Working with Feature Layers, Dynamic Map Services, and OGC in the ArcGIS API for JavaScript

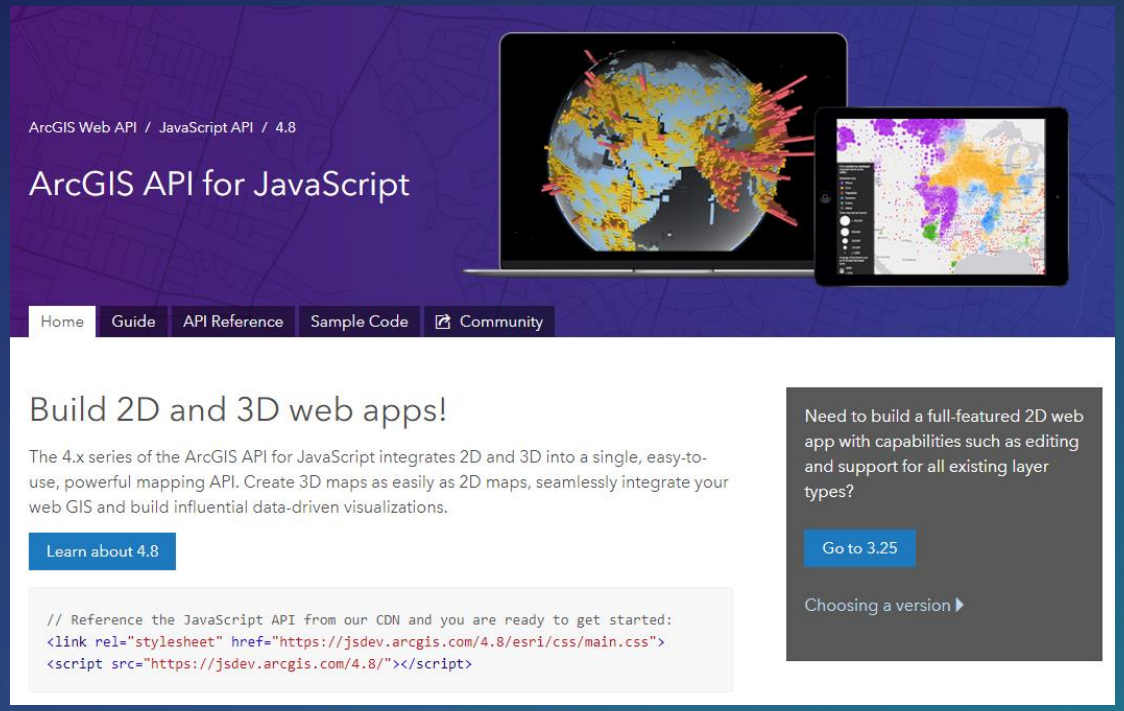
Yann Cabon

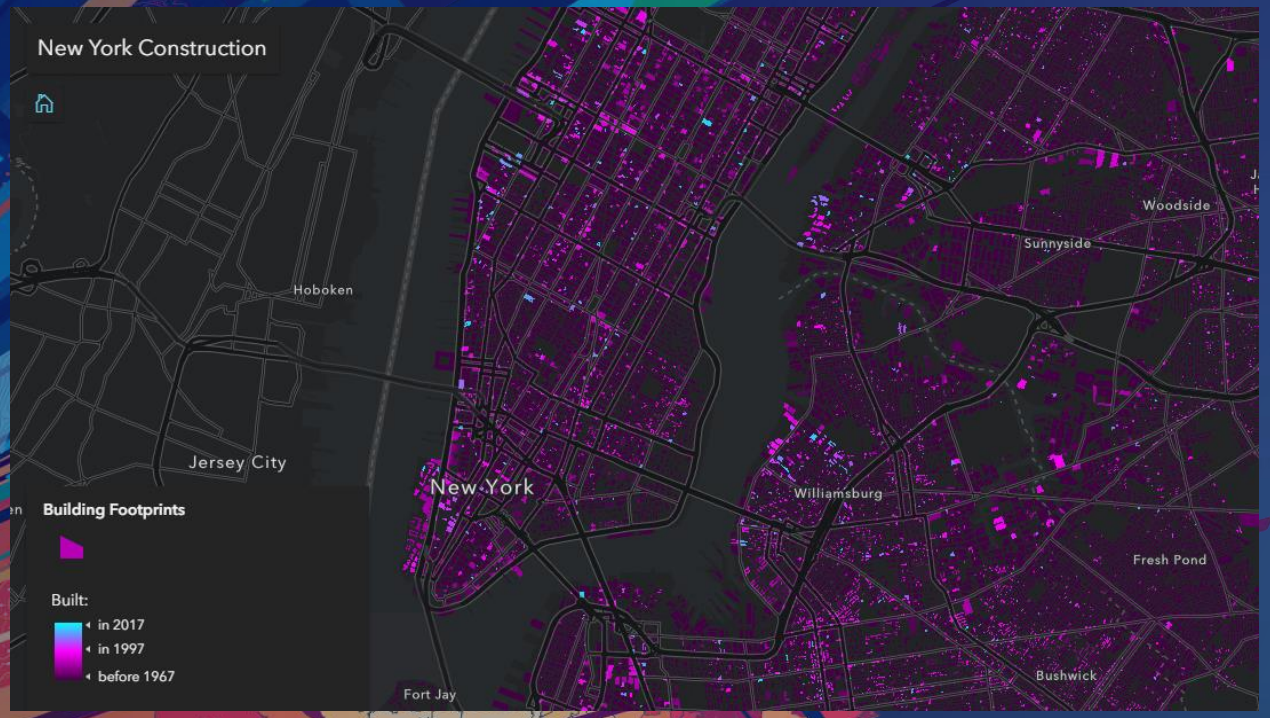
Noah Sager

Agenda

- Version 4.x
  - Feature Layers
  - Map Image Layers
  - OGC Layers

Why / How / Wow





FeatureLayer

Main Characteristics

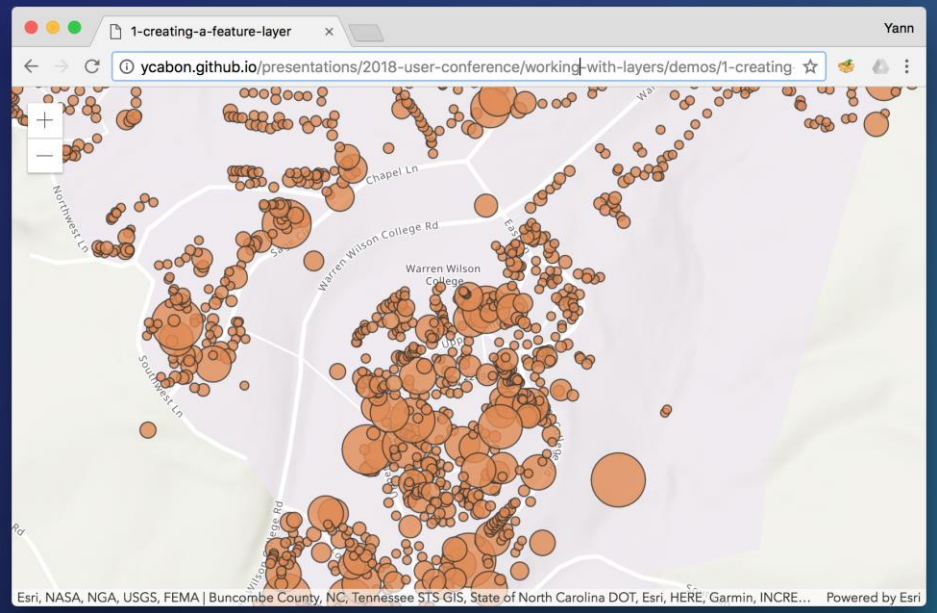
- Layer that renders features in a vector format.
- Direct access to features
  - From a feature service
  - Or created by the application
- Can be queried spatially, by attributes, and for statistics
- The application draws the features on the screen
  - Can change dynamically the rendering
  - Can add labels to features
- Features can be added, removed, or modified

Getting started

Getting started with a WebMap and MapView

```
import WebMap = require("esri/WebMap");
import MapView = require("esri/views/MapView");

const map = new WebMap({
  basemap: "topo-vector"
});

const view = new MapView({
  container: "viewDiv",
  map: map
});
```

Add a FeatureLayer



Add a FeatureLayer

Import the FeatureLayer module

```
import WebMap = require("esri/WebMap");
import FeatureLayer = require("esri/layers/FeatureLayer");
import MapView = require("esri/views/MapView");

const map = new WebMap({
  basemap: "topo-vector"
});

const view = new MapView({
  container: "viewDiv",
  map: map
});

const layer = new FeatureLayer({
  url: "https://services.arcgis.com/arcgis/rest/services/Trees/FeatureServer/0"
});

map.add(layer);
```

Add a FeatureLayer

Create a FeatureLayer by providing the URL to the service

```
import WebMap = require("esri/WebMap");
import FeatureLayer = require("esri/layers/FeatureLayer");
import MapView = require("esri/views/MapView");

const map = new WebMap({
  basemap: "topo-vector"
});

const view = new MapView({
  container: "viewDiv",
  map: map
});

const layer = new FeatureLayer({
  url: "https://services.arcgis.com/arcgis/rest/services/Trees/FeatureServer/0"
});

map.add(layer);
```

Add a FeatureLayer

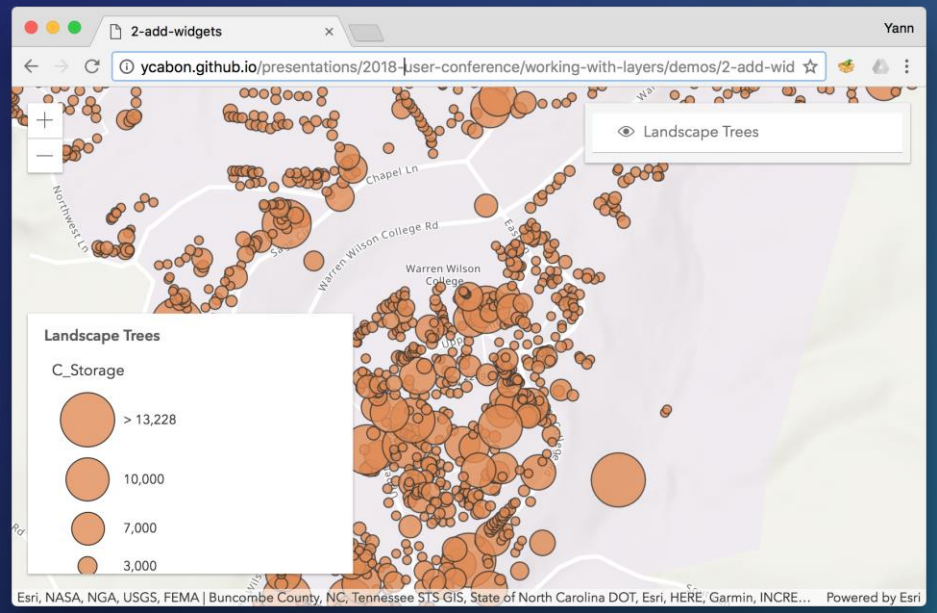
Add the layer to the map

```
import WebMap = require("esri/WebMap");
import FeatureLayer = require("esri/layers/FeatureLayer");
import MapView = require("esri/views/MapView");

const map = new WebMap({
  basemap: "topo-vector"
});

const view = new MapView({
  container: "viewDiv",
  map: map
});

const layer = new FeatureLayer({
  url: "https://services.arcgis.com/arcgis/rest/services/Trees/FeatureServer/0"
});

map.add(layer);
```

Add a FeatureLayer

```
import WebMap = require("esri/WebMap");
import FeatureLayer = require("esri/layers/FeatureLayer");
import MapView = require("esri/views/MapView");

const map = new WebMap({
  basemap: "topo-vector"
});

const view = new MapView({
  container: "viewDiv",
  map: map
});

const layer = new FeatureLayer({
  url: "https://services.arcgis.com/arcgis/rest/services/Trees/FeatureServer/0"
});

map.add(layer);
```

Add Widgets


2-add-widgets
Yann
ycabon.github.io/presentations/2018-user-conference/working-with-layers/demos/2-add-wid
Landscape Trees
Chapel Ln
Northwest Ln
Warren Wilson College Rd
Warren Wilson College
Landscape Trees
C_Storage
> 13,228
10,000
7,000
3,000
Esri, NASA, NGA, USGS, FEMA | Buncombe County, NC, Tennessee STS GIS, State of North Carolina DOT, Esri, HERE, Garmin, INCRE...
Powered by Esri

Add Widgets

Import Legend and LayerList modules

```
import Legend = require("esri/widgets/Legend");
import LayerList = require("esri/widgets/LayerList");

const legend = new Legend({
  view: view
});

const layerList = new LayerList({
  view: view
});

view.ui.add(legend, "bottom-left");
view.ui.add(layerList, "top-right");
```

Add Widgets

Create the widgets

```
import Legend = require("esri/widgets/Legend");
import LayerList = require("esri/widgets/LayerList");

const legend = new Legend({
  view: view
});

const layerList = new LayerList({
  view: view
});

view.ui.add(legend, "bottom-left");
view.ui.add(layerList, "top-right");
```

Add Widgets

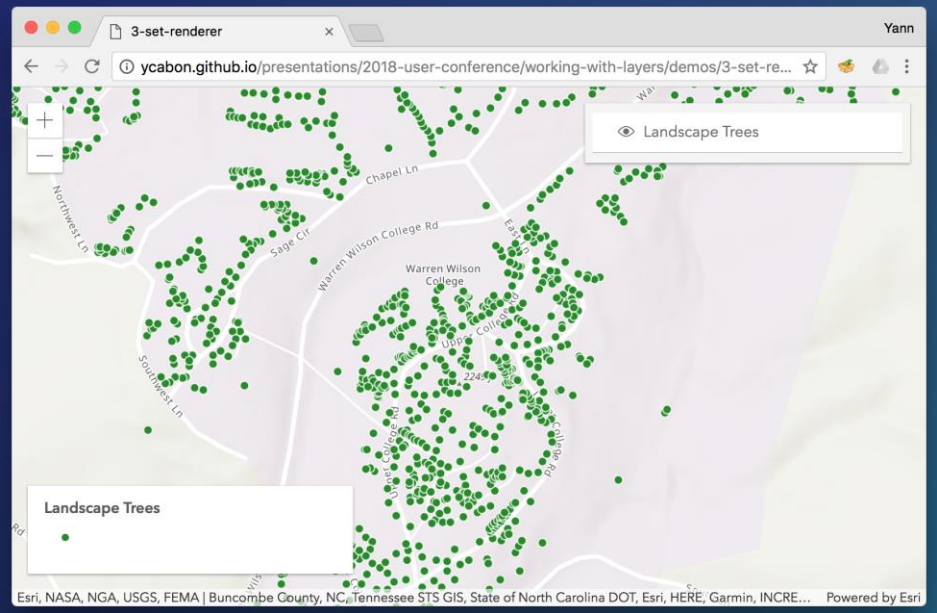
Add them to the view UI

```
import Legend = require("esri/widgets/Legend");
import LayerList = require("esri/widgets/LayerList");

const legend = new Legend({
  view: view
});

const layerList = new LayerList({
  view: view
});

view.ui.add(legend, "bottom-left");
view.ui.add(layerList, "top-right");
```

Add Widgets

```
import Legend = require("esri/widgets/Legend");
import LayerList = require("esri/widgets/LayerList");

const legend = new Legend({
  view: view
});

const layerList = new LayerList({
  view: view
});

view.ui.add(legend, "bottom-left");
view.ui.add(layerList, "top-right");
```

Change the Rendering



Change the Rendering

Import the SimpleRenderer and SimpleMarkerSymbol

```
import { SimpleRenderer } from "esri/renderers";
import { SimpleMarkerSymbol } from "esri/symbols";


const renderer = new SimpleRenderer({
  symbol: new SimpleMarkerSymbol({
    style: "circle",
    size: 6,
    color: "forestgreen",
    outline: {
      color: "white",
      width: 1
    }
  })
});

layer.renderer = renderer;
```

Change the Rendering

Create the renderer

```
import { SimpleRenderer } from "esri/renderers";
import { SimpleMarkerSymbol } from "esri/symbols";


const renderer = new SimpleRenderer({
  symbol: new SimpleMarkerSymbol({
    style: "circle",
    size: 6,
    color: "forestgreen",
    outline: {
      color: "white",
      width: 1
    }
  })
});

layer.renderer = renderer;
```

Change the Rendering

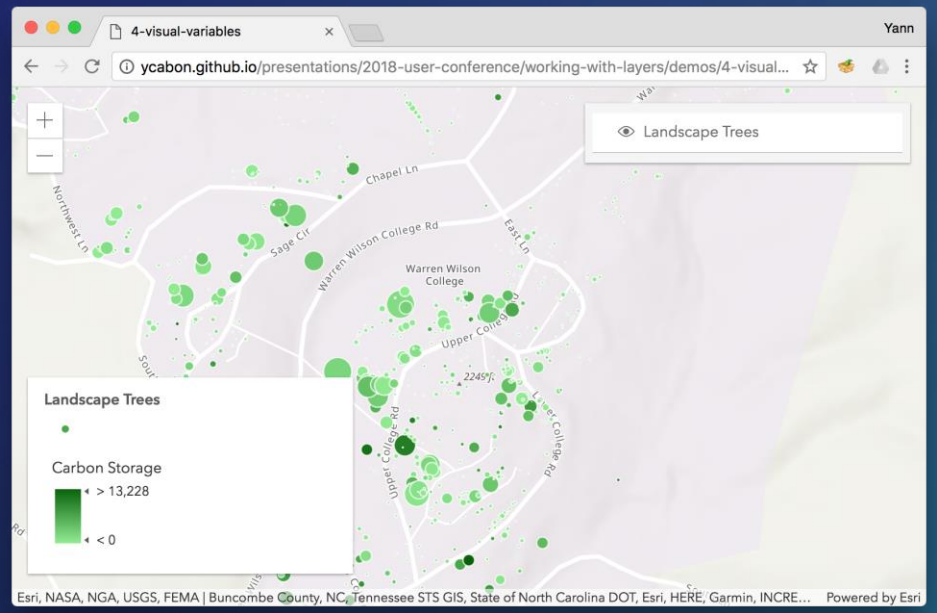
Assign the renderer

```
import { SimpleRenderer } from "esri/renderers";
import { SimpleMarkerSymbol } from "esri/symbols";


const renderer = new SimpleRenderer({
  symbol: new SimpleMarkerSymbol({
    style: "circle",
    size: 6,
    color: "forestgreen",
    outline: {
      color: "white",
      width: 1
    }
  })
});

layer.renderer = renderer;
```

Change the Rendering

```
import { SimpleRenderer } from "esri/renderers";
import { SimpleMarkerSymbol } from "esri/symbols";


const renderer = new SimpleRenderer({
  symbol: new SimpleMarkerSymbol({
    style: "circle",
    size: 6,
    color: "forestgreen",
    outline: {
      color: "white",
      width: 1
    }
  })
});

layer.renderer = renderer;
```

Data-Driven Rendering



Data-Driven Rendering

Define a size visual variable

```
renderer.visualVariables = [
  {
    // Size the tree symbol based on the Crown_Base attribute
    type: "size",
    field: "Crown_Base",
    valueUnit: ”feet",
    valueRepresentation: ”radius"
  },
  {
    // Color the tree symbol continously based on its carbon storage
    type: "color",
    field: "C_Storage",
    // values from statistics
    stops: [
      { value: 0, color: "lightgreen" },
      { value: 13228, color: "darkgreen" }
    ],
    legendOptions: { title: "Carbon Storage" }
  }
]
```

Data-Driven Rendering

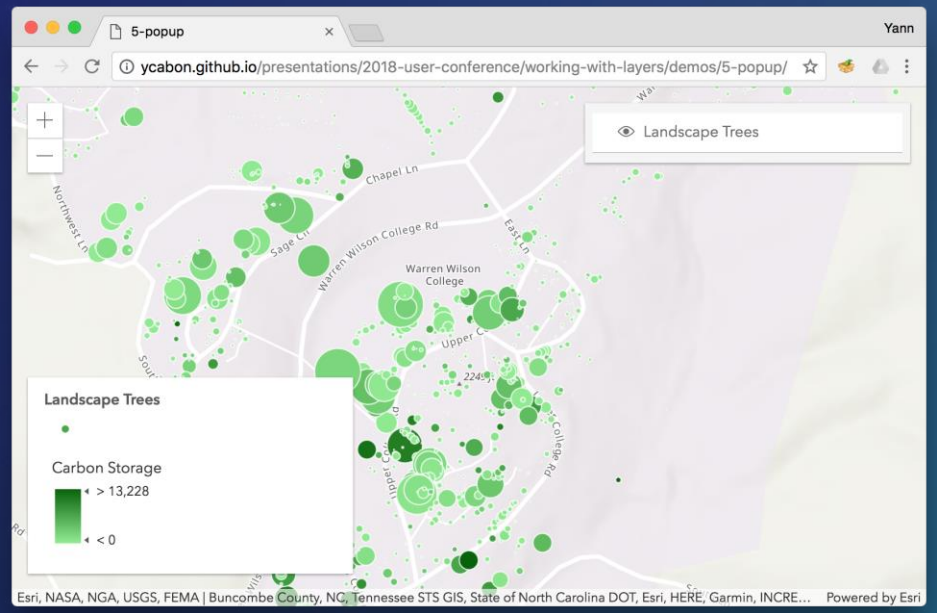
Define a color visual variable

```
renderer.visualVariables = [
  {
    // Size the tree symbol based on the Crown_Base attribute
    type: "size",
    field: "Crown_Base",
    valueUnit: ”feet",
    valueRepresentation: ”radius"
  },
  {
    // Color the tree symbol continously based on its carbon storage
    type: "color",
    field: "C_Storage",
    // values from statistics
    stops: [
      { value: 0, color: "lightgreen" },
      { value: 13228, color: "darkgreen" }
    ],
    legendOptions: { title: "Carbon Storage" }
  }
]
```

Data-Driven Rendering

```
renderer.visualVariables = [
  {
    // Size the tree symbol based on the Crown_Base attribute
    type: "size",
    field: "Crown_Base",
    valueUnit: ”feet",
    valueRepresentation: ”radius"
  },
  {
    // Color the tree symbol continously based on its carbon storage
    type: "color",
    field: "C_Storage",
    // values from statistics
    stops: [
      { value: 0, color: "lightgreen" },
      { value: 13228, color: "darkgreen" }
    ],
    legendOptions: { title: "Carbon Storage" }
  }
]
```

Add Popup


5-popup
Yann
ycabon.github.io/presentations/2018-user-conference/working-with-layers/demos/5-popup/
Landscape Trees
Chapel Ln
Warren Wilson College Rd
Warren Wilson College
Northwest Ln
Sage Ln
East Ln
Upper Ln
Landscape Trees
Carbon Storage
> 13,228
< 0
Esri, NASA, NGA, USGS, FEMA | Buncombe County, NC, Tennessee STS GIS, State of North Carolina DOT, Esri, HERE, Garmin, INCRE...
Powered by Esri

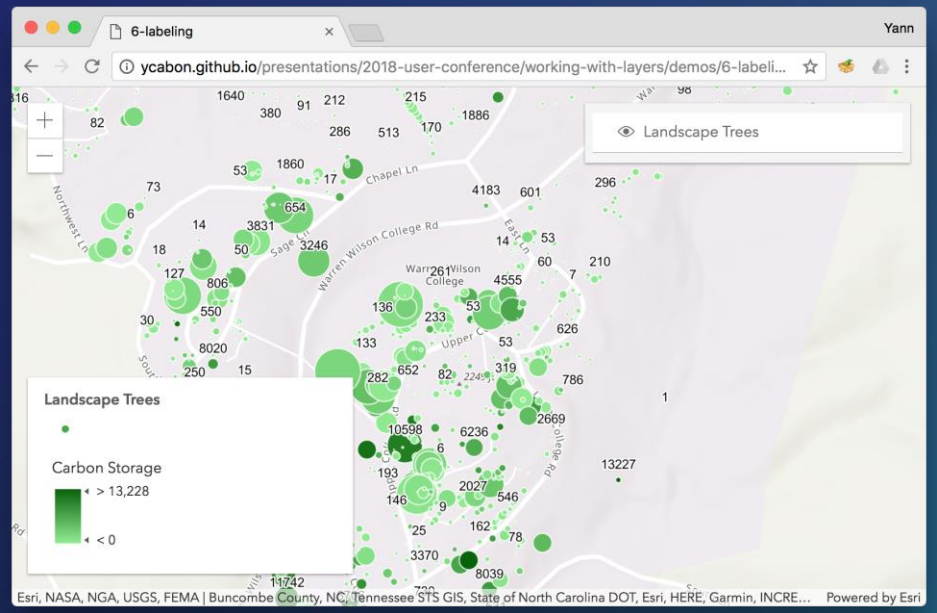
Add Popup

Add a PopupTemplate

```
import PopupTemplate = require("esri/PopupTemplate");


layer.popupTemplate = new PopupTemplate({
  title: `{Sci_Name}`,
  content: `<b>Carbon Storage:</b> {C_Storage}`
});
```

- **Multiple content options available**
  - **Formatted content**
  - **List of all the layer's fields**
  - **List of some fields**
- **Attachments**
- **Arcade Expression**

Add Labels


6-labeling
Yann
ycabon.github.io/presentations/2018-user-conference/working-with-layers/demos/6-labeli...
Landscape Trees
Landscape Trees
Carbon Storage
> 13,228
< 0
Esri, NASA, NGA, USGS, FEMA | Buncombe County, NC, Tennessee STS GIS, State of North Carolina DOT, Esri, HERE, Garmin, INCRE...
Powered by Esri

Add Labels

Import LabelClass and the TextSymbol

```
import LabelClass = require("esri/layers/support/LabelClass");
import { SimpleMarkerSymbol, TextSymbol } from "esri/symbols";


layer.labelingInfo = [
  new LabelClass({
    labelExpression: "{C_Storage}",
    labelPlacement: "above-center",
    symbol: new TextSymbol({
      color: "black",
      haloColor: "white",
      haloSize: 1
    }),
    where: "C_Storage > 0"
  })
];
```

Add Labels

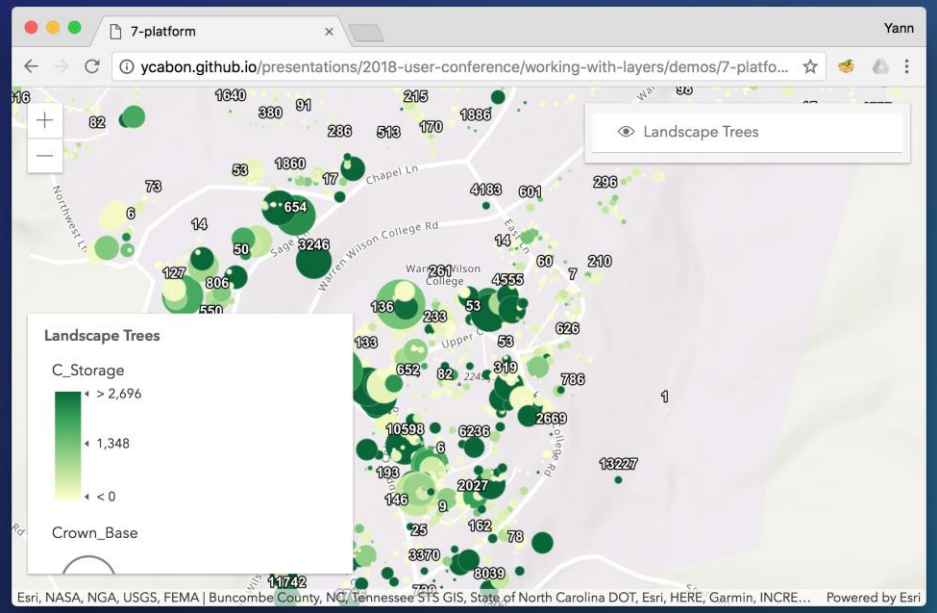
Define LabelClasses

```
import LabelClass = require("esri/layers/support/LabelClass");
import { SimpleMarkerSymbol, TextSymbol } from "esri/symbols";


layer.labelingInfo = [
  new LabelClass({
    labelExpression: "{C_Storage}",
    labelPlacement: "above-center",
    symbol: new TextSymbol({
      color: "black",
      haloColor: "white",
      haloSize: 1
    }),
    where: "C_Storage > 0"
  })
];
```

Add Labels

```
import LabelClass = require("esri/layers/support/LabelClass");
import { SimpleMarkerSymbol, TextSymbol } from "esri/symbols";


layer.labelingInfo = [
  new LabelClass({
    labelExpression: "{C_Storage}",
    labelPlacement: "above-center",
    symbol: new TextSymbol({
      color: "black",
      haloColor: "white",
      haloSize: 1
    }),
    where: "C_Storage > 0"
  })
];
```

Platform Integration



Platform Integration

Scrap everything we just did

```
const map = new WebMap({
  basemap: "topo-vector"
});

Layer.fromPortalItem({
  portalItem: new PortalItem({
    id: "ae449d81e9da47dbb1969afc3021bdf7"
  })
}).then(layer => {
  map.add(layer);
});
```

- **Configure the layer on the MapViewer**
- **Load the layer from the item id**
- ✨

Working with the Features

- Querying
  - queryFeatures()
  - queryObjectIds()
  - queryExtent()
  - queryFeatureCount()
- highlight
- Client-side Querying: New at 4.7 and enhanced in 4.8
  - Ultra fast to work with features available in memory
- Pushing edits to the service

MapImageLayer

Demos:
https://noashx.github.io/presentations/UC_2018/MapImageLayer/index.html

Why use MapImageLayer?

- Flexible
  - Created from: URL, Portal Item
- Powerful
  - Sublayer: labeling, popups, renderers, workspaces
- Server-side
  - Fast, light, and able to display lots of features

Add a MapImageLayer

```
require(["esri/layers/MapImageLayer"], function(MapImageLayer){

  // references an ArcGIS Enterprise server Map Service
  var layer = new MapImageLayer({
    url: "https://sampleserver6.arcgisonline.com/arcgis/rest/services/USA/MapServer"
  });

  map.add(layer);  // adds the layer to the map
});
```

```
require(["esri/layers/MapImageLayer"], function(MapImageLayer){

  // references a PortalItem from ArcGIS Online
  var layer = new MapImageLayer({
    portalItem: { // autocasts as new PortalItem()
      id: "08b59e3aa73e4200acb610c724449cda"
    }
  });

  map.add(layer);  // adds the layer to the map
});
```

Renderers

```
var layer = new MapImageLayer({
  portalItem: { // autocasts as new PortalItem()
    id: "e7e03ad8f72b42709e7d63dde8c6c3f5"
  },
  sublayers: [
  {
    id: 2,
    renderer: statesRenderer
  },
  {
    id: 0,
    renderer: citiesRenderer
  }]
});
```

```
  },
  label: "State boundaries"
};
```

Popups

```
var layer = new MapImageLayer({
  portalItem: { // autocasts as new PortalItem()
    id: "e7e03ad8f72b42709e7d63dde8c6c3f5"
  },
  sublayers: [
  {
    id: 2,
    visible: true,
    renderer: statesRenderer,
    popupTemplate: {
      title: "{state_name}",
      content: "{pop2000} people live in {state_abbr}"
    }
  },
  {
    id: 1,
    visible: true,
    renderer: highwaysRenderer,
  }
});
```

Labeling

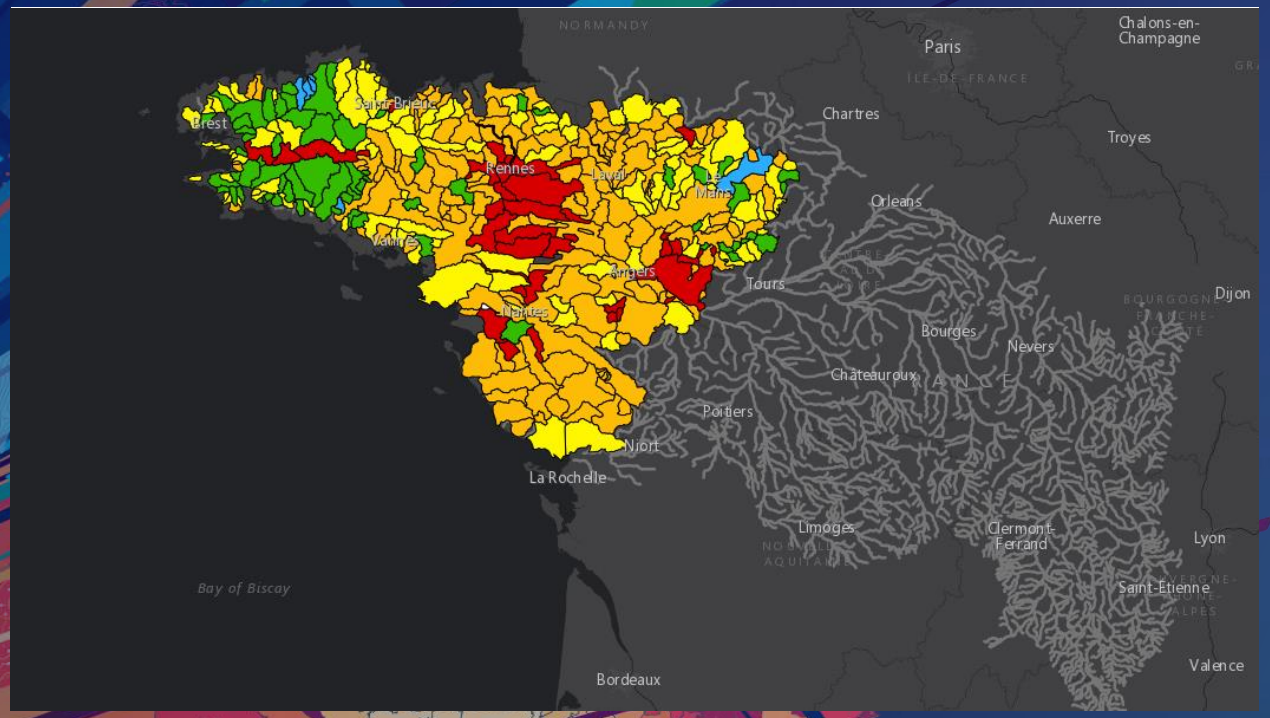
```
var layer = new MapImageLayer({
  portalItem: { // autocasts as new PortalItem()
      id: "e7e03ad8f72b42709e7d63dde8c6c3f5"
    },
  sublayers: [{
    id: 3,
    // labels are visible by default
    // labelingInfo autocasts to an array of LabelClass objects
    labelingInfo: [{
      labelExpression: "[name]",
      labelPlacement: "always-horizontal",
      symbol: {
        type: "text", // autocasts as new TextSymbol()
        color: [255, 255, 255, 0.7],
        haloColor: [0, 0, 0, 0.7],
        haloSize: 1,
        font: {
          size: 11
        }
      }
    }]
  }]
});
```

Query Table

```
// The base SQL statement used to query features.
// We'll select all rows from the Places feature class table
var queryString = "SELECT * FROM ss6.gdb.Places";

var layer = new MapImageLayer({
  url: "https://sampleserver6.arcgisonline.com/arcgis/rest/services/USA/MapServer",
  sublayers: [{
    title: "Places",
    renderer: renderer,
    id: 0,
    source: {
      // indicates the source of the sublayer is a dynamic data layer
      type: "data-layer",
      // this object defines the data source of the layer
      // in this case it's a table that will be queried
      // using a SQL WHERE clause
      dataSource: {
        type: "query-table",
        workspaceId: "MyDatabaseWorkspaceIDSSR2",
        query: queryString + document.getElementById(
          "layer-select").value,
        oidFields: "objectid"
      }
    }
  }]
});
```

Integration with the ArcGIS Platform

```
var usaLayer = new MapImageLayer({
  portalItem: { // autocasts as new PortalItem()
    id: "08b59e3aa73e4200acb610c724449cda"
  }
});


var map = new Map({
  basemap: "gray",
  layers: [usaLayer]
});


var view = new SceneView({
  container: "viewDiv",
  map: map
});
```


NORMANDY
Paris
Chalons-en-Champagne
ÎLE-DE-FRANCE
Chartres
Troyes
Brest
Saint-Brieuc
Rennes
Laval
Le Mans
Orleans
Auxerre
Vannes
Angers
Tours
Dijon
Nantes
Bourges
Nevers
Châteauroux
Poitiers
Niort
La Rochelle
Limoges
Clermont-Ferrand
Lyon
Bay of Biscay
Saint-Étienne
Valence
Bordeaux

OGC Layers

OGC Layers

- Open Geospatial Consortium
  - Defines a set of interoperable services specifications
  - JS API 4.x has an improve support and consistent API with ArcGIS layers
- Support in the API
  - WMS – Web Map Service
  - WMTS – Web Map Tile Service
  - WFS – Web Feature Service – available in 3.x only at the moment
  - KML – Keyhole Markup Language, some level of support
  - I3S - Indexed 3D Scene Layer

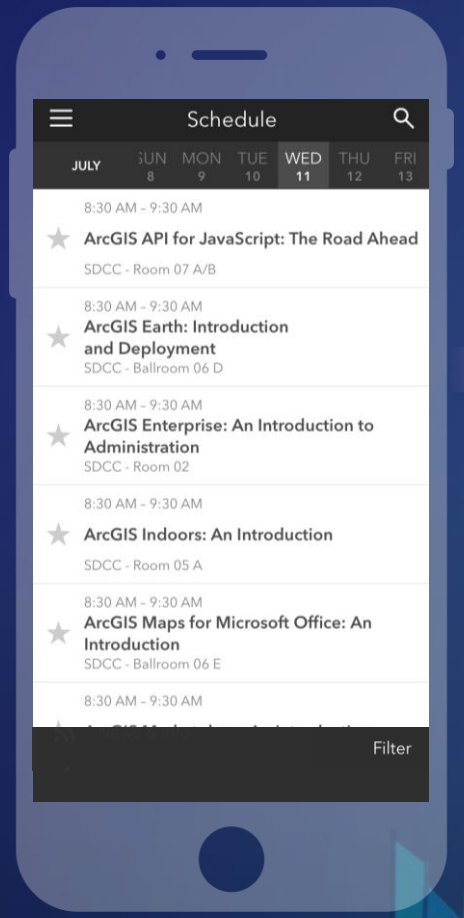
Conclusion

- Feature Layers, Map Image Layers, OGC Layers are powerful, useful, and elegant.
- There are even more layer types to explore with the 4.x version of the ArcGIS API for JavaScript.

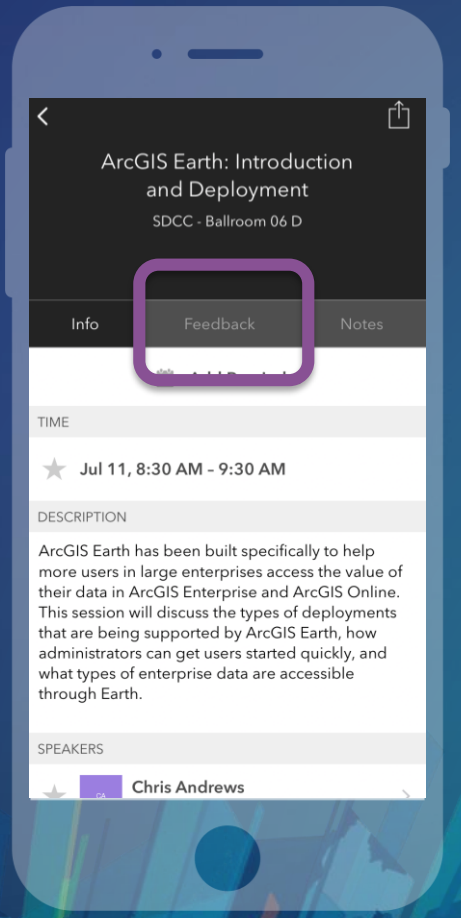
Please Take Our Survey on the App

Download the Esri Events app and find your event

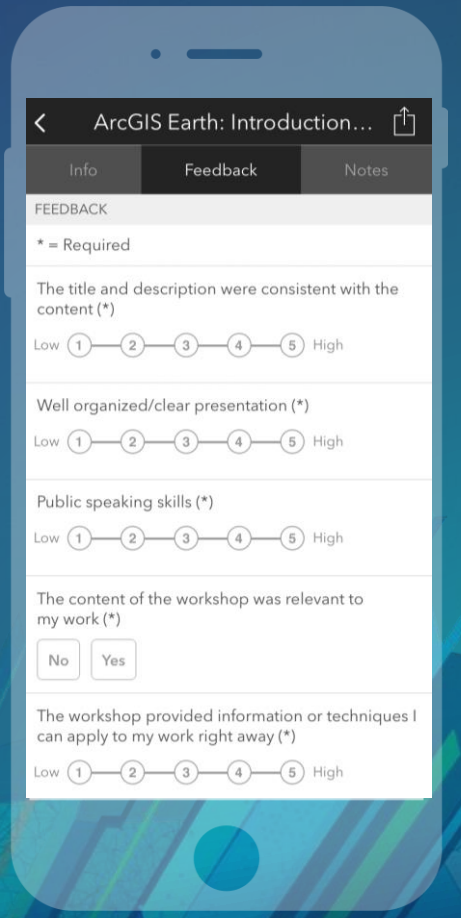
Select the session you attended



Scroll down to find the feedback section



Complete answers and select “Submit”



esri | THE SCIENCE OF WHERE