



ESRI DEVELOPER SUMMIT 2023

Widget Styling and Custom Themes

Agenda

- Components and Widget Styling
- Themes
- Activity

Components and Widget Styling

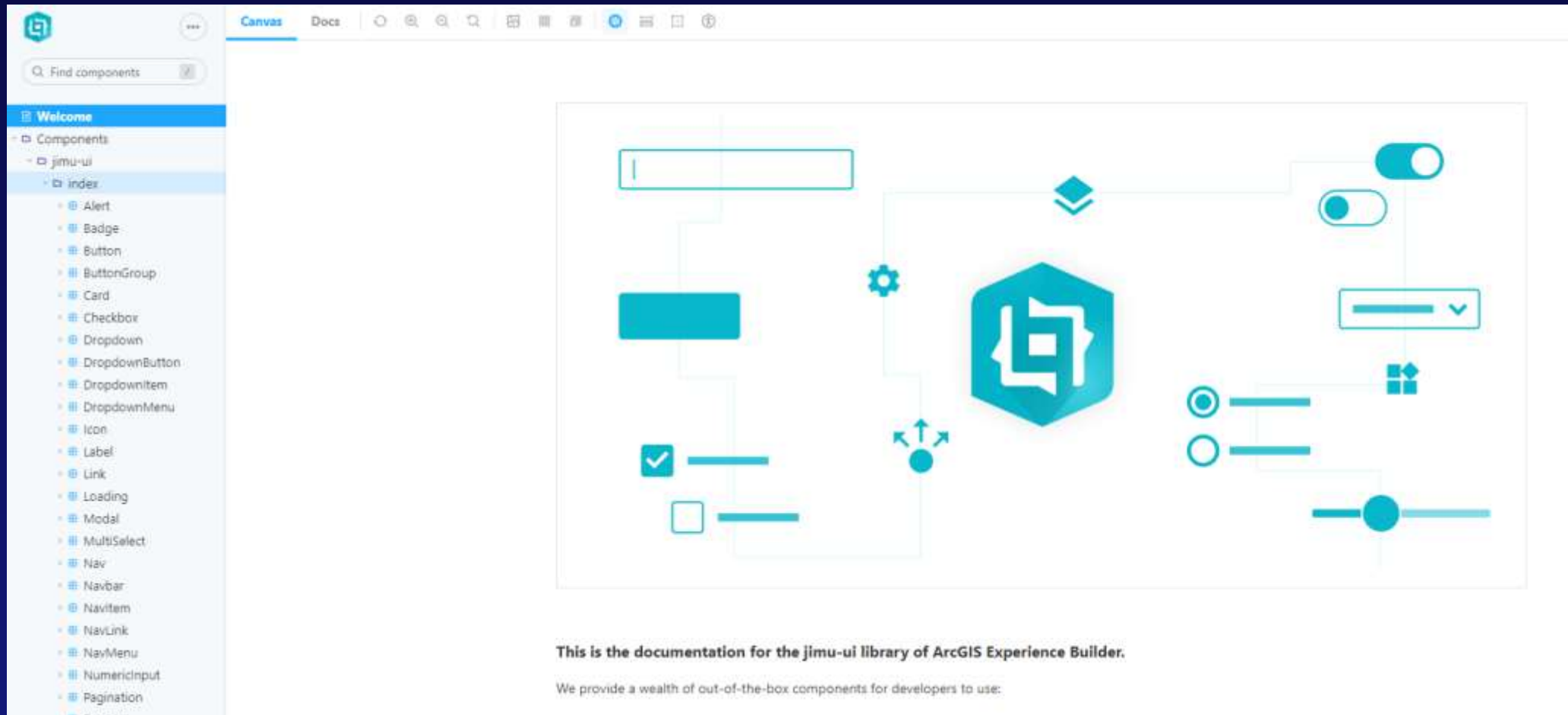
```
current_layer = QgsMapLayerRegistry.instance().mapLayersByName('DEM')[0]  
view.setMapLayer(current_layer)  
if there were problems with  
the styling, print it
```

```
current_layer = QgsMapLayerRegistry.instance().mapLayersByName('DEM')[0]  
view.setMapLayer(current_layer)  
if there were problems with  
the styling, print it
```

Jimu UI Library

- UI library of components for developers to use:
 - client\jimu-ui\index.d.ts
- Basic UI components:
 - button, dropdown, form controls,
 - icon, navigation, modal,
 - grid layout container, etc.
- Advanced UI components:
 - date picker,
 - resource selector,
 - expression builder, etc.

Storybook Documentation



developers.arcgis.com/experience-builder/api-reference/jimu-ui/

Using UI Components

```
3  // in widget.tsx:
4  import { React, AllWidgetProps } from 'jimu-core';
5  import { Button, Icon } from 'jimu-ui'; // import components
6
7  // Create an svg icon using Icon component:
8  const iconNode = <Icon icon={require('jimu-ui/lib/icons/star.svg')} />;
9
10 export default class Widget extends React.PureComponent<AllWidgetProps<IMConfig>, any> {
11   render() {
12     // Add Button component containing an icon to the widget:
13     return <Button type='primary'>{iconNode} primary button</Button>;
14   }
15 }
16
```

Styling your custom widget

- Utility Classes
- Inline CSS
- CSS/Sass style files
- CSS-in-JS
- Styled Components

Utility Classes

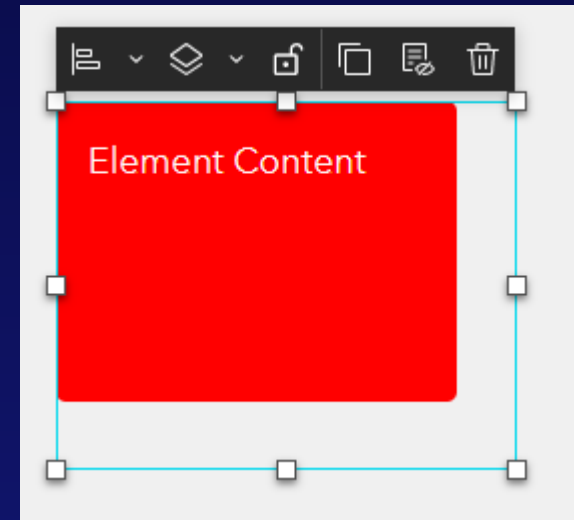
- Jimu UI provides the same CSS utility classes as Bootstrap to quickly apply styles to UI elements.
 - <https://getbootstrap.com/docs/4.3/utilities>

```
9   render() {  
10    return (  
11      <div className='w-100 p-3 bg-primary text-white'>  
12        <p>Were using utility classes!</p>  
13      </div>  
14    );  
15  }
```

Were using utility classes!

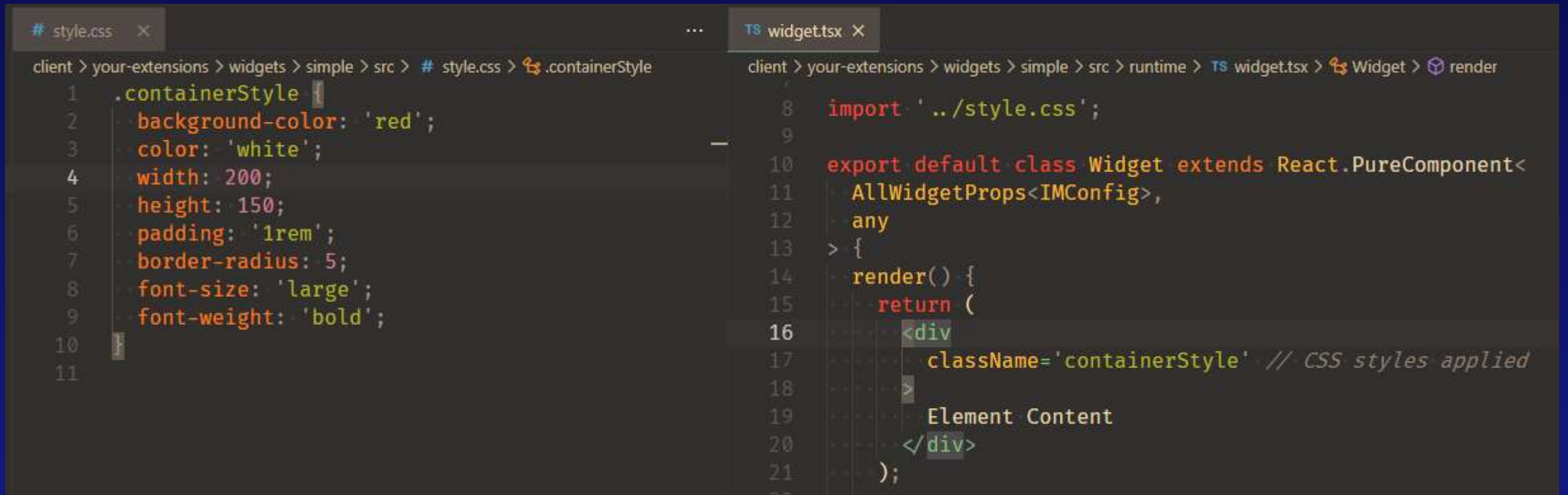
Inline CSS

```
12  render() {
13    ... const containerStyle = {
14    ...   background: 'red',
15    ...   color: 'white',
16    ...   width: 200,
17    ...   height: 150,
18    ...   padding: '1rem',
19    ...   borderRadius: 5,
20    ...   fontSize: 'large',
21    ...   FontWeight: 'bold',
22    ... };
23
24    ... return (
25    ...   <div
26    ...     style={containerStyle} // CSS styles applied
27    ...   >
28    ...     Element Content
29    ...   </div>
30    ... );
31  }
```



Using CSS Style Sheets

- Define CSS styles in external stylesheet files and import them separately in the widget.
- Accepted stylesheet file extensions are: .css, .sass and .scss.



The screenshot shows a code editor with two files open: `style.css` and `widget.tsx`. The `style.css` file defines a `.containerStyle` class with various styling properties. The `widget.tsx` file imports this style and uses it in a React component's render method.

```
# style.css
1 .containerStyle {
2   background-color: 'red';
3   color: 'white';
4   width: 200;
5   height: 150;
6   padding: '1rem';
7   border-radius: 5;
8   font-size: 'large';
9   font-weight: 'bold';
10 }
11

TS widget.tsx
client > your-extensions > widgets > simple > src > runtime > TS widget.tsx > Widget > render
8 import './style.css';
9
10 export default class Widget extends React.PureComponent<
11   AllWidgetProps<IMConfig>,
12   any
13 > {
14   render() {
15     return (
16       <div
17         className='containerStyle' // CSS styles applied
18       >
19         Element Content
20       </div>
21     );
22   }
23 }
```

CSS in JS

- The “CSS prop”
 - CSS styles can be written in template literals, which allows you to write JS logics inside of CSS.

```
100  
101     const over = this.state.scale > 5000000;  
102     const scaleStyle = css`  
103       font-weight: ${over ? 'bold' : 'normal'};  
104       color: ${over ? 'red' : 'black'};  
105     `;  
106
```

```
<span css={scaleStyle}>Scale 1:{this.state.scale}</span>
```

CSS in JS

- Styled components

- Like css prop, but for components
- Style components using Template Literals
- The "styled" approach is perfect for creating reusable components within your widget

```
39 // A styled button component:  
40 const StyledButton = styled.button`  
41   color: white;  
42   border: none;  
43   border-radius: 5px;  
44   background-color: blue;  
45   transition: 0.15s ease-in all;  
46   &:hover {  
47     background-color: green;  
48   }  
49 `;
```

```
... //<Button type='primary' onClick={this.resetView}>  
... // Reset  
... //</Button>  
... // A styled button component:  
... <StyledButton onClick={this.resetView}>Reset</StyledButton>
```


Custom Themes

```
currentLayer = view.map.allLayers[0]  
view.whenLayerViewed(layers)  
  {layer:layerViewed, console.log}  
  // if there were problems with  
  // the current layer, print it
```

```
currentLayer = view.map.allLayers[0]  
currentLayer = view.map.allLayers[0]  
mapView = view.map.allLayers[0]  
view.whenLayerViewed(layers)  
  {layer:layerViewed, console.log}
```


What is available

Basic:

- Colors
- Typography
- Spacing

Advanced

- Style override via Sass CSS
- Assets: fonts, images, etc.

Files

- variables.json
- thumbnail.png
- manifest.json

```

1111 public void write() {
1112     StreamWriter sw = new StreamWriter("MyLog.txt",
1113     true);
1114     sw.WriteLine("1111");
1115     sw.Close();
1116 }
1117
1118 public void read() {
1119     StreamReader sr = new StreamReader("MyLog.txt");
1120     string line;
1121     while ((line = sr.ReadLine()) != null)
1122     {
1123         Console.WriteLine(line);
1124     }
1125     sr.Close();
1126 }
1127
1128 }
1129
1130 }
1131
1132 }
1133
1134 }
1135
1136 }
1137
1138 }
1139
1140 }
1141
1142 }
1143
1144 }
1145
1146 }
1147
1148 }
1149
1150 }
1151
1152 }
1153
1154 }
1155
1156 }
1157
1158 }
1159
1160 }
1161
1162 }
1163
1164 }
1165
1166 }
1167
1168 }
1169
1170 }
1171
1172 }
1173
1174 }
1175
1176 }
1177
1178 }
1179
1180 }
1181
1182 }
1183
1184 }
1185
1186 }
1187
1188 }
1189
1190 }
1191
1192 }
1193
1194 }
1195
1196 }
1197
1198 }
1199
1200 }
1201
1202 }
1203
1204 }
1205
1206 }
1207
1208 }
1209
1210 }
1211
1212 }
1213
1214 }
1215
1216 }
1217
1218 }
1219
1220 }
1221
1222 }
1223
1224 }
1225
1226 }
1227
1228 }
1229
1230 }
1231
1232 }
1233
1234 }
1235
1236 }
1237
1238 }
1239
1240 }
1241
1242 }
1243
1244 }
1245
1246 }
1247
1248 }
1249
1250 }
1251
1252 }
1253
1254 }
1255
1256 }
1257
1258 }
1259
1260 }
1261
1262 }
1263
1264 }
1265
1266 }
1267
1268 }
1269
1270 }
1271
1272 }
1273
1274 }
1275
1276 }
1277
1278 }
1279
1280 }
1281
1282 }
1283
1284 }
1285
1286 }
1287
1288 }
1289
1290 }
1291
1292 }
1293
1294 }
1295
1296 }
1297
1298 }
1299
1300 }
1301
1302 }
1303
1304 }
1305
1306 }
1307
1308 }
1309
1310 }
1311
1312 }
1313
1314 }
1315
1316 }
1317
1318 }
1319
1320 }
1321
1322 }
1323
1324 }
1325
1326 }
1327
1328 }
1329
1330 }
1331
1332 }
1333
1334 }
1335
1336 }
1337
1338 }
1339
1340 }
1341
1342 }
1343
1344 }
1345
1346 }
1347
1348 }
1349
1350 }
1351
1352 }
1353
1354 }
1355
1356 }
1357
1358 }
1359
1360 }
1361
1362 }
1363
1364 }
1365
1366 }
1367
1368 }
1369
1370 }
1371
1372 }
1373
1374 }
1375
1376 }
1377
1378 }
1379
1380 }
1381
1382 }
1383
1384 }
1385
1386 }
1387
1388 }
1389
1390 }
1391
1392 }
1393
1394 }
1395
1396 }
1397
1398 }
1399
1400 }
1401
1402 }
1403
1404 }
1405
1406 }
1407
1408 }
1409
1410 }
1411
1412 }
1413
1414 }
1415
1416 }
1417
1418 }
1419
1420 }
1421
1422 }
1423
1424 }
1425
1426 }
1427
1428 }
1429
1430 }
1431
1432 }
1433
1434 }
1435
1436 }
1437
1438 }
1439
1440 }
1441
1442 }
1443
1444 }
1445
1446 }
1447
1448 }
1449
1450 }
1451
1452 }
1453
1454 }
1455
1456 }
1457
1458 }
1459
1460 }
1461
1462 }
1463
1464 }
1465
1466 }
1467
1468 }
1469
1470 }
1471
1472 }
1473
1474 }
1475
1476 }
1477
1478 }
1479
1480 }
1481
1482 }
1483
1484 }
1485
1486 }
1487
1488 }
1489
1490 }
1491
1492 }
1493
1494 }
1495
1496 }
1497
1498 }
1499
1500 }
1501
1502 }
1503
1504 }
1505
1506 }
1507
1508 }
1509
1510 }
1511
1512 }
1513
1514 }
1515
1516 }
1517
1518 }
1519
1520 }
1521
1522 }
1523
1524 }
1525
1526 }
1527
1528 }
1529
1530 }
1531
1532 }
1533
1534 }
1535
1536 }
1537
1538 }
1539
1540 }
1541
1542 }
1543
1544 }
1545
1546 }
1547
1548 }
1549
1550 }
1551
1552 }
1553
1554 }
1555
1556 }
1557
1558 }
1559
1560 }
1561
1562 }
1563
1564 }
1565
1566 }
1567
1568 }
1569
1570 }
1571
1572 }
1573
1574 }
1575
1576 }
1577
1578 }
1579
1580 }
1581
1582 }
1583
1584 }
1585
1586 }
1587
1588 }
1589
1590 }
1591
1592 }
1593
1594 }
1595
1596 }
1597
1598 }
1599
1600 }
1601
1602 }
1603
1604 }
1605
1606 }
1607
1608 }
1609
1610 }
1611
1612 }
1613
1614 }
1615
1616 }
1617
1618 }
1619
1620 }
1621
1622 }
1623
1624 }
1625
1626 }
1627
1628 }
1629
1630 }
1631
1632 }
1633
1634 }
1635
1636 }
1637
1638 }
1639
1640 }
1641
1642 }
1643
1644 }
1645
1646 }
1647
1648 }
1649
1650 }
1651
1652 }
1653
1654 }
1655
1656 }
1657
1658 }
1659
1660 }
1661
1662 }
1663
1664 }
1665
1666 }
1667
1668 }
1669
1670 }
1671
1672 }
1673
1674 }
1675
1676 }
1677
1678 }
1679
1680 }
1681
1682 }
1683
1684 }
1685
1686 }
1687
1688 }
1689
1690 }
1691
1692 }
1693
1694 }
1695
1696 }
1697
1698 }
1699
1700 }
1701
1702 }
1703
1704 }
1705
1706 }
1707
1708 }
1709
1710 }
1711
1712 }
1713
1714 }
1715
1716 }
1717
1718 }
1719
1720 }
1721
1722 }
1723
1724 }
1725
1726 }
1727
1728 }
1729
1730 }
1731
1732 }
1733
1734 }
1735
1736 }
1737
1738 }
1739
1740 }
1741
1742 }
1743
1744 }
1745
1746 }
1747
1748 }
1749
1750 }
1751
1752 }
1753
1754 }
1755
1756 }
1757
1758 }
1759
1760 }
1761
1762 }
1763
1764 }
1765
1766 }
1767
1768 }
1769
1770 }
1771
1772 }
1773
1774 }
1775
1776 }
1777
1778 }
1779
1780 }
1781
1782 }
1783
1784 }
1785
1786 }
1787
1788 }
1789
1790 }
1791
1792 }
1793
1794 }
1795
1796 }
1797
1798 }
1799
1800 }
1801
1802 }
1803
1804 }
1805
1806 }
1807
1808 }
1809
1810 }
1811
1812 }
1813
1814 }
1815
1816 }
1817
1818 }
1819
1820 }
1821
1822 }
1823
1824 }
1825
1826 }
1827
1828 }
1829
1830 }
1831
1832 }
1833
1834 }
1835
1836 }
1837
1838 }
1839
1840 }
1841
184
```

</STYLE>

Basic Theme

Demo

```
console.log(layerView.getLayer(index));
view.on('layer', function(layer) {
    console.log(layerView)
    // If you interact with the layerview, you'll get an error here
    console.log(layerView);
});
```

Override CSS

- style.ts
 - Use the Emotion library for CSS-in-JS
- style.scss
 - More traditional way
 - Imported last

```
let view = new ScrollView({
  container: "viewport",
  map: map,
  style: {
    width: 1000,
    height: 1000,
    backgroundColor: "white"
  }
});
```

[@ {

</STYLE>

Override CSS

Demo

```
const layerView = new LayerView({
  map: map,
  style: {
    width: 1000,
    height: 1000,
    backgroundColor: "white"
  }
});
```


Custom Fonts

- Use the “Override CSS” method:

```
@import url('https://fonts.googleapis.com/css?family=Open+Sans');
```

```
let view = new ScrollView({
  container: "viewport",
  map: map,
  style: {
    width: 100%,
    height: 100%,
    overflow: "hidden"
  }
});
```

[@ {

</STYLE>

Custom Fonts

Demo

```
const layerView = map.layers.getLayer(index);
const layer = layerView.getLayer();
const style = layer.getStyle();
// If you're using the layerView, you'll get an error here
const style = layer.getStyle();
```

Walk through

Custom Theme

```

const layer = viewmodel.flavorView()
view.viewModel.flavorView()
.then(layerView => console.log(
  'if there were problems with
  this promise, it would

```

Activity

30 Minutes

- Add some custom styles to your widget from earlier today.
- Create a custom theme, using your organization's brand and colors.
 - If you don't have one, use the Olympics 2024 colors/theme:
<https://www.paris2024.org/en/design>
- Bonus:
 - Also include the Paris2024 font (*is this possible?*)
- Documentation:
<https://developers.arcgis.com/experience-builder/guide/theme-development/>



esri®

THE
SCIENCE
OF
WHERE®

Copyright © 2023 Esri. All rights reserved.

</SCRIPT>

LIVE
BY
THE
CODE }