JOHN BRYCE
Leading in IT Education
a matrix company

Angular Components

Advanced Drill Down on Angular Components

Our Goal

JOHN BRYCE
Leading in IT Education
a matrix company

- Create a new Angular CLI app
- Add father child components
- Understand how to style our app
- Understand component lifecycle
- Advanced component template

New Angular Project

JOHN BRYCE
Leading in IT Education
a matrix company

- let's create a new @angular/cli project
  - **ng new advanced-components-tutorial**
- the default project creates an app with a single component: **app.component.ts**
- use @angular/cli to create a child component for our root component
  - **ng generate component Child**
- this will create the component files and also update our module to include the new component
- modify the **app.component.html** and add the child selector in the root component template
- run your app using: > **ng serve** you should see the child component is displayed

Styling Our App - Global Styles

JOHN BRYCE
Leading in IT Education
a matrix company

- In **.angular-cli.json** there is a property located in **apps[0].styles** with a list of files containing global styles for our app
- By default Angular CLI creates a global styles file called **styles.css**
- When you build your app Angular creates from that file a js file called: **styles.bundle.js** which appends the styles you set to the head section of the html
- you can use the global style file along with **@import** statement to add global styling for your app.
- Let's use the global styling file to install the **css** files of **bootstrap**

Styling Our App - Component Styles

JOHN BRYCE
Leading in IT Education
a matrix company

- In the **@Component** metadata there are a couple of options to control the style of a component
- **Styles** - string array of css styles
- **styleUrls -** array of styling files, relative to the src file.
- The styles will also be added to the head section in the index
- The styles will only affect the component and won't be inherited by child components
- Let's view the head section after adding some styles

Styling Our App - Component Styles

JOHN BRYCE
Leading in IT Education
a matrix company

- Angular place the styles in the head
- Angular attach an attribute for every component (ngcontent-c0 nghost-c1)
- The styles will be placed in the head bug with the proper attribute so they will only affect the component and not the entire app
- We can control how the styles affect the app with the **encapsulation** option in the component metadata
- To understand the options we first have to understand what is the **Shadow DOM**

Special Selectors

JOHN BRYCE
Leading in IT Education
a matrix company

- **:host -** target styles in the in the host element (not the parent)
- **:host-context -** conditional css based on parent styling

Shadow DOM

JOHN BRYCE
Leading in IT Education
a matrix company

- With Shadow DOM we can define a custom dom element let's call that element **<john-bryce></john-bryce>**
- The custom element can have a dom tree of it's own, and it's own styling
- This way we hide the inner parts of the component and create encapsulation where we hide the inner implementation
- Supported in chrome but not in other browsers

Shadow DOM

JOHN BRYCE
Leading in IT Education
a matrix company

```
class JohnBryce extends HTMLElement {
 constructor() {
   super();
   var shadow = this.attachShadow({mode: 'closed'});
   var hello = document.createElement('span');
   hello.innerText = 'hello world';
   shadow.appendChild(hello);
 }
}
customElements.define('john-bryce', JohnBryce);
```

Component Encapsulation

JOHN BRYCE
Leading in IT Education
a matrix company

- going back to component styles, in the component metadata we have an option called **encapsulation** which control how the styles for the component will be set
- **ViewEncapsulation.None** - will place the styles as they are in the head and will effect all the components in our application
- **ViewEncapsulation.Native -** will create our component as a shadow dom using the native api of the browser (not all browsers support this)
- **ViewEncapsulation.Emulated** - emulating shadow dom where the styles only affect the component

Component Template

JOHN BRYCE
Leading in IT Education
a matrix company

- A component in an Angular app represents a UI block with view and logic
- The view is written in Angular templating language
- The view can be on a seperate file on inline in the component class.
- Let’s go over advanced features of the component template

Template Reference Variable

JOHN BRYCE
Leading in IT Education
a matrix company

- With template reference variable you can get dom element or instance of a component or directive
- We can use it to get the instance of the child component
- The variable scope is the template
- The syntax is:
  - **<jb-child #someNameThatWillGetTheInstance …>**
- You can transfer the variable to functions
- You can add a name for the variable using the **exportAs** metadata then the syntax will be:
  - **<jb-child #someNameThatWillGetTheInstance=”nameSetInExportAs” …>**
  - This is useful when there is a component and directive on the same element and you want to select a specific one

@ViewChild

JOHN BRYCE
Leading in IT Education
a matrix company

- Property decorator
- Takes a class or a string as selectors
- Try to match the first element in the view that match the selector and populate the property with that value
- Can be used to grab instance of the child component in the parent
- If you have more than one component instance you can place the template reference variable as the string of the decorator and it will find the specific component
- Let’s try and grab the child component instance using the @ViewChild

@ContentChild

JOHN BRYCE
Leading in IT Education
a matrix company

- Property decorator
- Takes a class or a string as selector
- Try to match the first element inside projected content

Communication Parent Child Components

JOHN BRYCE
Leading in IT Education
a matrix company

- To summarize, these are the ways to communicate parent child components:
  - @Input
  - @Output
  - Template reference variable
  - @ViewChild
- Another way to transfer data from parent to child is via content projection (ng-content)

ng-content

JOHN BRYCE
Leading in IT Education
a matrix company

- When the parent use the child selector if we write html in that selector by default we won't see it

  <jb-child>

  <h1> hello world </h1> <!-- this won't show -->

  </jb-child>

- To display the content passed from the parent we can use in the child the <**ng-content></ng-content>** tag

ng-content

JOHN BRYCE
Leading in IT Education
a matrix company

- You can use the **select** attribute in the **ng-content** tag to cherry pick the part of content you want
- Example:

<ng-content select=”[title]”></ng-content>

Component Lifecycle

JOHN BRYCE
Leading in IT Education
a matrix company

- A component has a lifecycle managed by Angular
  - Angular creates the component
  - renders it
  - creates and renders it's children
  - checks it when it's data bound properties change
  - destroys it before removing the component from the DOM
- Angular supplies hooks to these events where you can supply logic of your own to run when the occur
- To act on a hook you have to implement the hook interface
- Let's go over the hooks in the order that they are called

ngOnChanges()

JOHN BRYCE
Leading in IT Education
a matrix company

- Run every time there is a change in the input of the component
- What do u think, changing a property in a non primitive input, will it trigger the **ngOnChanges?**
- Will receive an object describing the changes
  - key is the property
  - value is an object with the following
    - previous value
    - current value
    - is first time

ngOnInit

- Runs only once
- Runs after the first **ngOnChanges**
- Run after the input properties are set
- Good place for initialization logic

ngDoCheck

- Called multiple times
- Called during change detection run
- The first time it’s called after **ngOnInit**

ngAfterContentInit

JOHN BRYCE
Leading in IT Education
a matrix company

- Called once
- After projected content is displayed
- Will be called after properties with @ContentChild will be filled

ngAfterContentChecked

JOHN BRYCE
Leading in IT Education
a matrix company

- Called multiple times
- After Angular checks the projected content

ngAfterViewInit

JOHN BRYCE
Leading in IT Education
a matrix company

- Runs once
- After Angular finished initializing the component view and child component views
- Hook will run after @**ViewChild** is initiated

ngAfterViewChecked

JOHN BRYCE
Leading in IT Education
a matrix company

- Runs after every change detection
- After Angular finish to check the view of the component and all the children
- For example if a child has **ngModel** then the parent will run this hook after the directive finished binding

ngOnDestroy

JOHN BRYCE
Leading in IT Education
a matrix company

- Calls once
- Before Angular destroys the component
- Good place for cleanup

Dynamic Components

JOHN BRYCE
Leading in IT Education
a matrix company

- There are some cases where we don’t know the components we want to place
- The component to place needs to be decided at run time
- We need to programmatically insert new component
- Let’s try to practice this kind of scenario in the following example:
  - we have 3 todo task components
  - the app component need to decide randomly which one of them to place

Dynamic Components - ViewContainerRef

JOHN BRYCE
Leading in IT Education
a matrix company

- We need to tell Angular where we want our dynamic components to be loaded
- To do this we can use a service called **ViewContainerRef**
- If **ViewContainerRef** is dependency injected it represents the view container of the component
- **ViewContainerRef** is a placeholder where you can add one or more **ViewRef**
- **ViewRef** represents and Angular view which represents a grouping of Elements
- We can add a div and using template reference variable we can ask for it's view container
- Let's add that div in the app component
- We can also use **ng-template** or insert a custom directive

Dynamic Components - Todo Components

JOHN BRYCE
Leading in IT Education
a matrix company

- Let’s create 3 task components to be randomized
- We can use **ng generate component Task1/2/3**

Dynamic components - ComponentFactoryResolver

JOHN BRYCE
Leading in IT Education
a matrix company

- Injected service
- Has a method called **resolveComponentFactory** which given a component can dynamically create that component
- In our app component let's add the array of the tasks component and in the on init hook load a random component
- Dynamically created components need to be inserted in the **entryComponents** of the module decorator

Dynamic components - createComponent

JOHN BRYCE
Leading in IT Education
a matrix company

- The last thing to do is call the **createComponent** on the **ViewContainerRef**
- We need to pass this method the **ComponentFactory** we created

ng-template

- ng-template directive represents an Angular template
- Content of this tag will contain part of the template
- This part of template can be injected multiple times
- Consider a use case where instead of copy pasting part of you html code in the component template, you create a variable the holds that template and inject it.
- Let's create an example of a component with a form, we will call this component **LongFormComponent**
- Each form input is made from the following html:

```
<div class="form-group">
    <label>Changing Label</label>
    <input type="text" class="form-control" />
</div>
```

ng-container

- After creating the component and our **ng-template** it's time to inject the template
- We use **ng-container** and a directive ***ngTemplateOutlet** to place the template
- Let's create our long form

ng-template context

JOHN BRYCE
Leading in IT Education
a matrix company

- There is a problem with our form
- All labels are identical and we want them to change between each **ng-template**
- To change them we need to understand the context of **ng-template**
- The context is the same as the component the **ng-template** resides in
- But we can also send additional context available only to the **ng-template**
- **ngTemplateOutlet** directive accepts additional key named **context** with object that will be passed to the **ng-template** scope and will be available only there

```
*ngTemplateOutlet="formControl;context:{labelForInput: 'hello'}"
```

- While in the **ng-template** to access those scope variables we need to define variables and to which key in the object they are attached to

```
<ng-template #formControl let-label="labelForInput">
```

TemplateRef

JOHN BRYCE
Leading in IT Education
a matrix company

- Sometime we need the **ng-template** available in our component code
- We can access the **ng-template** we created using the **@ViewChild** decorator
- This could be helpful if you want to dynamically inject the template.
- For example let's add a small form below the form we just created
- The bottom form will have a text input and a button
- The form will be able to create new elements using the **ng-template** in the upper form
- We need to create a placeholder div in the form and get the **ViewContainerRef** of that element, how do we do that?
- We need to call **ViewContainerRef.createEmbeddedView** and pass the template and context

Web Workers

JOHN BRYCE
Leading in IT Education
a matrix company

- In the next section we will learn about web workers
- What are web workers?
- How can I combine them with my Angular app to improve performance?

What are Web Workers?

JOHN BRYCE
Leading in IT Education
a matrix company

- Web workers allow web content to run scripts in background threads
- The worker thread can perform tasks without interfering with the UI
- Workers can fetch data from the server
- Web content and workers can communicate by posting messages to event handlers
- To learn about web workers we will do the following:
  - create a simple web application (without Angular this time)
  - create a worker thread to grab our tasks from the server
  - pass the data back to the web content
  - display the content

Hello World Worker

JOHN BRYCE
Leading in IT Education
a matrix company

- A worker is created using the contructor **Worker()**
- The constructor is given the name of a JS script file to run in the worker thread
- Workers run in another global context not **window** the global context is: **DedicatedWorkerGlobalScope**
- A worker can't directly manipulate the DOM
- Communication is with messages and event handlers.
  - we send a message via **postMessage()** method
  - we respond to message via **onmessage** event handler and the data is received as an argument to the handler function
- First let's experiment with the worker and create a worker which sends back an hello world message to the main thread

Todo Worker

JOHN BRYCE
Leading in IT Education
a matrix company

- Now let’s try to make our worker grab data from the server and send it back to the main thread
- **worker.terminate** will kill the worker
- Worker can also close themselves by calling the **close()**
- Make sure to close the worker after you send the data

Shared Workers

JOHN BRYCE
Leading in IT Education
a matrix company

- The worker covered until now id called dedicated worker
- A dedicated worker has its own context and can only communicate with the parent that spawned it
- **SharedWorker** communicate through a **port** object and can share data between the parent and other shared workers
- The parent or shared worker can attach an **onmessage** or call **postmessage** on the port object

Angular and Web Workers

JOHN BRYCE
Leading in IT Education
a matrix company

- How can web workers help us optimize our Angular app?
- We can run Angular in a web worker thread
  - we keep the main thread clear from Angular logic, the main thread is just passing events to the worker and passing updates from the worker to the dom
  - we run Angular logic and our application logic in a worker keeping the main thread clear and not blocking the UI
  - as a result the app feels much more responsive for the user

@angular/platform-webworker

JOHN BRYCE
Leading in IT Education
a matrix company

- First we need to install the Angular packages for web workers

  npm install --save @angular/platform-webworker @angular/platform-webworker-dynamic

- We can't use @angular/cli so we will have to **eject** the webpack configuration and go manual
  - the reason is we need to bundle a seperate file for the worker and angular CLI doesn't support it yet

app.module.ts

JOHN BRYCE
Leading in IT Education
a matrix company

- In our AppModule we switch **BrowserModule** with **WorkerAppModule**

main.ts

- In main.ts we need to replace **platformBrowserDynamic** with **bootstrapWorkerUi** and give it the name of the bundle file to run

Entry Point Worker

JOHN BRYCE
Leading in IT Education
a matrix company

- Create the worker entry point file

```
import 'polyfills.ts';
import '@angular/core';
import '@angular/common';


import { platformWorkerAppDynamic } from '@angular/platform-webworker-dynamic';
import { AppModule } from './app/app.module';


platformWorkerAppDynamic().bootstrapModule(AppModule);
```

Modify webpack.config.js

JOHN BRYCE
Leading in IT Education
a matrix company

- You will need to modify the webpack.config.js file to include the entry for the worker
- You will need to the **HtmlWebpackPlugin** to not include the worker script since it will be loaded by the worker
- **CommonChunksPlugin** will have to be modified manually to not include the worker bundle file
- **AngularCompilerPlugin** we will have to set the new entry module manually
- Run: **npm start** to see the app

Example of working example of Angular with web workers is in this url

https://github.com/ywarezk/ng2-web-workers

Do Not Use Angular with Web Workers

JOHN BRYCE
Leading in IT Education
a matrix company

- The API is still experimental
- Does not support routing yet
- Does not work well with lazy loaded modules
- Does not support server side rendering
- Does not combine well with @angular/cli
- The changes in the actual Angular app are minimal, so we can develop our app like we use to, and keep track on this feature and embed it when it’s more mature