



Building Your Own Widget with the ArcGIS API for JavaScript

Matt Driscoll – @driskull

JC Franco – @arfnocode

Agenda

- About Widgets
- Widget Framework
- Styling
- Build A Widget!

Widgets

About

- What?
 - Encapsulated UI components
 - Cohesive (integrated, unified)
 - Single-purpose pieces of functionality
- Why?
 - Reusable
 - Interchangeable
- How?
 - `esri/Widgets/Widget`

Widget Framework

Architecture

- Views + ViewModels
 - Separation of concerns
 - Reusable
 - UI replacement
 - Easier integration
- Built with TypeScript

Views

- Extend `esri/widgets/Widget`
- Rely on `ViewModel`
- Focus on UI

ViewModels

- Extend `esri/core/Accessor`
- Provide APIs to support view
- Focus on business logic

View + ViewModel in action

- View renders its state
 - state = view + ViewModel props
- View calls VMs APIs
 - causes a change (e.g., property or result)
- View updates

esri/widgets/Widget

- Lifecycle
- API consistency
 - Unified object constructor
 - Properties
 - Watching

Lifecycle

- constructor
- postInitialize
- render
- destroy

render

- Defines UI
- Reacts to state
- Uses JSX
- VDOM

render example

```
render() {
  const { x, y, scale } = this;

  return (
    <div bind={this} class={CSS.base} onclick={this._handleClick}
      title="map info" tabIndex={0}>
      <p>x: {x}</p>
      <p>y: {y}</p>
      <p>scale: {scale}</p>
    </div>
  );
}
```

Widget rendering

Implementing Decorators

- @subclass + declared
- @property
 - autocast
 - computed
 - read-only
 - aliased
- @aliasOf
- @renderable
- @accessibleHandler

Implementing

- Extend `esri/widgets/Widget`

```
/// <amd-dependency path="esri/core/tsSupport/declareExtendsHelper" name="__extends" />
/// <amd-dependency path="esri/core/tsSupport/decorateHelper" name="__decorate" />

@subclass("MyWidget")
class MyWidget extends declared(Widget) {

}

export = MyWidget;
```

Implementing

- Implement render

```
// ...  
class MyWidget extends declared(Widget) {  
  render() {  
    return (  
      <div>I'm a widget</div>  
    );  
  }  
}  
// ...
```


Implementing

- Define properties

```
// ...
@property()
@renderable()
name: string = "I'm a widget";

render() {
  return (
    <div>{this.name}</div>
  );
}
// ...
```

New in 4.7

- Unified CSS classes
- Animation hooks
- Default `iconClass` and `label` properties

Unified CSS classes

- Use `class` attribute
- `Widget#classes` builds node class
- `join` utility is deprecated
- `classes` attribute is deprecated

Unified CSS classes

```
// 4.6
render() {
  const dynamicClasses = { [CSS.active]: this.isActive };

  return (
    <div class={join(CSS.base, CSS.mixin)} classes={dynamicClasses}>{/* ... */}</div>
  );
}
```

Unified CSS classes

```
// 4.7
render() {
  const dynamicClasses = { [CSS.active]: this.isActive };

  return (
    <div class={this.classes(CSS.base, CSS.mixin, dynamicClasses)}>{/* ... */}</div>
  );
}
```

Animation hooks

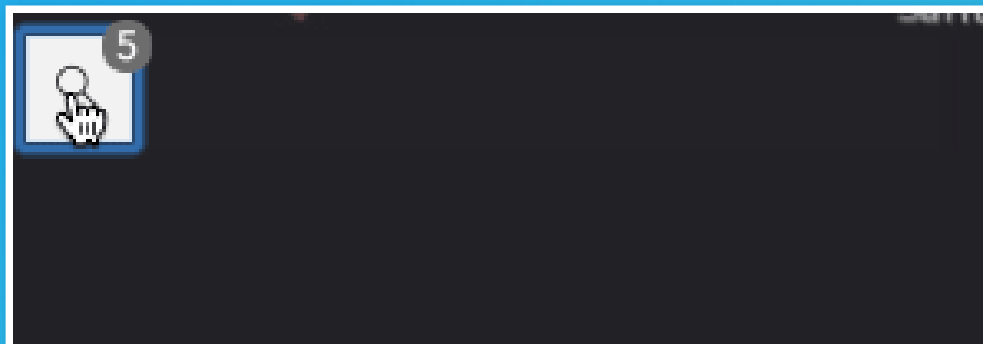
- CSS or JS
- Node attributes
 - `enterAnimation`
 - `exitAnimation`
 - `updateAnimation` (JS-only)
- `cssTransition` (CSS-only)

Animation hooks

```
render() {
  const { visible } = this;

  const content = visible ? (
    <div enterAnimation={cssTransition("enter", CSS.fadeIn)}
      exitAnimation={cssTransition("exit", CSS.fadeOut)}>{/*...*/}</div>
  ) : null;

  return (
    <div class={CSS.base}>
      {content}
    </div>
  );
}
```



iconClass and label

- UI hints for container widgets
 - Expand uses iconClass (example)
- iconClass - Esri icon font class name
- label - localized widget label

```
@subclass("MyWidget")
class MyWidget extends declared(Widget) {

    @property()
    iconClass: string = "esri-icon-basemap";

    @property()
    label: string = i18n.widgetLabel;

}
```


Recap

- Views + ViewModels
- `esri/widgets/Widget`
- `render()`

Styling

How?

- BEM
- Sass

Naming CSS classes

Block Element Modifier (BEM)

- Scopes styles to blocks
- Semantic
- Low specificity

```
// block
.example-widget {}

// block_element
.example-widget__input {}

// block--modifier
.example-widget--loading {}

// block_element--modifier
.example-widget__input--disabled {}
```

Styling with Sass

- CSS preprocessor
- Powered-up CSS
 - Nesting
 - Variables
 - Functions
 - Mixins
 - Inheritance

Sass makes it easier to...

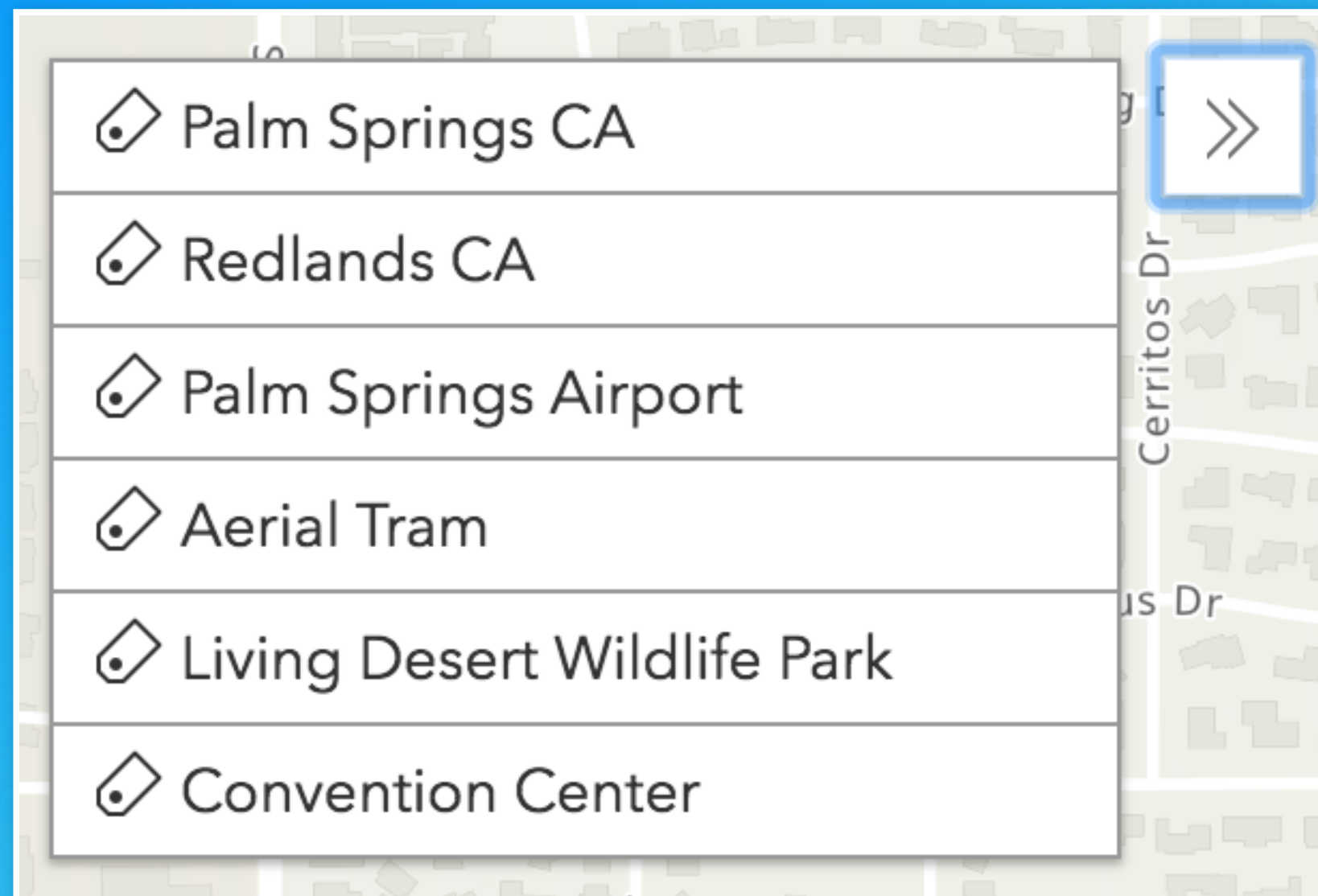
- Restyle
- Organize
- Write less code :)

Recap

- BEM
- Sass

Let's build a widget!

- Bookmarks Doc
- Bookmarks Completed Demo



VM: API Design

```
interface BookmarksViewModel {  
  bookmarkItems: Collection<BookmarkItem>;  
  state: "loading" | "ready" | "disabled"; // will be computed property  
  view: MapView;  
  goTo(item: BookmarkItem): IPromise<any>;  
}  
  
interface BookmarkItem {  
  active: Boolean;  
  extent: Extent;  
  name: string;  
}
```

Build Steps

- Demo Start
- HTML Steps
- ViewModel Steps
- View Steps
- Sass Steps

Let's Recap

- Widgets are single functionality UI components
- We use them for reusability/interchangeability
- Widget Framework
- Constructing a widget
 - ViewModels
 - Views
- Styling
 - BEM
 - Sass

Suggested Session

- [ArcGIS API for JavaScript: Customizing Widgets](#)

Additional Resources

- [Implementing Accessor](#)
- [Setting up TypeScript](#)
- [Widget Development](#)
- [JS API SDK](#)
- [Styling](#)
- [Widget Patterns](#)

Questions?

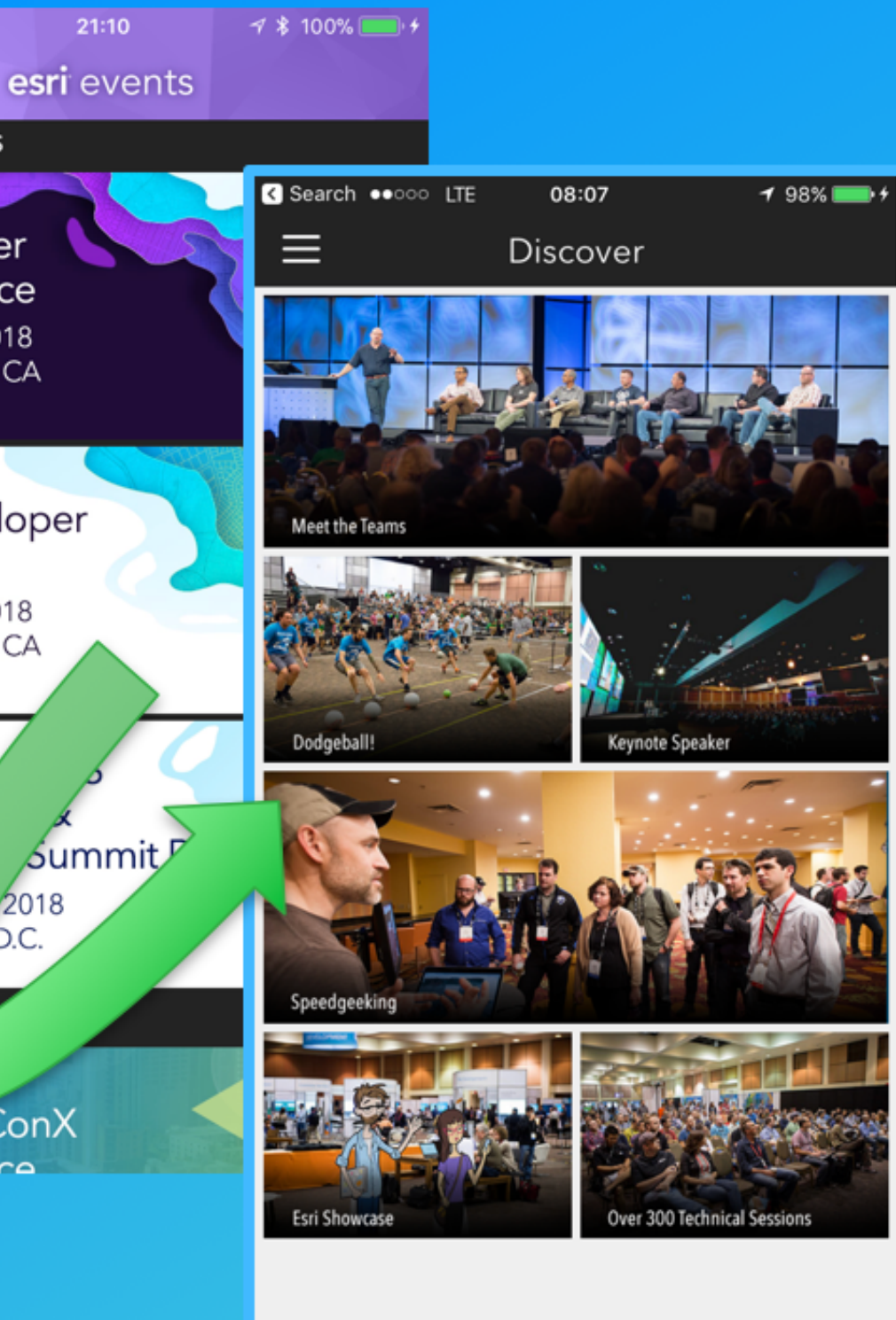
For example

🤔 *Where can I find the slides/source?*

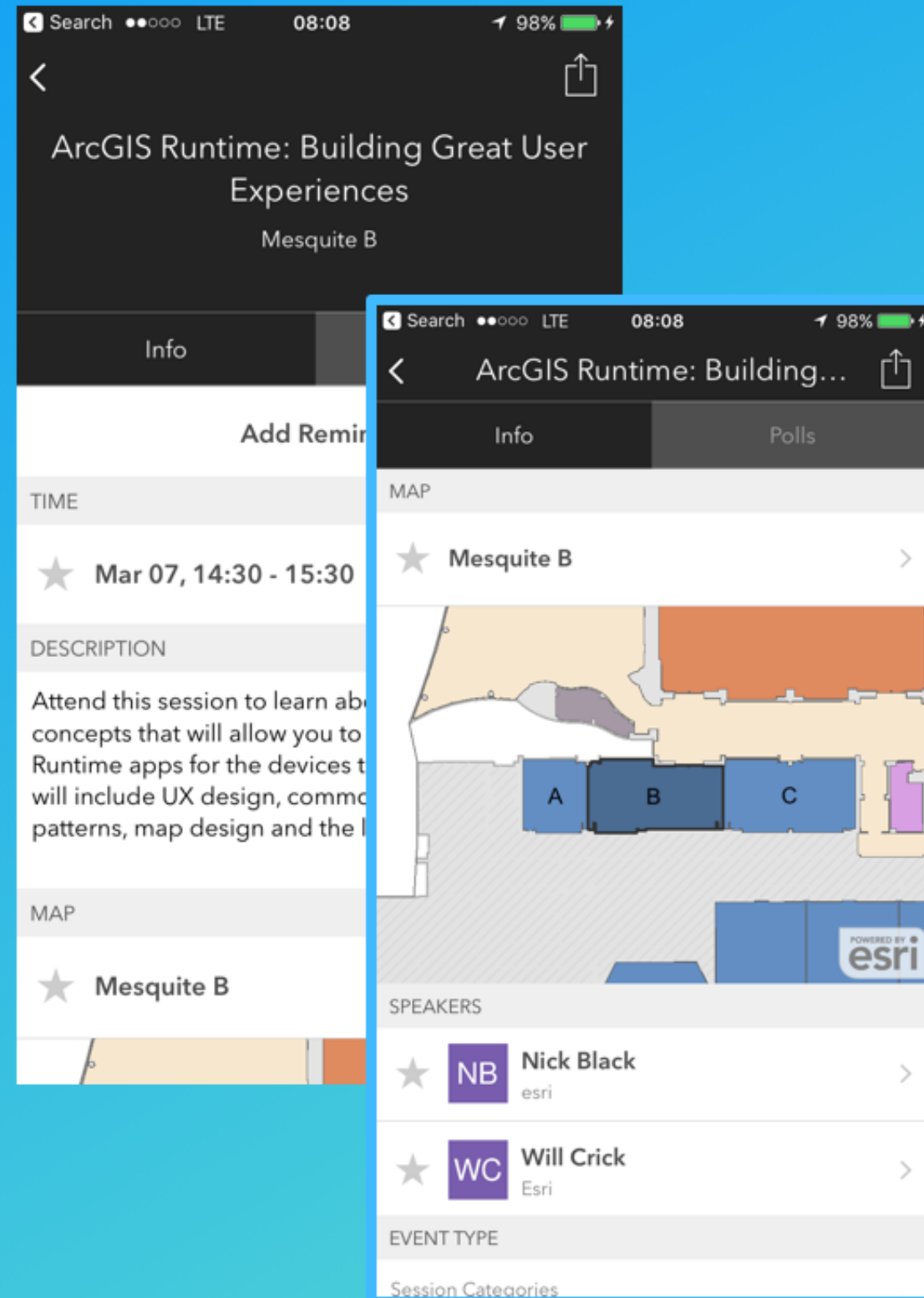
👉 esriurl.com/buildwidgetsds2018 👈

Please Take Our Survey!

Load the Esri Events app
and find your event

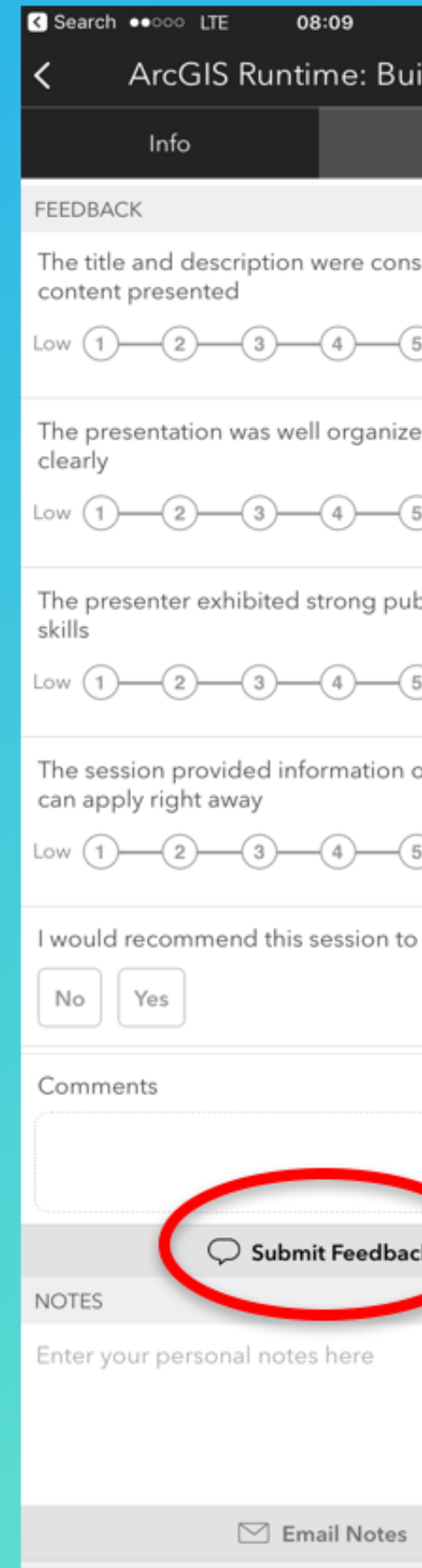


Select the session you
attended



Scroll down to the
“Feedback” section

Complete Answers,
add a Comment,
and Select “Submit”



Thank you!

