Retrieving Resource Type Information in AEM Angular SPA

In AEM Angular SPA, you don't directly use Sightly-like syntax within your Angular component's HTML. Instead, you leverage the Sling Model API and the AEM SPA Editor SDK to access and render components.

Here's how you can achieve a similar effect to `data-sly-resource` with `resourceType` in an AEM Angular SPA:

1. Sling Model Definition:

First, you need to create a Sling Model that exposes the necessary information, including the resource type of the component you want to include.

Java

```java
import org.apache.sling.api.resource.Resource;
import org.apache.sling.models.annotations.Model;
import org.apache.sling.models.annotations.injectorspecific.Self;

@Model(adaptables = Resource.class)
public class MyComponentModel {

    @Self
    private Resource resource;

    public String getResourceType() {
        return resource.getResourceType();
    }
}
```

2. Component Policy (Optional):

If you need to pass additional configuration values, you can use component policies. For example, if you want to dynamically set the resource type.

3. Angular Component:

In your Angular component, you'll use the AEM SPA Editor SDK's `MapTo` and `ModelManager` to fetch the Sling Model data and dynamically render the resource.

TypeScript

```typescript
import { Component, OnInit } from '@angular/core';
import { MapTo, ModelManager } from '@adobe/cq-angular-editable-components';

interface MyComponentModel {
  resourceType: string;
}

@Component({
  selector: 'app-my-component',
  template: `
    <div *ngIf="model">
      <div *ngIf="model.resourceType">
        <cq-placeholder [resourceType]="model.resourceType"></cq-placeholder>
      </div>
    </div>
  `,
})
export class MyComponent implements OnInit {
  model: MyComponentModel;

  constructor() {}
```

```
    ngOnInit(): void {
      this.model = ModelManager.model;
    }
  }

  MapTo('myapp/components/mycomponent')(MyComponent);
```

Explanation:

- Sling Model: The Sling Model `MyComponentModel` adapts from the resource and exposes the `resourceType` property.
- Angular Component:
  - The component uses `ModelManager.model` to retrieve the Sling Model data.
  - It then accesses the `resourceType` from the model.
  - `cq-placeholder` from the AEM SPA Editor SDK is used to dynamically render the component based on the retrieved `resourceType` .
  - `MapTo` registers the component with the AEM SPA Editor, allowing it to be editable.

Important Considerations:

- `cq-placeholder` : The `cq-placeholder` component is crucial for dynamically rendering components based on the `resourceType` provided.
- Sling Model Mapping: Ensure that your Sling Model is correctly mapped to the component in AEM.
- AEM SPA Editor SDK: The AEM SPA Editor SDK provides the necessary tools and components for integrating your Angular application with AEM.

- Dynamic Resource Type: If you want to dynamically set the `resourceType` based on component policies or other data, you can extend the Sling Model and the Angular component accordingly.
- Security: Be cautious when dynamically rendering components based on user-supplied data to avoid potential security vulnerabilities. Sanitize and validate input as needed.

By using this approach, you can achieve similar functionality to Sightly's `data-sly-resource` with `resourceType` in your AEM Angular SPA, enabling dynamic component rendering based on resource type information.

give a specific example with the resourceType is the Image Component

Java

```java
// MyImageContainerModel.java (Sling Model)
import org.apache.sling.api.resource.Resource;
import org.apache.sling.models.annotations.Model;
import org.apache.sling.models.annotations.injectorspecific.Self;

@Model(adaptables = Resource.class)
public class MyImageContainerModel {

    @Self
    private Resource resource;
    public String getImageResourceType() {
        return "core/wcm/components/image/v3/image"; // Explicitly setting the image resou
    }
}
```

TypeScript

```
// my-image-container.component.ts (Angular Component)
import { Component, OnInit } from '@angular/core';
import { MapTo, ModelManager } from '@adobe/cq-angular-editable-components';

interface MyImageContainerModel {
  imageResourceType: string;
}

@Component({
  selector: 'app-my-image-container',
  template: `
    <div *ngIf="model">
      <div *ngIf="model.imageResourceType">
        <cq-placeholder [resourceType]="model.imageResourceType"></cq-placeholder>
      </div>
    </div>
  `,
})
export class MyImageContainerComponent implements OnInit {
  model: MyImageContainerModel;

  constructor() {}

  ngOnInit(): void {
    this.model = ModelManager.model;
  }
}

MapTo('myapp/components/myimagecontainer')(MyImageContainerComponent);
```

Explanation of the Specific Example:

1. Sling Model (`MyImageContainerModel.java`):
   - The `getImageResourceType()` method now explicitly returns `"core/wcm/components/image/v3/image"` . This is the resource type of the AEM Core Components Image component.
   - This shows how you can hardcode the resource type, or retrieve it dynamically based on other logic.
2. Angular Component (`my-image-container.component.ts`):
   - The `MyImageContainerModel` interface now defines `imageResourceType` instead of the generic `resourceType` .
   - The template uses `cq-placeholder` with `[resourceType]="model.imageResourceType"` . This will render the AEM Image component within the placeholder.
   - The `MapTo` function maps the angular component to the AEM component `myapp/components/myimagecontainer` .
3. AEM Component (`myapp/components/myimagecontainer`):
   - You need to create the 'myimagecontainer' component in AEM.
   - The myimagecontainer component's HTL or JSP file is not used to render the image. Instead, the angular component and sling model handle the rendering.

- The AEM myimagecontainer component's dialog can contain configuration, that the sling model could then expose, if needed.

How it Works:

- When the `my-image-container` component is rendered in the AEM SPA Editor, the Angular component is instantiated.
- The `ModelManager.model` fetches the `MyImageContainerModel` data.
- The Angular component uses the `cq-placeholder` and the `imageResourceType` to dynamically render the AEM Image component within the placeholder.
- The AEM SPA Editor handles the communication between the AEM backend and the Angular frontend, ensuring that the Image component is rendered correctly and that it is editable within the SPA editor.

This example demonstrates how to specifically render the AEM Image component dynamically within your Angular SPA. You can adapt this pattern to render other AEM components as needed.

will i see the <img> tag in the html when it populates to the browser with the above code?

Yes, you will see the `<img>` tag in the rendered HTML in your browser when the AEM Image component is populated using the provided code. However, the `<img>` tag is not directly generated by the Angular component's template. Instead, it's rendered by the AEM SPA Editor's `cq-placeholder` and the AEM Image component itself.

Here's a breakdown of how it works:

1. Angular Component and `cq-placeholder` :
    - The Angular component's template contains the `cq-placeholder` tag. This placeholder acts as a container where the AEM Image component will be rendered.
    - The Angular component provides the `resourceType` ("core/wcm/components/image/v3/image") to the `cq-placeholder` .
2. AEM SPA Editor and Image Component:
    - The AEM SPA Editor, based on the `resourceType` , dynamically renders the AEM Image component within the `cq-placeholder` .
    - The AEM Image component, in turn, generates the `<img>` tag based on the image's path and other properties (alt text, etc.).
3. Browser Rendering:
    - The browser receives the final HTML, which includes the `<img>` tag generated by the AEM Image component.
    - The browser then displays the image.

In essence:

- The Angular component's template only contains the `cq-placeholder` .
- The AEM Image component (rendered dynamically by the AEM SPA Editor) generates the `<img>` tag.

Therefore, when you inspect the rendered HTML in your browser's developer tools, you will see the `<img>` tag within the `cq-placeholder` 's generated div.

Example of the generated HTML (simplified):

HTML

```html
<div data-cq-data-path="myapp/components/myimagecontainer/cq:responsive/child/image">
  <img src="/content/dam/your-project/your-image.jpg" alt="Your Image Alt Text" class="cmp
</div>
```

The `<img>` tag will be inside the `div` that the `cq-placeholder` generates. The attributes of the `<img>` tag are populated by the AEM Image component, based on the data stored in AEM.