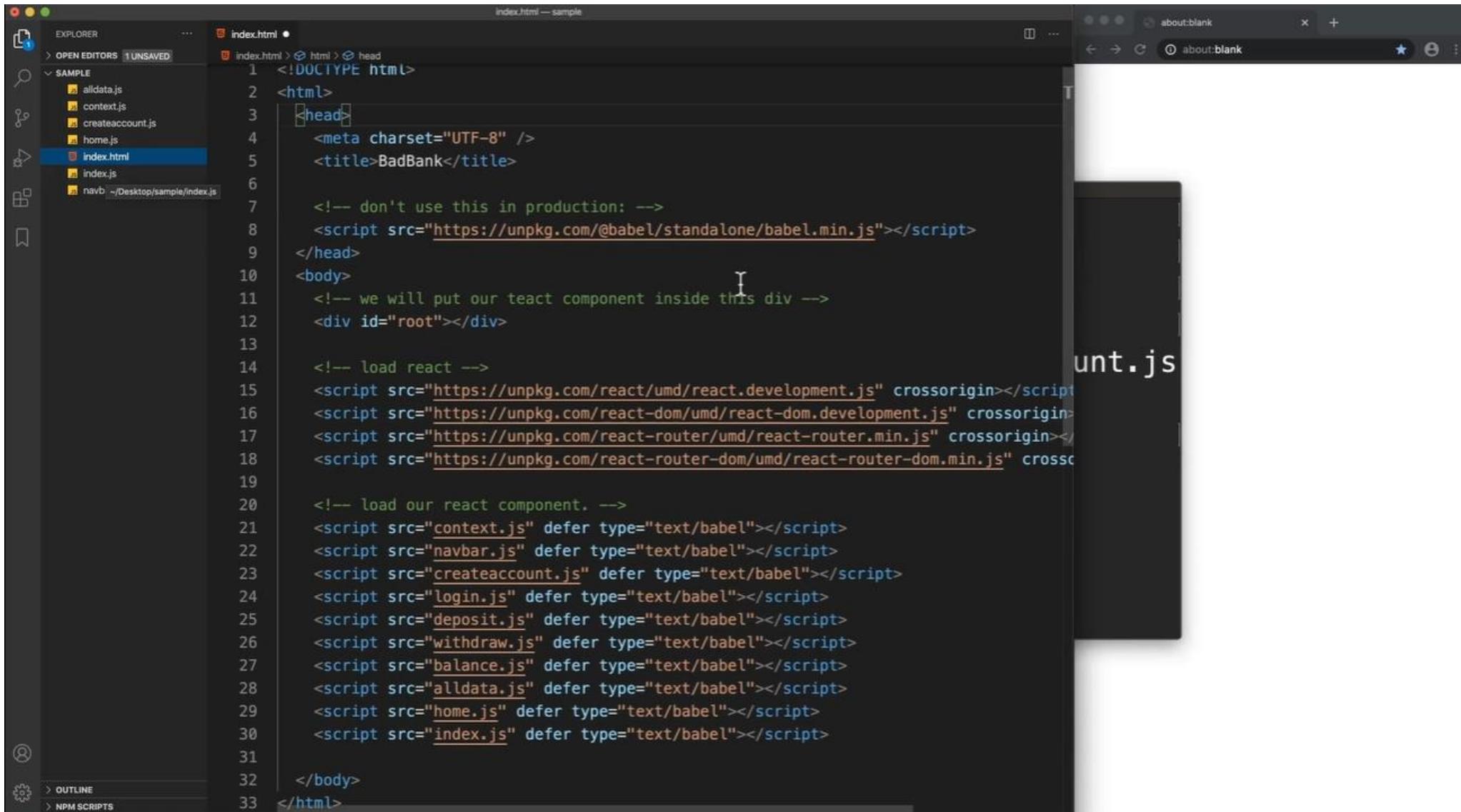
**Code Editor (index.html):**

```
index.html — sample
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8" />
5     <title>BadBank</title>
6
7     <!-- don't use this in production: -->
8     <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
9   </head>
10  <body>
11    <!-- we will put our react component inside this
12    <div id="root"></div>
13
14    <!-- load react -->
15    <script src="https://unpkg.com/react/umd/react.development.js"></script>
16    <script src="https://unpkg.com/react-dom/umd/react-dom.development.js"></script>
17    <script src="https://unpkg.com/react-router/umd/react-router.development.js"></script>
18    <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.development.js"></script>
19
20    <!-- load our react component. -->
21    <script src="context.js" defer type="text/babel"></script>
22    <script src="navbar.js" defer type="text/babel"></script>
23    <script src="createaccount.js" defer type="text/babel"></script>
24    <script src="login.js" defer type="text/babel"></script>
25    <script src="deposit.js" defer type="text/babel"></script>
26    <script src="withdraw.js" defer type="text/babel"></script>
27    <script src="balance.js" defer type="text/babel"></script>
28    <script src="alldata.js" defer type="text/babel"></script>
29    <script src="home.js" defer type="text/babel"></script>
30    <script src="index.js" defer type="text/babel"></script>
31
32  </body>
33 </html>
```

**Terminal (about:blank):**

```
sample % touch context.js
sample % touch navbar.js
sample % touch index.js
sample % touch home.js
sample % touch createaccount.js
sample % touch alldata.js
sample %
```

# Components Loaded: Demonstration (3/6)



The image shows a code editor interface with a dark theme. On the left is the Explorer sidebar showing a file tree under a 'SAMPLE' folder. The 'index.html' file is open in the main editor area. The browser window on the right shows a blank page titled 'about:blank'. The code in 'index.html' is as follows:

```
index.html — sample
index.html •
OPEN EDITORS 1 UNSAVED
SAMPLE
  alldata.js
  context.js
  createaccount.js
  home.js
  index.html
  index.js
  navb ~/Desktop/sample/index.js

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>BadBank</title>

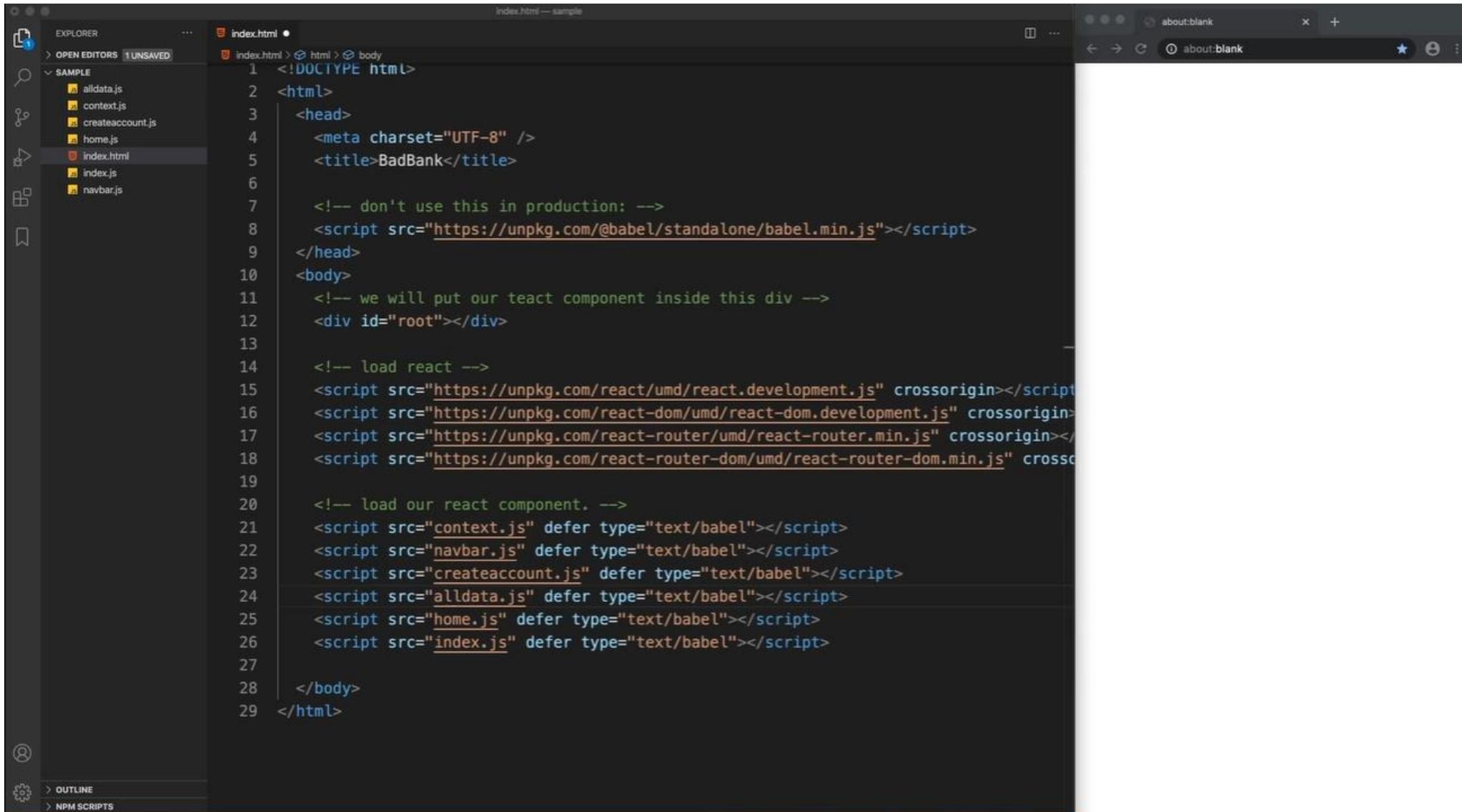
    <!-- don't use this in production: -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>
    <!-- we will put our react component inside this div -->
    <div id="root"></div>

    <!-- load react -->
    <script src="https://unpkg.com/react/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom/umd/react-dom.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-router/umd/react-router.min.js" crossorigin></script>
    <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.js" crossorigin></script>

    <!-- load our react component. -->
    <script src="context.js" defer type="text/babel"></script>
    <script src="navbar.js" defer type="text/babel"></script>
    <script src="createaccount.js" defer type="text/babel"></script>
    <script src="login.js" defer type="text/babel"></script>
    <script src="deposit.js" defer type="text/babel"></script>
    <script src="withdraw.js" defer type="text/babel"></script>
    <script src="balance.js" defer type="text/babel"></script>
    <script src="alldata.js" defer type="text/babel"></script>
    <script src="home.js" defer type="text/babel"></script>
    <script src="index.js" defer type="text/babel"></script>

  </body>
</html>
```

# Components Loaded: Demonstration (4/6)



The image shows a screenshot of a code editor (VS Code) and a web browser side-by-side.

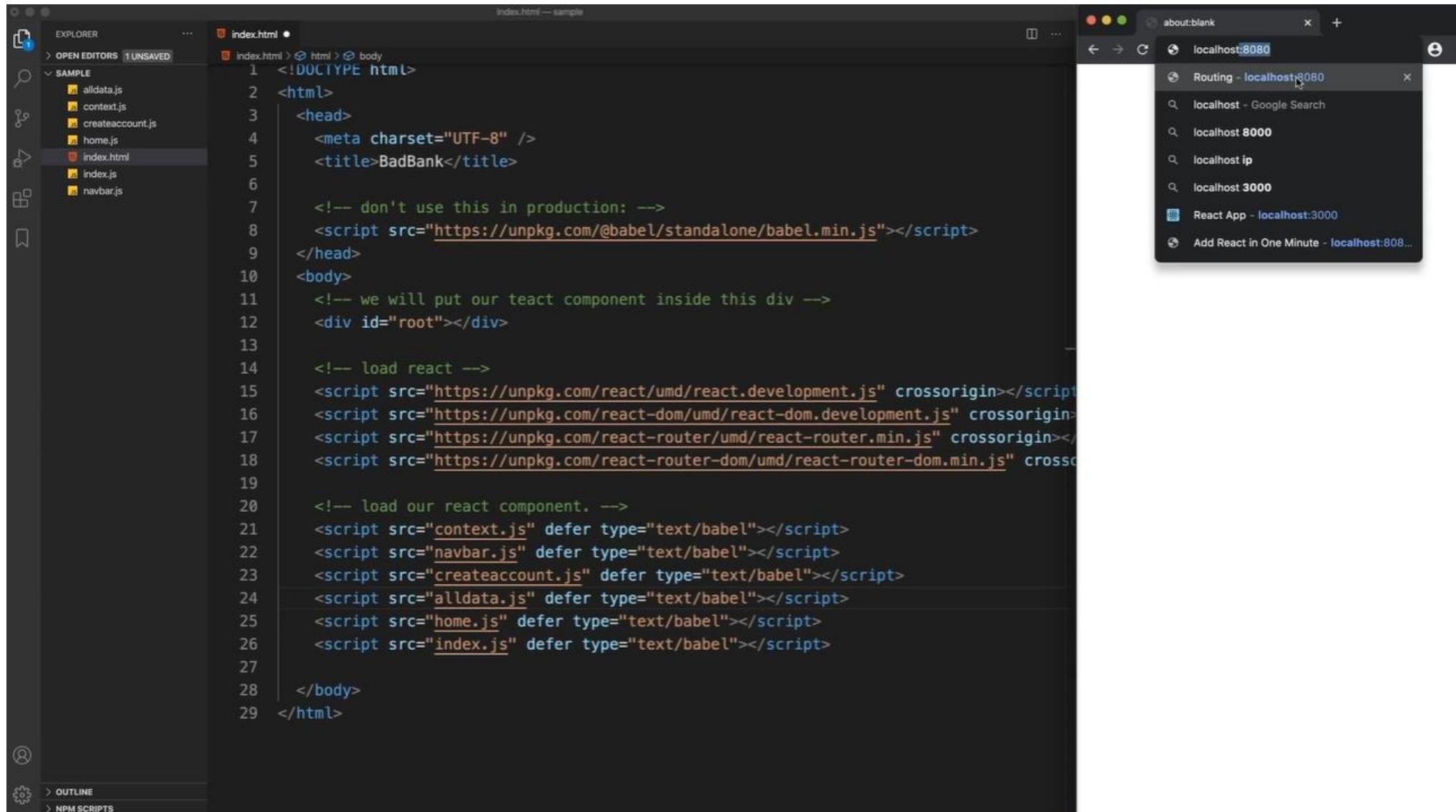
**Code Editor (index.html — sample):**

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8" />
5      <title>BadBank</title>
6
7      <!-- don't use this in production: -->
8      <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
9  </head>
10 <body>
11     <!-- we will put our react component inside this div -->
12     <div id="root"></div>
13
14     <!-- load react -->
15     <script src="https://unpkg.com/react/umd/react.development.js" crossorigin></script>
16     <script src="https://unpkg.com/react-dom/umd/react-dom.development.js" crossorigin></script>
17     <script src="https://unpkg.com/react-router/umd/react-router.min.js" crossorigin></script>
18     <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.js" crossorigin></script>
19
20     <!-- load our react component. -->
21     <script src="context.js" defer type="text/babel"></script>
22     <script src="navbar.js" defer type="text/babel"></script>
23     <script src="createaccount.js" defer type="text/babel"></script>
24     <script src="alldata.js" defer type="text/babel"></script>
25     <script src="home.js" defer type="text/babel"></script>
26     <script src="index.js" defer type="text/babel"></script>
27
28     </body>
29 </html>
```

**Browser (about:blank):**

The browser window shows a blank white page, indicating that the application has loaded but no content is currently displayed.

# Components Loaded: Demonstration (5/6)



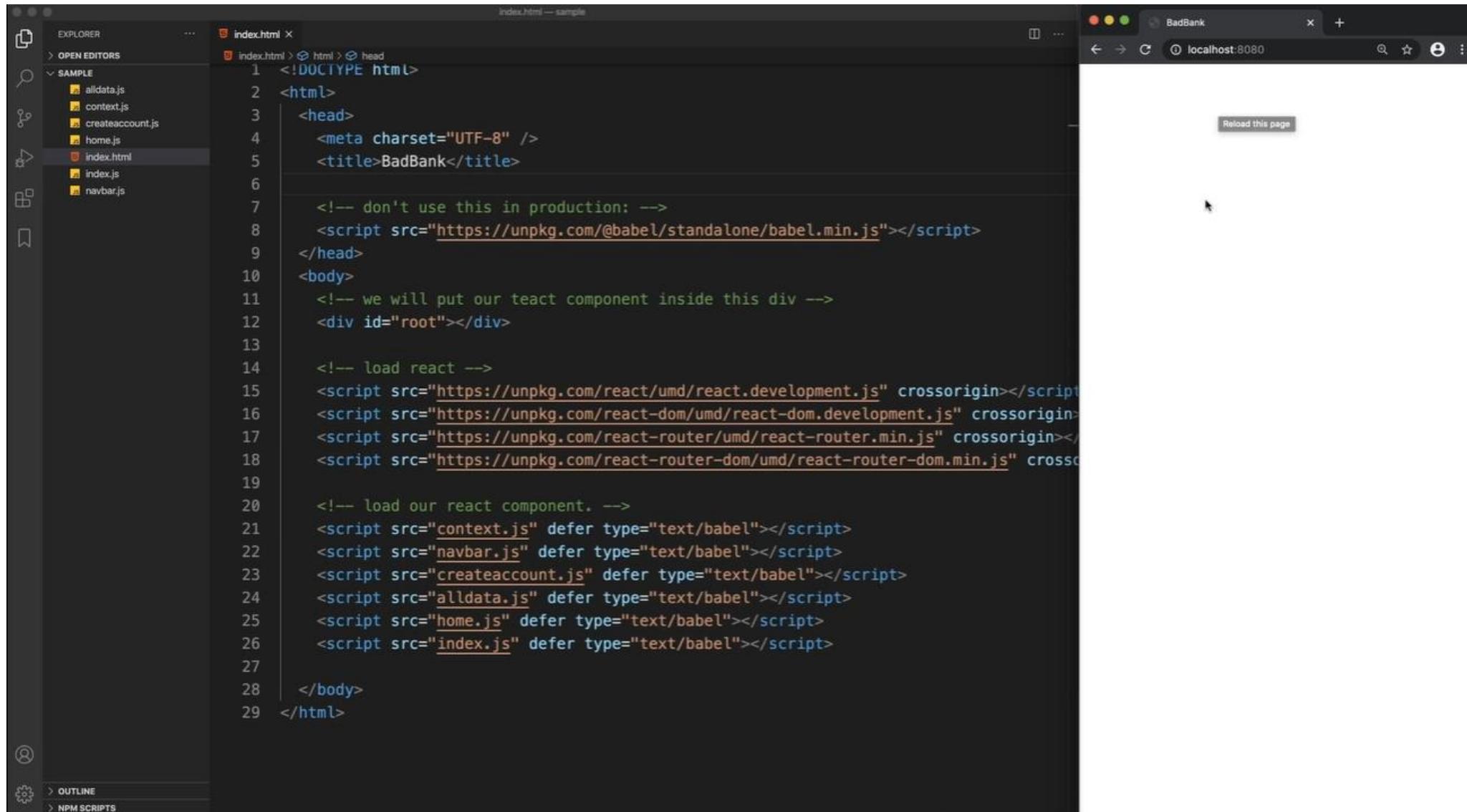
The image shows a split-screen view. On the left is a code editor window titled "index.html — sample". The file contains the following HTML code:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8" />
5          <title>BadBank</title>
6
7          <!-- don't use this in production: -->
8          <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
9      </head>
10     <body>
11         <!-- we will put our React component inside this div -->
12         <div id="root"></div>
13
14         <!-- load React -->
15         <script src="https://unpkg.com/react/umd/react.development.js" crossorigin></script>
16         <script src="https://unpkg.com/react-dom/umd/react-dom.development.js" crossorigin></script>
17         <script src="https://unpkg.com/react-router/umd/react-router.min.js" crossorigin></script>
18         <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.js" crossorigin></script>
19
20         <!-- load our React component. -->
21         <script src="context.js" defer type="text/babel"></script>
22         <script src="navbar.js" defer type="text/babel"></script>
23         <script src="createaccount.js" defer type="text/babel"></script>
24         <script src="alldata.js" defer type="text/babel"></script>
25         <script src="home.js" defer type="text/babel"></script>
26         <script src="index.js" defer type="text/babel"></script>
27
28     </body>
29 </html>
```

On the right is a web browser window titled "localhost:8080" showing a search results page with several items listed:

- Routing - localhost:8080
- localhost - Google Search
- localhost 8000
- localhost ip
- localhost 3000
- React App - localhost:3000
- Add React in One Minute - localhost:808...

# Components Loaded: Demonstration (6/6)



The image shows a code editor and a web browser side-by-side. The code editor on the left displays the file `index.html` with line numbers, showing the HTML structure and imports for a React application. The browser on the right shows the rendered page titled "BadBank" at `localhost:8080`, which is a simple landing page with a "Reload this page" button.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>BadBank</title>
    <!-- don't use this in production: -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>
    <!-- we will put our react component inside this div -->
    <div id="root"></div>
    <!-- load react -->
    <script src="https://unpkg.com/react/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom/umd/react-dom.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-router/umd/react-router.min.js" crossorigin></script>
    <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.js" crossorigin></script>
    <!-- load our react component. -->
    <script src="context.js" defer type="text/babel"></script>
    <script src="navbar.js" defer type="text/babel"></script>
    <script src="createaccount.js" defer type="text/babel"></script>
    <script src="alldata.js" defer type="text/babel"></script>
    <script src="home.js" defer type="text/babel"></script>
    <script src="index.js" defer type="text/babel"></script>
  </body>
</html>
```

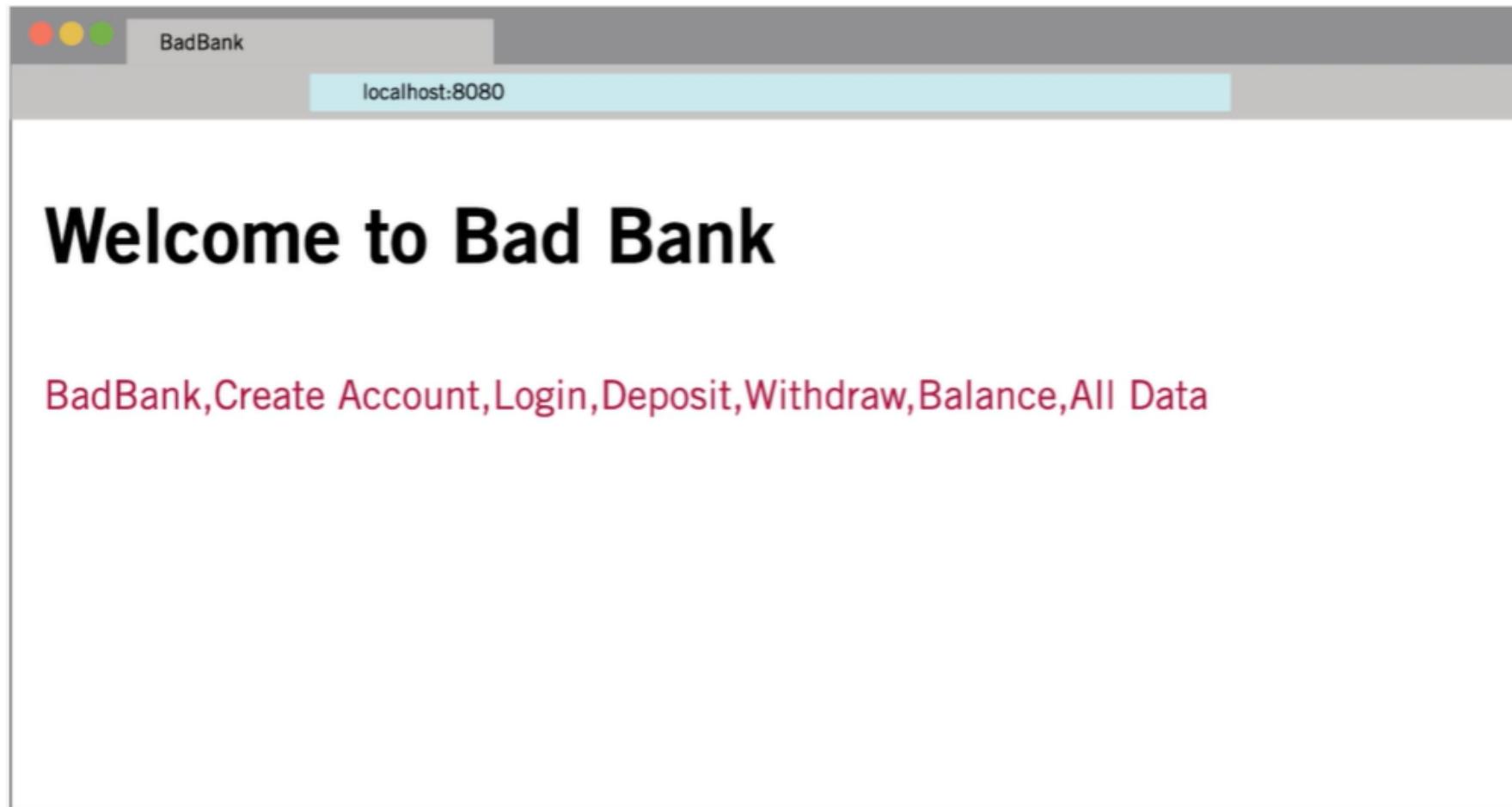
# Navigation Bar

The diagram illustrates a navigation bar for a bank application. At the top, there is a horizontal menu bar with the following items: Badbank, Create Account, Login, Deposit, Withdraw, Balance, and All Data. Below this menu bar is a main content area. The content area has a header section labeled "Badbank Landing Module". Inside this header, there is a welcome message: "Welcome to the bank. You can move around using the navigation bar." Below the welcome message is a stylized icon of a bank building, featuring a red roof and black columns. A single black arrow points from the word "Badbank" in the menu bar down to the "Badbank Landing Module" header.

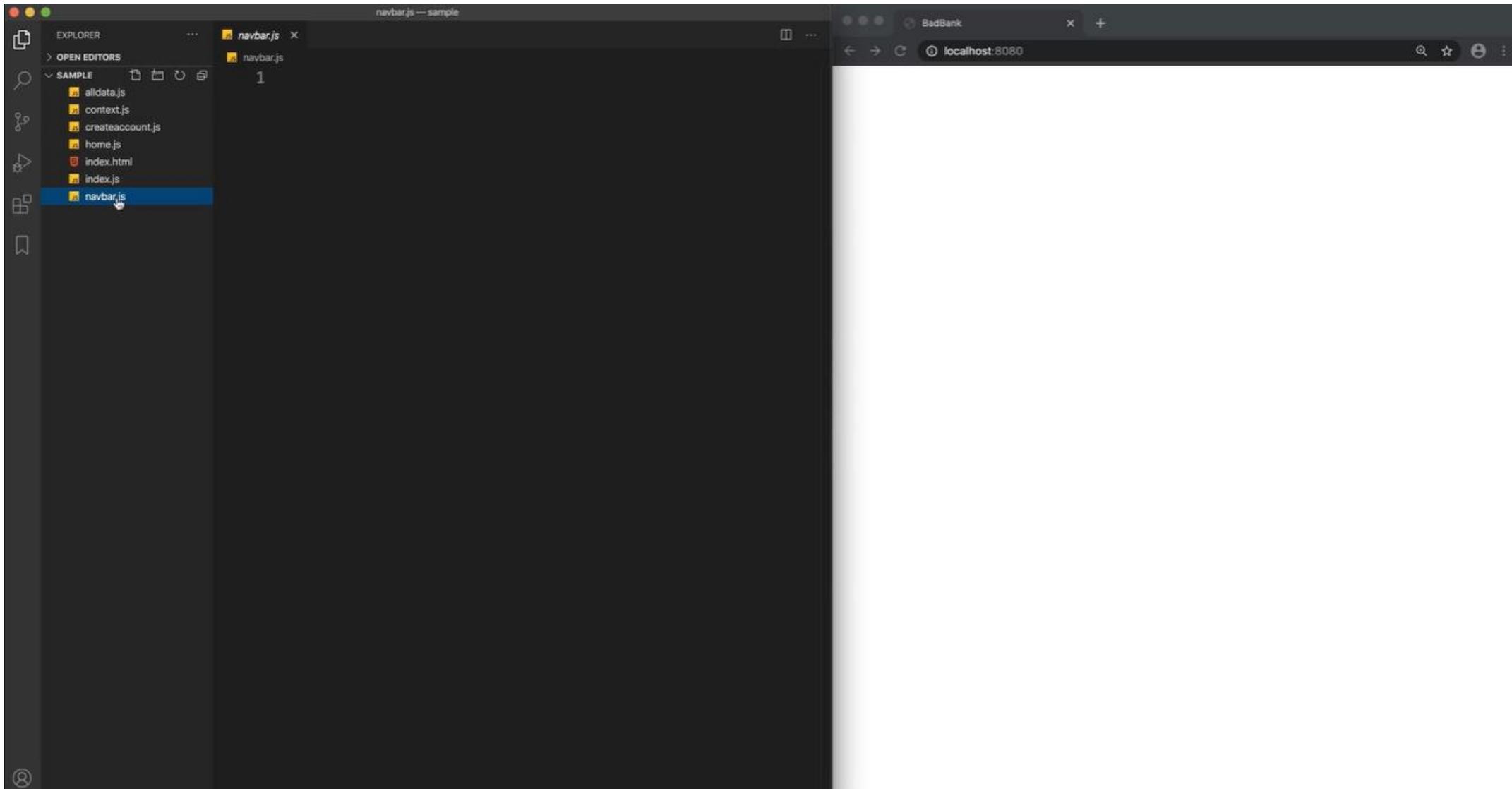
**Components:**

- Home
- Create account
- Login
- Deposit
- Withdraw
- Balance
- All data

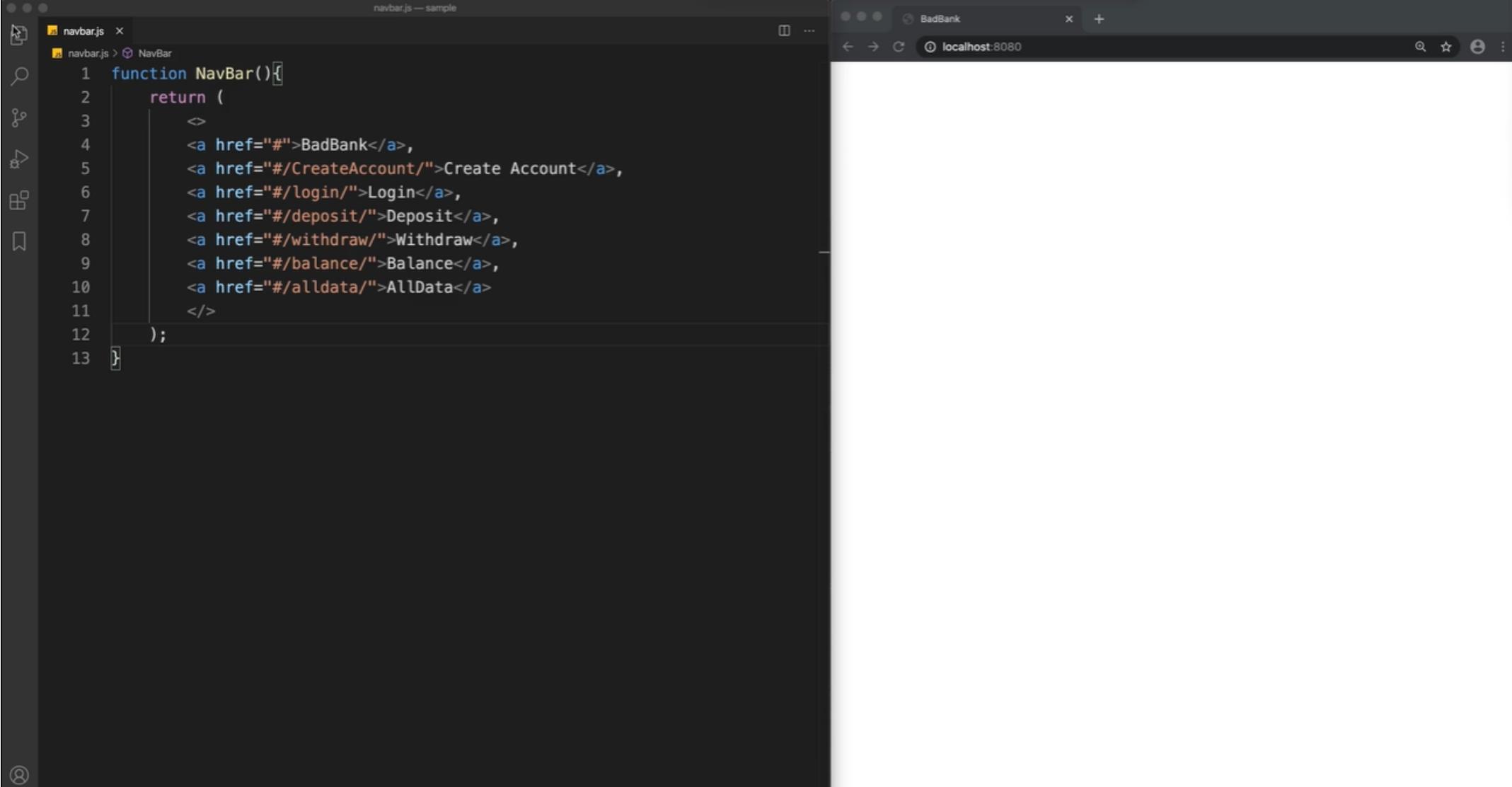
## Output Preview



# Navigation bar: Demonstration (1/5)



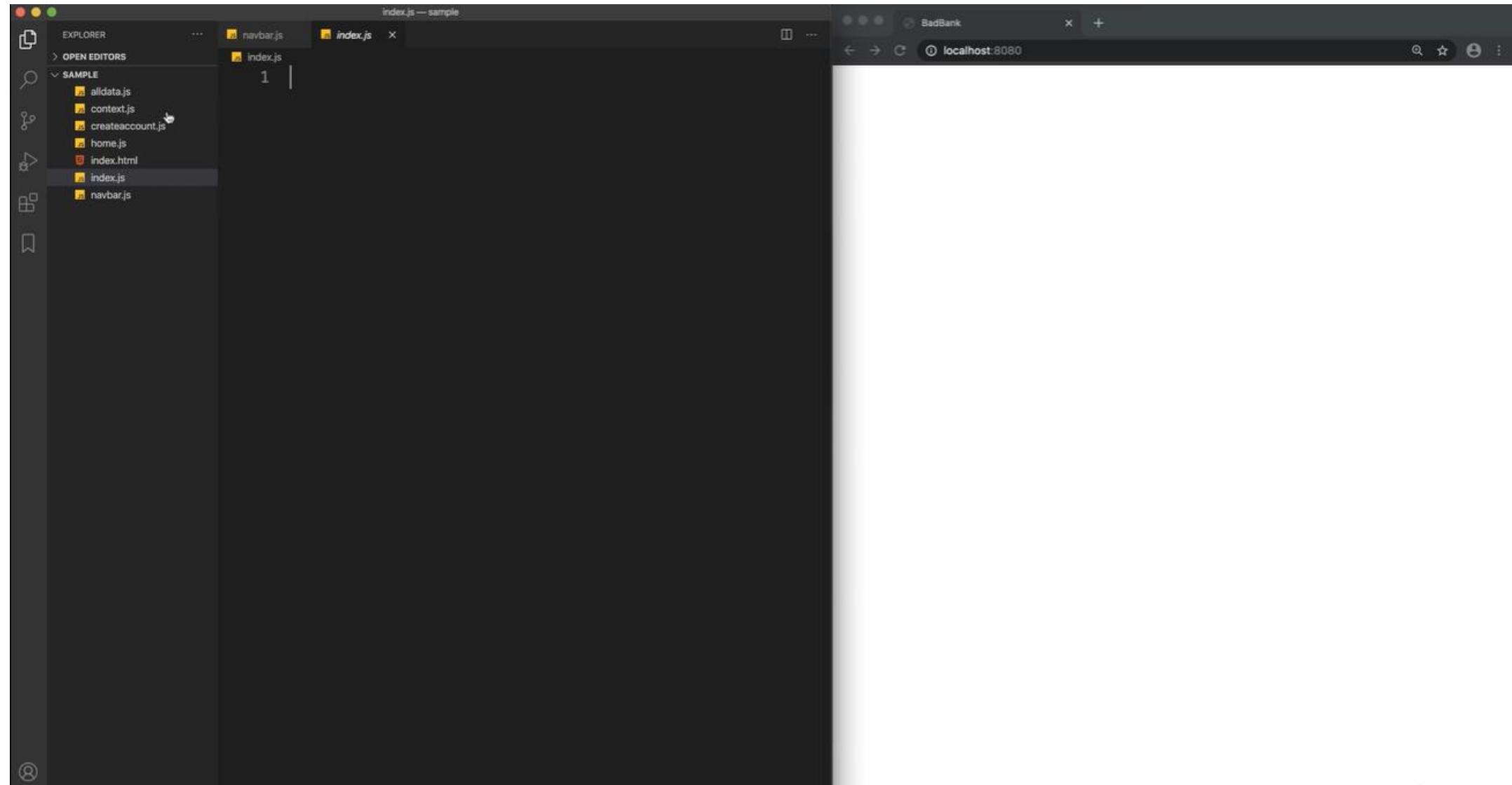
## Navigation bar: Demonstration (2/5)



The image shows a split-screen view. On the left is a code editor with a dark theme, displaying a file named `navbar.js`. The code defines a `NavBar` function that returns a JSX fragment (`<>`) containing several `a` tags with hrefs like `#/CreateAccount/`, `#/login/`, etc. On the right is a web browser window titled `BadBank` with the URL `localhost:8080`. The browser displays the rendered navigation bar with the same links.

```
navbar.js — sample
navbar.js > NavBar
1  function NavBar(){
2    return (
3      <>
4        <a href="#">BadBank</a>,
5        <a href="#/CreateAccount/">Create Account</a>,
6        <a href="#/login/">Login</a>,
7        <a href="#/deposit/">Deposit</a>,
8        <a href="#/withdraw/">Withdraw</a>,
9        <a href="#/balance/">Balance</a>,
10       <a href="#/alldata/">AllData</a>
11     </>
12   );
13 }
```

# Navigation bar: Demonstration (3/5)



# Navigation bar: Demonstration (4/5)

The image shows a split-screen environment. On the left is a code editor with a dark theme, displaying the file `index.js`. The code defines a function `Spa()` which returns a component structure. It includes an `<h1>` element with the text "Welcome to bad bank", a `<NavBar/>` component, and a closing bracket. Below this is a `ReactDOM.render` call that renders the `<Spa/>` component into the `document.getElementById('root')` element. On the right is a web browser window titled "BadBank" showing the URL `localhost:8080`. The page displays the text "Welcome to bad bank" above a navigation bar, which is the output of the `<NavBar/>` component.

```
index.js — sample
navbar.js index.js ...
index.js > ...
1 function Spa(){
2     return (
3         <>
4             <h1>Welcome to bad bank</h1>
5             <NavBar/>
6         </>
7     );
8 }
9
10 ReactDOM.render(
11     <Spa/>,
12     document.getElementById('root')
13 )
```

# Navigation bar: Demonstration (5/5)

The screenshot illustrates a development environment where a single-page application (SPA) is being built and tested. On the left, a code editor displays the file `index.js` with the following content:

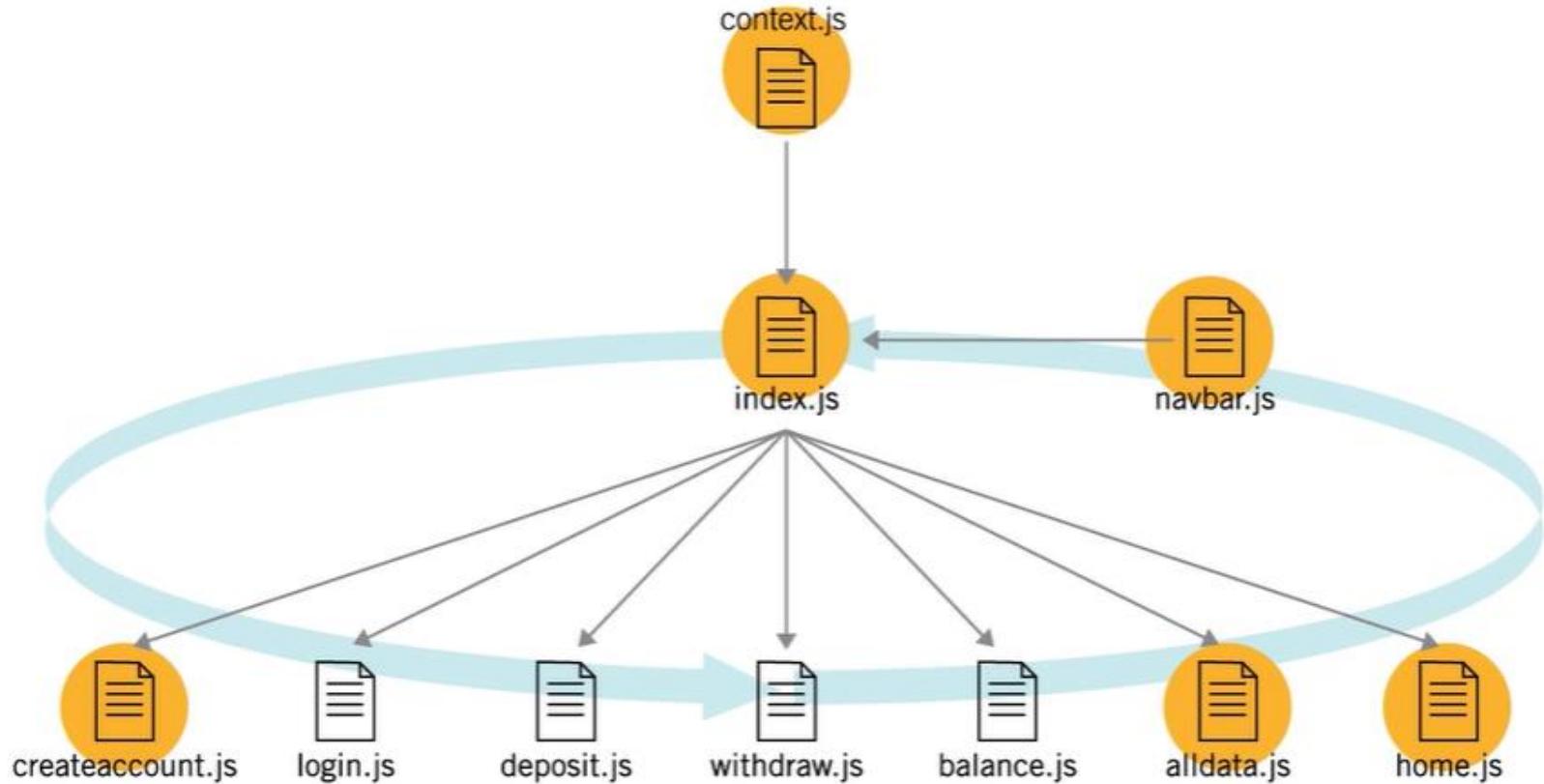
```
index.js — sample
1  function Spa(){
2      return (
3          <>
4              <h1>Welcome to bad bank</h1>
5              <NavBar/>
6          </>
7      );
8  }
9
10 ReactDOM.render(
11     <Spa/>,
12     document.getElementById('root')
13 )
```

On the right, a web browser window titled "BadBank" shows the rendered SPA at `localhost:8080`. The page displays the text "Welcome to bad bank" and a navigation bar with the following links:

Welcome to bad bank

[BadBank](#), [Create](#)  
[Account](#), [Login](#), [Deposit](#), [Withdraw](#), [Balance](#), [AllData](#)

# Routing



# Routing – Route, Link, HashRouter

```
const Route = ReactRouterDOM.Route;  
const Link = ReactRouterDOM.Link;  
const HashRouter = ReactRouterDOM.HashRouter;
```

# Navbar Routing: Demonstration (1/9)

The image shows a split-screen development environment. On the left, the code editor displays the file `context.js` with the following content:

```
const Route = ReactRouterDOM.Route;
const Link = ReactRouterDOM.Link;
const HashRouter = ReactRouterDOM.HashRouter;
```

The file is part of a project structure named "SAMPLE" which includes files like `alldata.js`, `context.js`, `createaccount.js`, `home.js`, `index.html`, `index.js`, and `navbar.js`. The code editor interface includes a sidebar with icons for file operations and a status bar at the bottom.

On the right, a web browser window titled "BadBank" is open at the URL `localhost:8080`. The page content is:

# Welcome to bad bank

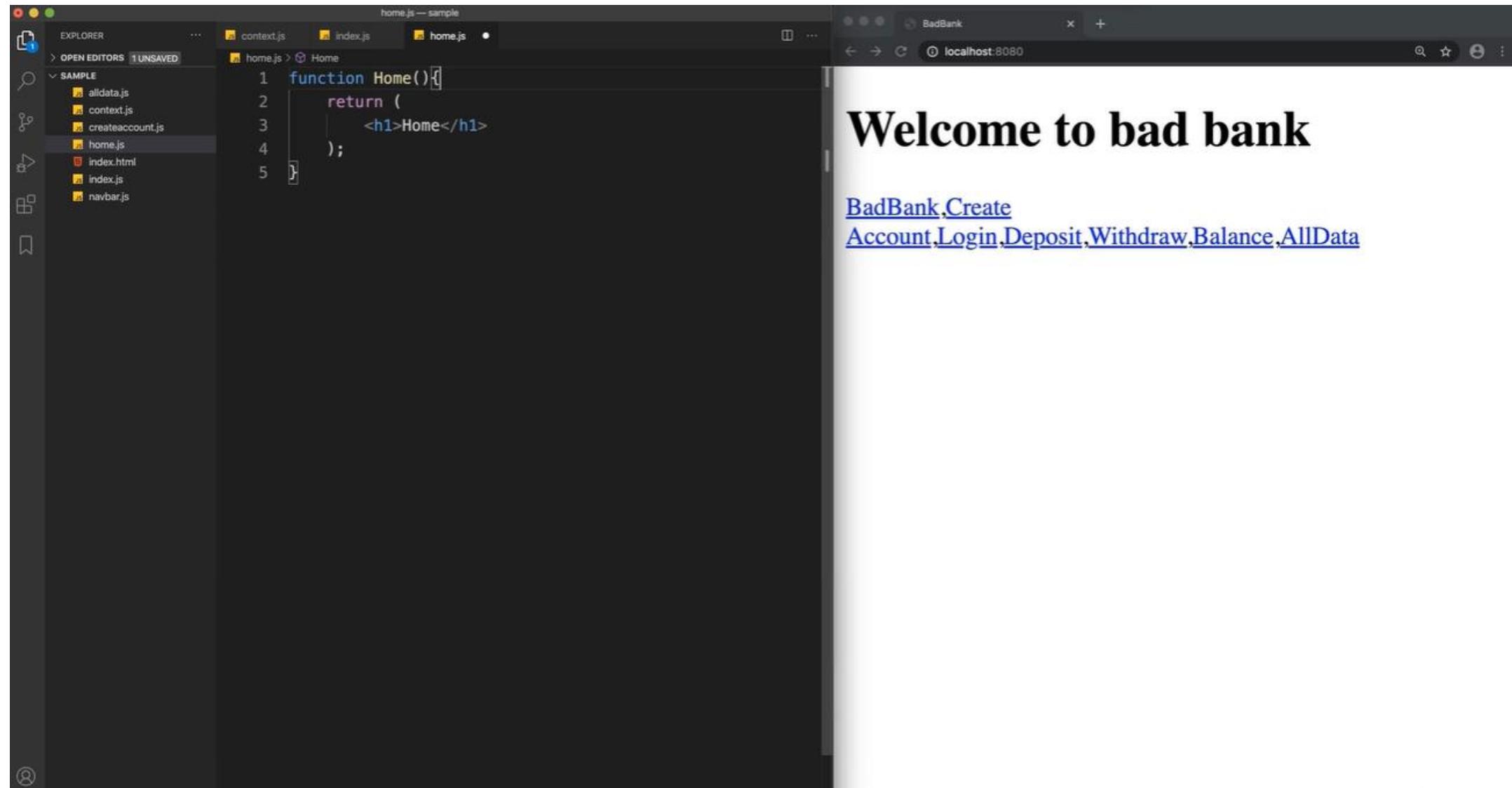
[BadBank.Create](#)  
[Account](#) [Login](#) [Deposit](#) [Withdraw](#) [Balance](#) [AllData](#)

## Navbar Routing: Demonstration (2/9)

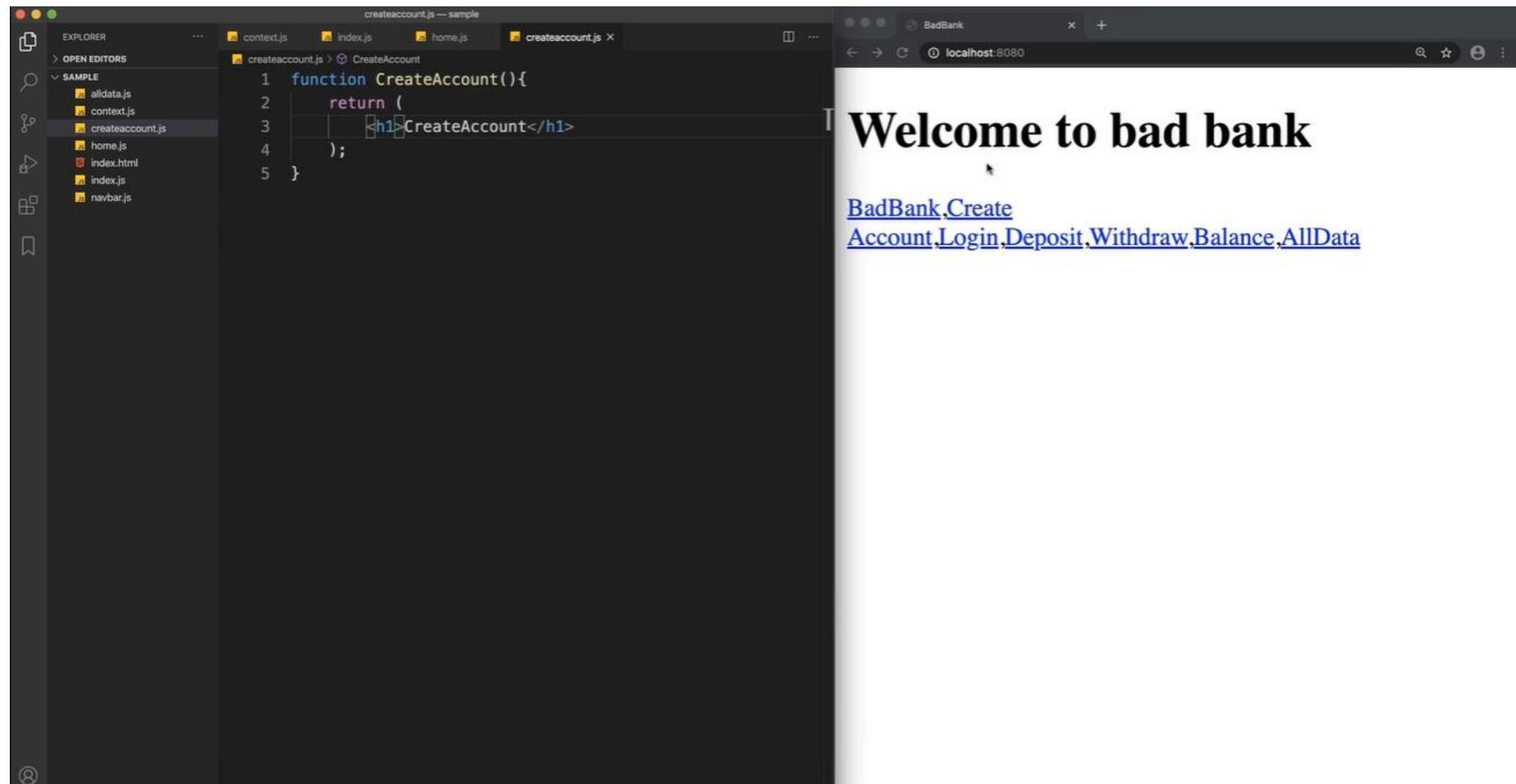
The image shows a split-screen environment. On the left is a code editor with a dark theme, displaying the file `index.js`. The code defines a function `Spa()` which returns a `<HashRouter>` component. Inside the router, there is a `<NavBar/>` component and several `<Route>` components mapping specific URLs to different components: `/` to `home`, `/CreateAccount/` to `CreateAccount`, `/login/` to `Login`, `/deposit/` to `Deposit`, `/withdraw/` to `Withdraw`, `/balance/` to `Balance`, and `/alldata/` to `AllData`. The code ends with `</HashRouter>` and `ReactDOM.render` calling `<Spa/>` with the root element. On the right is a web browser window titled "BadBank" with the URL `localhost:8080`. The page displays the text "Welcome to bad bank" and a list of underlined links: `BadBank`, `Create`, `Account`, `Login`, `Deposit`, `Withdraw`, `Balance`, and `AllData`.

```
index.js — sample
1  function Spa(){
2      return (
3          <HashRouter>
4              <NavBar/>
5              <Route path="/" exact component={home} />
6              <Route path="/CreateAccount/" component={CreateAccount} />
7              <Route path="/login/" component={Login} />
8              <Route path="/deposit/" component={Deposit} />
9              <Route path="/withdraw/" component={Withdraw} />
10             <Route path="/balance/" component={Balance} />
11             <Route path="/alldata/" component={AllData} />
12         </HashRouter>
13     );
14 }
15
16 ReactDOM.render(
17     <Spa/>,
18     document.getElementById('root')
19 )
```

# Navbar Routing: Demonstration (3/9)



# Navbar Routing: Demonstration (4/9)



# Navbar Routing: Demonstration (5/9)

The image shows a split-screen environment. On the left, a code editor displays the file `alldata.js` with the following content:

```
function AllData(){
    return (
        <h1>AllData</h1>
    );
}
```

The code editor's sidebar shows other files like `context.js`, `index.js`, `home.js`, and `createaccount.js`. On the right, a web browser window titled "BadBank" is open at `localhost:8080`. The page content is:

# Welcome to bad bank

[BadBank](#), [Create](#)  
[Account](#), [Login](#), [Deposit](#), [Withdraw](#), [Balance](#), [AllData](#)

# Navbar Routing: Demonstration (6/9)

The screenshot shows a development environment with a code editor and a browser window.

**Code Editor (VS Code):**

- Explorer:** Shows a project structure with files: context.js, index.js, home.js, createaccount.js, and alldata.js.
- Open Editors:** The file `alldata.js` is open, displaying the following code:

```
function AllData(){
  return (
    <h1>AllData</h1>
  );
}
```

**Browser Window:**

- Title: BadBank
- URL: localhost:8080
- Content: An empty white page.

**Console (React DevTools):**

- Elements tab is selected.
- Console tab is selected.
- Message: "Some messages have been moved to the Issues panel." with a link to "View issues".
- Warning message: "react-dom.development.js:25129 Download the React DevTools for a better development experience: <https://fb.me/react-devtools>".
- Warning message: "You are using the in-browser Babel transformer. Be sure to precompile your scripts for production – <https://babeljs.io/docs/setup/>".
- Error message: "Uncaught ReferenceError: home is not defined" with stack trace:

```
at Spa (<anonymous>:7:16)
at renderWithHooks (react-dom.development.js:14938)
at mountIndeterminateComponent (react-dom.development.js:17617)
at beginWork (react-dom.development.js:18731)
at HTMLUnknownElement.callCallback (react-dom.development.js:182)
```

# Navbar Routing: Demonstration (7/9)

The screenshot shows a code editor on the left and a browser window on the right.

**Code Editor (VS Code):**

- Explorer:** Shows files: context.js, index.js, home.js, createaccount.js, alldata.js.
- Editor:** Opened file: index.js — sample. The code defines a `Spa()` function that returns a `<HashRouter>` component. Inside it, there's a `<NavBar/>` component and three `<Route>` components for paths: "/", "/CreateAccount", and "/alldata". Finally, it closes with a `</HashRouter>`.
- Console:** Shows developer tools for the browser.

**Browser:** Title: BadBank. Address: localhost:8080/#/. The page displays the word "Home" in large, bold, black font. At the top, there are several blue links: BadBank, Create, Account, Login, Deposit, Withdraw, Balance, AllData.

**Console (Browser):**

- Elements tab is selected.
- Console tab shows some messages:
  - A warning message: "Some messages have been moved to the Issues panel." with a link to "View issues".
  - A message from "react-dom.development.js:25129": "Download the React DevTools for a better development experience: <https://fb.me/react-devtools>".
  - A warning message: "You are using the in-browser Babel transformer. Be sure to precompile your scripts for production - <https://babeljs.io/docs/setup/>".

# Navbar Routing: Demonstration (8/9)

The image shows a development environment with a code editor and a browser window.

**Code Editor (VS Code):**

- Explorer:** Shows files: context.js, index.js, home.js, createaccount.js, alldata.js.
- Editor:** Opened file: index.js (sample). The code defines a `Spa()` function that returns a `<HashRouter>` component. Inside it, there's a `<NavBar/>`, followed by three `<Route>` components: one for the root path (exact) pointing to `Home`, one for the `/CreateAccount` path pointing to `CreateAccount`, and one for the `/alldata` path pointing to `AllData`. Finally, it closes with a `</HashRouter>`.
- Console:** Shows a warning message: "Some messages have been moved to the Issues panel." It also includes links to "react-dom.development.js:25129", "View issues", and "Download the React DevTools for a better development experience: <https://fb.me/react-devtools>".

**Browser:**

- Title Bar:** BadBank
- Address Bar:** localhost:8080/#/CreateAccount/
- Content:** Displays the heading "CreateAccount".
- Console:** Shows the same warning message as the code editor, along with a yellow box containing a warning about using the in-browser Babel transformer and a link to "babel.min.js:1".

# Navbar Routing: Demonstration (9/9)

The screenshot displays a development environment with several components:

- Code Editor:** Shows the `index.js` file containing the following code:

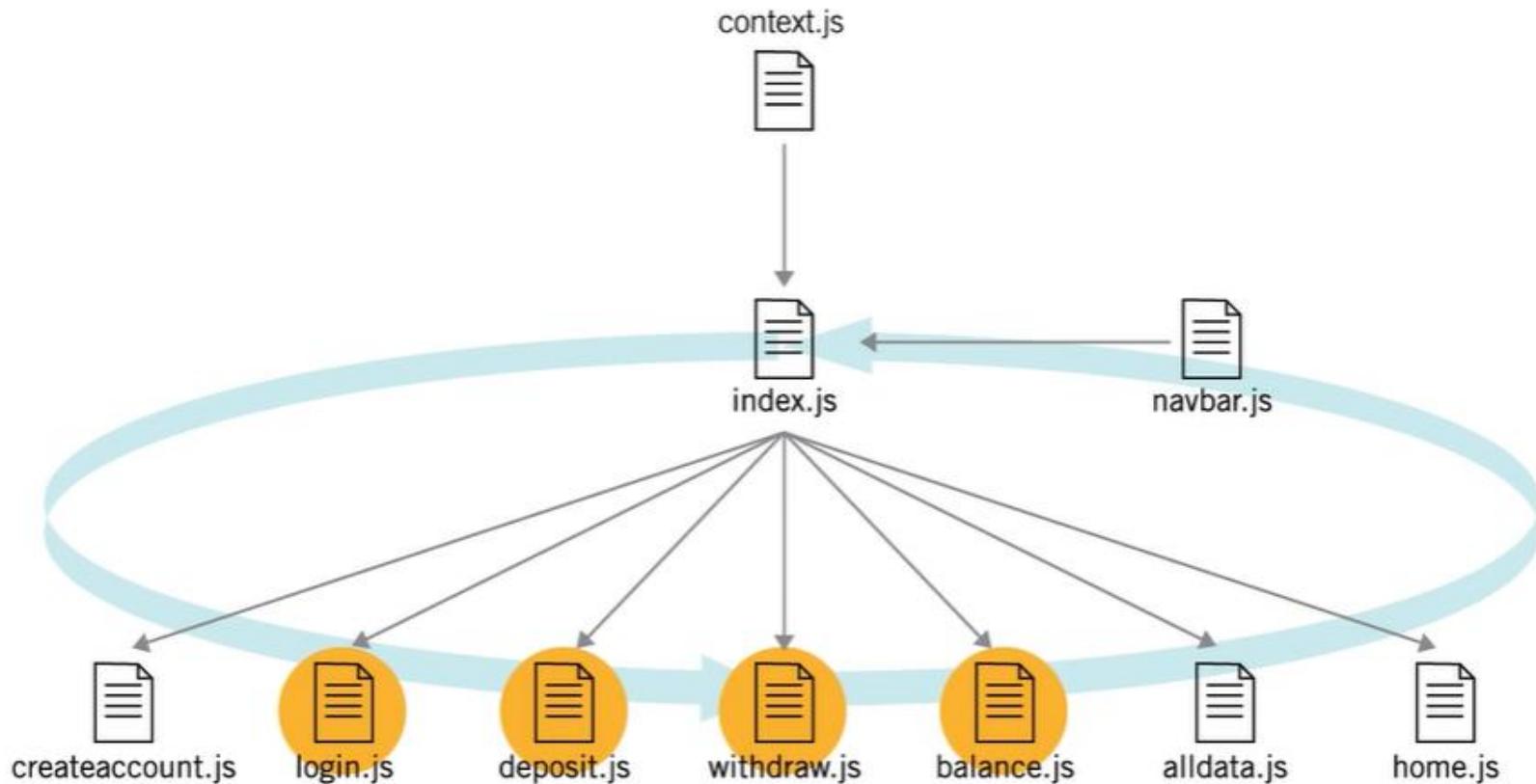
```
function Spa(){
  return (
    <HashRouter>
      <NavBar/>
      <Route path="/" exact component={Home} />
      <Route path="/CreateAccount/" component={CreateAccount} />
      <Route path="/alldata/" component={AllData} />
    </HashRouter>
  );
}

ReactDOM.render(
  <Spa/>,
  document.getElementById('root')
)
```
- Browser Preview:** A browser window titled "BadBank" shows the URL `localhost:8080/#/alldata/`. The page content includes:

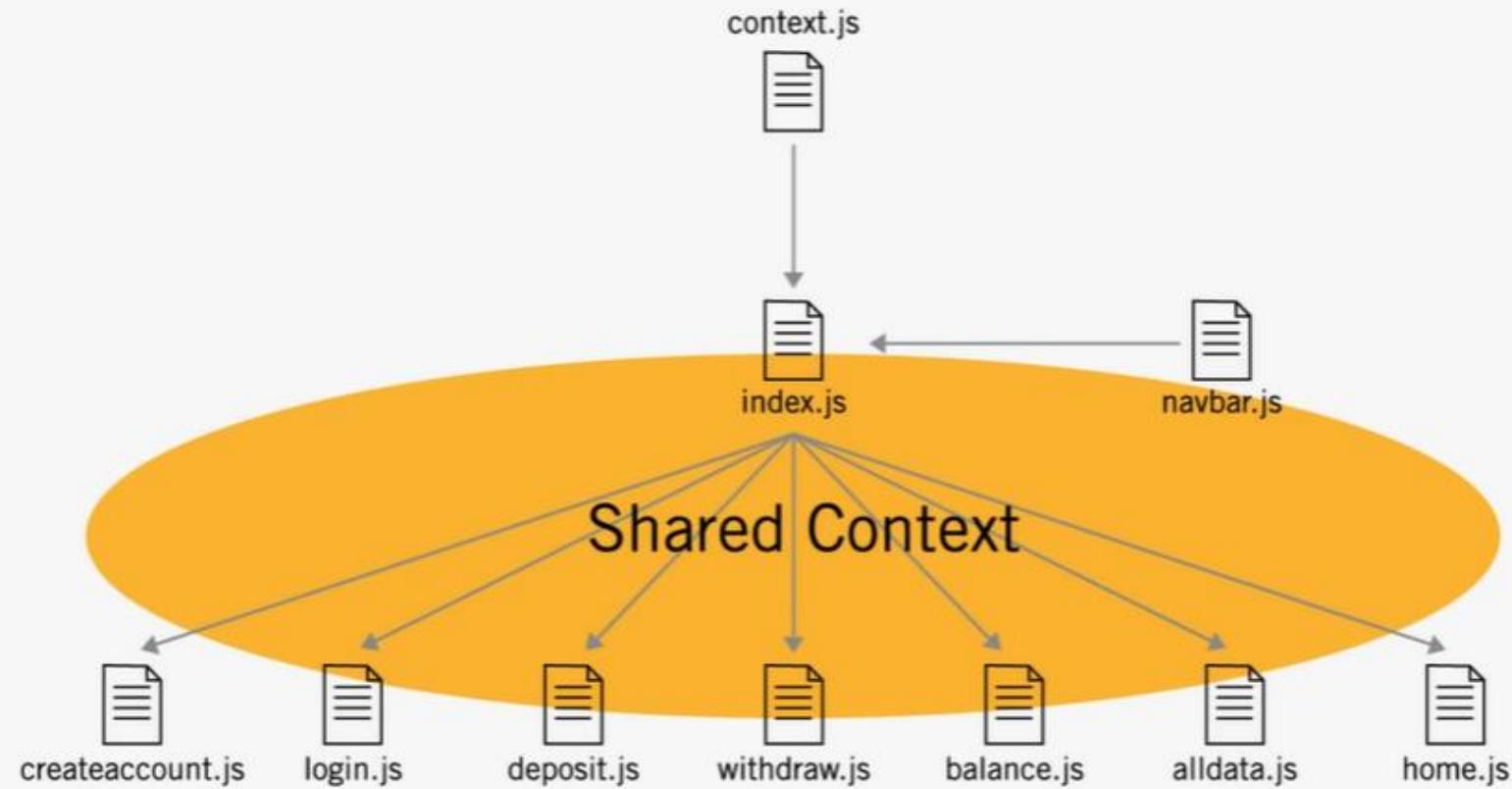
BadBank.Create  
Account,Login,Deposit,Withdraw,Balance,AllData

# AllData
- Developer Tools:** An open console window shows the following messages:
  - A message indicating some messages have been moved to the Issues panel, with a link to [View issues](#).
  - A warning from `react-dom.development.js:25129` encouraging the download of [React DevTools](#) for better development experience, with a link to <https://fb.me/react-devtools>.
  - A yellow warning message about using the in-browser Babel transformer, advising precompilation for production, with a link to <https://babeljs.io/docs/setup/>.

# Navbar Routing: Your Turn



# Shared Context



# Create Context

```
Const Route = ReactRouterDOM.Route;  
Const Link = ReactRouterDOM.link;  
Const HashRouter = ReactRouterDOM.HashRouter;  
Const UserContext = React.createContext(null);
```

- Context object comes with a provider that takes a value.
- The value is passed to consuming components.

# Shared Context: Demonstration (1/8)

The image shows a split-screen view. On the left, a code editor displays the file `context.js` with the following code:

```
const Route      = ReactRouterDOM.Route;
const Link      = ReactRouterDOM.Link;
const HashRouter = ReactRouterDOM.HashRouter;
const UserContext = React.createContext(null);
```

The code editor's sidebar shows other files in the `SAMPLE` folder: `alldata.js`, `context.js`, `createaccount.js`, `home.js`, `index.html`, `index.js`, and `navbar.js`. The `context.js` file is currently selected.

On the right, a web browser window titled "BadBank" shows the URL `localhost:8080/#`. The page content includes the word "Home" and several underlined links: BadBank, Create, Account, Login, Deposit, Withdraw, Balance, and AllData.

# Shared Context: Demonstration (2/8)

The image shows a split-screen environment. On the left, a code editor displays the file `index.js` with the following content:

```
index.js — sample
1 function Spa(){
2     return (
3         <HashRouter>
4             <NavBar/>
5             <UserContext.Provider value={users:[{name:'abel',
6                 email:'abel@mit.edu',password:'secret',balance:100}]}>
7                 <Route path="/" exact component={Home} />
8                 <Route path="/CreateAccount/" component={CreateAccount} />
9                 <Route path="/alldata/" component={AllData} />
10            </UserContext.Provider>
11        </HashRouter>
12    );
13
14 ReactDOM.render(
15     <Spa/>,
16     document.getElementById('root')
17 )
```

On the right, a web browser window titled "BadBank" shows the application running at `localhost:8080/#`. The page displays a large heading "Home" and a list of navigation links:

[BadBank](#), [Create Account](#), [Login](#), [Deposit](#), [Withdraw](#), [Balance](#), [AllData](#)

# Shared Context: Demonstration (3/8)

The image shows a split-screen demonstration. On the left, a code editor displays the file `home.js` with the following code:

```
function Home(){
  const ctx = React.useContext(UserContext);
  return (
    <h1>Home<br/>
    |   {JSON.stringify(ctx)}
    </h1>
  );
}
```

The code uses the `useContext` hook to access the `UserContext`. On the right, a web browser window titled "BadBank" shows the rendered output at `localhost:8080/#/`. The page displays the word "Home" in large bold letters, with the context value printed below it as a JSON string: `BadBank,Create Account,Login,Deposit,Withdraw,Balance,AllData`.

# Shared Context: Demonstration (4/8)

The screenshot illustrates the use of shared context in a React application. On the left, the code editor shows the file `createaccount.js` with the following content:

```
1 function CreateAccount(){
2     const ctx = React.useContext(UserContext);
3     return (
4         <h1>Create Account<br/>
5             {JSON.stringify(ctx)}
6     );
7 }
8 }
```

The file is part of a project structure under the `SAMPLE` folder, which also includes `alldata.js`, `context.js`, `home.js`, `index.html`, `index.js`, and `navbar.js`. The `context.js` file is currently selected in the Explorer sidebar.

On the right, a browser window displays the application at `localhost:8080/#/`. The page title is `BadBank`. The content area contains the text `BadBank,Create Account,Login,Deposit,Withdraw,Balance,AllData` followed by a large `Home` heading.

## Shared Context: Demonstration (5/8)

The image shows a split-screen view. On the left is a code editor (VS Code) displaying a file named `alldata.js` with the following content:

```
function AllData(){
  const ctx = React.useContext(UserContext);
  return (
    <h1>All Data<br/>
    {JSON.stringify(ctx)}
  );
}
```

The code editor's sidebar shows other files: `context.js`, `index.js`, `home.js`, `createaccount.js`, and `index.html`. The browser window on the right shows the URL `localhost:8080/#` and displays the rendered output of the `AllData` component. The output is a large block of JSON text: `BadBank,Create  
Account,Login,Deposit,Withdraw,Balance,AllData`. Below this, the word `Home` is displayed.

## Shared Context: Demonstration (6/8)

The image shows a code editor on the left and a browser window on the right.

**Code Editor (VS Code):**

- File: `alldata.js — sample`
- Editor tab: `alldata.js`
- Content:

```
1 function AllData(){
2   const ctx = React.useContext(UserContext);
3   return (
4     <h1>All Data<br/>
5       {JSON.stringify(ctx)}
6     </h1>
7   );
8 }
```

**Browser Window:**

- Title: BadBank
- URL: `localhost:8080/#`
- Content:

BadBank,Create  
Account,Login,Deposit,Withdraw,Balance,AllData

# Home

{"users":  
[{"name":"abel","email":"abe@badbank.com"}]}

# Shared Context: Demonstration (7/8)

The image shows a split-screen demonstration. On the left, a code editor displays the file `createaccount.js` with the following code:

```
function CreateAccount(){
  const ctx = React.useContext(UserContext);
  return (
    <h1>Create Account<br/>
    {JSON.stringify(ctx)}
  );
}
```

The code uses the `React.useContext` hook to access the `UserContext`. On the right, a web browser window titled "BadBank" shows the rendered output at `localhost:8080/#/CreateAccount/`. The page title is "BadBank,Create". Below it, there are several blue underlined links: "Account", "Login", "Deposit", "Withdraw", "Balance", and "AllData". The main content area displays the heading "Create Account" and a JSON object:

**Create Account**

{"users":  
[{"name":"abel","email":"abe@badbank.com"}]}

# Shared Context: Demonstration (8/8)

The image shows a split-screen environment. On the left, a code editor displays the file `createaccount.js` with the following code:

```
function CreateAccount(){
  const ctx = React.useContext(UserContext);
  return (
    <h1>Create Account<br/>
    {JSON.stringify(ctx)}
  );
}
```

The code editor's sidebar shows other files: `context.js`, `index.js`, `home.js`, `alldata.js`, `createaccount.js`, `home.js`, `index.html`, `index.js`, and `navbar.js`. The `createaccount.js` file is currently selected.

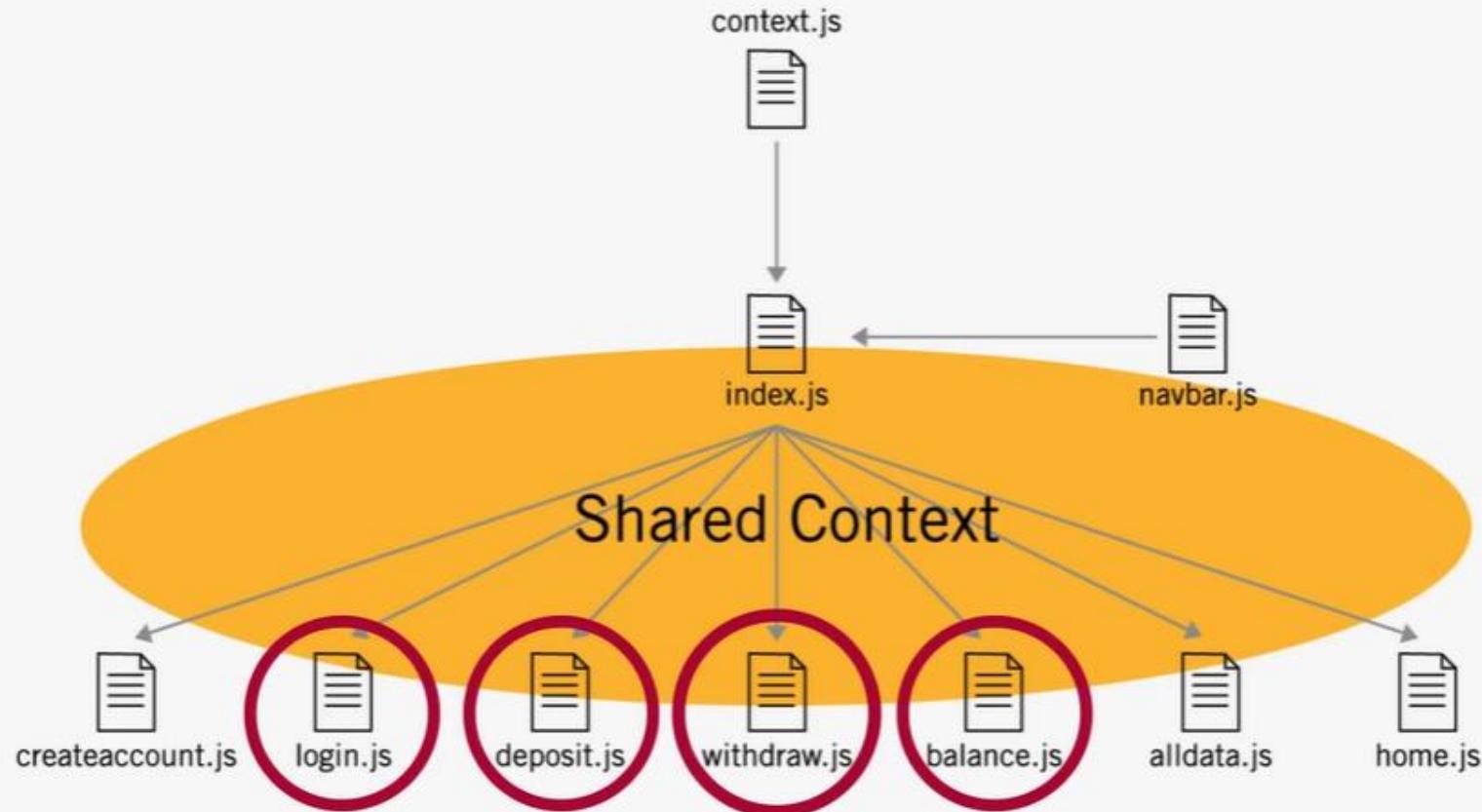
On the right, a web browser window titled "BadBank" shows the URL `localhost:8080/#/alldata/`. The page content includes some navigation links and the following JSON data:

BadBank,Create  
Account,Login,Deposit,Withdraw,Balance,AllData

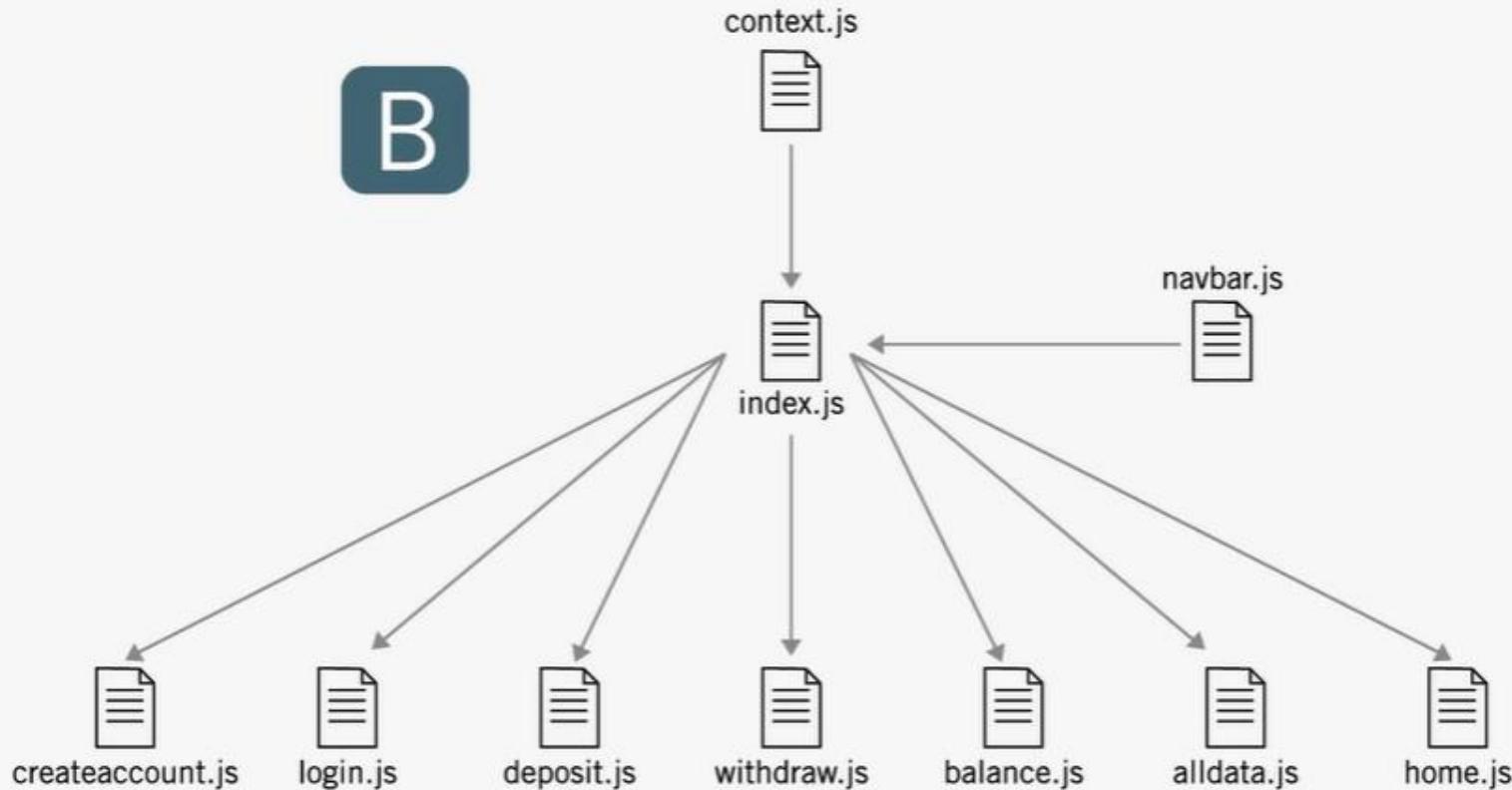
**All Data**

{"users":  
[{"name":"abel","email":"abe@badbank.com"}]}

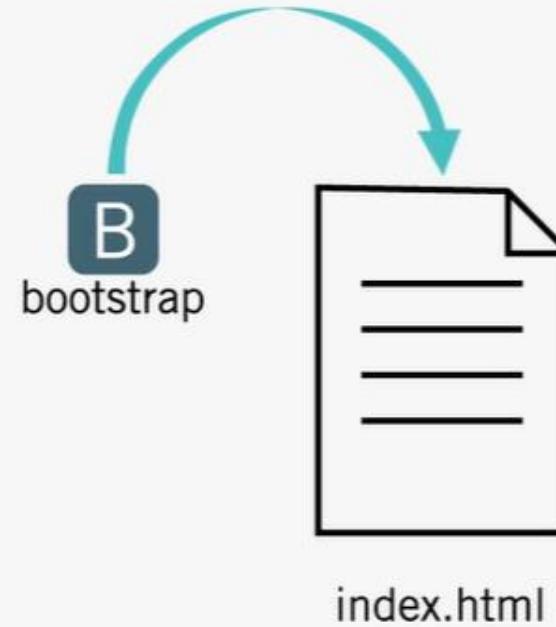
# Shared Context: Your Turn



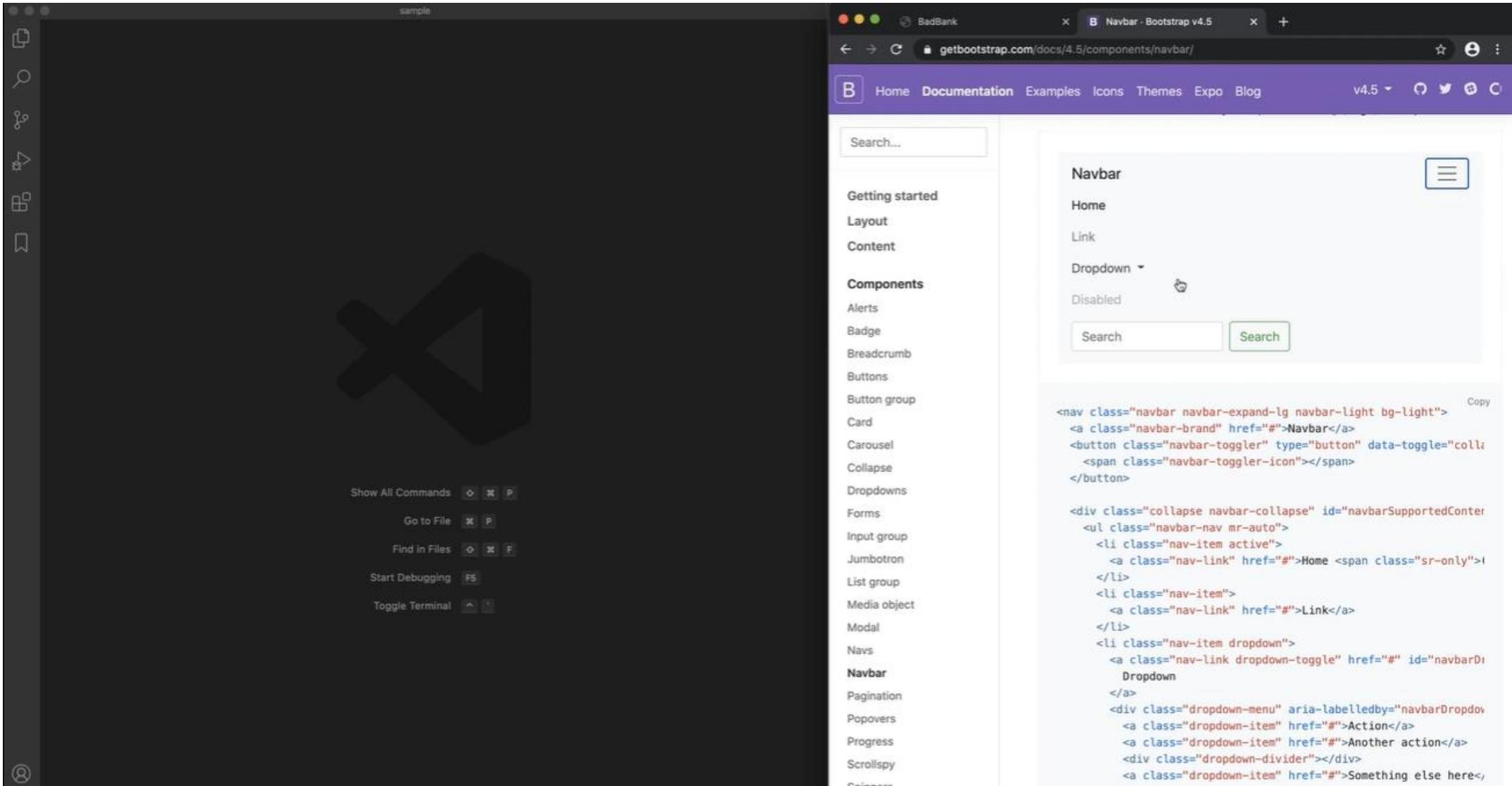
# Styles – Bootstrap



# Add Bootstrap Styles



# Add Bootstrap Styles: Demonstration (1/7)

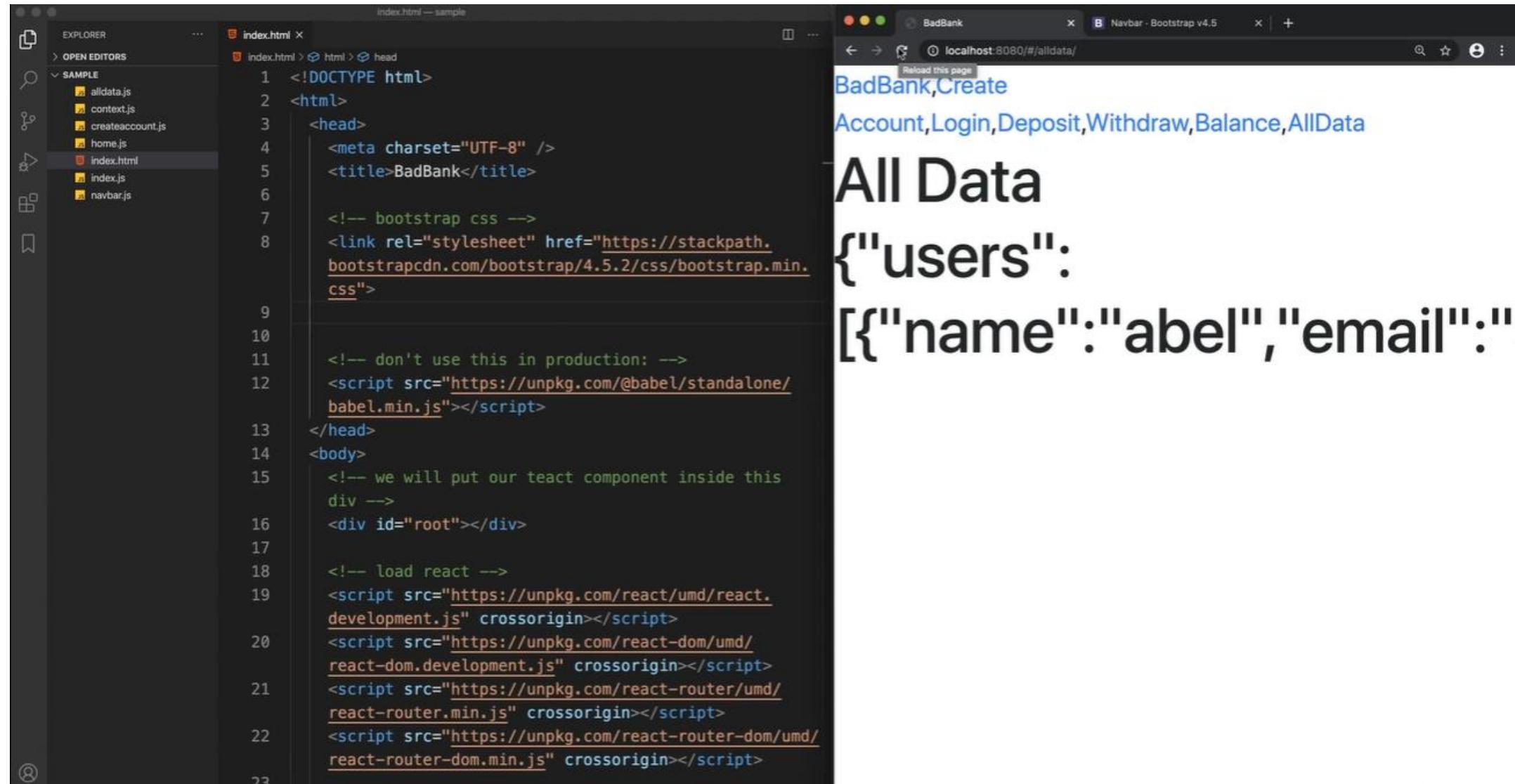


The image shows a split-screen view. On the left is a dark-themed Microsoft Visual Studio Code interface with a large 'X' logo in the center. On the right is a web browser window displaying the Bootstrap documentation at [getbootstrap.com/docs/4.5/components/navbar/](https://getbootstrap.com/docs/4.5/components/navbar/). The browser has a purple header bar with the Bootstrap logo and navigation links. The main content area shows the 'Navbar' component documentation, featuring a search bar and a sidebar with navigation links. Below the sidebar is a code snippet for the Navbar component:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Dropdown</a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li>
    </ul>
  </div>
</nav>
```

## Add Bootstrap Styles: Demonstration (2/7)



The image shows a split-screen development environment. On the left, the code editor displays the file `index.html` with the following content:

```
index.html — sample
index.html
  index.html > head
    1  <!DOCTYPE html>
    2  <html>
    3    <head>
    4      <meta charset="UTF-8" />
    5      <title>BadBank</title>
    6
    7      <!-- bootstrap css -->
    8      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    9
   10     <!-- don't use this in production: -->
   11     <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
   12
   13    </head>
   14    <body>
   15      <!-- we will put our react component inside this div -->
   16      <div id="root"></div>
   17
   18      <!-- load react -->
   19      <script src="https://unpkg.com/react/umd/react.development.js" crossorigin></script>
   20      <script src="https://unpkg.com/react-dom/umd/react-dom.development.js" crossorigin></script>
   21      <script src="https://unpkg.com/react-router/umd/react-router.min.js" crossorigin></script>
   22      <script src="https://unpkg.com/react-router-dom/umd/react-router-dom.min.js" crossorigin></script>
```

On the right, a browser window titled "BadBank" shows the rendered page at `localhost:8080/#/alldata/`. The page displays the following content:

BadBank,Create  
Account,Login,Deposit,Withdraw,Balance,AllData

# All Data

{"users":  
[{"name": "abel", "email": "

# Add Bootstrap Styles: Demonstration (3/7)

The image shows a side-by-side comparison between a code editor and a web browser.

**Code Editor (Left):** A screenshot of a code editor window titled "navbar.js — sample". It contains the following JavaScript code:

```
function NavBar(){
    return (
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <a class="navbar-brand" href="#">Navbar</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
                <div class="navbar-nav">
                    <a class="nav-link active" href="#">Home <span class="sr-only">(current)</span>
                    <a class="nav-link" href="#">Features</a>
                    <a class="nav-link" href="#">Pricing</a>
                    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
                </div>
            </div>
        </nav>
    );
}
```

**Browser (Right):** A screenshot of a web browser window titled "BadBank" showing the "Navbar - Bootstrap v4.5" page at [getbootstrap.com/docs/4.5/components/navbar/](https://getbootstrap.com/docs/4.5/components/navbar/). The page displays the Bootstrap 4.5 Navbar component documentation. It includes a search bar, navigation links, and a sidebar with a list of components. Two examples of navbar code are shown:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
            <a class="nav-link active" href="#">Home <span class="sr-only">(current)</span>
            <a class="nav-link" href="#">Features</a>
            <a class="nav-link" href="#">Pricing</a>
            <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
        </div>
    </div>
</nav>
```

Below the first example, there is explanatory text: "And because we use classes for our navs, you can avoid the list-based approach entirely if you like."

Below the second example, there is explanatory text: "You may also utilize dropdowns in your navbar nav. Dropdown menus require a wrapping element for positioning, so be sure to use separate and nested elements for .nav-item and .nav-link as shown below."

## Add Bootstrap Styles: Demonstration (4/7)

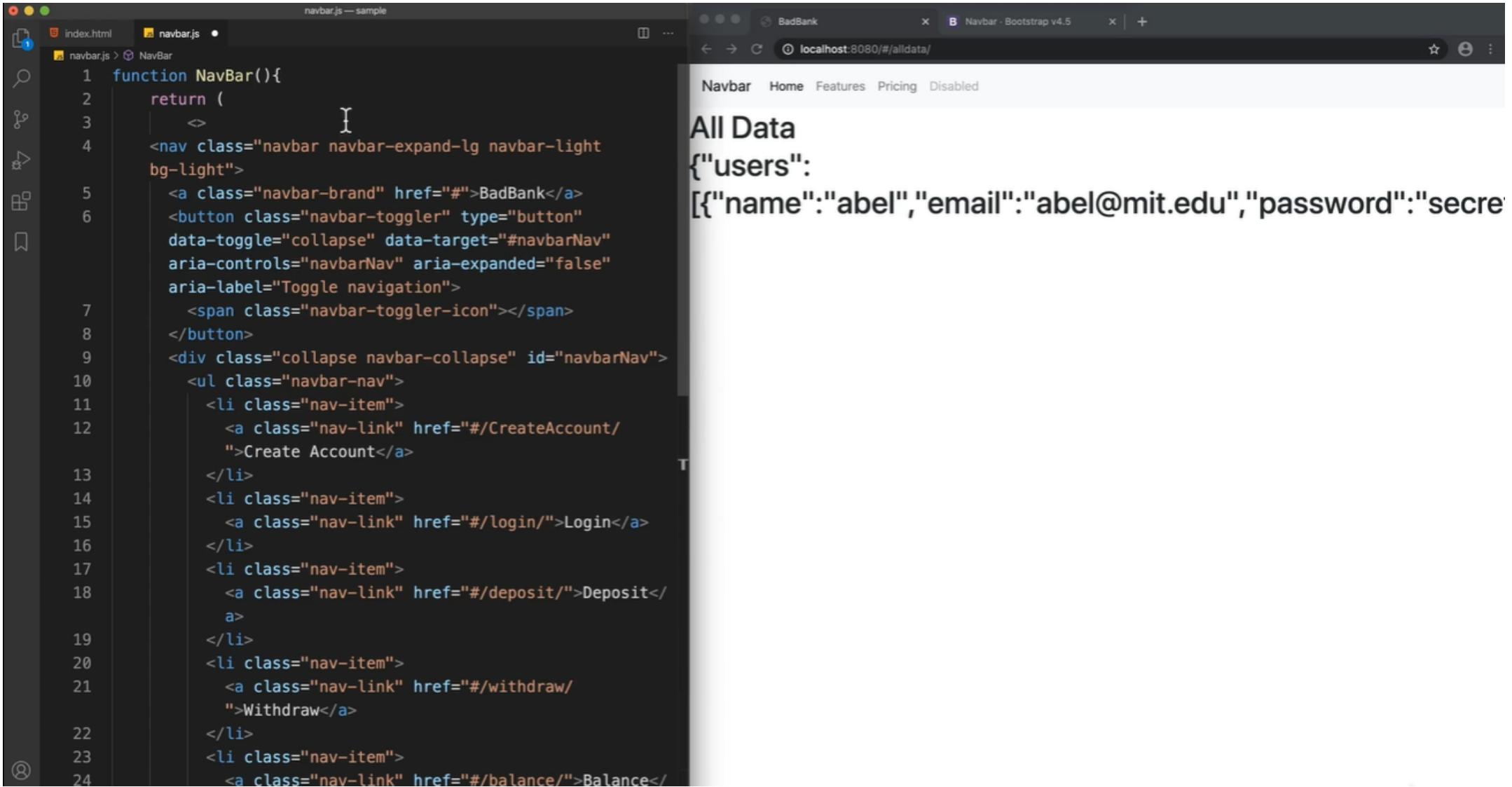
The image shows a split-screen environment. On the left, a code editor displays the file `navbar.js` with the following content:

```
navbar.js — sample
index.html navbar.js
NavBar
3
4 <nav class="navbar navbar-expand-lg navbar-light bg-light">
5   <a class="navbar-brand" href="#">NavBar</a>
6   <button class="navbar-toggler" type="button" data-toggle="c
7     <span class="navbar-toggler-icon"></span>
8   </button>
9   <div class="collapse navbar-collapse" id="navbarNavAltMarku
10    <div class="navbar-nav">
11      <a class="nav-link active" href="#">Home <span class="s
12      <a class="nav-link" href="#">Features</a>
13      <a class="nav-link" href="#">Pricing</a>
14      <a class="nav-link disabled" href="#" tabindex="-1" aria
15    </div>
16  </div>
17 </nav>
18 |  );
19 );
20 }
```

On the right, a web browser window titled "BadBank" shows the URL `localhost:8080/#/alldata/`. The page displays a navigation bar with links for "Navbar", "Home", "Features", "Pricing", and "Disabled". Below the navigation bar, the text "All Data" is displayed, followed by JSON data:

```
All Data
{"users": [
  {"name": "abel", "email": "abel@mit.edu", "password": "secret", "id": 1}
]}
```

# Add Bootstrap Styles: Demonstration (5/7)



The image shows a code editor on the left and a web browser on the right. The code editor displays the file `navbar.js` with the following content:

```
1  function NavBar(){
2      return (
3          <>           [
4              <nav class="navbar navbar-expand-lg navbar-light bg-light">
5                  <a class="navbar-brand" href="#">BadBank</a>
6                  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
7                      <span class="navbar-toggler-icon"></span>
8                  </button>
9                  <div class="collapse navbar-collapse" id="navbarNav">
10                     <ul class="navbar-nav">
11                         <li class="nav-item">
12                             <a class="nav-link" href="#/CreateAccount/">Create Account</a>
13                         </li>
14                         <li class="nav-item">
15                             <a class="nav-link" href="#/login/">Login</a>
16                         </li>
17                         <li class="nav-item">
18                             <a class="nav-link" href="#/deposit/">Deposit</a>
19                         </li>
20                         <li class="nav-item">
21                             <a class="nav-link" href="#/withdraw/">Withdraw</a>
22                         </li>
23                         <li class="nav-item">
24                             <a class="nav-link" href="#/balance/">Balance</a>
25                     </ul>
26                 </div>
27             </nav>
28         )
29     )
30 }
```

The browser window on the right shows a page titled "BadBank" at `localhost:8080/#/alldata/`. The page features a Bootstrap 4.5 navbar with links for Home, Features, Pricing, and Disabled. Below the navbar, the text "All Data" is displayed, followed by JSON data for users:

```
All Data
{"users": [{"name": "abel", "email": "abel@mit.edu", "password": "secret", "id": 1}, {"name": "charles", "email": "charles@mit.edu", "password": "secret", "id": 2}, {"name": "david", "email": "david@mit.edu", "password": "secret", "id": 3}, {"name": "elena", "email": "elena@mit.edu", "password": "secret", "id": 4}, {"name": "frank", "email": "frank@mit.edu", "password": "secret", "id": 5}, {"name": "gina", "email": "gina@mit.edu", "password": "secret", "id": 6}, {"name": "hannah", "email": "hannah@mit.edu", "password": "secret", "id": 7}, {"name": "ivan", "email": "ivan@mit.edu", "password": "secret", "id": 8}, {"name": "jessica", "email": "jessica@mit.edu", "password": "secret", "id": 9}, {"name": "karen", "email": "karen@mit.edu", "password": "secret", "id": 10}, {"name": "laura", "email": "laura@mit.edu", "password": "secret", "id": 11}, {"name": "michael", "email": "michael@mit.edu", "password": "secret", "id": 12}, {"name": "nancy", "email": "nancy@mit.edu", "password": "secret", "id": 13}, {"name": "oliver", "email": "oliver@mit.edu", "password": "secret", "id": 14}, {"name": "peter", "email": "peter@mit.edu", "password": "secret", "id": 15}, {"name": "quentin", "email": "quentin@mit.edu", "password": "secret", "id": 16}, {"name": "ronald", "email": "ronald@mit.edu", "password": "secret", "id": 17}, {"name": "sophia", "email": "sophia@mit.edu", "password": "secret", "id": 18}, {"name": "timothy", "email": "timothy@mit.edu", "password": "secret", "id": 19}, {"name": "ulrich", "email": "ulrich@mit.edu", "password": "secret", "id": 20}, {"name": "valerie", "email": "valerie@mit.edu", "password": "secret", "id": 21}, {"name": "wendy", "email": "wendy@mit.edu", "password": "secret", "id": 22}, {"name": "xavier", "email": "xavier@mit.edu", "password": "secret", "id": 23}, {"name": "yvonne", "email": "yvonne@mit.edu", "password": "secret", "id": 24}, {"name": "zachary", "email": "zachary@mit.edu", "password": "secret", "id": 25}]} 
```

# Add Bootstrap Styles: Demonstration (6/7)

The screenshot shows a development setup with two main windows. On the left is a code editor with a dark theme, displaying the file `navbar.js`. The code is a React component for a navigation bar, specifically a Bootstrap navbar. It includes logic for handling account creation, login, deposit, withdrawal, balance checks, and viewing all data. On the right is a web browser window titled "BadBank" running at `localhost:8080/#/alldata/`. The browser displays the text "All Data" followed by JSON data for a user named "abel". Below the browser is a developer tools console window. The console has tabs for Elements, Console, and Sources, with the Console tab selected. It shows several messages: a warning about moving messages to the Issues panel, a link to download React DevTools, and two errors from react-dom.development.js. One error is a warning about using the 'class' DOM property, suggesting 'className'. The other error is a specific warning about 'class' being invalid. The browser's status bar indicates there are 1 error, 1 warning, and 1 fix available.

```
<div class="collapse navbar-collapse" id="navbarNav">
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" href="#/CreateAccount/">Create Account</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#/login/">Login</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#/deposit/">Deposit</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#/withdraw/">Withdraw</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#/balance/">Balance</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#/alldata/">AllData</a>
    </li>
  </ul>
</div>
</nav>
```

All Data

{"users": [{"name": "abel", "email": "abel@mit.edu", "password": "secret"}]}

Elements    Console    Sources    >

Console

Some messages have been moved to the Issues panel. [View issues](#)

react-dom.development.js:25129

Download the React DevTools for a better development experience: <http://fb.me/react-devtools>

babel.min.js:1

You are using the in-browser Babel transformer. Be sure to precompile your scripts for production - <https://babeljs.io/docs/setup/>

react-dom.development.js:82

Warning: Invalid DOM property `class`. Did you mean `className`?

in a (created by NavBar)  
in nav (created by NavBar)  
in NavBar (created by Spa)  
in t (created by t)  
in t (created by Spa)  
in Spa

# Add Bootstrap Styles: Demonstration (7/7)

The screenshot shows a development environment with two main panes. On the left, the code editor displays `navbar.js` with the following content:

```
index.html navbar.js — sample
navbar.js > NavBar
  10 <ul className="nav">
  11   <li className="nav-item">
  12     <a className="nav-link" href="#/CreateAccount/">Create Account</a>
  13   </li>
  14   <li className="nav-item">
  15     <a className="nav-link" href="#/login/">Login</a>
  16   </li>
  17   <li className="nav-item">
  18     <a className="nav-link" href="#/deposit/">Deposit</a>
  19   </li>
  20   <li className="nav-item">
  21     <a className="nav-link" href="#/withdraw/">Withdraw</a>
  22   </li>
  23   <li className="nav-item">
  24     <a className="nav-link" href="#/balance/">Balance</a>
  25   </li>
  26   <li className="nav-item">
  27     <a className="nav-link" href="#/alldata/">AllData</a>
  28   </li>
  29 </ul>
  30 </div>
  31 </nav>
  32
  33 </>
  34 );
```

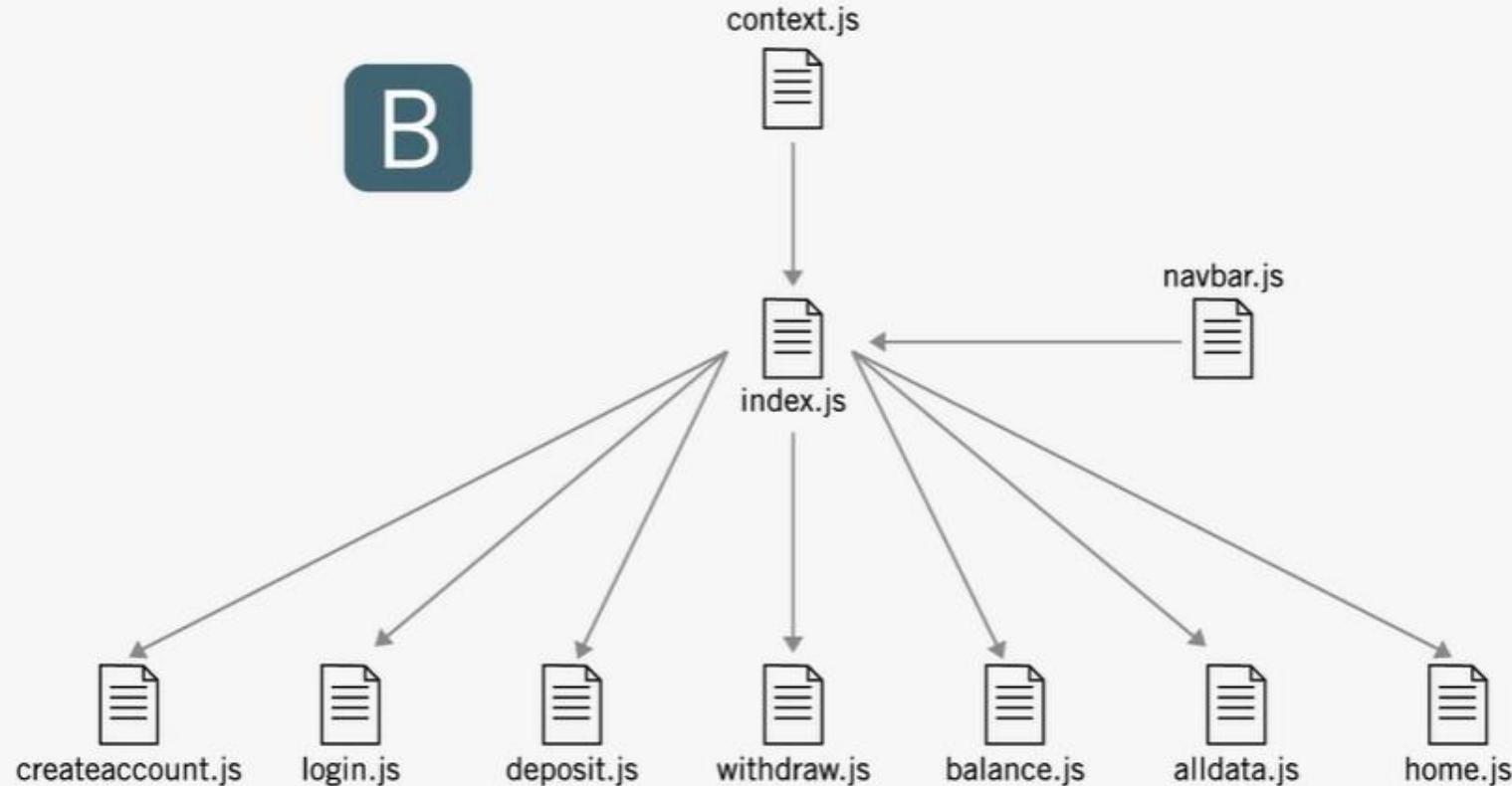
On the right, a browser window titled "BadBank" shows the application's interface. The navigation bar includes links for "Create Account", "Login", "Deposit", "Withdraw", "Balance", and "AllData". The "All Data" page is displayed, showing the following JSON response:

```
All Data
{"users": [
  {"name": "abel", "email": "abel@mit.edu", "password": "secret", "id": 1}
]}
```

The browser's developer tools Console tab is open, displaying several messages:

- "Some messages have been moved to the Issues panel." with a link to "View issues".
- A message from "react-dom.development.js:25129" encouraging the download of React DevTools.
- A warning message about using the in-browser Babel transformer, with a link to "babel.min.js:1" and instructions to precompile scripts for production.

# Add Bootstrap Styles: Your Turn



# Bootstrap Style – Card

**BadBank** Create Account Login Deposit Withdraw Balance AllData

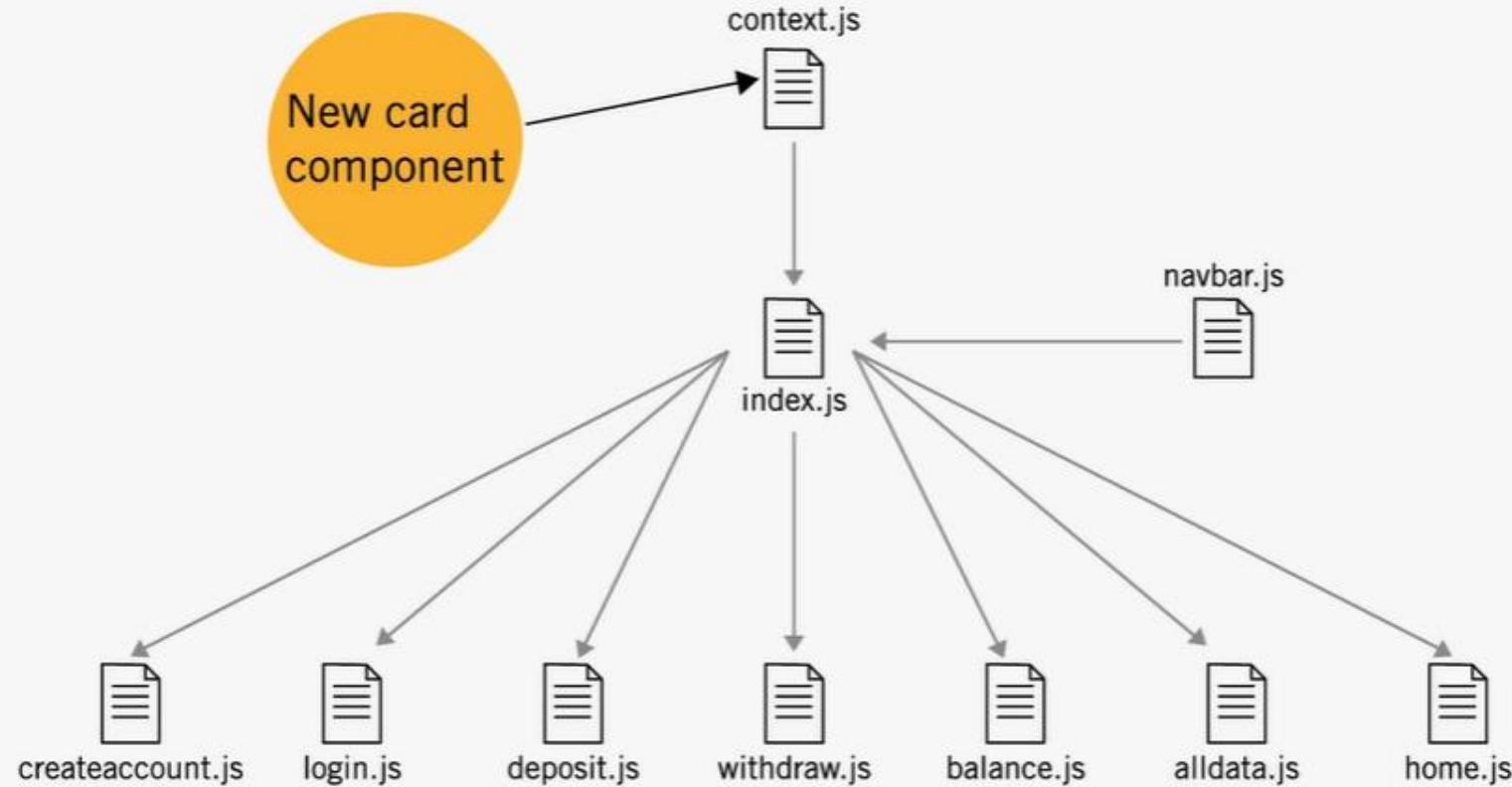
BadBank Landing Module

Welcome to the bank

You can move around using the navigation bar.



# Styles – Bootstrap



# Card – For All Components

Featured

## Special title treatment

With supporting text below as a natural lead-in to additional content

# Syntax Components

```
<div class="card">
  <div class="card-header">
    Featured
  </div>
  <div class="card-body">
    <h5 class="card-title">Special title treatment</h5>
    <p class="card-title">With supporting text below as
      a natural lead-in to additional content.</p>
  </div>
</div>
```

# Colors



light

dark

white

# Bootstrap Style – Card: Demonstration (1/5)

The image shows a code editor on the left and a web browser on the right.

**Code Editor (context.js):**

```
context.js — sample
context.js > Card > classes
1 const Route      = ReactRouterDOM.Route;
2 const Link       = ReactRouterDOM.Link;
3 const HashRouter = ReactRouterDOM.HashRouter;
4 const UserContext = React.createContext(null);
5
6 function Card(props){
7     function classes(){
8         const bg  = props.bgcolor ? 'bg-' + props.bgcolor : '';
9         const txt = props.txtcolor ? 'text-' + props.txtcolor: 'text-white';
10        return 'card mb-3 ' + bg + txt;
11    }
12
13    return (
14        <div className={classes()} style={{maxWidth: "18rem"}}>
15            <div className="card-header">{props.header}</div>
16            <div className="card-body">
17                {props.title && (<h5 className="card-title">{props.title}</h5>)}
18                {props.text && (<p className="card-text">{props.text}</p>)}
19                {props.body}
20                {props.status && (<div id='createStatus'>{props.status}</div>)}
21            </div>
22        </div>
23    );
24}
```

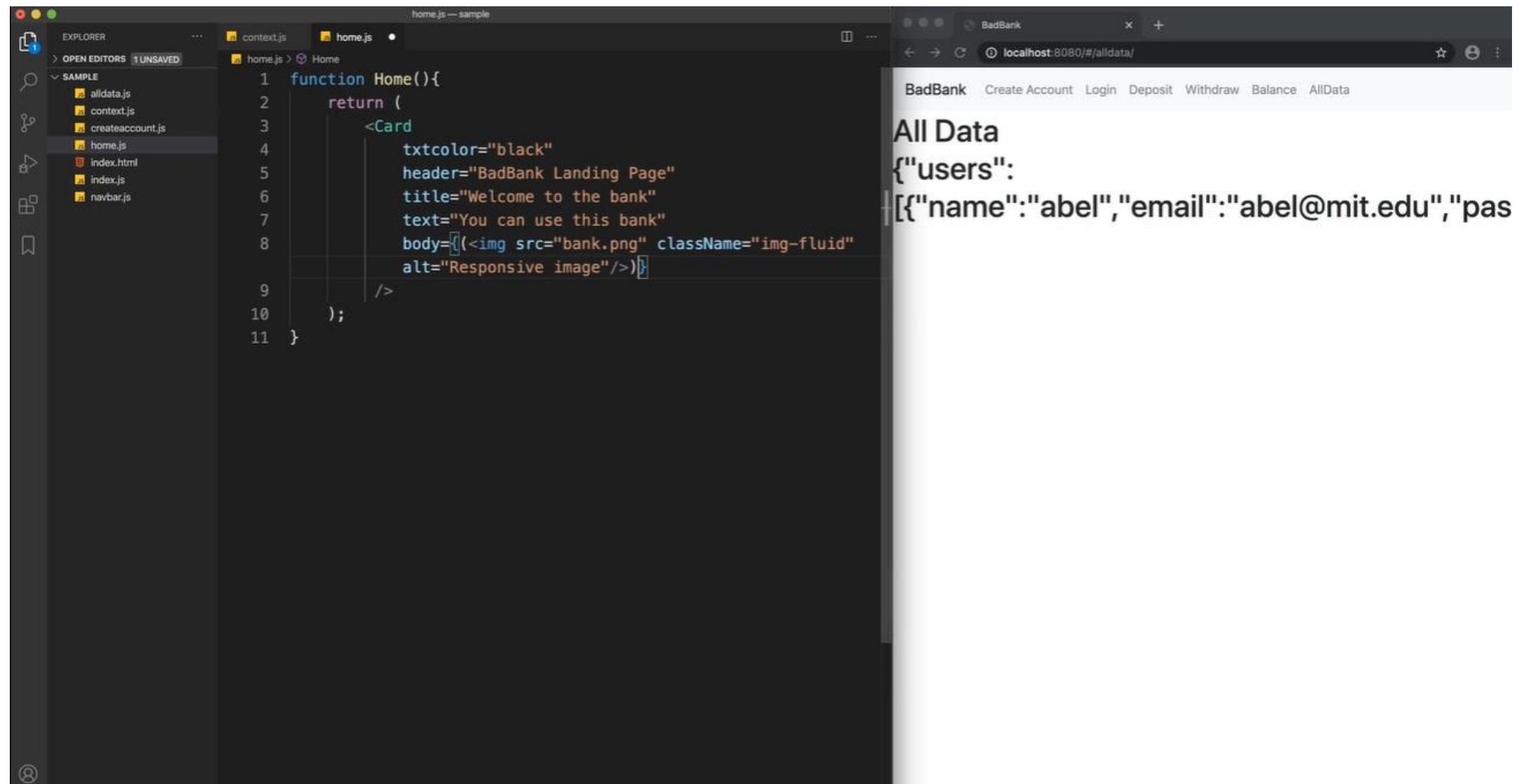
**Browser (BadBank):**

BadBank Create Account Login Deposit Withdraw Balance AllData

All Data

{"users": [{"name": "abel", "email": "abel@mit.edu", "pas

# Bootstrap Style – Card: Demonstration (2/5)



The screenshot shows a code editor (VS Code) on the left and a browser window on the right.

**Code Editor (VS Code):**

```
home.js — sample
OPEN EDITORS 1 UNSAVED
SAMPLE
  alldata.js
  context.js
  createaccount.js
  home.js
  index.html
  index.js
  navbar.js

1  function Home(){
2    return (
3      <Card
4        txtcolor="black"
5        header="BadBank Landing Page"
6        title="Welcome to the bank"
7        text="You can use this bank"
8        body={()}
10      );
11    }

```

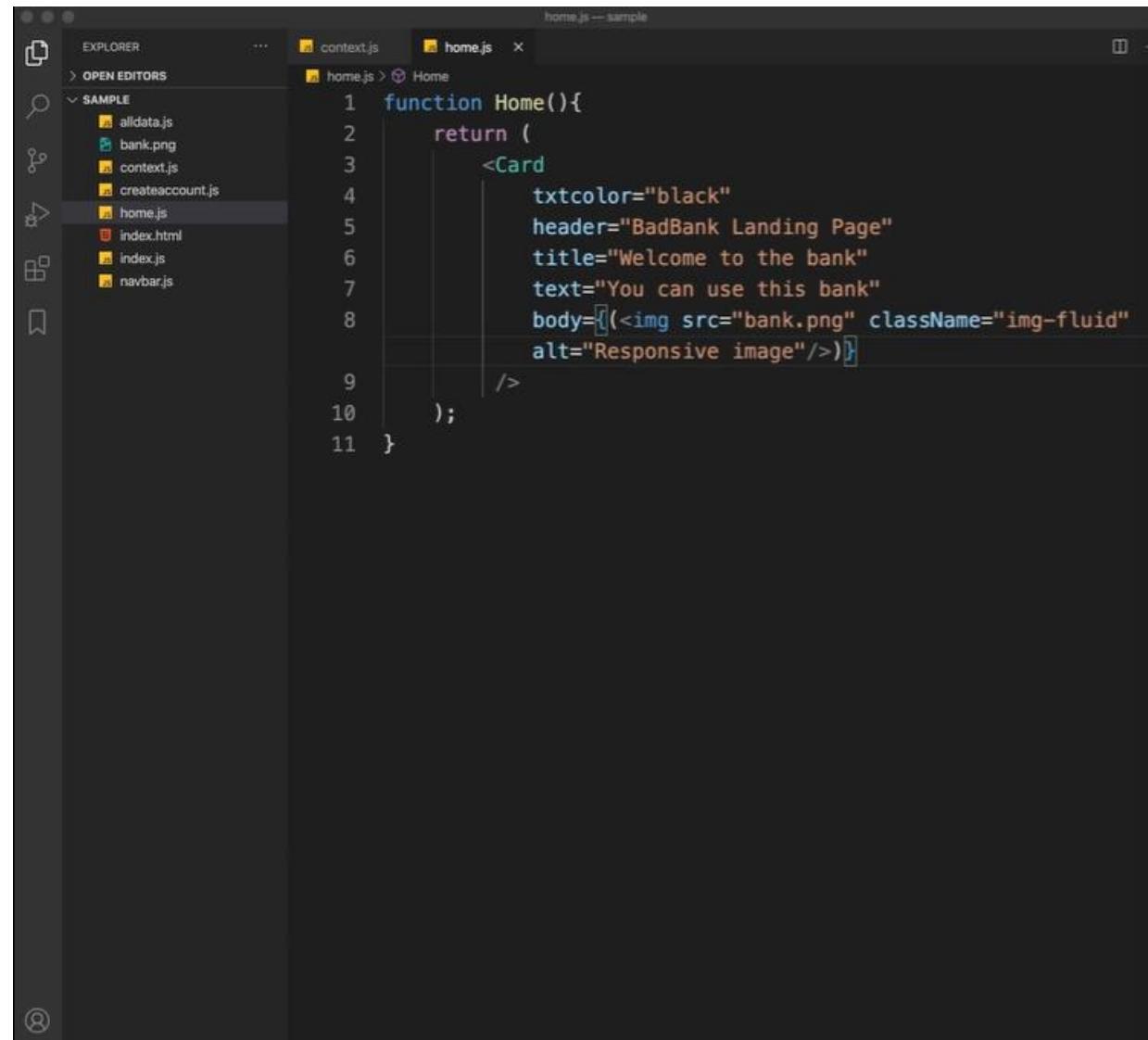
**Browser Window:**

BadBank Create Account Login Deposit Withdraw Balance AllData

All Data

```
{"users": [{"name": "abel", "email": "abel@mit.edu", "pas
```

# Bootstrap Style – Card: Demonstration (3/5)



The screenshot shows a code editor interface with a dark theme. On the left, the Explorer sidebar lists files: context.js, home.js (which is currently open), and several other files like alldata.js, bank.png, and index.html. The home.js file contains the following code:

```
function Home(){
    return (
        <Card
            txtcolor="black"
            header="BadBank Landing Page"
            title="Welcome to the bank"
            text="You can use this bank"
            body={()}
        >
    );
}
```

On the right, a browser window titled "BadBank" is displayed at the URL localhost:8080. The page has a header with links for Create Account, Login, Deposit, Withdraw, Balance, and AllData. Below the header, there's a section titled "BadBank Landing Page" with the text "Welcome to the bank" and "You can use this bank". A large black icon of a classical building with four columns and a triangular pediment is centered on the page.

# Bootstrap Style – Card: Demonstration (4/5)

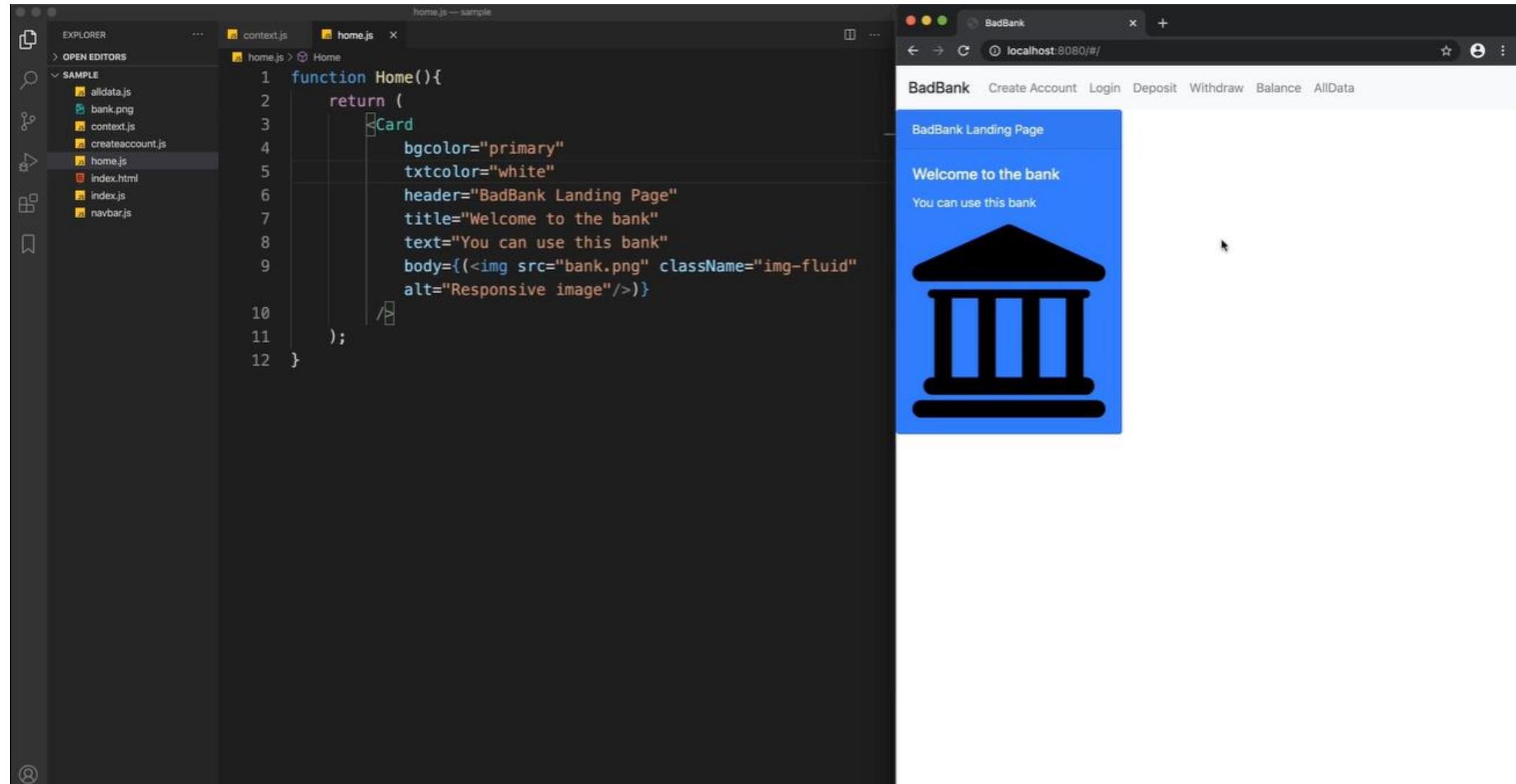
The image shows a split-screen development environment. On the left, a code editor displays a file named `home.js` with the following content:

```
function Home(){
  return (
    <Card
      bgcolor="primary"
      txtcolor="black"
      header="BadBank Landing Page"
      title="Welcome to the bank"
      text="You can use this bank"
      body={()}
    >
  );
}
```

The code uses the `Card` component from Bootstrap to create a card with a primary background color, black text, and a large image of a bank building in the body.

On the right, a web browser window titled "BadBank" is open at `localhost:8080/#/`. The page displays the "BadBank Landing Page" with the title "Welcome to the bank" and the text "You can use this bank". A large, stylized black icon of a classical building with four columns is centered on the page.

# Bootstrap Style – Card: Demonstration (5/5)



The image shows a side-by-side comparison between a code editor and a web browser.

**Code Editor (Left):** The 'home.js — sample' file contains the following code:

```
function Home(){
    return (
        <Card
            bgcolor="primary"
            txtcolor="white"
            header="BadBank Landing Page"
            title="Welcome to the bank"
            text="You can use this bank"
            body={()}
    );
}
```

**Browser (Right):** The browser window displays the 'BadBank' landing page at `localhost:8080/#/`. The page has a blue background and features a large black icon of a classical building with four columns and a pediment. The text on the page includes 'BadBank Landing Page', 'Welcome to the bank', 'You can use this bank', and a large black bank building icon.

# Bootstrap Style – Card: Your Turn

**BadBank** Create Account Login Deposit Withdraw Balance AllData

BadBank Landing Module

Welcome to the bank

You can move around using the navigation bar.



# Code Blocks

```
show {  
  if  
  else}
```

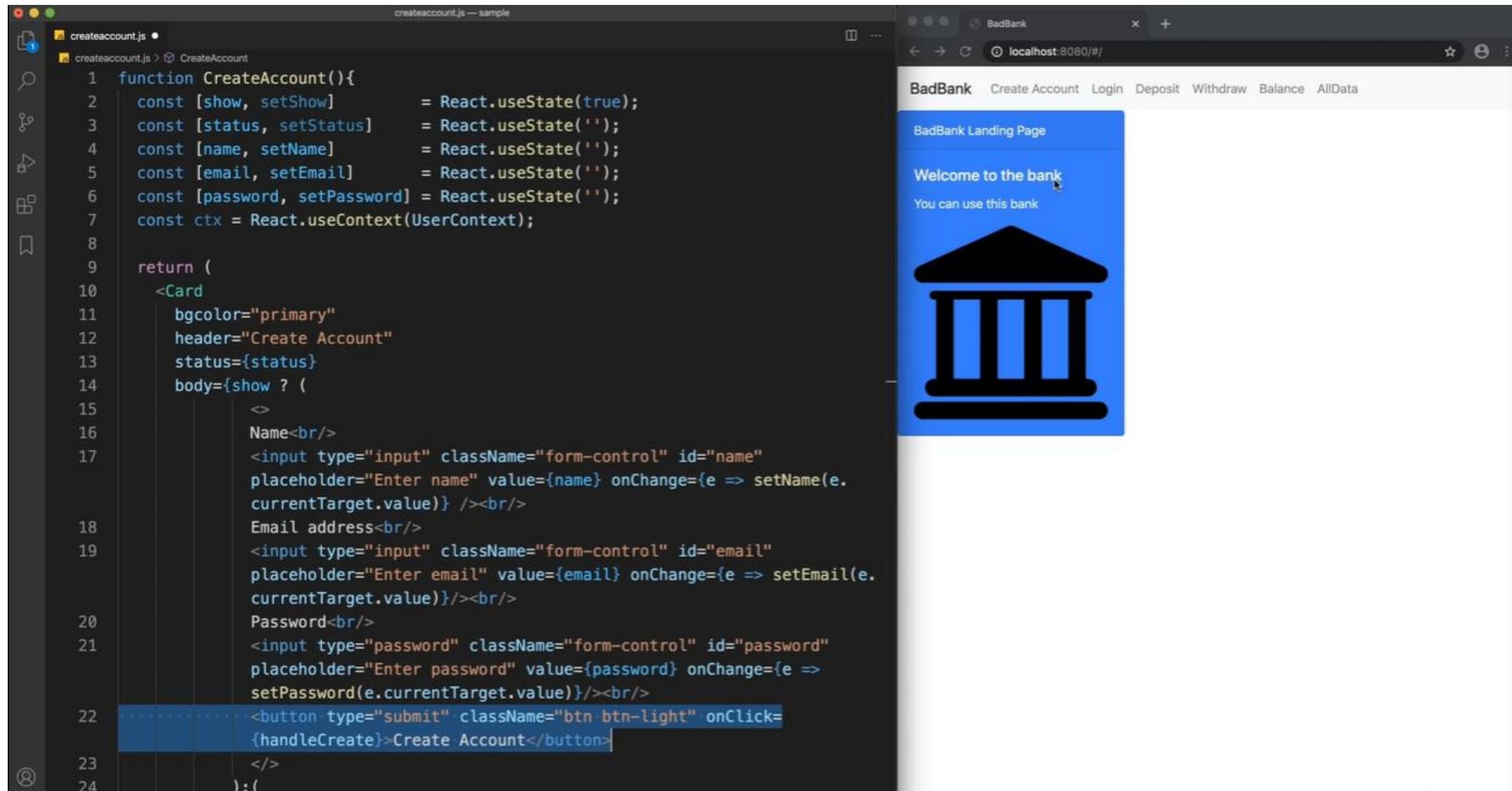
State: status, name, email, password, UserContext

Event: handleCreate, validate, clearform

Fields: styling, value, onChange event  
Button: handle event

Fields: styling, value, onChange event  
Button: handle event

# Creating Interactive Form (1/8)



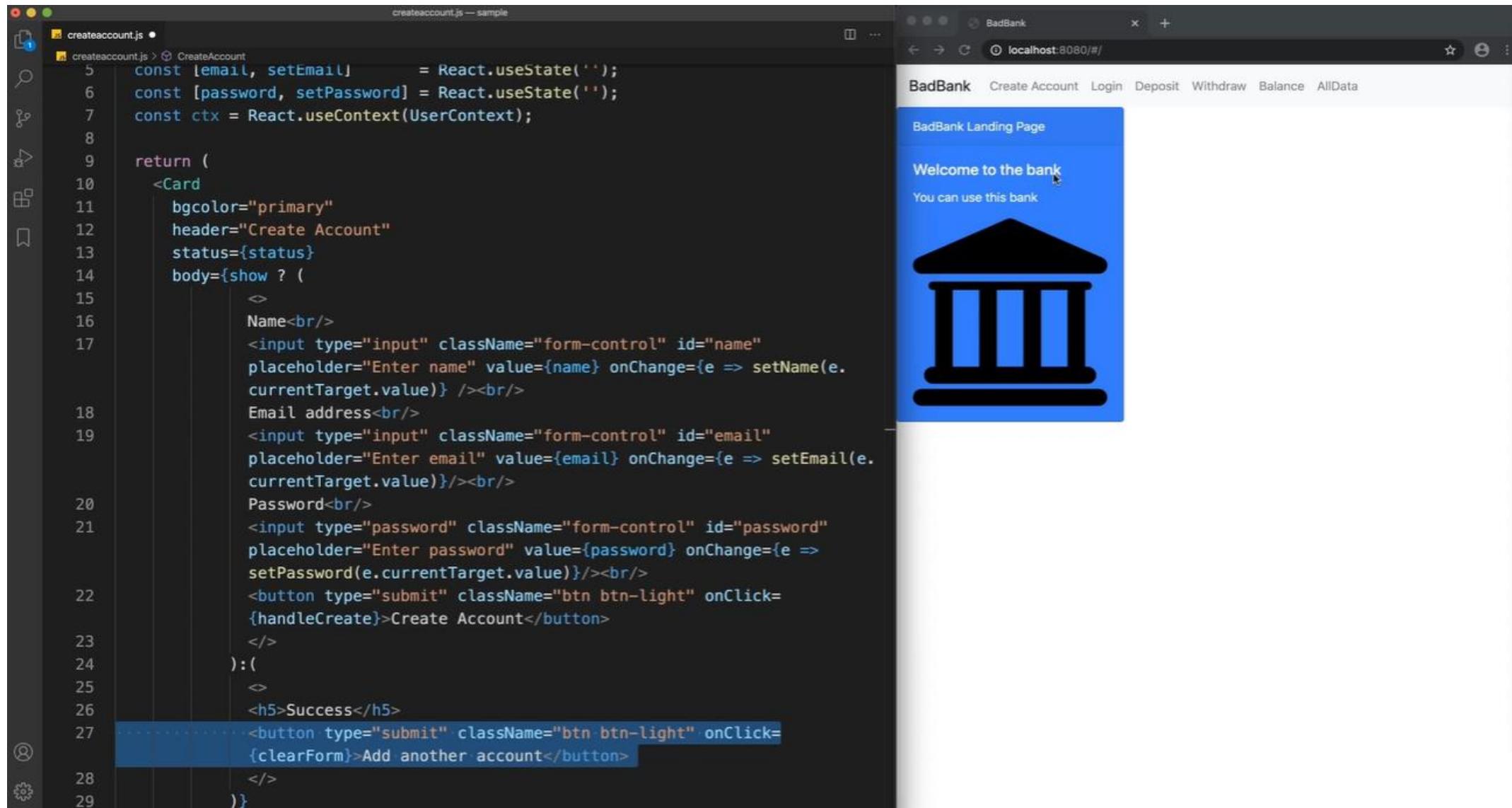
The image shows a split-screen development environment. On the left, a code editor displays the file `createaccount.js` with the following content:

```
createaccount.js — sample
createaccount.js > CreateAccount

1 function CreateAccount(){
2     const [show, setShow] = React.useState(true);
3     const [status, setStatus] = React.useState('');
4     const [name, setName] = React.useState('');
5     const [email, setEmail] = React.useState('');
6     const [password, setPassword] = React.useState('');
7     const ctx = React.useContext(UserContext);
8
9     return (
10       <Card
11         bgcolor="primary"
12         header="Create Account"
13         status={status}
14         body={show ? (
15           <>
16             Name<br/>
17             <input type="input" className="form-control" id="name"
18               placeholder="Enter name" value={name} onChange={e => setName(e.
19                 currentTarget.value)} /><br/>
20             Email address<br/>
21             <input type="input" className="form-control" id="email"
22               placeholder="Enter email" value={email} onChange={e => setEmail(e.
23                 currentTarget.value)} /><br/>
24             Password<br/>
25             <input type="password" className="form-control" id="password"
26               placeholder="Enter password" value={password} onChange={e =>
27                 setPassword(e.currentTarget.value)} /><br/>
28             <button type="submit" className="btn btn-light" onClick=
29               {handleCreate}>Create Account</button>
30           </>
31         );
32     );
33 }
```

On the right, a web browser window titled "BadBank" is open at `localhost:8080/#/`. The page displays the "BadBank Landing Page" with the text "Welcome to the bank" and "You can use this bank". Below the text is a large black icon of a classical bank building with four columns.

# Creating Interactive Form (2/8)

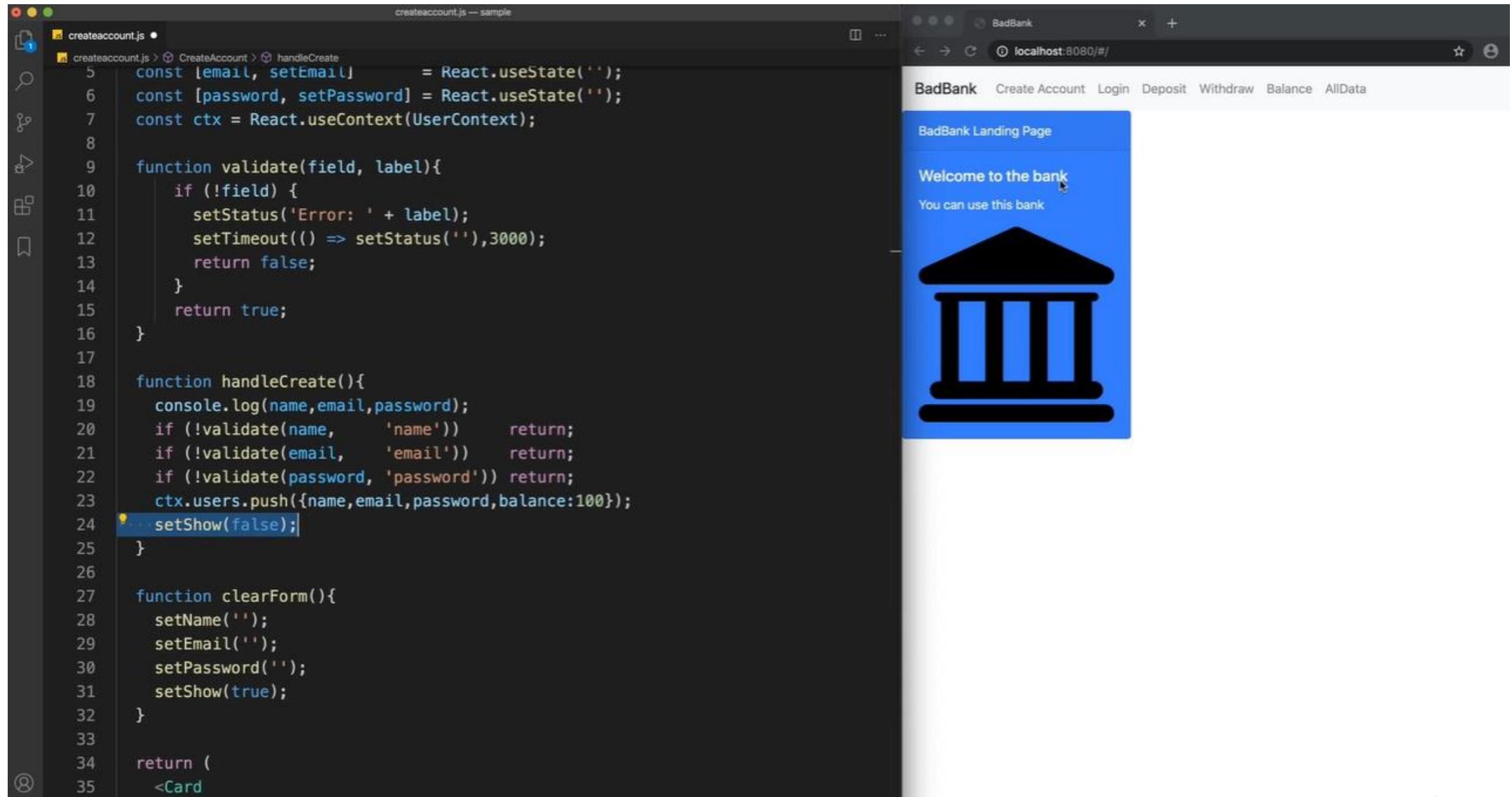


The image shows a split-screen development environment. On the left, a code editor displays the file `createaccount.js` with the following content:

```
createaccount.js — sample
createaccount.js > CreateAccount
  5  const [email, setEmail]      = React.useState('');
  6  const [password, setPassword] = React.useState('');
  7  const ctx = React.useContext(UserContext);
  8
  9  return (
 10    <Card
 11      bgcolor="primary"
 12      header="Create Account"
 13      status={status}
 14      body={show ? (
 15        <>
 16          Name<br/>
 17          <input type="input" className="form-control" id="name"
 18            placeholder="Enter name" value={name} onChange={e => setName(e.
 19              currentTarget.value)} /><br/>
 20          Email address<br/>
 21          <input type="input" className="form-control" id="email"
 22            placeholder="Enter email" value={email} onChange={e => setEmail(e.
 23              currentTarget.value)}/><br/>
 24          Password<br/>
 25          <input type="password" className="form-control" id="password"
 26            placeholder="Enter password" value={password} onChange={e =>
 27              setPassword(e.currentTarget.value)}/><br/>
 28          <button type="submit" className="btn btn-light" onClick=
 29            {handleCreate}>Create Account</button>
 30        </>
 31      ):(
 32        <>
 33          <h5>Success</h5>
 34          <button type="submit" className="btn btn-light" onClick=
 35            {clearForm}>Add another account</button>
 36        </>
 37      )}
 38    )
 39  )
 40}
```

On the right, a web browser window titled "BadBank" is open at `localhost:8080/#/`. The page displays the "BadBank Landing Page" with the text "Welcome to the bank" and "You can use this bank". It features a large icon of a classical bank building.

# Creating Interactive Form (3/8)



The image shows a split-screen development environment. On the left, a code editor displays the file `createaccount.js` with the following content:

```
createaccount.js — sample
createaccount.js > CreateAccount > handleCreate
  5  const [email, setEmail]      = React.useState('');
  6  const [password, setPassword] = React.useState('');
  7  const ctx = React.useContext(UserContext);
  8
  9  function validate(field, label){
 10    if (!field) {
 11      setStatus('Error: ' + label);
 12      setTimeout(() => setStatus(''),3000);
 13      return false;
 14    }
 15    return true;
 16  }
 17
 18  function handleCreate(){
 19    console.log(name,email,password);
 20    if (!validate(name, 'name')) return;
 21    if (!validate(email, 'email')) return;
 22    if (!validate(password, 'password')) return;
 23    ctx.users.push({name,email,password,balance:100});
 24    setShow(false);
 25  }
 26
 27  function clearForm(){
 28    setName('');
 29    setEmail('');
 30    setPassword('');
 31    setShow(true);
 32  }
 33
 34  return (
 35    <Card
```

On the right, a web browser window titled "BadBank" is open at `localhost:8080/#`. The page displays a blue header with the title "BadBank" and a navigation menu: Create Account, Login, Deposit, Withdraw, Balance, AllData. Below the header is a large blue box containing the text "BadBank Landing Page" and "Welcome to the bank". It also features a stylized black and blue bank building icon.

## Creating Interactive Form (4/8)

show {  
  if  
  else

State: status, name, email, password, UserContext



Event: handleCreate, validate, clearform



Fields: styling, value, onChange event  
Button: handle event



Fields: styling, value, onChange event  
Button: handle event



# Creating Interactive Form (5/8)

The image displays a split-screen view. On the left is a code editor window titled "createaccount.js — sample" containing the source code for a React component. On the right is a web browser window titled "BadBank" showing the rendered form.

**Code Editor (createaccount.js):**

```
1 function CreateAccount(){
2     const [show, setShow] = React.useState(true);
3     const [status, setStatus] = React.useState('');
4     const [name, setName] = React.useState('');
5     const [email, setEmail] = React.useState('');
6     const [password, setPassword] = React.useState('');
7     const ctx = React.useContext(UserContext);
8
9     function validate(field, label){
10         if (!field) {
11             setStatus('Error: ' + label);
12             setTimeout(() => setStatus(''),3000);
13             return false;
14         }
15         return true;
16     }
17
18     function handleCreate(){
19         console.log(name,email,password);
20         if (!validate(name, 'name')) return;
21         if (!validate(email, 'email')) return;
22         if (!validate(password, 'password')) return;
23         ctx.users.push({name,email,password,balance:100});
24         setShow(false);
25     }
26
27     function clearForm(){
28         setName('');
29         setEmail('');
30         setPassword('');
31         setShow(true);
32     }
}
```

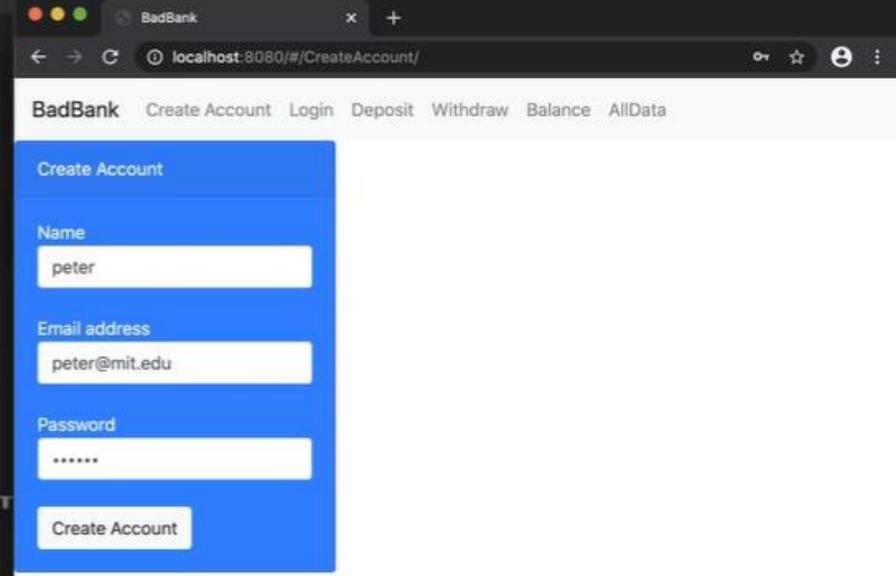
**Browser Preview:**

The browser window shows a "Create Account" form with three input fields: "Name", "Email address", and "Password", each with a placeholder text field. Below the inputs is a "Create Account" button.

# Creating Interactive Form (6/8)

```
createaccount.js — sample
createaccount.js @ CreateAccount
1 function CreateAccount(){
2   const [show, setShow] = React.useState(true);
3   const [status, setStatus] = React.useState('');
4   const [name, setName] = React.useState('');
5   const [email, setEmail] = React.useState('');
6   const [password, setPassword] = React.useState('');
7   const ctx = React.useContext(UserContext);
8
9   function validate(field, label){
10     if (!field) {
11       setStatus('Error: ' + label);
12       setTimeout(() => setStatus(''), 3000);
13       return false;
14     }
15     return true;
16   }
17
18   function handleCreate(){
19     console.log(name,email,password);
20     if (!validate(name, 'name')) return;
21     if (!validate(email, 'email')) return;
22     if (!validate(password, 'password')) return;
23     ctx.users.push({name,email,password,balance:100});
24     setShow(false);
25   }
26
27   function clearForm(){
28     setName('');
29     setEmail('');
30     setPassword('');
31     setShow(true);

```



The screenshot shows a web application window titled "BadBank" with a URL of "localhost:8080/#/CreateAccount/". The page has a header with links for Create Account, Login, Deposit, Withdraw, Balance, and AllData. Below the header is a "Create Account" form. The form contains three input fields: "Name" with the value "peter", "Email address" with the value "peter@mit.edu", and "Password" with the value "\*\*\*\*\*". At the bottom of the form is a blue "Create Account" button.

# Creating Interactive Form (7/8)

The image shows a development environment with three main components:

- Code Editor:** On the left, a code editor displays the `createaccount.js` file. The code defines a `CreateAccount` function that handles account creation logic using React hooks like `useState` and `useContext`. It includes validation functions for name, email, and password, and a `handleCreate` function that logs the user data and pushes it into the `ctx.users` array.
- Browser Preview:** In the center, a browser window titled "BadBank" shows the result of a successful account creation. The page title is "Create Account" and the content area displays "Success" with a button labeled "Add another account". The URL is `localhost:8080/#/CreateAccount/`.
- Developer Tools Console:** On the right, the browser's developer tools are open, specifically the "Console" tab. The console output includes:
  - A message indicating that some messages have been moved to the Issues panel, with a link to "View issues".
  - An error message from `react-dom.development.js:25129` suggesting to download React DevTools for better development experience, with a link to `https://fb.me/react-devtools`.
  - A warning message about using an in-browser Babel transformer, with a link to `https://babeljs.io/docs/setup/`.
  - User input: `peter peter@mit.edu secret` followed by the file path `createaccount.js:19`.

# Creating Interactive Form (8/8)

The image shows a development environment with two main windows. On the left is a code editor displaying the file `createaccount.js`. The code contains logic for validating input and creating users, along with a React component definition for a card-based form. On the right is a web browser window titled "BadBank" showing the application's interface. The browser's developer tools console tab is open, displaying logs related to the application's execution.

`createaccount.js — sample`

```
10  if (!validate(name, 'name')) return;
11  if (!validate(email, 'email')) return;
12  if (!validate(password, 'password')) return;
13  ctx.users.push({name,email,password,balance:100});
14  setShow(false);
15 }
16
17 function clearForm(){
18     setName('');
19     setEmail('');
20     setPassword('');
21     setShow(true);
22 }
23
24 return (
25     <Card
26         bgcolor="primary"
27         header="Create Account"
28         status={status}
29         body={show ? (
30             <>
31             Name<br/>
32             <input type="input" className="form-control" id="name"
33                 placeholder="Enter name" value={name} onChange={e => setName(e.
34                 currentTarget.value)} /><br/>
35             Email address<br/>
36             <input type="input" className="form-control" id="email"
37                 placeholder="Enter email" value={email} onChange={e => setEmail(e.
38                 currentTarget.value)} /><br/>
39             Password<br/>
40             <input type="password" className="form-control" id="password"
```

localhost:8080#/alldata/

BadBank Create Account Login Deposit Withdraw Balance AllData

## All Data

```
{"users": [{"name": "abel", "email": "abel@mit.edu", "password": "secret"}, {"name": "peter", "email": "peter@mit.edu", "password": "secret"}]}
```

Elements Console > 1 1 Filter 2 hidden

Some messages have been moved to the Issues panel. [View issues](#)

[react-dom.development.js:25129](#)  
Download the React DevTools for a better development experience: <https://fb.me/react-devtools>

**⚠** You are using the in-browser Babel [babel.min.js:1](#) transformer. Be sure to precompile your scripts for production - <https://babeljs.io/docs/setup/>

peter peter@mit.edu secret [createaccount.js:19](#)

# Creating Interactive Form: Your Turn

Login

Email

Enter email

Email

Enter password

Login

Deposit

Balance \$100

Deposit Amount

Deposit Amount

Deposit

Withdraw

Balance \$100

Withdraw Amount

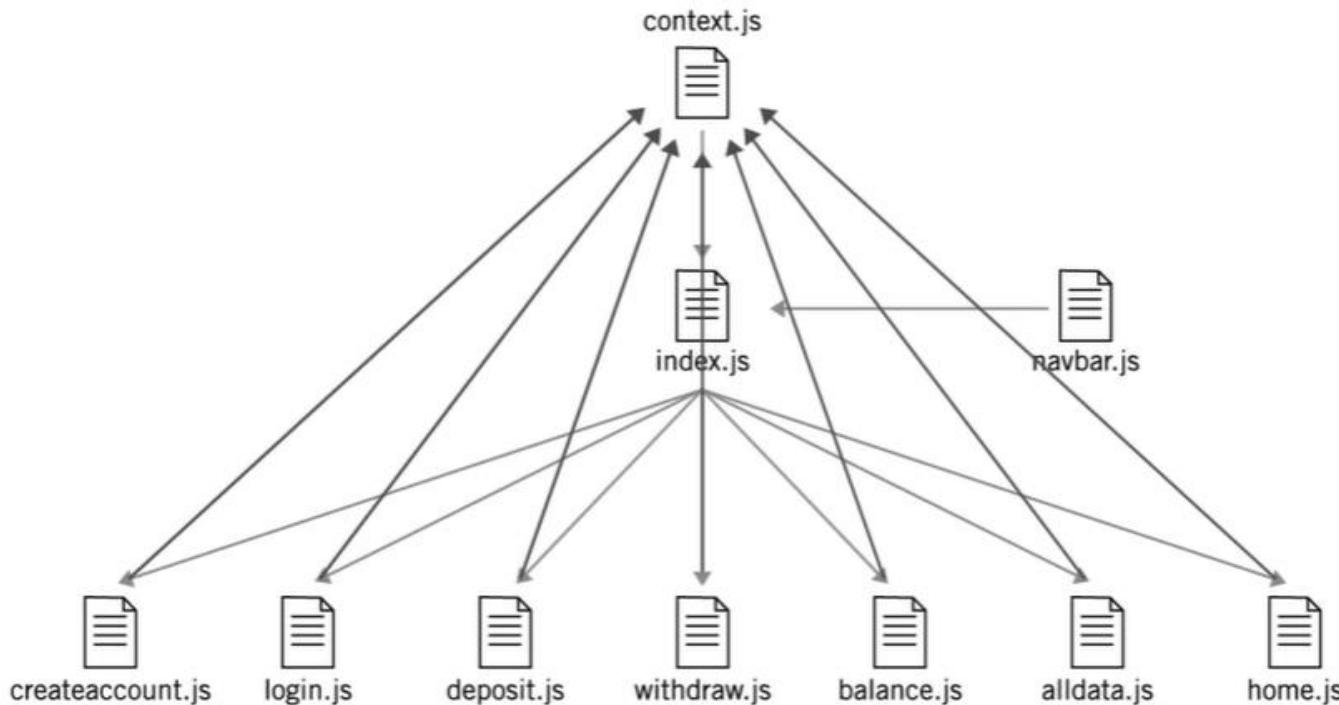
Withdraw Amount

Withdraw

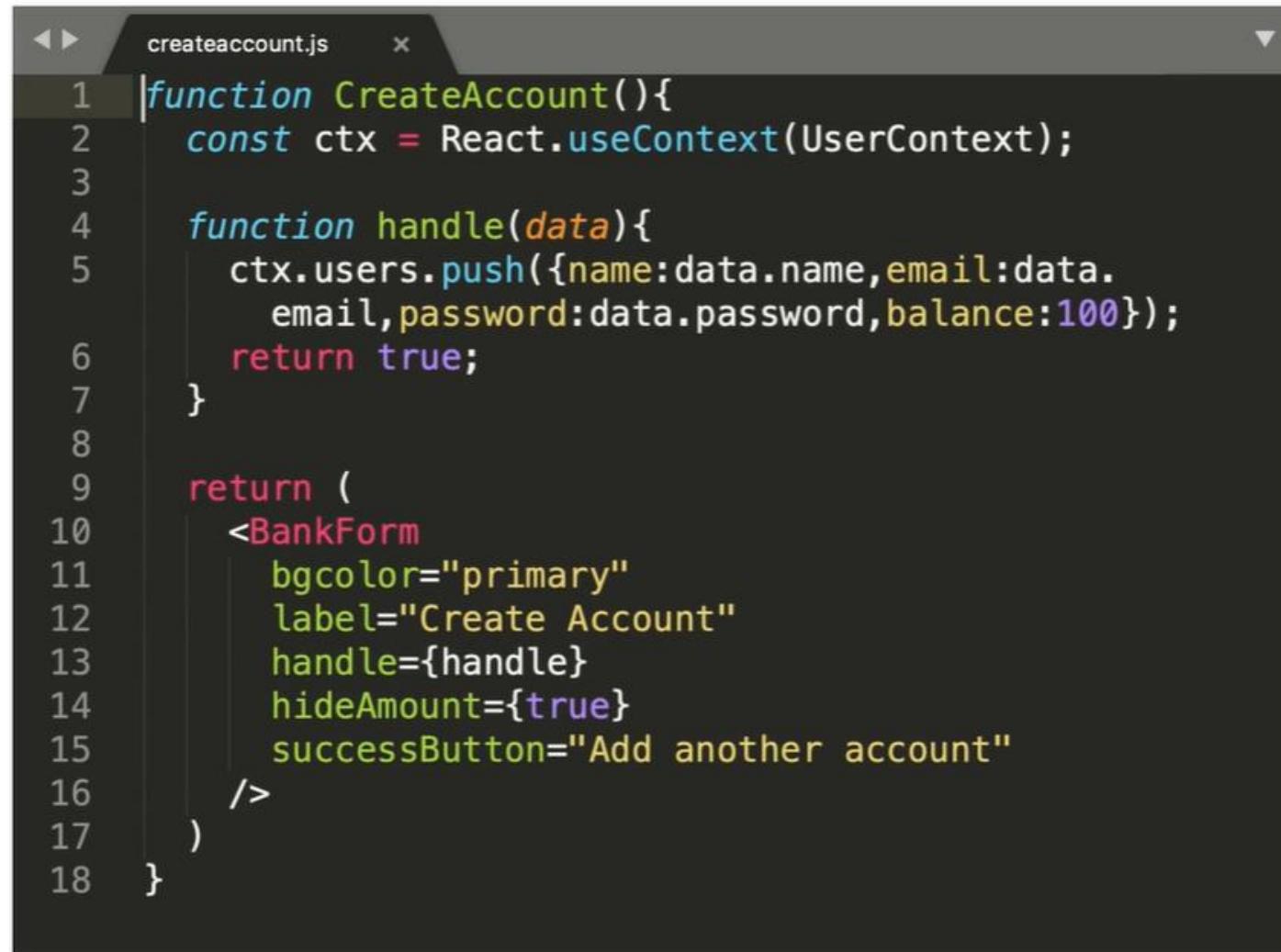
# How To Re-Design Your Application?



# Bad Bank – How Can You Improvise Work Done ?



# Code Snippet For Account Creation



A screenshot of a code editor window titled "createaccount.js". The code is written in JavaScript and uses JSX syntax. It defines a function named "CreateAccount" which returns a `<BankForm>` component. The `BankForm` component has several props: `bgcolor="primary"`, `label="Create Account"`, `handle={handle}`, `hideAmount={true}`, and `successButton="Add another account"`. The `handle` function pushes a new user object into the `ctx.users` array. The user object contains `name`, `email`, `password`, and `balance` properties, with `balance` set to 100.

```
function CreateAccount(){
  const ctx = React.useContext(UserContext);

  function handle(data){
    ctx.users.push({name:data.name,email:data.
      email,password:data.password,balance:100});
    return true;
  }

  return (
    <BankForm
      bgcolor="primary"
      label="Create Account"
      handle={handle}
      hideAmount={true}
      successButton="Add another account"
    />
  )
}
```



© MIT xPRO 2021. All rights reserved.