Class 4: Expression Placeholders & Props

- Review: Last Class
- Static Resources & URIs
- JSX Expression Placeholders
- Props
- Styling Components as Objects

Review: JSX → JavaScript

```jsx
export default function ComponentName() {
  return <div>Hello</div>
}
```

In JSX, the *HTML-like* elements are transpiled into JavaScript function calls to `React.createElement`:

```jsx
export default function ComponentName() {
  return React.createElement('div', null, 'Hello')
}
```

Review: Components are Objects

```
export default function ComponentName() {
  return <div>Hello</div>
}
```

Calling `ComponentName()` returns a JavaScript object:

```
{
  type: 'div',
  props: { children: 'Hello' },
  // ...
}
```

Review: JSX Component Invocation

```
<ComponentName />
```

`<ComponentName />` is transpiled into a JS function call:

```
ComponentName()
```

When your SPA is rendering `<ComponentName />`, it calls the function `ComponentName()` to get the object that describes what to render.

Review: JavaScript Modules

A JavaScript module is a **file** that contains JavaScript code.

Importing a Module

`src/App.jsx`

```
import ComponentName from "./ComponentName";
```

Omit the file extension when importing modules (`.js` / `.jsx`).

Exporting a Module

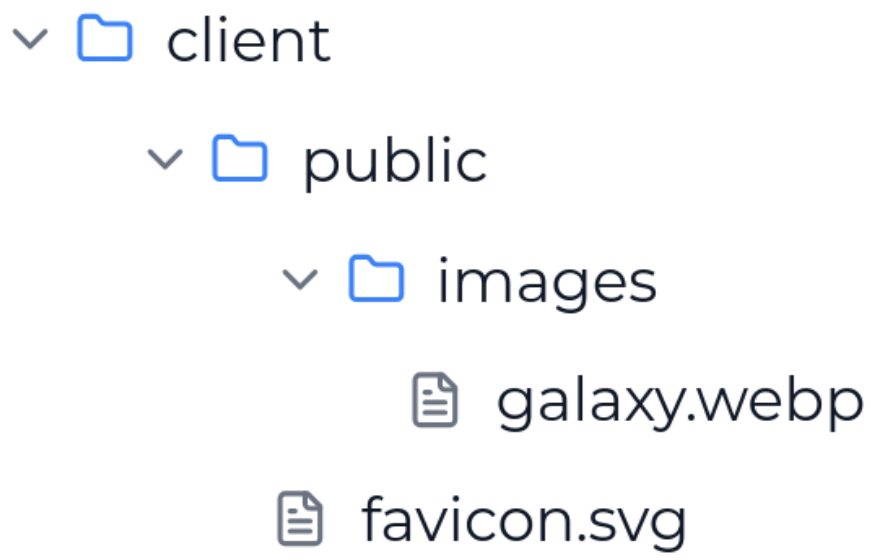
`src/components/ComponentName.jsx`

```
export default function ComponentName() {
  return (
    <div>Hello</div>
  )
}
```

Static Resources & URIs

Static Resources

Store **all** static resources (i.e. images, fonts, and other files that are not code) in the `public` directory.



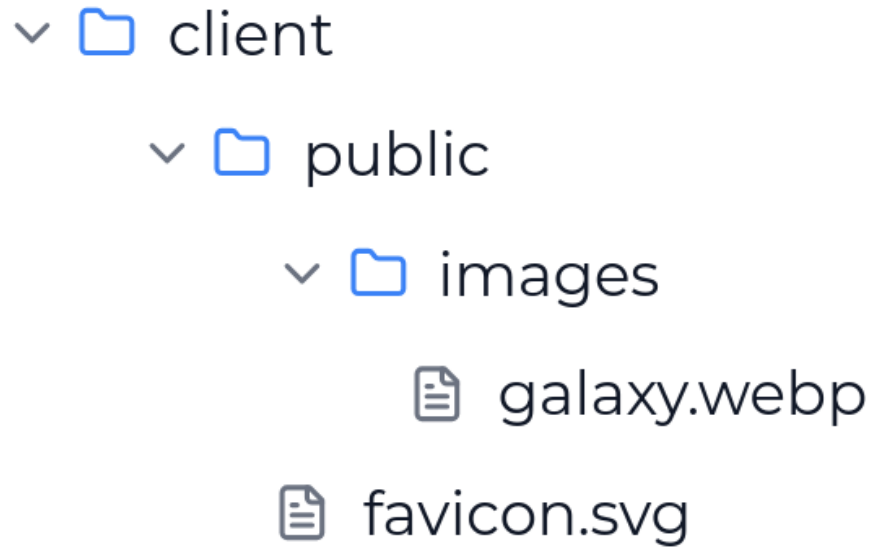
These files are served as static files by the web server.

Think: *INFO 1300*

Static Resource URIs

Reference static resources from the `public` directory using a Uniform Resource Identifier (URI) that begins with: `/`

```
<img src="/images/galaxy.webp" />
```



Gotcha: Paths vs. URIs

Paths

Used in **code** to locate files on the file system.

```
import ComponentName
  from "./components/ComponentName"
```

`./components/ComponentName` is the file path on the developer's computer.

URIs

Used in **HTML** to locate static resources served by the web server.

```
<img src="/images/galaxy.webp" />
```

`/images/galaxy.webp` is the URI served by the web server to the user's browser.

Activity: Static Resources & URIs

Working with your peers (2-4), complete the following tasks:

1. Move all images from the **root** of the activity repository to the `public/images` directory.
2. Test that you can access the images in your browser using the URL:

   `http://CODESPACE_NAME-5173.app.github.dev/images/pulsar.webp`

   Example:

   `https://zanny-umbrella-5173.app.github.dev/images/pulsar.webp`

Expression Placeholders

JSX Expression Placeholders

Use **curly braces** `{}` to *embed* JavaScript *expressions* inside the HTML-like syntax of JSX.

```
export default function ComponentName() {
  const name = "Prof. Harms"

  return <div>Hello, {name}!</div>
}
```

Gotcha: Expression placeholders **only** exists inside the HTML-like portions of a JSX file.

JavaScript Variable Declaration

Declaring a **constant** variable in JavaScript:

```
const name = "Prof. Harms"
```

Declaring a **mutable** variable in JavaScript:

```
let count = 0
```

Demo: JSX Expression Placeholders

```
export default function Citation() {
  const citation = "Microsoft Copilot"

  return (
    <p className="citation">
      Source: <cite>{citation}</cite>
    </p>
  )
}
```

Activity: JSX Expression Placeholders

The `Card` component's image is hard-coded to `/images/galaxy.webp`.

1. Create a **constant** named `imgUri` that stores the URI for the static image resource in `Card`.
2. Modify the `Card` component to render the image using an expression placeholder and `imgUri`.
3. Create another constant for the `alt` text.

```
export default function ComponentName() {
  const name = "Prof. Harms"

  return <div>Hello, {name}!</div>
}
```

Props

Props

Props (short for *properties*) are a way to pass data into a React component.

```
<ComponentName propName={propValue} />
```

Think: HTML *attributes*, but for JSX components

Example:

```
<Citation source="Microsoft Copilot" />
```

Defining Props

Defined as parameters to the component function using **camelCase**.

```
export default function ComponentName({propName1, propName2, propName3}) {
  // ...
}
```

Use the prop names to access the prop values inside the component.

```
export default function ComponentName({propName}) {
  return <div>{propName}</div>
}
```

Demo: Defining & Using Props

```
export default function Citation({source}) {
  return (
    <p className="citation">
      Source: <cite>{source}</cite>
    </p>
  )
}
```

Activity: Defining & Using Props

1. Add a prop for the image URI to the `Card` component.
2. Modify the `Card` component to use the image URI prop.
3. Update the `App` component to pass the image URI prop.
4. Add a prop for the `alt` text to the `Card` component.
5. Render 3 cards with different images and alt text.

```
export default function
  ComponentName({propName}) {

  return <div>{propName}</div>
}
```

Gotcha: Props are Read-Only

Props are **read-only** inside a component. You **cannot** modify prop values.

Gotcha: JSX Provides Some Props

For HTML elements, JSX provides some props by default, like `className`, `style`, and `id`.

```
<div className="warning">Warning!</div>
```

Why not `class`? Because props must be valid JavaScript identifiers, and `class` is a reserve word!

Style Object

Review: Style Object

```
<div style="color: red; font-size: 20px; font-weight: bold;">Hello</div>
```

`font-size` and `font-weight` are not valid JS identifiers (they contain hyphens). Use **camelCase** instead:

```
{
  color: 'red',
  fontSize: '20px',
  fontWeight: 'bold',
}
```

Activity: Style Practice

Working with your peers (2-4), complete Part I of the handout.

Example:

```
{
  color: 'red',
  fontSize: '20px',
  fontWeight: 'bold',
}
```

Gotcha: Expression Placeholders & Object Literals

When using an object literal inside a JSX expression placeholder, you need to use **double** curly braces `{{ ... }}`.

```
<div className="warning" style={{ color: 'red', fontSize: '20px' }}>
  Hello
</div>
```

The **outside** pair of curly braces is for the JSX expression placeholder.
The **inside** pair of curly braces is for the JavaScript object literal.

Demo: Style Object as Prop

```
export default function Citation({ citation }) {
  return (
    <p className="citation" style={{ textAlign: 'center' }}>
      Source: <cite>{citation}</cite>
    </p>
  )
}
```

Activity: Style Object as Prop

Set the background color of the `<div>` in `Card` to white.

`Citation.jsx`:

```
export default function Citation({ citation }) {
  return (
    <p className="citation" style={{ textAlign: 'center' }}>
      Source: <cite>{citation}</cite>
    </p>
  )
}
```

Summary

- Static resources go in the `public` directory and are referenced using URIs that start with `/`.
- Expression placeholders `{}` allow embedding JavaScript expressions inside JSX.
- Props are a way to pass data into components as function parameters.
- Style objects are JavaScript objects that define CSS styles using camelCase property names.
- When using object literals in JSX expression placeholders, use double curly braces `{{ ... }}`.

What's Next

Due Today: Project 1, Milestone 1

Due Tuesday: Class 5 Preparation