Visual Reporting With Connected Design Data

Philippe Leefsma

Principal Developer Advocate, Forge Partner Development

@F3lipek

Class summary

Viewing 3D models on a webpage or in a mobile App is great, but visualizing design data isn't just about meshes and textures

In this class I will show you how to use JavaScript to turn the Forge Viewer into a powerful visual reporting tool, giving you access to valuable data associated with each model extracted from the original design file and connect your viewing application to external databases &Web APIs to add rich visualization experience

Basic Web development experience is preferable - HTML5 JavaScript, CSS, Web Services, REST API's, …

Key learning objectives

At the end of this class, you will be able to:

- Access and manipulate the data which is available out of the box with your Forge Viewer models
- Explore various way of overlaying and embedding custom graphics into the Viewer
- Learn how to create powerful, highly customized visual applications integrating elements from multiple data sources
- Leverage the Forge Model Derivative API to access and manipulate design data in the cloud

Agenda

- Accessing Forge Design Data
- UI Customization
- Connecting your App to the Cloud

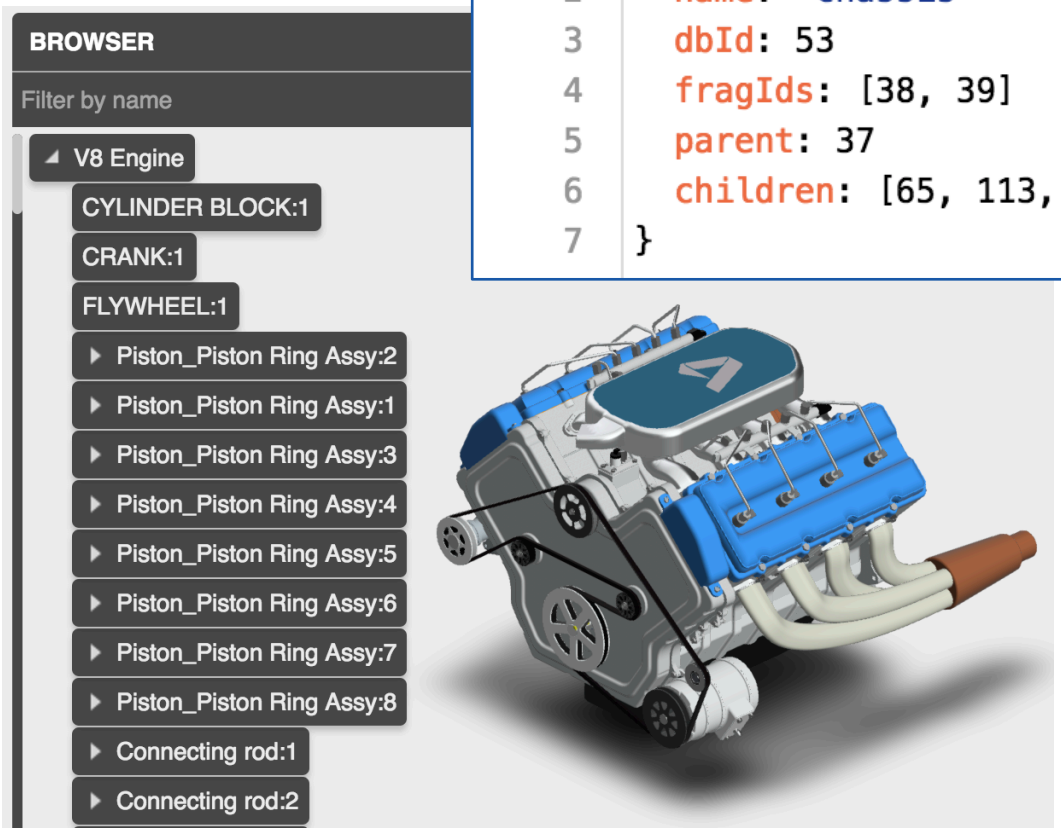
Accessing Forge Design Data

Accessing the Viewer API

```
1  html:
2  <div id='viewer'> </div>
3
4  js:
5
6    loadDocument(urn, ..., function(doc) {
7
8      var viewerContainer = document.getElementById('viewer')
9
10     var viewer = Autodesk.Viewing.Viewer3D(viewerContainer)
11
12     viewer.loadModel(getViewablePath(doc))
13
14     // call API's
15
16     viewer.setBackgroundColor( ... )
17
18     viewer.loadExtension('MyExtId')
19
20     // ...
21   })
```

Viewer Component Model

```
{
  name: 'chassis'           //display name of the component
  dbId: 53                  //unique id for the component in the model
  fragIds: [38, 39]         //reference the three.js meshes
  parent: 37                //dbId of the parent node
  children: [65, 113, 146] //array of children nodes ids
}
```



Model Structure

```
1  function buildModelTreeRec (node) {
2
3      instanceTree.enumNodeChildren (node.dbId, function (childId) {
4
5          node.children = node.children || []
6
7          var childNode = {
8              dbId: childId,
9              name: instanceTree.getNodeName(childId)
10         }
11
12         node.children.push(childNode)
13
14         buildModelTreeRec(childNode)
15     })
16 }
```

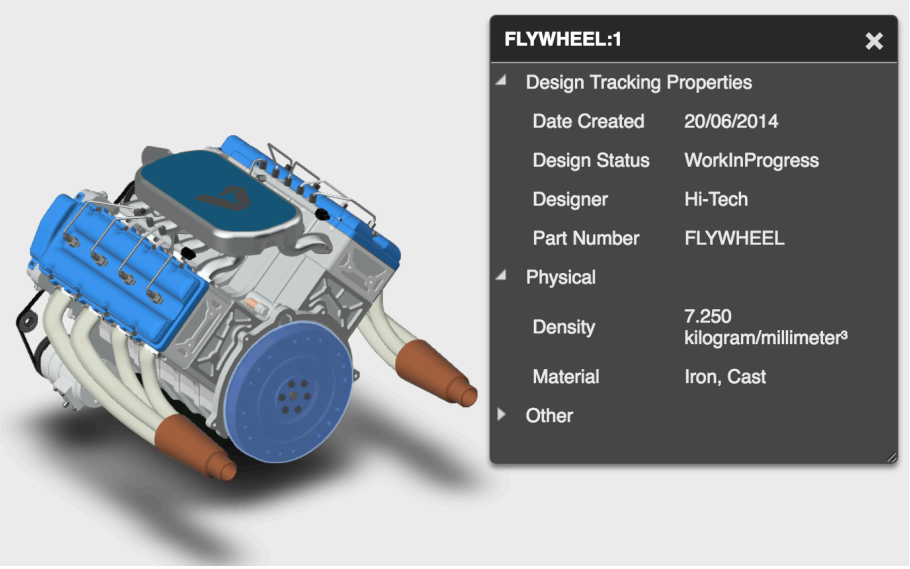
Node Fragments

```
function buildModelTreeRec (node) {

  node.fragIds = []

  instanceTree.enumNodeChildren(node.dbId, function(childId) {
    var childNode = {
      dbId: childId,
      name: instanceTree.getNodeName(childId)
    }

    node.children.push(childNode)

    buildModelTreeRec(childNode)
  })
}
```

```
{
  name: 'chassis'          //display name of the component
  dbId: 53                 //unique id for the component in the model
  fragIds: [38, 39]        //reference the three.js meshes
  parent: 37               //dbId of the parent node
  children: [65, 113, 146] //array of children nodes ids
}
```

Component Properties

```
1  model.getProperties(dbId, function(result) {
2
3      if (result.properties){
4
5          result.properties.forEach( function (prop) {
6
7              console.log(prop)
8          })
9      }
10 }
```

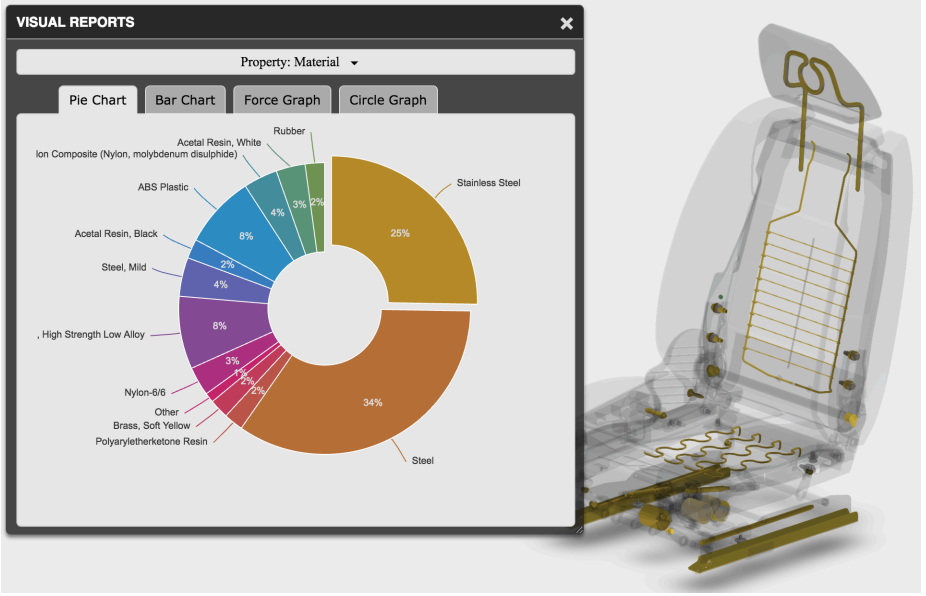


UI Customization

2D Overlays

Custom Panel

```
1  class CustomPanel extends Autodesk.Viewing.UI.DockingPanel {
2
3   constructor(container, title, options = {}) {
4
5    super(container, panelId, title, options)
6
7    this.container.appendChild(...)
8   }
9
10 }
```



Custom Property Panel

```
1  class CustomPropertyPanel extends Autodesk.Viewing.UI.ViewerPropertyPanel {
2
3    constructor(viewer) {
4
5      super(viewer)
6    }
7
8    setProperties (properties) {
9
10     properties.push({ ... custom property ... })
11
12   super.setProperties(properties)
13   }
14 }
```

Fragments Overrides

```
1  // access fragment proxy i.e. THREE.Mesh
2  var fragProxy = viewer.impl.getFragmentProxy(
3      model, fragId)
4
5  fragProxy.getAnimTransform()
6
7  fragProxy.position = new THREE.Vector3(x, y, z)
8
9  //Not a standard three.js quaternion
10 fragProxy.quaternion._x = qx;
11 fragProxy.quaternion._y = qy;
12 fragProxy.quaternion._z = qz;
13 fragProxy.quaternion._w = qw;
14
15 fragProxy.updateAnimTransform()
16
17 viewer.impl.invalidate(true)
```

SVG

- *Scalable Vector Graphics (SVG) is an XML-based markup language for describing two-dimensional vector graphics. SVG is essentially to graphics what HTML is to text*

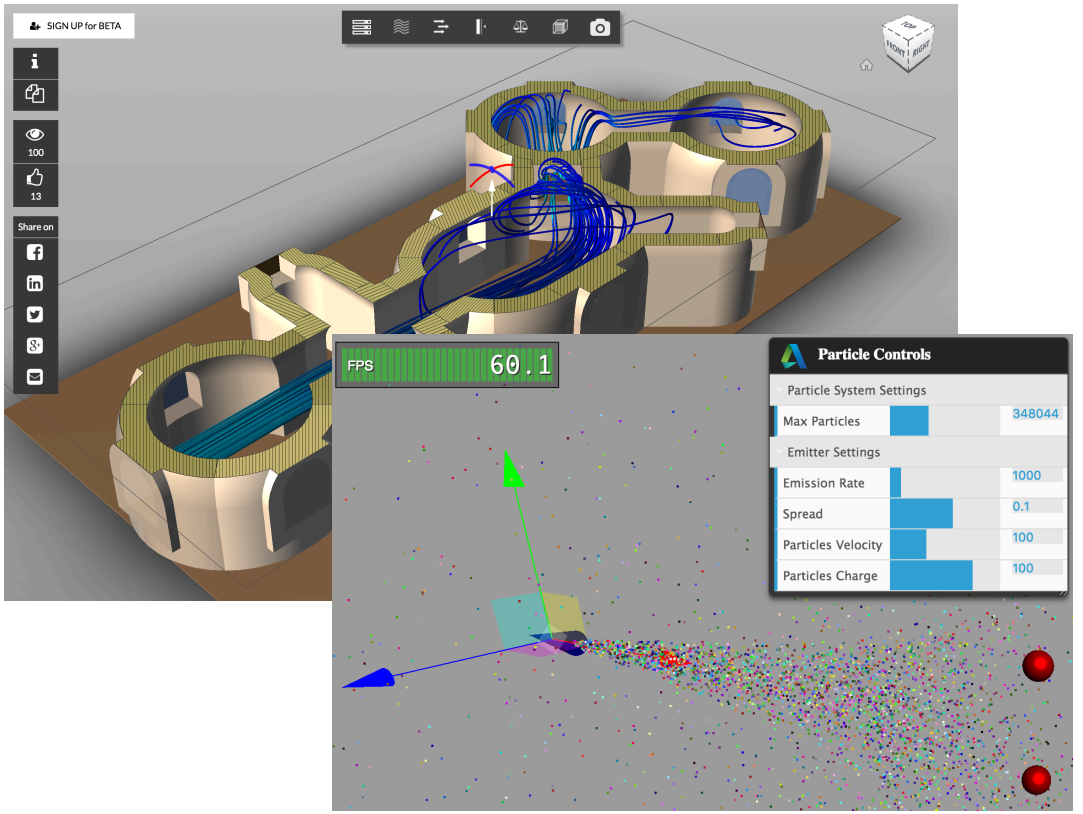
Coordinates Conversion

```
1  function worldToScreen(worldPoint) {
2
3    var p = new THREE.Vector4()
4
5    p.x = worldPoint.x
6    p.y = worldPoint.y
7    p.z = worldPoint.z
8    p.w = 1
9
10   var camera = viewer.navigation.getCamera()
11
12   p.applyMatrix4(camera.matrixWorldInverse)
13   p.applyMatrix4(camera.projectionMatrix)
14
15   var screenPoint = viewer.impl.viewportToClient(p.x, p.y)
16
17   //snap pixel centre
18   screenPoint.x = Math.floor(screenPoint.x) + 0.5
19   screenPoint.y = Math.floor(screenPoint.y) + 0.5
20
21   return screenPoint
22 }
```

```
nt) {

ion.getScreenViewport()


ort.left) / viewport.width,
ort.top) / viewport.height


ties.getHitPoint(n.x, n.y)
```

3D Overlays & Custom meshes

```
1  var geometry = new THREE.SphereGeometry(size, 4, 4)
2
3  var mesh = new THREE.Mesh(
4    geometry,
5    material)
6
7  mesh.position.set(x, y, z)
8
9  viewer.impl.scene.add(mesh)
```



CSS3DRenderer & CSS3DObject

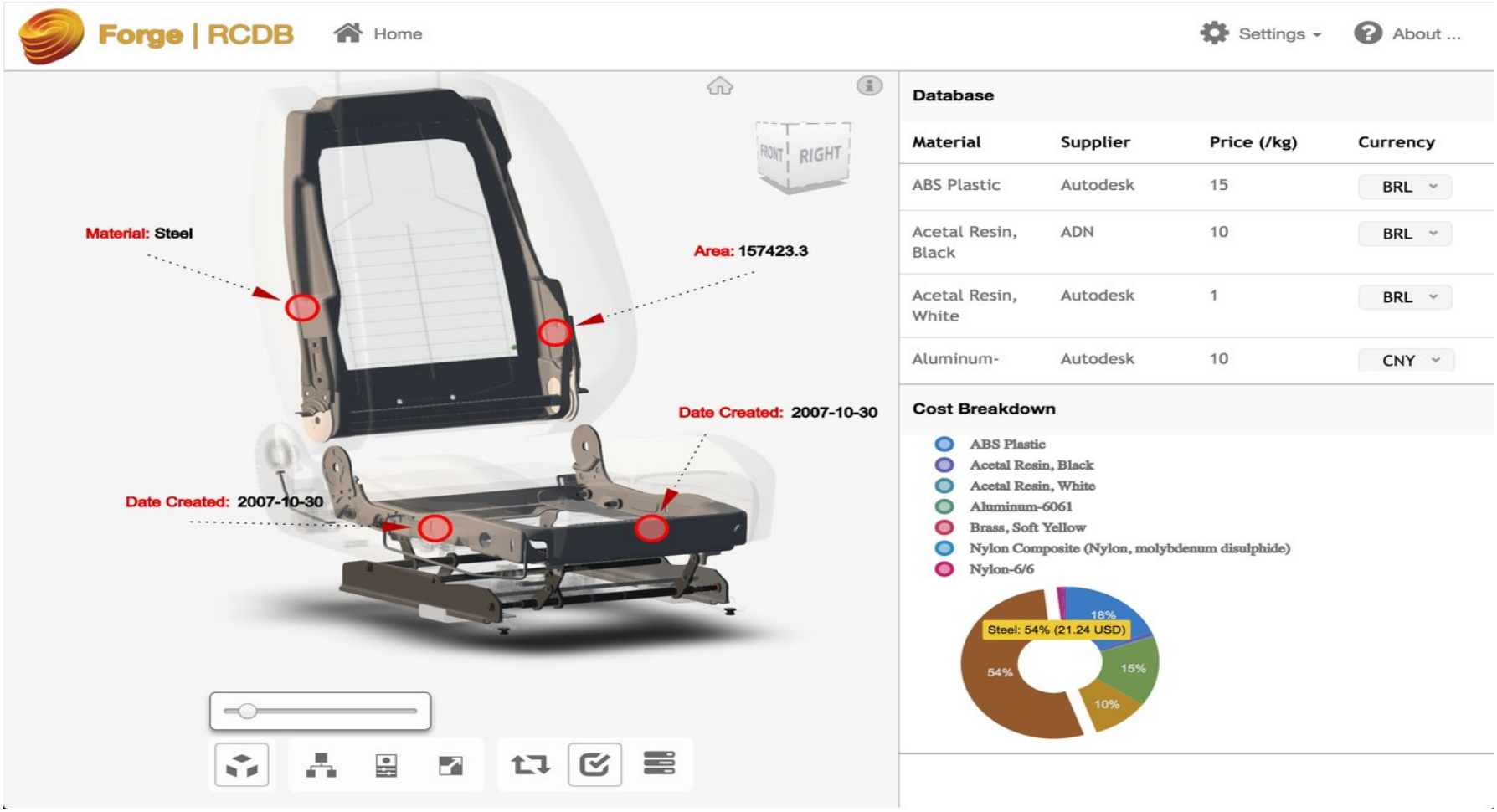
```
1  var cssRenderer = new THREE.CSS3DRenderer()
2
3  viewer.container.appendChild(
4    cssRenderer.domElement)
5
6  var glScene = new THREE.Scene()
7
8  var iFrame = document.createElement('iframe')
9
10 var cssObj = new THREE.CSS3DObject(iFrame)
11
12 cssObj.position.set(x, y, z)
13
14 cssObj.scale.set(sx, sy, sz)
15
16 glScene.add(cssObj)
```

Connecting your App to the Cloud

REST API - Node.Js

```
1  function getItem(id, onSuccess, onError) {
2
3    fetch('/api/items/' + id).then(function (response) {
4
5      response.json(function (item) {
6
7      onSuccess(item)
8      })
9
10   }, function (error) {
11
12     onError(error)
13   })
14 }
```

Forge RCDB - https://forge-rcdb.autodesk.io



How did I do?

- Your class feedback is critical. Fill out a **class survey** now.
- Use the AU mobile app or fill out a class survey online.
- Give feedback after each session.
- AU speakers will get feedback in real-time.
- **Your feedback results in better classes and a better AU experience.**

More Questions? Visit the AU Answer Bar

- Seek answers to all of your technical product questions by visiting the **Answer Bar**.
- Open daily from **8am-6pm Tuesday** and **Wednesday**; **8am-4:30pm Thursday**.
- Located outside **Hall C, Level 2**.
- Meet Autodesk developers, testers, & support engineers ready to help with your most challenging technical questions.