Web AppBuilder for ArcGIS
Dive Into Mobile Development

Yiwei Ma & Kevin Gao
Yiwei Ma
Software Developer
Kevin Gao
User Interface Engineer
Theme
A theme in WAB is a template framework representing the look and feel of an app.
And a theme is responsible for handling the experience in both desktop and mobile views.
Out-of-Box Themes

- Billboard
- Box
- Dart
- Foldable
- Dashboard
- Jewelry Box
- Launchpad
- Plateau
- Tab
All Out-of-Box Themes are optimized for mobile
Launchpad Theme: Desktop VS Mobile
What about Custom Themes?
It is designers and developers’ responsibility to provide a mobile experience for custom themes.
Themes: Components
Major Components In A WAB Theme

• Layout

• Panel

• Style

• Controller
What Composes a WAB Theme?

- Layout
- Panel
- Style
- Controller
What Composes a WAB Theme?

- Layout
- Panel
- Style
- Controller
What Composes a WAB Theme?

- Layout
- Panel
- Style
- Controller
What Composes a WAB Theme?

- Layout
- Panel
- Style
- Controller
Each of these four components is capable of providing extra, mobile-specific configurations and functionalities to together deliver a complete mobile experience.
How WAB Determines Mobile View
Mobile Breakpoint

• The default threshold of the screen size to enter into mobile view is: < 600px eight in width or height
• The number is stored in `window.jimuConfig.breakPoints` which can be reset

```
window.jimuConfig.breakPoints
(2) [600, 1280]
```
Key Config and Properties In Mobile Mode:

- The properties defined in the `mobileLayout` section from the selected layout will be applied.

- An extra class `jimu-ismobile` will be added to the `jimu-layout-manager` dom node.

```html
    ▼<div id="jimu-layout-manager" widgetid="jimu-layout-manager" class="jimu-ismobile LaunchpadTheme_default"></div>
```

- App wide property `appInfo.isRunInMobile` is turned to true.

```javascript
    appInfo.isRunInMobile
    → true
```
How Launchpad theme creates its mobile UI
Layout (default)
"mobileLayout": {
  "widgetOnScreen": {
    "widgets": {
      "themes/LaunchpadTheme/widgets/AnchorBarController/Widget":{
        "position": {
          "bottom": 0,
          "height": 40,
          "width": "100%",
          "zIndex": 0
        },
        "version": "2.7"
      },
      "themes/LaunchpadTheme/widgets/Header/Widget":{
        "position": {
          "left": 0,
          "right": 0,
          "top": 0,
          "height": 46,
          "relativeTo": "map"
        },
        "version": "2.7"
      },
      "widgets/Search/Widget":{
        "position": {
          "left": 10,
          "top": 10,
          "right": 10,
          "zIndex": 0
        },
        "version": "2.7"
      }
    }
  }
}

mobileLayout Section

```
"widgets/ZoomSlider/Widget": {  
  "position": {  
    "right": 10,  
    "bottom": 160  
  },  
  "version": "2.7"  
},
"widgets/HomeButton/Widget":{  
  "position": {  
    "right": 10,  
    "bottom": 75  
  },  
  "version": "2.7"  
},
"widgets/MyLocation/Widget":{  
  "inPanel": false,  
  "position": {  
    "right": 10,  
    "bottom": 115  
  },  
  "version": "2.7"  
},
"widgets/AttributeTable/Widget":{  
  "version": "2.7",  
  "position": {  
    "left": 10,  
    "bottom": 70  
  }  
},
```
Panel (LaunchpadPanel)
Panel (LaunchpadPanel)

The panel takes half of screen height by default and takes full screen when maximized in mobile mode. This logic is applied after checking whether the app is running in mobile mode by calling `window.appInfo.isRunInMobile`:

```javascript
setPosition: function(position){
  ...
  if(window.appInfo.isRunInMobile){
    position.left = 0;
    position.top = box.h / 2;
    position.width = box.w;
    position.height = box.h / 2;
  }
  ...
},
```
Style (CSS)
In mobile mode, the border radiuses of the search widget are made into squares instead of rounded corners, and box shadow is also applied.

In Launchpad.css, you can find the following CSS styles that override some of the properties when in mobile mode by using mobile only class name *is-mobile*:

```
.claro.is-mobile .jimu-widget-search {
  -webkit-box-shadow: 0 1px 0.5px rgba(0,0,0.3), 0 2px 2px rgba(0,0,0.2);
  box-shadow: 0 1px 0.5px rgba(0,0,0.3), 0 2px 2px rgba(0,0,0.2);
}
.clsr.is-mobile .jimu-widget-search .arcgisSearch .searchToggle, 
.claro.is-mobile .jimu-widget-search .arcgisSearch .searchGroup .searchInput, 
.claro.is-mobile .jimu-widget-search .arcgisSearch .searchSubmit {
  -webkit-border-radius: 0;
  border-radius: 0;
}
```

*Note: Launchpad theme adds another class “is-mobile” to the body tag in mobile mode.*
It is worth mentioning...
What Makes Up The WAB UI?

- Dojo dijits
- Jimu widgets
- ArcGIS API widgets
How UI Libraries Work In WAB:

**Theme:** common.css, style.css

**Jimu:** jimu.css, jimu-override.css, etc.

**ArcGIS API for JavaScript:** esri.css

**Dojo dijits:** claro.css
Controller(s)

• Header:
  - The header controller is minimized to only display the link dropdown button by setting the visibility of other UI elements to hidden:

```
.claro.is-mobile .jimu-widget-header .logo,  
.claro.is-mobile .jimu-widget-header .titles {
  visibility: hidden;
}
```

  - And additional CSS styles are applied to make the links dropdown look like the other map controls (zoom, home, etc.)
Controller(s)

- **AnchorBarController**:
  - The controller is positioned at the bottom of the screen and takes full width. This is defined in the `mobileLayout` section of the layout config:

```json
"themes/LaunchpadTheme/widgets/AnchorBarController/Widget": {
  "position": {
    "bottom": 0,
    "height": 40,
    "width": "100%",
    "zIndex": 0
  },
```

- Any more widget icons that cannot be displayed at the same time will be wrapped in a group button.
Let’s Create A Custom Mobile UI
Major Changes: Controller

- Display the sidebar controller as a horizontal header at the top of the app
- The list of widget icons are wrapped in a side menu and is initially hidden
- It does not require at least one widget to be opened
Major Changes: Panel

- Always takes full screen
- Has a close button
Step 1: Update Layout
Add "mobileLayout" Section

- Add `mobileLayout` at the end of layout’s `config.json`
- Also add `widgetOnScreen` and `map` attributes to `mobileLayout`, since they will have their custom layouts in mobile

```json
"mobileLayout": {
    "widgetOnScreen": {
        "widgets": {}
    },
    "map": {}
}
```
Reposition Widgets

Using controller as an example:

```javascript
"themes/SidebarTheme/widgets/SidebarController/Widget":{
  "position": {
    "left": 0,
    "right": 0,
    "top": 0,
    "height": 110
  },
},
```
Step 2: Controller Widget
Controller Widget

• The controller in mobile view will:
  - Not require at least one widget to be opened
  - Have a menu toggle that only appears in mobile view to toggle on/off the menu of widgets (from widget pool)
Update HTML Template

- Modify Widget.html to add UI elements needed for mobile view:

```html
<nav>
  <div class="menu-toggle">
    <span data-dojo-attach-point="menuToggle" class="menu-toggle-btn">
      <img src="${folderUrl}/images/menu.svg" alt="menu toggle"/>
    </span>
  </div>
  <div class="logo">
    <img data-dojo-attach-point="logoNode"/>
  </div>
  <div data-dojo-attach-point="containerNode" class="container-section menu">
    <div data-dojo-attach-point="iconsNode" class="icon-list jimu-main-background">
    </div>
  </div>
  <div data-dojo-attach-point="userNode" class="user-section">
  </div>
</nav>
```

Read more on creating widgets with Dojo Toolkit
Update Widget.js

1. Modify JS code to reflect HTML template changes
2. Do not display the first widget from the widget pool when app launches

```javascript
_initUI: function() {
  ...
  if(!hasOpenAtStartWidget && !appInfo.isRunInMobile) {
    this._showWidgetContent(allConfigs[0]);
    this._activateIconNode(firstIconNode);
  }
  ...
},
```

Do not show the first widget if it is running in mobile mode
Update Widget.js

3. Add mobile specific CSS rules to controller’s style.css:

```html
/* Mobile View */
.jimu-ismobile .jimu-widget-sidebar-controller {
  display: flex;
  flex-direction: row;
}
.jimu-ismobile .jimu-widget-sidebar-controller .menu-toggle-btn {
  display: block;
  width: 32px;
  height: 32px;
  padding: 4px;
  margin: 10px;
}
.jimu-ismobile .jimu-widget-sidebar-controller .logo {
  padding: 0;
  min-height: 0;
  margin: 10px;
  flex: 1;
}
.jimu-ismobile .jimu-widget-sidebar-controller .menu {
  display: none;
}
...
Update Widget.js

4. Add logic to toggle menu in Widget.js:
   - Add a new method called `toggleMenuMobile` to handle showing / hiding the menu container

```javascript
toggleMenuMobile: function() {
  if(appInfo.isRunInMobile) {
    var layoutId = jimuConfig.layoutId,
        layoutContainer = document.getElementById(layoutId);
    if(!this._menuOpened) {
      domClass.remove(this.containerNode, 'jimu-widget-sidebar-controller-menu_opened');
      this.domNode.insertBefore(this.containerNode, this.userNode);
    } else {
      domClass.add(this.containerNode, 'jimu-widget-sidebar-controller-menu_opened');
      layoutContainer.appendChild(this.containerNode);
    }
    this._menuOpened = !this._menuOpened;
  }
}
```

Supporting classes are added for styling the menu

The menu container is temporarily appended to app’s root node to avoid z-index issues
Update Widget.js

- Attach onclick event to the menu toggle button and call toggleMenuMobile when clicked

```javascript
_initUI: function() {
    
    this.own(on(this.menuToggle, 'click', function () {
        this.toggleMenuMobile();
    }));
},
```
Update Widget.js

5. Add more CSS rules to style menu element:

```
.jimu-widget-sidebar-controller-menu .icon-node {
  text-align: left;
  padding-left: 30px;
}

.jimu-widget-sidebar-controller-menu .icon-node img {
  display: inline-block;
  margin-bottom: 0;
  margin-right: 20px;
  vertical-align: middle;
}
```

(Only part of style are shown)
Step 3: Panel Widget
Panel Widget

• Make the panel take full screen:
  - Method 1: via CSS*
    ```
    .jimu-ismobile .jimu-header-panel {
        width: auto !important;
        top: 0 !important;
        left: 0 !important;
        right: 0 !important;
    }
    ```
  - Method 2: vis JS
    ```
    function() {
      ... 
      if(window.appInfo.isRunInMobile) {
        this.domNode.style.left = "0";
        this.domNode.style.width = "100%";
      }
    }
    ```

*It is not recommended to use !important to override styles
Panel Widget

• Add close button:
  - Update Panel.html template to add a close button:

```html
<span role="button" class="close-btn" data-dojo-attach-point="closeButtonNode"></span>
```

  - Attach onclick event to the button and add logic to close the panel:

```javascript
postCreate: function() {
  this.inherited(arguments);

  var _this = this;
  this.own(on(this.closeButtonNode, 'click', function() {
    _this.panelManager.closePanel(_this);
  }));
},
```
Panel Widget

• Create a new folder called images and add a close icon.
• In panel’s style.css files, add CSS rules to style the close button:

```css
.jimu-header-panel .close-btn {
    display: none;
    width: 32px;
    height: 32px;
    cursor: pointer;
    background-image: url(images/close.svg);
    background-position: center center;
    background-repeat: no-repeat;
    position: absolute;
    top: 0;
    right: 0;
    margin: 10px;
}

.jimu-ismobile .jimu-header-panel .close-btn {
    display: block;
}
```
Step 4: Style
Style

• Apply CSS styles to remove border-radius from search widget:

```css
.jimu-ismobile .jimu-widget-search .arcgisSearch.searchToggle,
.jimu-ismobile .jimu-widget-search .arcgisSearch.searchGroup.searchInputGroup,
.jimu-ismobile .jimu-widget-search .arcgisSearch.searchGroup.hasMultipleSources.searchBtn,
.jimu-ismobile .jimu-widget-search .arcgisSearch.searchBtn {
    border-radius: 0;
}
```
• Add CSS overrides to restyle mobile popup:

```css
/* Mobile Popup */
.esriPopupMobile {
  bottom: 0 !important;
  top: auto !important;
  left: 0 !important;
  right: 0 !important;
  background-color: #fff;
  box-shadow: none;
  border-radius: 0;
  font-family: inherit;
  box-shadow: 0 -2px 7px rgba(0, 0, 0, 0.12);
}
.esriPopupMobile .sizer {
  width: auto;
}
...
Q&A