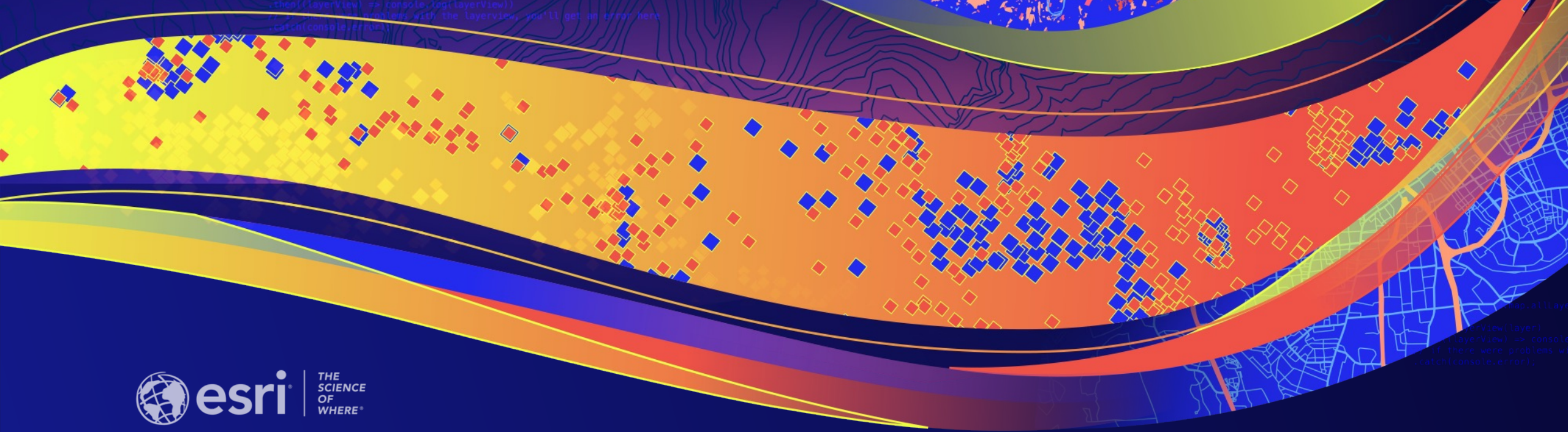


esri | THE SCIENCE OF WHERE

ESRI DEVELOPER SUMMIT 2023

ArcGIS Maps SDK for JavaScript: Dynamic Vector Symbology

Dario D'Amico & Anne Fitz

Session Overview

- What are CIM Symbols?
- Create a symbol from scratch
- CIM Symbol Builder
- Working with Web Styles
  - Publishing from ArcGIS Pro
  - Using in ArcGIS Online Map Viewer
- Data-driven CIM

What are CIM Symbols?

CIM Symbols

High quality, scalable

Scaled vector symbol

Scaled image

Symbol layers

Primitive Overrides

Dynamically update attributes of an individual symbol layer using Arcade

How is it used in the JavaScript SDK?

https://developers.arcgis.com/javascript/latest/api-reference/esri-symbols-CIMSymbol.html

```
// require(["esri/symbols/CIMSymbol"], function(CIMSymbol)
const cimSymbol = new CIMSymbol({
   data: {
      type: "CIMSymbolReference",
      symbol: {
         type: "CIMLineSymbol", // CIMPointSymbol or CIMPolygonSymbol
         symbolLayers: [{ ... }]
      },
      primitiveOverrides: [{ ... }]
   }
});
```

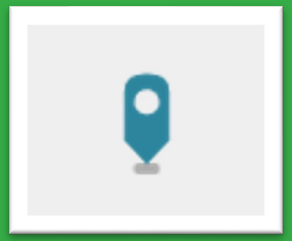
follows the cim-spec

CIM Point Symbol

CIMPictureMarker

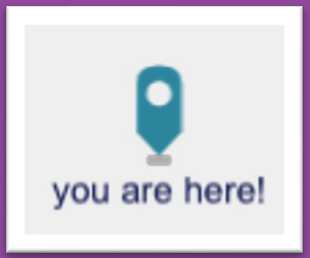
- Created from raster image file
- PNG / JPG / GIF
  - `animatedSymbolProperties` supported at v4.24

CIMVectorMarker



- Represents vector graphics
- Constructed from MarkerGraphics

CIMTextSymbol



- Supported as a marker graphic on a CIMVectorMarker symbol layer

CIM Line Symbol

CIMSolidStroke

CIMPictureStroke

- Geometric Effects
  - Dashes

  - Buffer

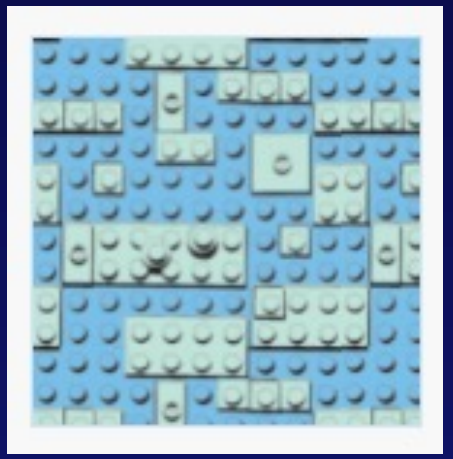
  - Arrows

- Marker placement

CIM Polygon Symbol

CIMSolidFill

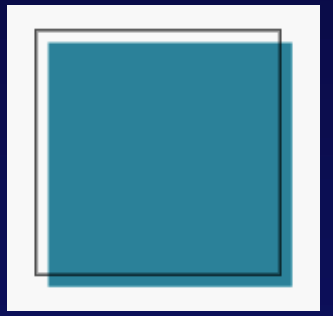
CIMPictureFill



CIMHatchFill

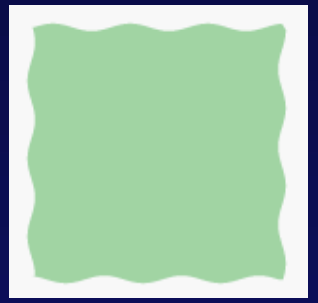
- **Geometric Effects**

Move

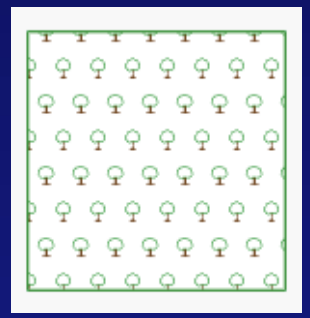


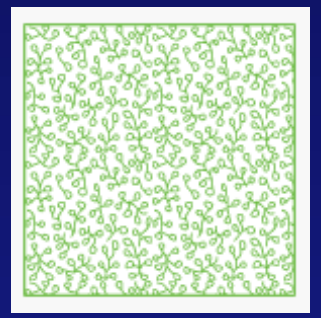
Offset

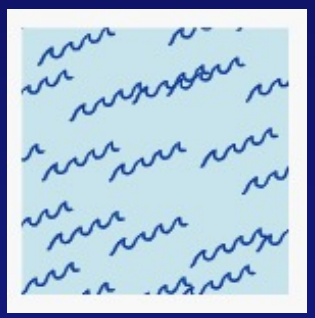
Wave



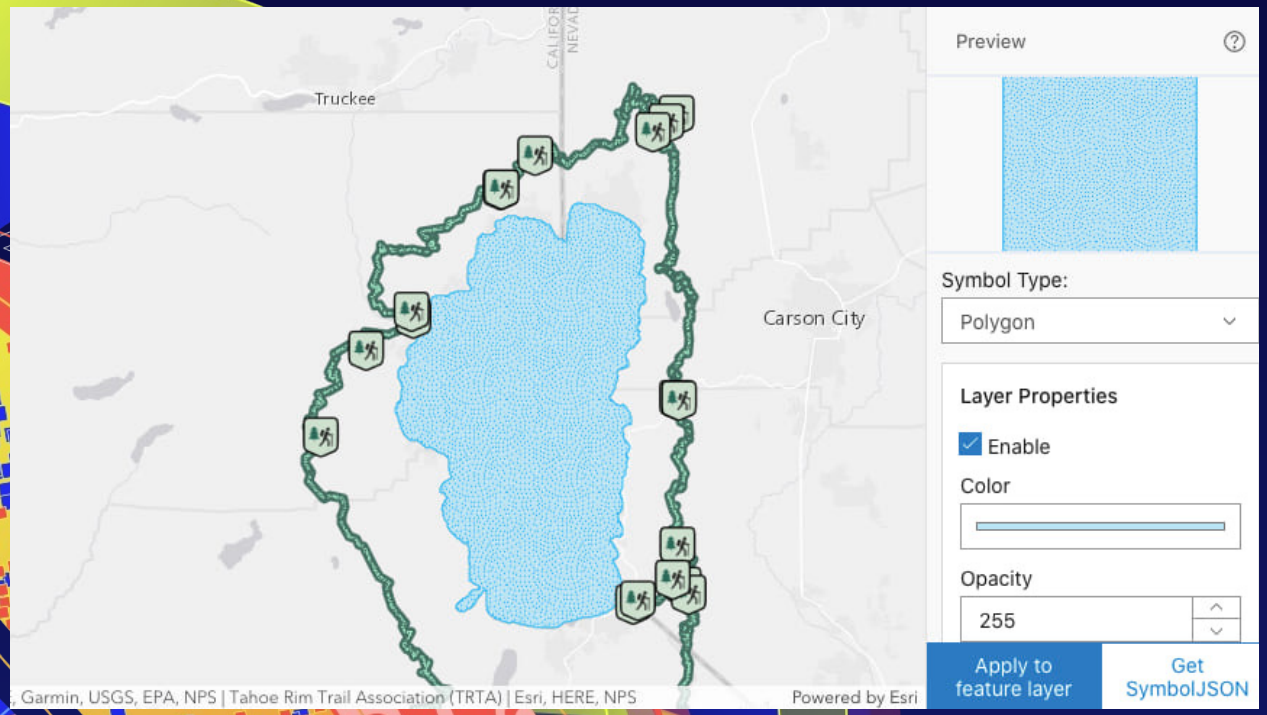
- **Marker placement**







Support for random marker placement new at v4.24!

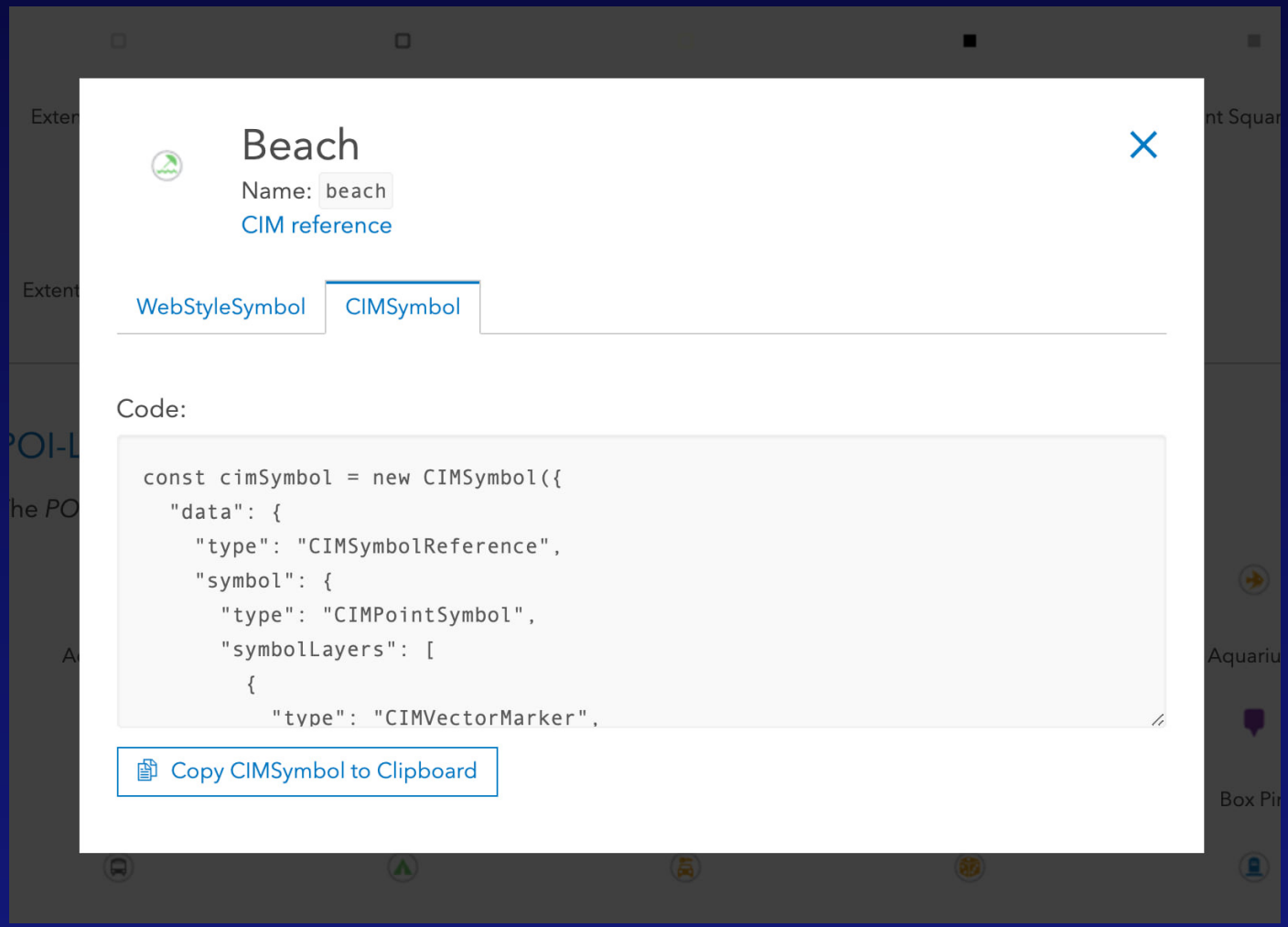


CIM Symbol Builder

Working with Web Styles

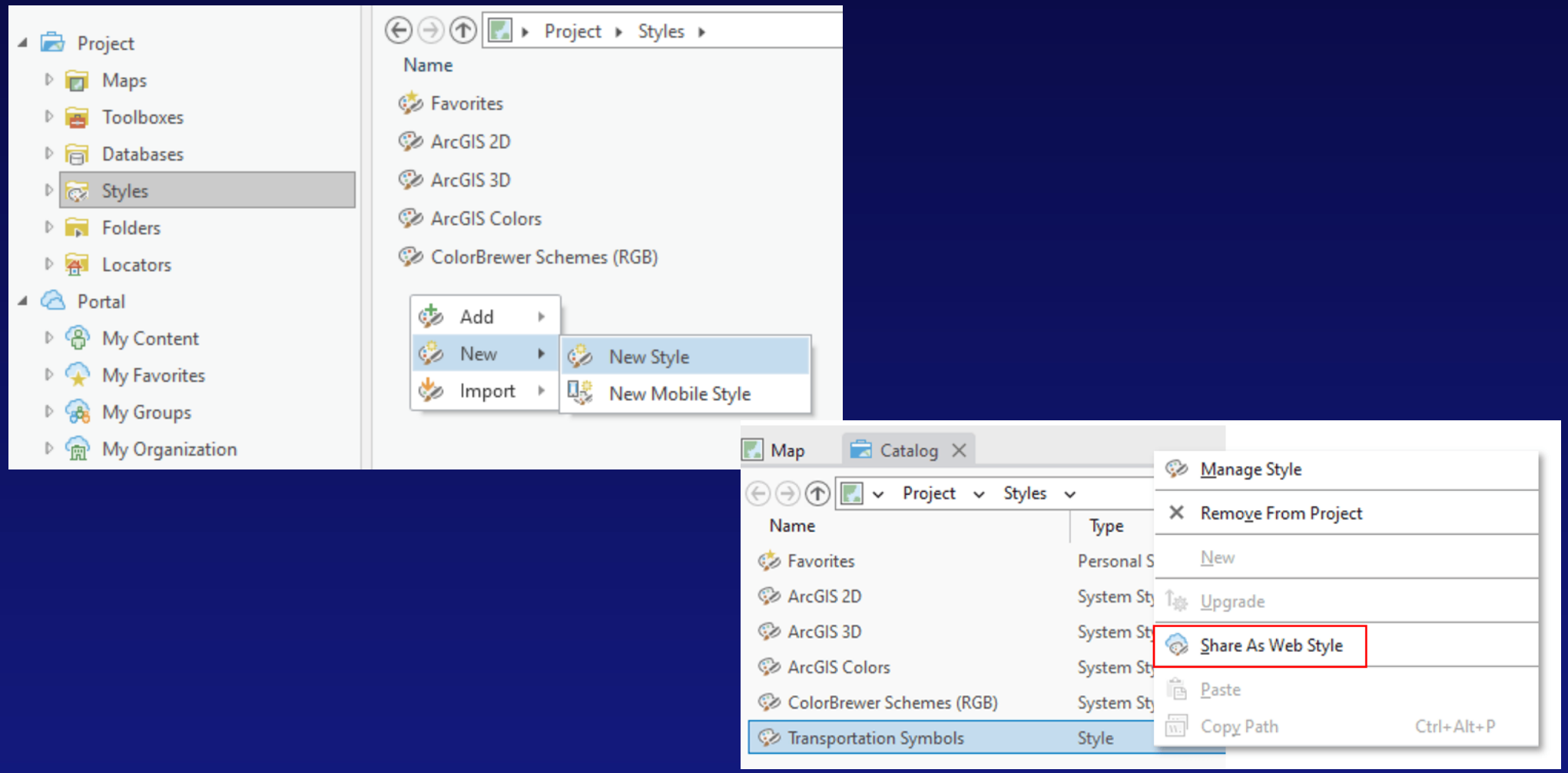
WebStyleSymbols

- Predefined 2D CIM Symbols that you can conveniently reference in your apps
- Esri WebStyleSymbol (2D) guide page
- Create your own web style symbols

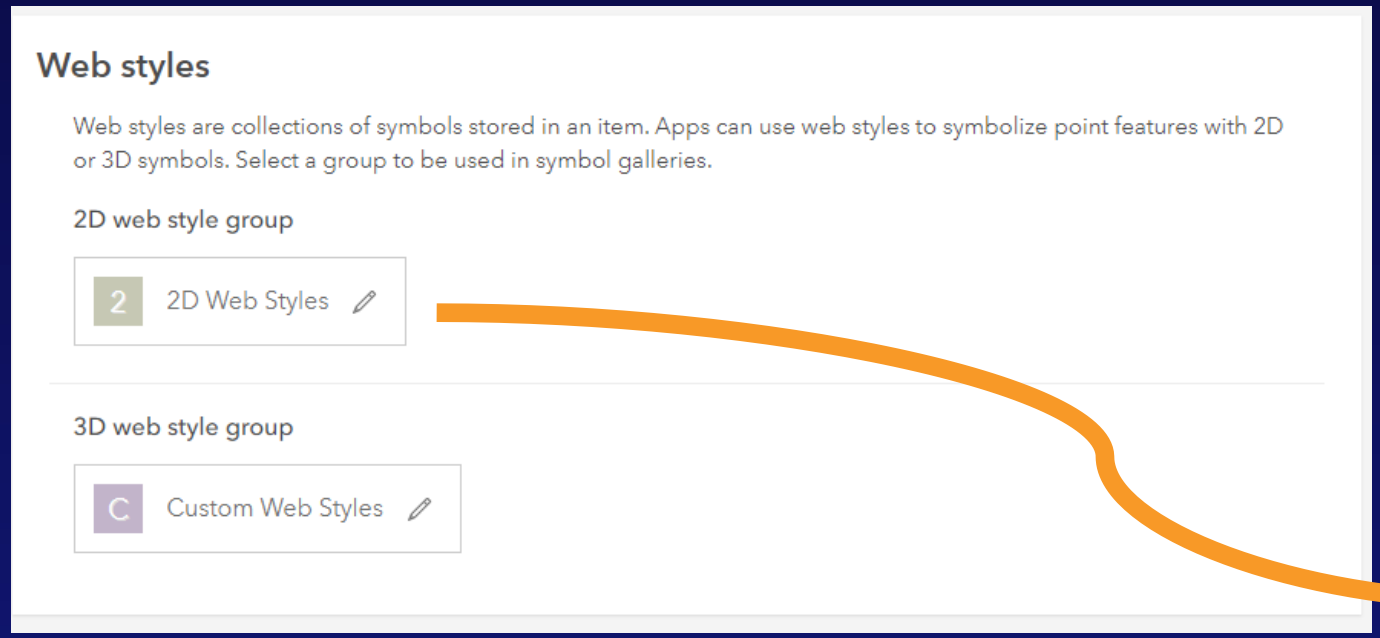


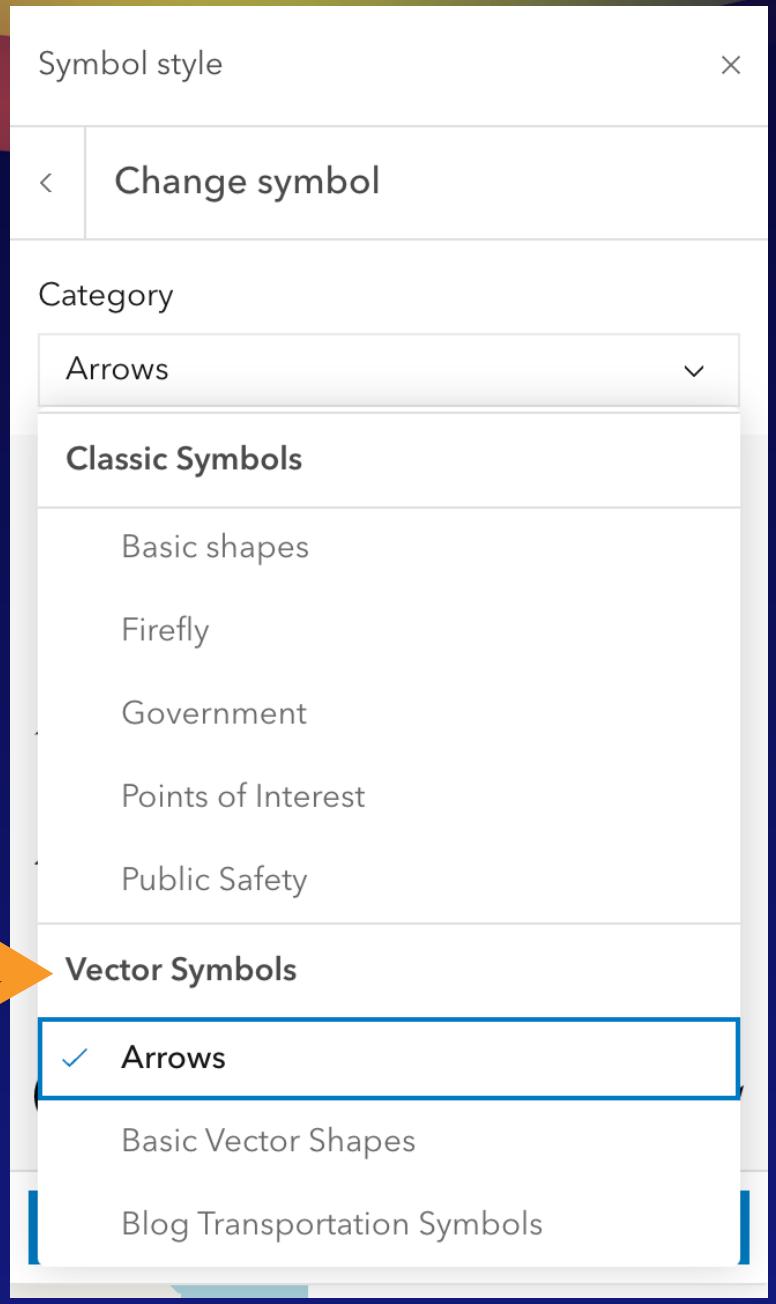
Creating & Publishing a Web Style

Blog: [Use Published 2D Symbols in ArcGIS Online]

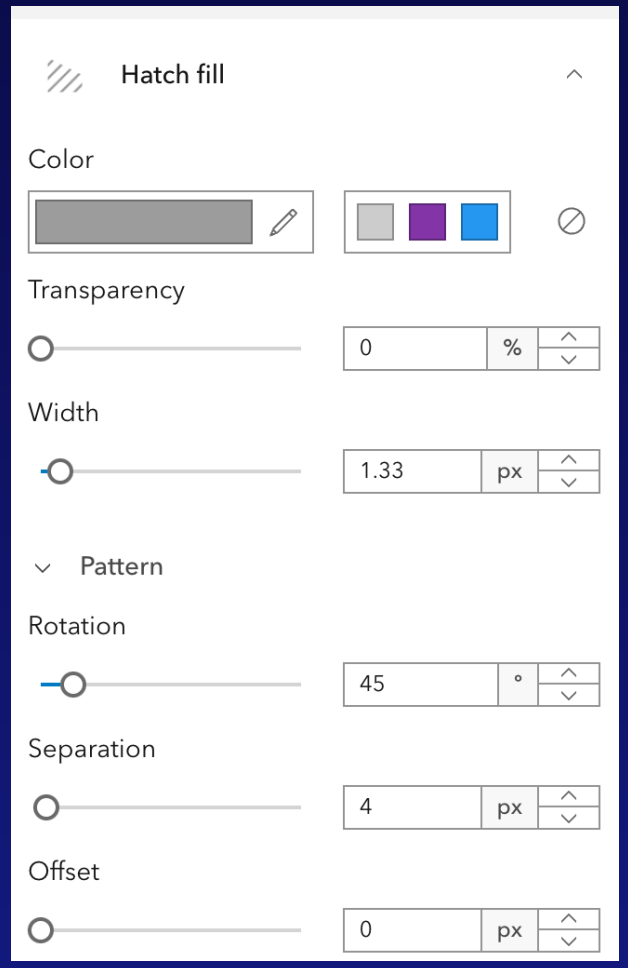


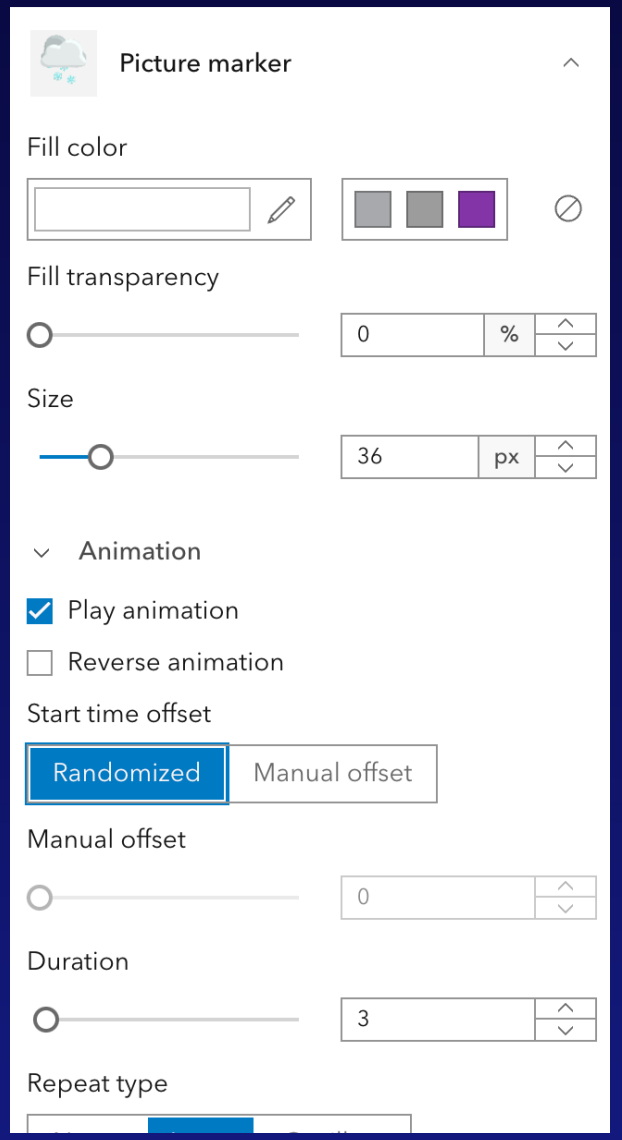
Using Web Styles in Map Viewer

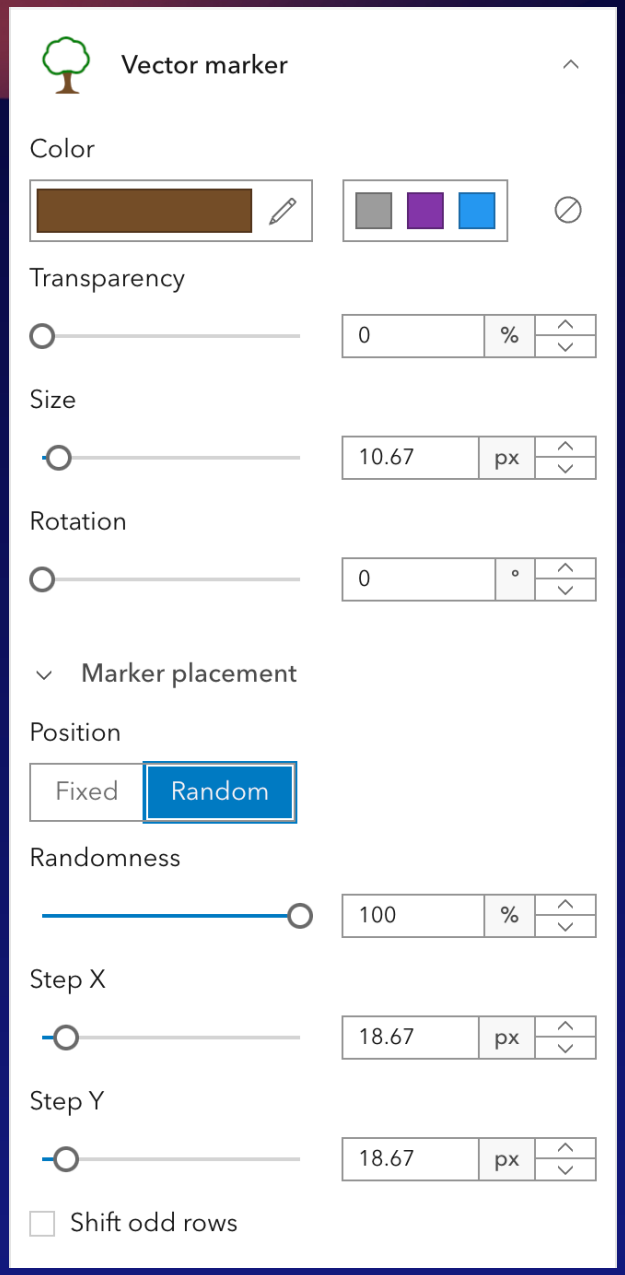




Adjust CIM properties from Map Viewer Symbol Styler





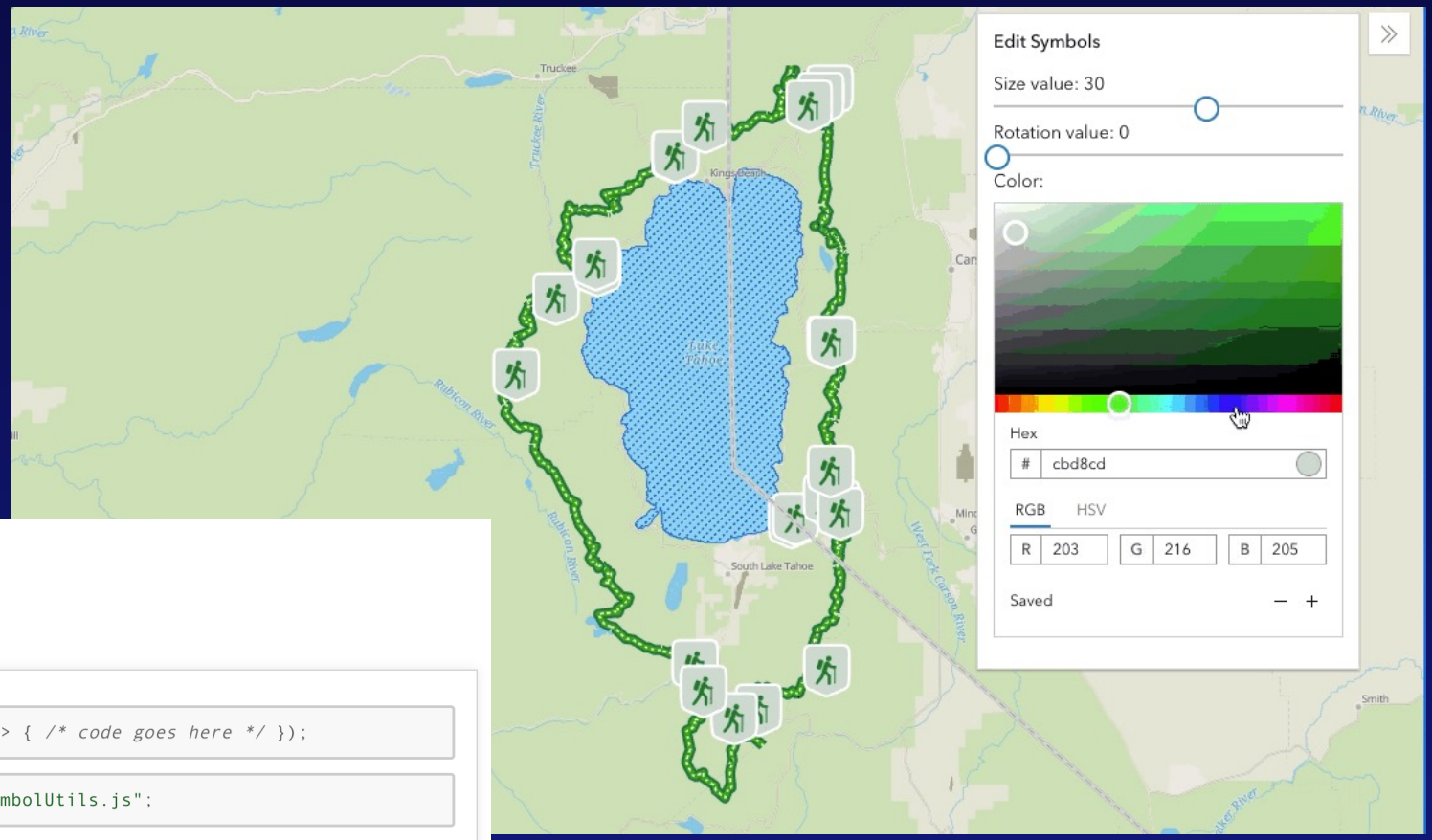


Utility methods for CIMSymbols / WebStyles

cimSymbolUtils

https://developers.arcgis.com/javascript/latest/api-reference/esri-symbols-support-cimSymbolUtils.html

- Utility functions for CIMSymbols
- Allows you to get / set the size, color, and rotation of a CIM symbol
- Scales proportionally, honors color locking



cimSymbolUtils

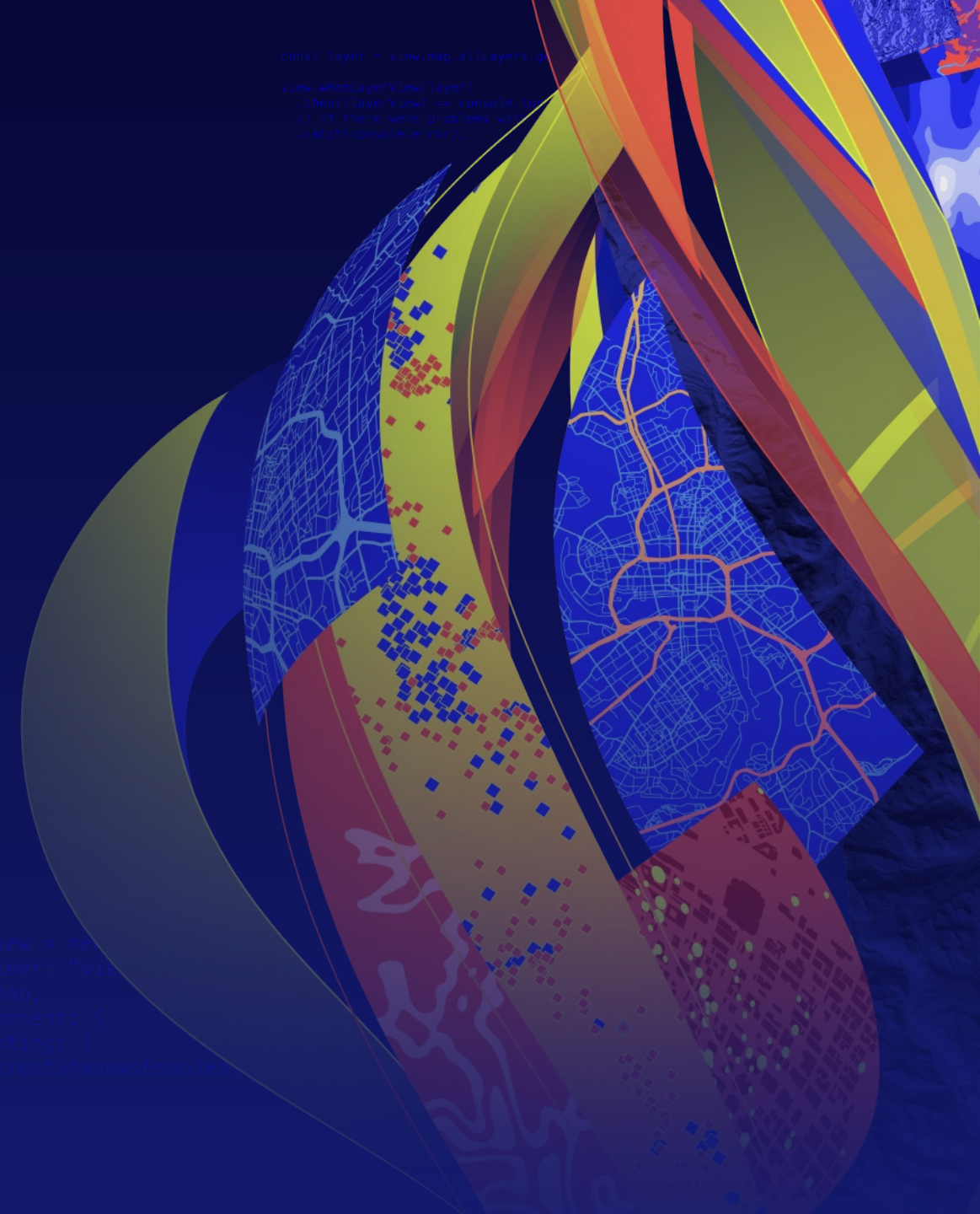
AMD:
```
require(["esri/symbols/support/cimSymbolUtils"], (cimSymbolUtils) => { /* code goes here */ });
```

ESM:
```
import * as cimSymbolUtils from "@arcgis/core/symbols/support/cimSymbolUtils.js";
```

Object: `esri/symbols/support/cimSymbolUtils`

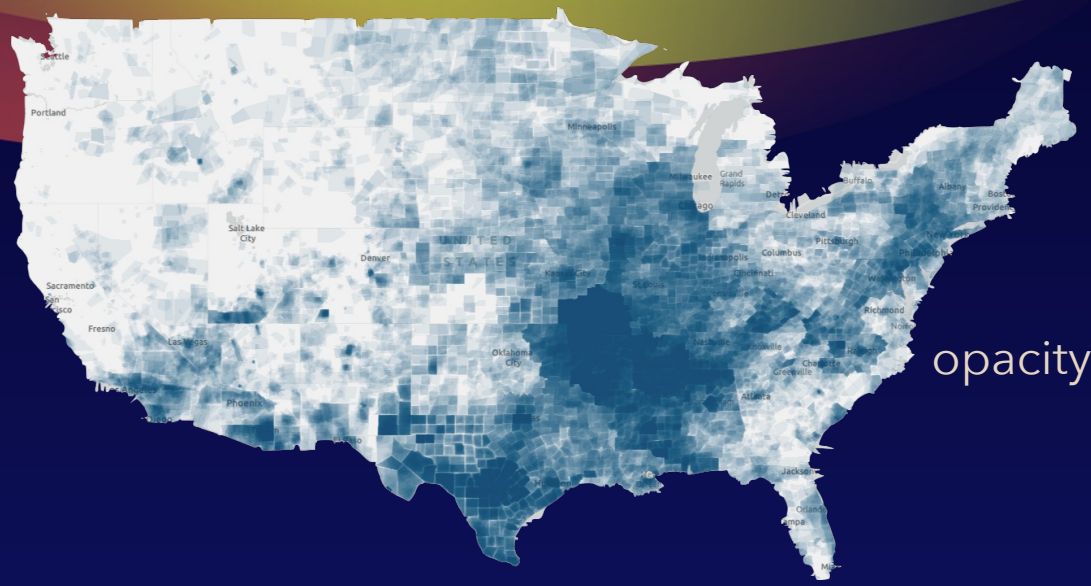
Since: ArcGIS Maps SDK for JavaScript 4.16

Data Driven CIM & Animations

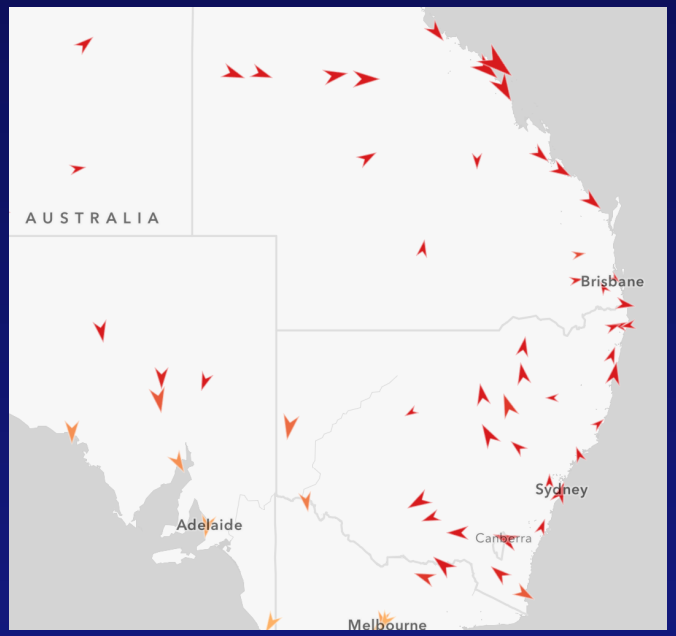


Data-driven mapping

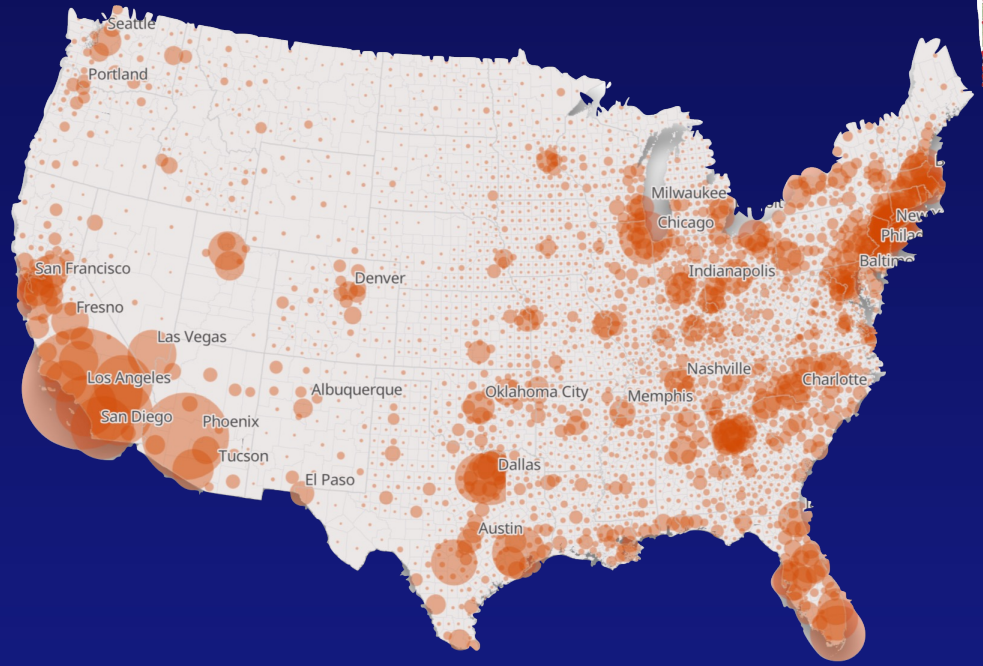
- Override a symbol's visual variable with a data value (either a field value, or an Arcade expression)

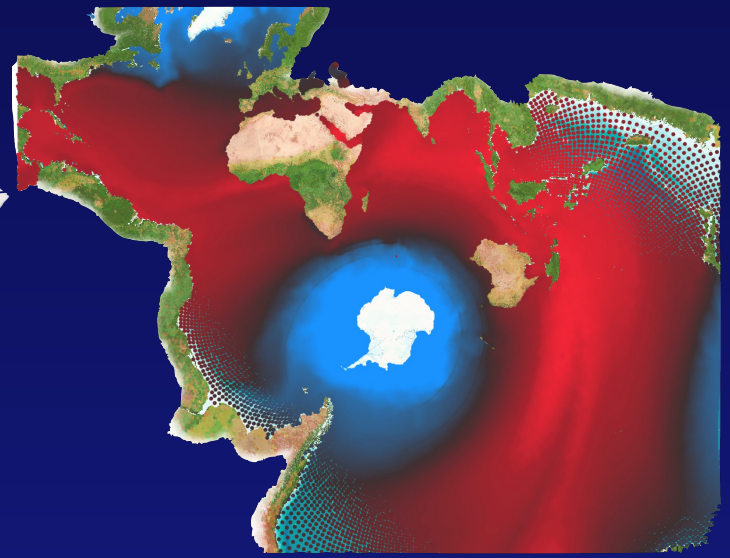


rotation



size





color

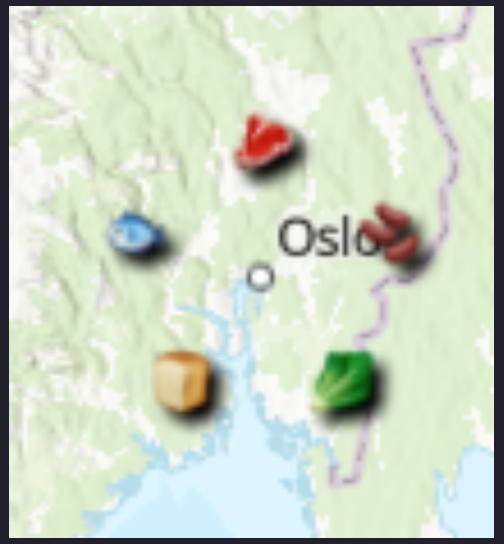
The “Diet Pentagon”

- Novel semi-dynamic symbology.
  - Irregular pentagon that stretches towards preferred items.
- Countries with similar pentagons have similar diets.
- Two steps process:
  - Precompute all possible pentagons as separate symbols.
  - Use a `UniqueValueRenderer` to match them to features.

Demo: https://beautiful-blancmange-5fda50.netlify.app/demo-1.html
Source: https://github.com/damix911/DS2023/blob/main/demo-1.html

The "vertex labels" (the 5 icons around the polygon)

```
{
  type: "CIMPointSymbol",
  symbolLayers: [
    ...labels,
    {
      type: "CIMVectorMarker",
      /*...*/
      markerGraphics: [
        {
          type: "CIMMarkerGraphic",
          /*...*/
        },
        {
          type: "CIMMarkerGraphic",
          /*...*/
        },
        {
          type: "CIMMarkerGraphic",
          /*...*/
        },
        {
          type: "CIMMarkerGraphic",
          /*...*/
        },
        {
          type: "CIMMarkerGraphic",
        }
      ]
    }
  ]
};
```



The Background Polygon

```
{
  type: "CIMMarkerGraphic",
  geometry: {
    rings: [back]
  },
  symbol: {
    type: "CIMPolygonSymbol",
    symbolLayers: [
      {
        type: "CIMSolidStroke",
        width: 1,
        color: [150, 150, 150, 240]
      },
      {
        type: "CIMSolidFill",
        color: [240, 240, 240, 200]
      }
    ]
  }
},
```

Inner Dashed Ring

```
{
  type: "CIMMarkerGraphic",
  geometry: {
    rings: [r1]
  },
  symbol: {
    type: "CIMPolygonSymbol",
    symbolLayers: [
      {
        type: "CIMSolidStroke",
        effects: [{
          type: "CIMGeometricEffectDashes",
          dashTemplate: [2, 2],
          lineDashEnding: "FullGap",
          controlPointEnding: "NoConstraint"
        }],
        width: 0.75,
        color: [255, 255, 255, 220]
      }
    ]
  }
},
```

Outer Dashed Ring

```
{
  type: "CIMMarkerGraphic",
  geometry: {
    rings: [r2]
  },
  symbol: {
    type: "CIMPolygonSymbol",
    symbolLayers: [
      {
        type: "CIMSolidStroke",
        effects: [{
          type: "CIMGeometricEffectDashes",
          dashTemplate: [2, 2],
          lineDashEnding: "FullGap",
          controlPointEnding: "NoConstraint"
        }],
        width: 0.75,
        color: [255, 255, 255, 220]
      }
    ]
  }
},
```

The Data Polygon

```
{
  type: "CIMMarkerGraphic",
  geometry: {
    rings: [data]
  },
  symbol: {
    type: "CIMPolygonSymbol",
    symbolLayers: [
      {
        type: "CIMSolidStroke",
        width: 1,
        color: [240, 94, 35, 240]
      },
      {
        type: "CIMSolidFill",
        color: [240, 94, 35, 60]
      }
    ]
  }
},
```

Country Name

```
{
  "type": "CIMMarkerGraphic",
  "geometry": {
    "x": 0,
    "y": 0
  },
  "primitiveName": "Text",
  "symbol": {
    "type": "CIMTextSymbol",
    "fontFamilyName": "Arial",
    "fontStyleName": "Bold",
    "height": 8,
    "horizontalAlignment": "Center",
    "offsetX": 0,
    "offsetY": -42,
    "symbol": {
      "type": "CIMPolygonSymbol",
      "symbolLayers": [
        {
          "type": "CIMSolidFill",
          "enable": true,
          "color": [0, 0, 0, 255]
        }
      ]
    },
    "verticalAlignment": "Center",
  }
}
```

Text Callout

```
"callout": {
  type: "CIMBackgroundCallout",
  backgroundSymbol: {
    "type": "CIMPolygonSymbol",
    "symbolLayers": [
      {
        "type": "CIMSolidStroke",
        "enable": true,
        "width": 1,
        "color": [20, 40, 120, 255]
      },
      {
        "type": "CIMSolidFill",
        "enable": true,
        "color": [220, 230, 240, 255]
      }
    ]
  }
},
```

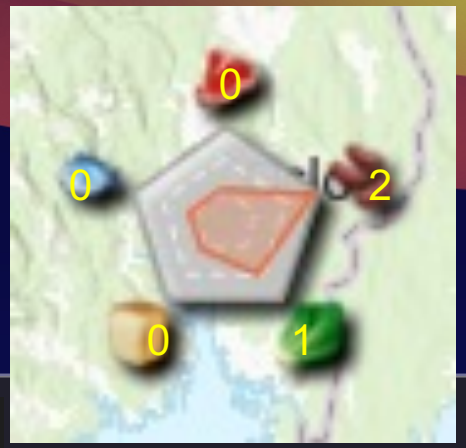
Creating the Pentagons (`createPentagon()` function)

Data-driven shape

Static shape

```
for (let i = 0; i < diet.length; i++) {
  const co = Math.cos(2 * Math.PI * i / diet.length);
  const si = Math.sin(2 * Math.PI * i / diet.length);

  const radius = diet[i] * 5;

  data.push([(radius + 5) * si, (radius + 5) * co]);
  back.push([15 * si, 15 * co]);
  r1.push([5 * si, 5 * co]);
  r2.push([10 * si, 10 * co]);

  labels.push({
    type: "CIMPictureMarker",
    enable: true,
    anchorPoint: { x: 22 * si, y: 22 * co },
    size: 14,
    scaleX: 1,
    tintColor: [255, 255, 255, 255],
    url: iconUrls[i]
  });
}
```

Creating the Renderer

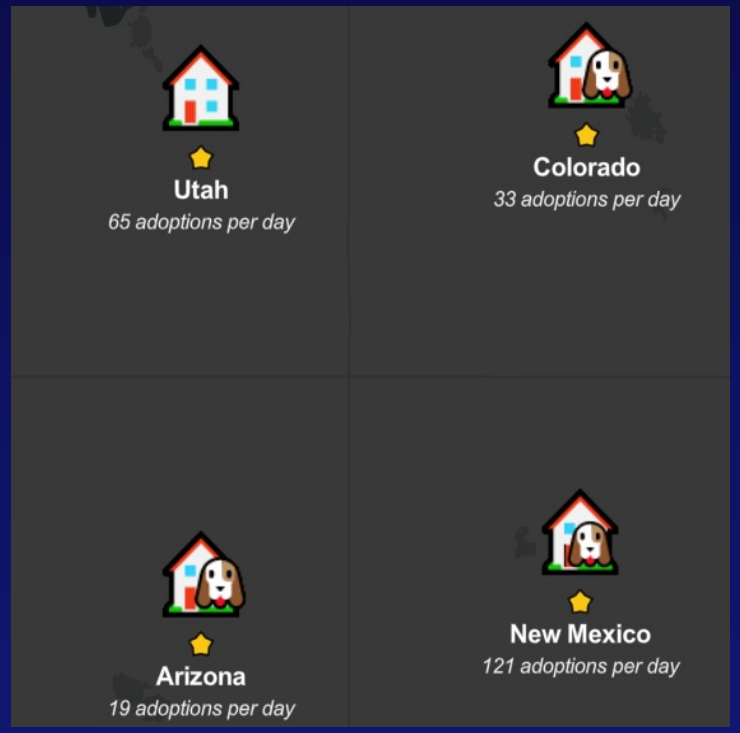



```
const DietCode = "Diet_" + m + f + c + g + b;

uniqueValueInfos.push({
  value: DietCode,
  symbol: {
    type: "cim",
    data: {
      type: "CIMSymbolReference",
      symbol: createPentagon(DietCode),
      primitiveOverrides: [
        {
          type: "CIMPrimitiveOverride",
          primitiveName: "Text",
          propertyName: "TextString",
          valueExpressionInfo: {
            type: "CIMExpressionInfo",
            expression: "$feature.Name"
          }
        }
      ]
    }
  },
});
```

[Meat , Fish, Carbs, Greens, Beans]
e.g. [0, 0, 0, 1, 2] --> "02100"

```
renderer: {
  type: "unique-value",
  valueExpression: `
    var m = round($feature.Meat / 100 * 2);
    var f = round($feature.Fish / 100 * 2);
    var c = round($feature.Carbs / 100 * 2);
    var g = round($feature.Greens / 100 * 2);
    var b = round($feature.Beans / 100 * 2);

    return "Diet_" + m + f + c + g + b;
  `,
  defaultSymbol: {
    type: "simple-marker"
  },
  uniqueValueInfos
},
```

Puppy Adoption Rate



- Floating puppy heads fly into loving homes!
- Animation speed is driven by the rate of adoptions per day.
- We use a composite CIM symbol with 4 parts.
  - An animated GIF;
  - A vector star;
  - 2 text symbols.

Demo: https://beautiful-blancmange-5fda50.netlify.app/demo-2.html
Source: https://github.com/damix911/DS2023/blob/main/demo-2.html

The animated GIF

```
{
  type: "cim",
  data: {
    type: "CIMSymbolReference",
    symbol: {
      type: "CIMPointSymbol",
      symbolLayers: [
        {
          type: "CIMPictureMarker",
          url: "images/puppy-and-house.gif",
          size: 48,
          offsetX: 0,
          offsetY: 30,
          animatedSymbolProperties: {
            primitiveName: "AdoptionsPerDay"
          }
        },
        {
          type: "CIMVectorMarker",
          /* ... */
          "scaleSymbolsProportionally": true,
          "respectFrame": true,
          size: 20,
          markerGraphics: [
            /* ... */
          ]
        }
      ]
    },
    primitiveOverrides: [
      /* ... */
    ]
  }
}
```

A Vector Star

```
{
  type: "CIMMarkerGraphic",
  geometry: {
    rings: [star]
  },
  symbol: {
    type: "CIMPolygonSymbol",
    symbolLayers: [
      {
        type: "CIMSolidStroke",
        color: [0, 0, 0, 255],
        width: 0.5
      },
      {
        type: "CIMSolidFill",
        color: [250, 200, 20, 255]
      }
    ]
  }
},
```

```
const star = [];

for (let i = 0; i <= 10; i++) {
  const radius = i % 2 ? 2 : 3;
  const radians = 2 * Math.PI * i / 10;

  star.push([
    radius * Math.sin(radians),
    radius * Math.cos(radians)
  ]);
}
```

State Name

```
{
  "type": "CIMMarkerGraphic",
  "geometry": {
    "x": 0,
    "y": 0
  },
  "primitiveName": "Name",
  "symbol": {
    "type": "CIMTextSymbol",
    "fontFamilyName": "Arial",
    "fontStyleName": "Bold",
    "height": 6,
    "horizontalAlignment": "Center",
    "offsetX": 0,
    "offsetY": -7,
    "symbol": {
      "type": "CIMPolygonSymbol",
      "symbolLayers": [
        {
          "type": "CIMSolidFill",
          "enable": true,
          "color": [255, 255, 255, 255]
        }
      ]
    },
    "verticalAlignment": "Center",
  }
},
```

Label with Adoption Count

```
{
  "type": "CIMMarkerGraphic",
  "geometry": {
    "x": 0,
    "y": 0
  },
  "primitiveName": "Stats",
  "symbol": {
    "type": "CIMTextSymbol",
    "fontFamilyName": "Arial",
    "fontStyleName": "Italic",
    "height": 5,
    "horizontalAlignment": "Center",
    "offsetX": 0,
    "offsetY": -15,
    "symbol": {
      "type": "CIMPolygonSymbol",
      "symbolLayers": [
        {
          "type": "CIMSolidFill",
          "enable": true,
          "color": [255, 255, 255, 255]
        }
      ]
    },
    "verticalAlignment": "Center",
  }
}
```

Primitive Overrides

```
primitiveOverrides: [
  {
    type: "CIMPrimitiveOverride",
    primitiveName: "AdoptionsPerDay",
    propertyName: "Duration",
    valueExpressionInfo: {
      type: "CIMExpressionInfo",
      expression: "100 / $feature.AdoptionsPerDay"
    }
  },
  {
    type: "CIMPrimitiveOverride",
    primitiveName: "Name",
    propertyName: "TextString",
    valueExpressionInfo: {
      type: "CIMExpressionInfo",
      expression: "$feature.Name"
    }
  },
  {
    type: "CIMPrimitiveOverride",
    primitiveName: "Stats",
    propertyName: "TextString",
    valueExpressionInfo: {
      type: "CIMExpressionInfo",
      expression: "$feature.AdoptionsPerDay + ' adoptions per day'"
    }
  }
```

Animation speed

State name

Adoption stats string

Flaming Polylines

- Visualization of fault lines using a fire animation.
- Uses animated GIF markers placed along a line.
- Thanks Anne for the idea!

Demo: https://beautiful-blancmange-5fda50.netlify.app/demo-3.html
Source: https://github.com/damix911/DS2023/blob/main/demo-3.html

Layer and Renderer

```
const featureLayer = new FeatureLayer({
  url: "https://services2.arcgis.com/FiaPA4ga0iQKduv3/arcgis/rest/services/Earthquake_Faults_and_Folds_in_the_USA/FeatureServer",
  effect: "bloom(2, 1px, 0)",
  blendMode: "plus",
  renderer: {
    type: "simple",
    symbol: {
      type: "cim",
      data: {
        type: "CIMSymbolReference",
        symbol: {
          type: "CIMLineSymbol",
          symbolLayers
        }
      }
    }
  }
});
```

We Use an Animated GIF

```
effect: "bloom(2, 1px, 0)",
blendMode: "plus",
```

Additive blending is good for *"light-like"* image with black background.

Using a Single Symbol Layer :-/

```
symbolLayers.push({
  "type": "CIMPictureMarker",
  "enable": true,
  "anchorPointUnits": "Relative",
  "dominantSizeAxis3D": "Y",
  "size": 8,
  "billboardMode3D": "FaceNearPlane",
  "markerPlacement": {
    "type": "CIMMarkerPlacementAlongLineSameSize",
    "placePerPart": true,
    "angleToLine": true,
    "endings": "NoConstraint",
    "placementTemplate": [
      4
    ]
  },
  "invertBackfaceTexture": true,
  "scaleX": 1,
  "textureFilter": "Picture",
  "url": "images/flames-2.gif"
});
```

Using Multiple Symbol Layers :-)

```
const symbolLayers = [];

const primeNumbers = [2, 3, 5, 7, 11, 13, 17, 19];

for (let i = primeNumbers.length - 1; i >= 0; i--) {
  symbolLayers.push({
    "type": "CIMPictureMarker",
    "enable": true,
    "anchorPointUnits": "Relative",
    "dominantSizeAxis3D": "Y",
    "size": 8,
    "billboardMode3D": "FaceNearPlane",
    "markerPlacement": {
      "type": "CIMMarkerPlacementAlongLineSameSize",
      "placePerPart": true,
      "angleToLine": true,
      "endings": "NoConstraint",
      "placementTemplate": [
        4 + primeNumbers[i]
      ]
    },
    "invertBackfaceTexture": true,
    "scaleX": 1,
    "textureFilter": "Picture",
    "url": "images/flames-2.gif",
    animatedSymbolProperties: {
      startTimeOffset: i / 5
    }
  });
}
```

Customized Highlight Effect

- The Highlight Effect that you always wanted!
- Uses the same technique as the previous sample.
  - Animated GIF markers along a line.
  - Makes everything pretty using the `bloom()` layer effect.

Demo: https://beautiful-blancmange-5fda50.netlify.app/demo-4.html
Source: https://github.com/damix911/DS2023/blob/main/demo-4.html

Basemap

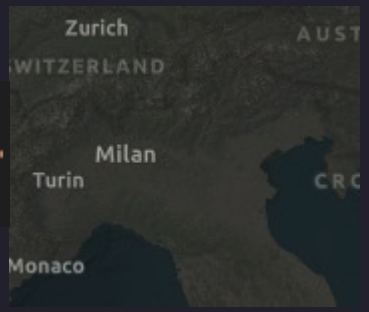
```
const imageryLayer = new TileLayer({
  url: "https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer"
});
```

Make it Darker

```
const imageryLayer = new TileLayer({
  url: "https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer",
  effect: "brightness(40%)"
});
```

Reference Vector Tile Layer

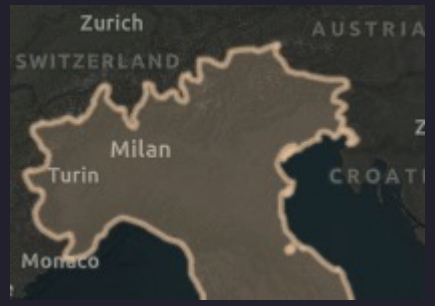
```
const vectorTileLayer = new VectorTileLayer({
  opacity: 0.7,
  url: "https://www.arcgis.com/sharing/rest/content/items/c11ce4f7801740b2905eb03ddc963ac8/resources/styles/root.json"
});
```



The "highlight" is a Graphic in the Topmost Layer

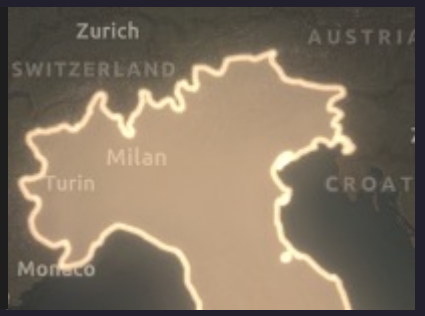
```
const graphicsLayer = new GraphicsLayer();
```

```
const basemap = new Basemap({
  baseLayers: [
    imageryLayer,
    vectorTileLayer
  ]
});

const map = new Map({
  basemap,
  layers: [
    graphicsLayer
  ]
});
```



BLOOOOOOOOOOM!!!

```
const graphicsLayer = new GraphicsLayer({
  effect: "bloom(1.6, 1px, 0)"
});
```



A custom `highlight()` function

```
async function highlight(feature) {
  const { geometry } = feature;

  const hl = new Graphic({
    geometry,
    symbol: {
      type: "cim",
      data: {
        type: "CIMSymbolReference",
        symbol: {
          type: "CIMLineSymbol",
          symbolLayers: [
            {
              "type": "CIMPictureMarker",
              "size": 4,
              "markerPlacement": {
                "type": "CIMMarkerPlacementAlongLineSameSize",
                "placementTemplate": [1]
              },
              "tintColor": [250, 210, 170, 100],
              "url": "images/pulse.gif"
            },
            {
              "type": "CIMSolidFill",
              "color": [250, 210, 170, 50],
              "enable": true
            }
          ]
        }
      }
    }
  });

  graphicsLayer.graphics.removeAll();
  graphicsLayer.graphics.add(hl);
}
```

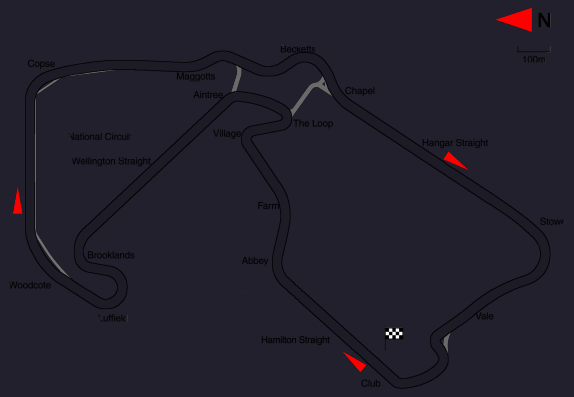
Futuristic Race Animation

- A luminous dot zooming through the Silverstone track.
  - The circuit is a georeferenced image in a MediaLayer.
  - The dots are point features placed individually by hand.
  - Every dot has a `Time` attribute that tells when to light up.
- Could be made data-driven using real telemetry data.
  - They did it IRL!
  - Google *"Ghost lap - Ayrton Senna at Suzuka 1989"*.
  - It's awesome!

Demo: https://beautiful-blancmange-5fda50.netlify.app/demo-5.html
Source: https://github.com/damix911/DS2023/blob/main/demo-5.html

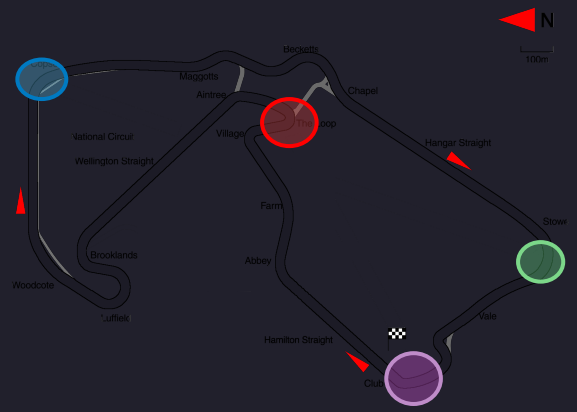
Georeference the Image

```
const p1 = {
  sourcePoint: { x: 103, y: 287 },
  mapPoint: new Point({ x: -112621.4753091779 - 10, y: 6814414.635010252 + 10, spatialReference: { wkid: 3857 } })
};

const p2 = {
  sourcePoint: { x: 1247, y: 1127 },
  mapPoint: new Point({ x: -113992.56450407869 + 10, y: 6812193.183875641 + 20, spatialReference: { wkid: 3857 } })
};

const p3 = {
  sourcePoint: { x: 805, y: 395 },
  mapPoint: new Point({ x: -112702.68965173334 - 10, y: 6813098.484929426, spatialReference: { wkid: 3857 } })
};

const p4 = {
  sourcePoint: { x: 1589, y: 767 },
  mapPoint: new Point({ x: -113278.35602102136, y: 6811612.740192077, spatialReference: { wkid: 3857 } })
};

const controlPoints = [p1, p2, p3, p4];
```



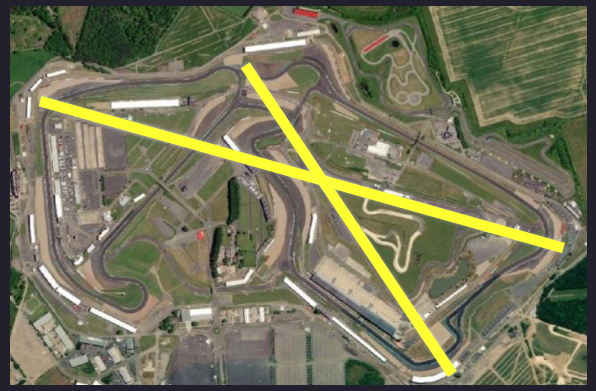
Georeference the Image

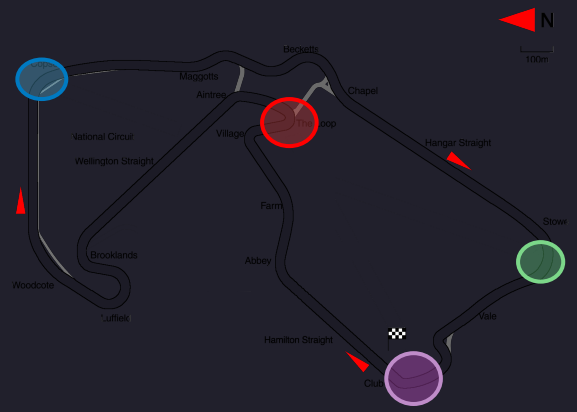
```
const p1 = {
  sourcePoint: { x: 103, y: 287 },
  mapPoint: new Point({ x: -112621.4753091779 - 10, y: 6814414.635010252 + 10, spatialReference: { wkid: 3857 } })
};

const p2 = {
  sourcePoint: { x: 1247, y: 1127 },
  mapPoint: new Point({ x: -113992.56450407869 + 10, y: 6812193.183875641 + 20, spatialReference: { wkid: 3857 } })
};

const p3 = {
  sourcePoint: { x: 805, y: 395 },
  mapPoint: new Point({ x: -112702.68965173334 - 10, y: 6813098.484929426, spatialReference: { wkid: 3857 } })
};

const p4 = {
  sourcePoint: { x: 1589, y: 767 },
  mapPoint: new Point({ x: -113278.35602102136, y: 6811612.740192077, spatialReference: { wkid: 3857 } })
};

const controlPoints = [p1, p2, p3, p4];
```



Georeference the Image

```
const p1 = {
  sourcePoint: { x: 103, y: 287 },
  mapPoint: new Point({ x: -112621.4753091779 - 10, y: 6814414.635010252 + 10, spatialReference: { wkid: 3857 } })
};

const p2 = {
  sourcePoint: { x: 1247, y: 1127 },
  mapPoint: new Point({ x: -113992.56450407869 + 10, y: 6812193.183875641 + 20, spatialReference: { wkid: 3857 } })
};

const p3 = {
  sourcePoint: { x: 805, y: 395 },
  mapPoint: new Point({ x: -112702.68965173334 - 10, y: 6813098.484929426, spatialReference: { wkid: 3857 } })
};

const p4 = {
  sourcePoint: { x: 1589, y: 767 },
  mapPoint: new Point({ x: -113278.35602102136, y: 6811612.740192077, spatialReference: { wkid: 3857 } })
};

const controlPoints = [p1, p2, p3, p4];
```





Georeference the Image

```
const georeference = new ControlPointsGeoreference({ controlPoints, width: 1665, height: 1200 });

const silverstoneTrack = new ImageElement({
  image: "images/silverstone.png",
  georeference
});

const mediaLayer = new MediaLayer({
  source: [silverstoneTrack]
});
```

Georeference the Image

```
const georeference = new ControlPointsGeoreference({ controlPoints, width: 1665, height: 1200 });

const silverstoneTrack = new ImageElement({
  image: "images/silverstone.png",
  georeference
});

const mediaLayer = new MediaLayer({
  effect: "invert()",
  source: [silverstoneTrack]
});
```

Georeference the Image

```
const georeference = new ControlPointsGeoreference({ controlPoints, width: 1665, height: 1200 });

const silverstoneTrack = new ImageElement({
  image: "images/silverstone.png",
  georeference
});

const mediaLayer = new MediaLayer({
  effect: "invert() bloom(0.4, 1px, 0)",
  source: [silverstoneTrack]
});
```

Animated Points Using Primitive Overrides

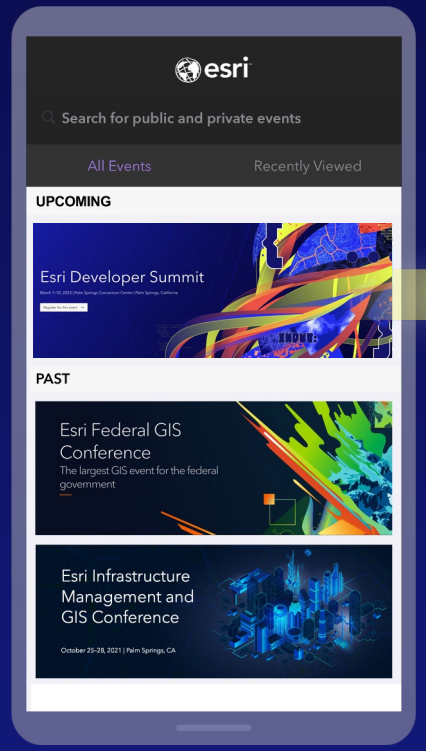
```
const featureLayer = new FeatureLayer({
  effect: "brightness(200%) bloom(2.5, 0.1px, 15%)",
  renderer: {
    type: "simple",
    symbol: {
      type: "cim",
      data: {
        type: "CIMSymbolReference",
        symbol: {
          type: "CIMPointSymbol",
          symbolLayers: [
            {
              "type": "CIMPictureMarker",
              "size": 8,
              "tintColor": [210, 230, 250, 40],
              "url": "images/flash.gif",
              animatedSymbolProperties: {
                primitiveName: "FlashTime",
                duration: 1
              }
            }
          ]
        },
        primitiveOverrides: [
          {
            type: "CIMPrimitiveOverride",
            primitiveName: "FlashTime",
            propertyName: "StartTimeOffset",
            valueExpressionInfo: {
              type: "CIMExpressionInfo",
              expression: "210 - $feature.time / 210"
            }
          }
        ]

        /* ... */
      }
```

Resources

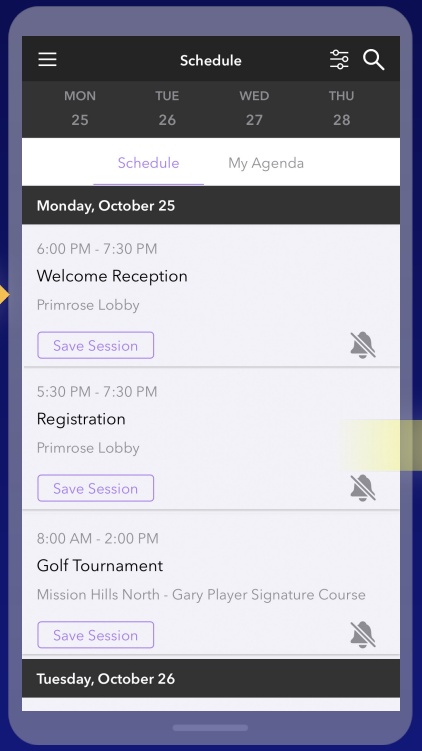
- CIMSymbol API Reference: https://developers.arcgis.com/javascript/latest/api-reference/esri-symbols-CIMSymbol.html
- CIM Symbol Builder: https://esri.github.io/cim-symbol-builder-js/
- 2D WebStyles: https://developers.arcgis.com/javascript/latest/visualization/symbols-color-ramps/esri-web-style-symbols-2d/
- Esri Community: https://community.esri.com/t5/arcgis-api-for-javascript/ct-p/arcgis-api-for-javascript
- Helpful Blogs:
  - https://www.esri.com/arcgis-blog/products/js-api-arcgis/mapping/create-points-lines-and-polygons-using-cimsymbols/
  - https://www.esri.com/arcgis-blog/products/js-api-arcgis/mapping/visualize-electoral-swing-using-composite-symbols/#how-the-map-is-made

Please Share Your Feedback in the App

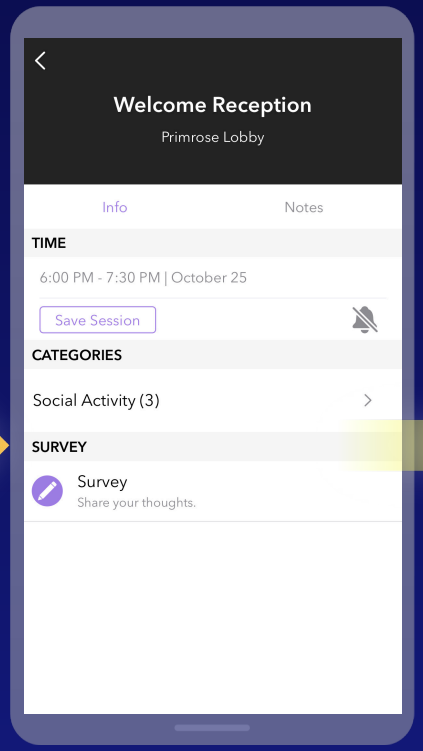
Download the Esri Events app and find your event



Select the session you attended



Scroll down to “Survey”



Log in to access the survey