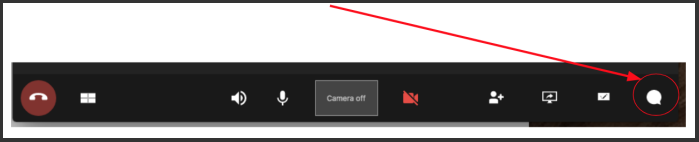
Web-based 3D scientific visualization

ALEX RAZOUMOV
alex.razoumov@westgrid.ca

To ask questions

- Vidyo: use the GROUP CHAT to ask questions



- Please mute your microphone unless you have a question
- Feel free to ask questions via audio at any time
- Websteam: email training@westgrid.ca

Why web visualization?

- ✔ Use it if you
  - want a portable platform: anyone with a browser can load your 3D dataset(s), or
  - want a much simpler/cleaner or more specialized interface than provided by standard desktop tools (ParaView, VisIt), or
  - want a mobile, touch-friendly interface
- ✘ Work with native desktop apps if you want full-featured local visualization
- ✘ Work with native desktop client + remote server if you want to perform 3D rendering of a large dataset on a big remote server or HPC cluster and display results interactively (single user) locally on your laptop
  - faster performance, more functionality, no JavaScript coding

3D "sine envelope wave function" inside a unit cube ($x_i \in [0,1]$) on a 30^3 Cartesian grid

$$f(x_1,x_2,x_3) = \sum_{i=1}^{2} \left[\frac{\sin^2\left(\sqrt{\xi_{i+1}^2 + \xi_i^2}\right) - 0.5}{\left[0.001(\xi_{i+1}^2 + \xi_i^2) + 1\right]^2} + 0.5 \right], \text{ where } \xi_i \equiv 30(x_i - 0.5)$$

```
from numpy import sin, sqrt, zeros
from tqdm import tqdm

n = 30
data = zeros((n,n,n), dtype=float)
for i in tqdm(range(n)):
    x = 15.*((i+0.5)/float(n)-0.5)
    for j in range(n):
        y = 15.*((j+0.5)/float(n)-0.5)
        for k in range(n):
            z = 15.*((k+0.5)/float(n)-0.5)
            data[i][j][k] = ((sin(sqrt(y*y+x*x)))**2-0.5)/(0.001*(y*y+x*x)+1.)**2 + \
                            ((sin(sqrt(z*z+y*y)))**2-0.5)/(0.001*(z*z+y*y)+1.)**2 + 1.

import pyevtk.hl as hl
hl.imageToVTK('sineEnvelope', pointData={"scalar": data})
```

This will generate `sineEnvelope.vti` (VTK ImageData format)

Open-source (commercially-supported) projects from Kitware, Inc.

- ParaViewWeb JavaScript library
  - covered in our March 2017 webinar `https://westgrid.github.io/trainingMaterials/tools/visualization`
  - few pre-built apps to demo its capabilities
  - learning curve to develop your own apps
- vtk.js JavaScript library
  - JavaScript API for many (not all) VTK classes
  - learning curve, but fairly easy to get started
- ParaView Glance is a web app for sharing pre-built 3D scenes on the web
  - the easiest, no programming required to use the base app

ParaViewWeb

- Lightweight JavaScript API for writing client-style HTML5 web applications to display 3D interactive visualizations in a web browser
  - open-source project from Kitware, Inc.
- Most PVW applications use a remote ParaView backend to process and render data
  - a handful of prebuilt applications available
  - the most complete app is Visualizer, providing most of ParaView Qt desktop application features within a web browser
  - in principle, can build your own apps
  - source `https://github.com/Kitware/paraviewweb`
- Small 3D geometry can be rendered locally on the client using WebGL
- PVW's core and several apps normally included with pre-compiled ParaView but their source codes hosted in separate repos

ParaViewWeb applications

- **Visualizer** provides an experience inside the browser very similar to the ParaView Qt desktop application, example of what can be built with ParaViewWeb

  `https://github.com/kitware/visualizer`

  `https://kitware.github.io/visualizer/docs`

- **LightViz** provides simpler, more intuitive visualization

  `https://github.com/kitware/light-viz`

  `https://kitware.github.io/light-viz/docs`

- **ArcticViewer** is a standalone (no PV server needed) JavaScript viewer for Cinema- or Catalyst- pregenerated images

  `https://kitware.github.io/arctic-viewer`

- Few other in development at Kitware

- Theoretically anyone can write their own (JavaScript)

Running Visualizer

Testing in single-user mode on a laptop:

- Two ways to start, both wait for incoming traffic on port 8080:
  (1) either a Python ParaViewWeb server application (serves Visualizer connected to ParaView)
  - ➥ included in a precompiled ParaView binary: (1) Python PVW server app `pvw-visualizer.py` and (2) static HTML content directory `web/visualizer/www` with Visualizer JS code inside
  - ➥ instructions for Linux and Windows at `https://kitware.github.io/visualizer/docs`

```
$ cd /Applications/ParaView-5.7.0.app/Contents     # 'start 1' on presenter's laptop
$ ./bin/pvpython Resources/web/visualizer/server/pvw-visualizer.py \
                 --content Resources/web/visualizer/www \
                 --data ~/talks/2017/03-pvweb/data --port 8080
$ open http://localhost:8080
```

(2) a standalone JavaScript Visualizer app (in Node.js runtime environment)

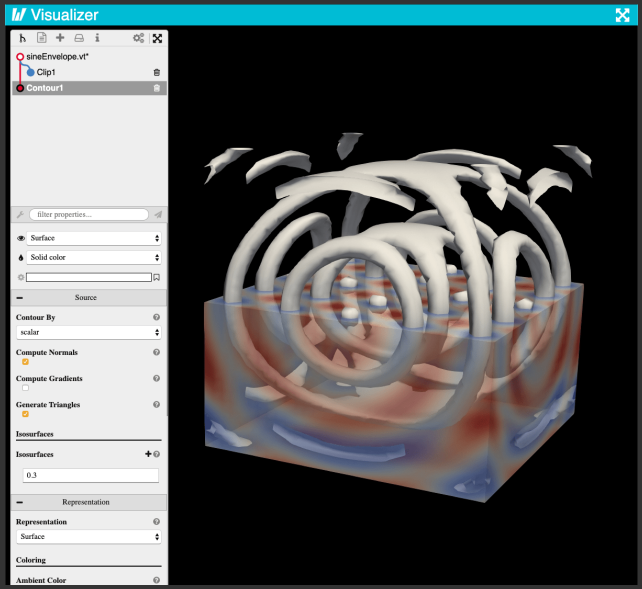
```
$ sudo npm install -g pvw-visualizer   # installs it into /usr/local/lib/node_modules/pvw-visualizer
                                       # and creates a symbolic link /usr/local/bin/Visualizer
$ Visualizer --paraview /Applications/ParaView-5.7.0.app --data ~/talks/2017/03-pvweb/data
```

Multi-user deployment on a production website:

- Configure a PVW launcher and a virtual host on your Apache server (steps detailed in our 2017 webinar)

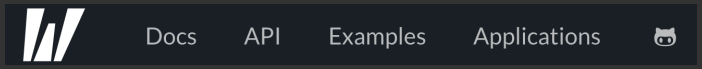
Visualizer GUI

- Main UI elements: toolbar at the top lets you show the pipeline browser, browse files, add elements (filters and objects), save screenshots and states, get dataset info
- Can hide the left panel entirely by clicking on the cyan Visualizer logo
- Controls very similar to ParaView's Properties; the Apply button is 
- Same mouse navigation as in ParaView
- To be able to load NetCDF, compile the backend PV server with NetCDF support, launch the PVW Visualizer server app with a proxy file `pvw-visualizer.py --proxies proxies.json` to define the reader based on the file extension
- VTK files load directly



Writing your own ParaViewWeb apps

- Is PVW right for you?
  - is your goal remote scientific visualization? ⇒ use client-server or batch offscreen visualization
  - do you want to simply share 3D models online? ⇒ use ParaView Glance, 3DHOP or a sharing platform such as `https://sketchfab.com`
- Use PVW to write a custom web app that talks to a remote ParaView server
  - it is JavaScript ⇒ steep learning curve if you've never coded in it
- Main resource `http://kitware.github.io/paraviewweb`



- Can play with Visualizer, LightViz, ArcticViewer apps (hosted in separate repos, linked from Applications)

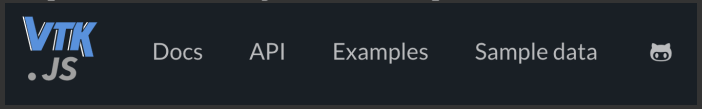
1. Let me know the application/functionality you have in mind, or
2. Talk directly to Kitware `https://www.kitware.com`, they'll be happy to develop apps for you (and please keep me in the loop)

VTK = Visualization Toolkit

- Software for 3D computer graphics, image processing, volume rendering, and scientific visualization
- In development since the early 1990s
- Open-source, multi-platform: Linux, Windows, Mac, the Web and mobile devices
- Core functionality written in C++, wrapped into other language bindings: Tcl, Python, Java
- Sits on top of a graphics library (typically OpenGL)
- Distributed-memory parallel processing via MPI
- Many-core and GPU architecture support via VTK-m (separate code base)

VTK.js

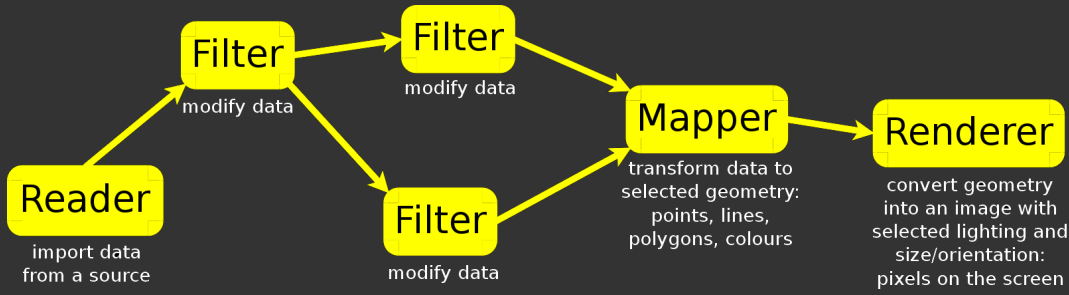
- Open-source ES6 JavaScript class library for sci-vis on the web
  - not all VTK classes implemented
  - more comlex applications: vtk.js ES6 code can be integrated into a web application in Node.js environment, typically requires a web server for local testing and for deployment
  - simpler usage: can be directly imported as a script tag inside live HTML pages from a global CDN (content delivery network) such as `https://unpkg.com`
- Uses WebGL (check your browser compatibility `https://get.webgl.org`)
  - WebGL2 for best performance `https://get.webgl.org/webgl2` (Chrome, Firefox)
- Variety of visualization algorithms
- Main resource `http://kitware.github.io/paraviewweb`



  - docs and tutorials assume JavaScript knoweldge and familiarity with browser devtools
  - check code examples under both API and Examples $\Rightarrow$ can run simpler examples inside live HTML pages

Data flow in VTK

https://vtk.org/Wiki/VTK/Tutorials/VTK_Terminology



- Data goes through Mapper which knows how to draw it, places that data into the rendered scene via a VTK Actor
  - `mapper.setInputConnection(object.getOutputPort())`
- Actor is an OpenGL object = the part that is rendered
  - the output of a Mapper: `actor.setMapper(mapper)`
  - passed to a Renderer
- Renderer can hold multiple actors
- RendererWindow (on the screen) can hold multiple renderers

Basic example: render a cone (`cone.html`)

```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript" src="https://unpkg.com/vtk.js"></script>
    <script type="text/javascript">

      // create a basic cone object
      var cone = vtk.Filters.Sources.vtkConeSource.newInstance();
      cone.setRadius(0.3);
      cone.setResolution(50);

      // map polygonal data into renderable geometry
      var coneMapper = vtk.Rendering.Core.vtkMapper.newInstance();
      coneMapper.setInputConnection(cone.getOutputPort());

      // create an OpenGL object
      var coneActor = vtk.Rendering.Core.vtkActor.newInstance();
      coneActor.setMapper(coneMapper);
      coneActor.getProperty().setEdgeVisibility(true);

      // create a full-webpage renderer
      var fullScreenRenderer = vtk.Rendering.Misc.vtkFullScreenRenderWindow.newInstance();

      // from which you create a renderer itself
      var renderer = fullScreenRenderer.getRenderer();
      renderer.addActor(coneActor);
      renderer.resetCamera();

      // and a render window
      var renderWindow = fullScreenRenderer.getRenderWindow();
      renderWindow.render();

    </script>
  </body>
</html>
```

Add a sphere (conesphere.html)

```
diff cone.html conesphere.html
```

```
    10a11,15
>       var sphere = vtk.Filters.Sources.vtkSphereSource.newInstance();
>       sphere.setRadius(0.3);
>       sphere.setThetaResolution(50);
>       sphere.setPhiResolution(50);
>       sphere.setCenter([0.8, 0, 0]);
14a20,21
>       var sphereMapper = vtk.Rendering.Core.vtkMapper.newInstance();
>       sphereMapper.setInputConnection(sphere.getOutputPort());
19c26,27
<       coneActor.getProperty().setEdgeVisibility(true);
---
>       var sphereActor = vtk.Rendering.Core.vtkActor.newInstance();
>       sphereActor.setMapper(sphereMapper);
26a35
>       renderer.addActor(sphereActor);
```

Add glyphs (`glyphs.html`)

```
diff cone.html glyphs.html
```

```
10a11,14
>       var glyph = vtk.Filters.Sources.vtkSphereSource.newInstance();
>       glyph.setRadius(0.015);
>       glyph.setThetaResolution(30);
>       glyph.setPhiResolution(30);
14a19,21
>       var glyphMapper = vtk.Rendering.Core.vtkGlyph3DMapper.newInstance(); // special mapper with 2 connect
>       glyphMapper.setInputConnection(cone.getOutputPort(), 0); // cone output goes to input port 0
>       glyphMapper.setInputConnection(glyph.getOutputPort(), 1); // glyph output goes to input port 1
19a27,28
>       var glyphActor = vtk.Rendering.Core.vtkActor.newInstance();
>       glyphActor.setMapper(glyphMapper);
26a36
>       renderer.addActor(glyphActor);
```

Readers

https://kitware.github.io/vtk-js/examples

PolyDataReader	HttpDataSetReader	PDBReader
XMLImageDataReader	HttpSceneLoader	ImageStream
OBJReader	STLReader	DracoReader
ZipHttpReader (json metadata + binary data files in ZIP format)	ElevationReader	JSONNucleoReader
	JSONNucleoReader	...

- In Node.js can include local files into your web app during build
- In live HTML pages can (1) load data files from public URLs and (2) drop your files into the page

vtkPDBReader (pdb.html)

https://kitware.github.io/vtk-js/examples/PDBReader.html

Dataset from VMD tutorials

```
const reader = vtk.IO.Misc.vtkPDBReader.newInstance();

const filter = vtk.Filters.General.vtkMoleculeToRepresentation.newInstance();
filter.setInputConnection(reader.getOutputPort());
filter.setHideElements(['O']); // also try H, N

const sphereMapper = vtk.Rendering.Core.vtkSphereMapper.newInstance();
sphereMapper.setInputConnection(filter.getOutputPort(0));
sphereMapper.setScaleArray(filter.getSphereScaleArrayName());
const stickMapper = vtk.Rendering.Core.vtkStickMapper.newInstance();
stickMapper.setInputConnection(filter.getOutputPort(1));
stickMapper.setScaleArray('stickScales');
stickMapper.setOrientationArray('orientation');

const sphereActor = vtk.Rendering.Core.vtkActor.newInstance();
sphereActor.setMapper(sphereMapper);
const stickActor = vtk.Rendering.Core.vtkActor.newInstance();
stickActor.setMapper(stickMapper);

const fullScreenRenderer=vtk.Rendering.Misc.vtkFullScreenRenderWindow.newInstance({background:[0,0.2,0.2]});
const renderer = fullScreenRenderer.getRenderer();
const renderWindow = fullScreenRenderer.getRenderWindow();

renderer.addActor(sphereActor);
renderer.addActor(stickActor);

reader.setUrl(`https://raw.githubusercontent.com/razoumov/publish/master/data/1lda.pdb`).then(() => {
    renderer.resetCamera();
    renderWindow.render();
});
```

vtkXMLImageDataReader (`xml.html`)

```
const fullScreenRenderer = vtk.Rendering.Misc.vtkFullScreenRenderWindow.newInstance({background:[0,0,0]});
const renderer = fullScreenRenderer.getRenderer();
const renderWindow = fullScreenRenderer.getRenderWindow();

const reader = vtk.IO.XML.vtkXMLImageDataReader.newInstance();

const mapper = vtk.Rendering.Core.vtkVolumeMapper.newInstance();
mapper.setInputConnection(reader.getOutputPort());

const actor = vtk.Rendering.Core.vtkVolume.newInstance();
actor.setMapper(mapper);

const ctfun = vtk.Rendering.Core.vtkColorTransferFunction.newInstance(); // color transfer function
ctfun.addRGBPoint(100.0, 0.1, 0, 0.9); // blue
ctfun.addRGBPoint(1500.0, 0.1, 0.9, 0); // green
actor.getProperty().setRGBTransferFunction(0, ctfun);
const ofun = vtk.Common.DataModel.vtkPiecewiseFunction.newInstance(); // opacity transfer function
ofun.addPoint(100.0, 0.9);     ofun.addPoint(387., 0.1);     ofun.addPoint(1500.0, 0.3);
actor.getProperty().setScalarOpacity(0, ofun);
actor.getProperty().setScalarOpacityUnitDistance(0, 4.5);
actor.getProperty().setInterpolationTypeToLinear();
actor.getProperty().setUseGradientOpacity(0, true);
actor.getProperty().setShade(true);
actor.getProperty().setAmbient(0.5);
actor.getProperty().setDiffuse(0.7);

reader.setUrl(`https://raw.githubusercontent.com/razoumov/publish/master/data/integerEnvelope.vti`).then(() => {
    reader.loadData().then(() => {
        renderer.addVolume(actor);
        renderer.resetCamera();
        renderer.getActiveCamera().zoom(1.5);
        renderer.getActiveCamera().elevation(70);
        renderer.updateLightsGeometryToFollowCamera();
        renderWindow.render();
    });
});
```

SceneExplorer

https://kitware.github.io/vtk-js/examples/SceneExplorer.html

- Drop `sineEnvelope.vtkjs` onto it
- Press “c” for menu (if available)
- Reload, drop `StanfordDragon.vtkjs` onto it (dataset linked from the page above)

VolumeViewer

https://kitware.github.io/vtk-js/examples/VolumeViewer.html

- Drop `headsq.vti` onto it
- Drop `~/Movies/publish/data/integerEnvelope.vti`
  - in the header I had to add Scalars="density" to <PointData ...> tag
- Quite sensitive to the VTI file format (could not read my **binary** `sineEnvelope.vti`)
- Edit the opacity transfer function (instructions in the page)

ParaView Glance

https://kitware.github.io/paraview-glance/app

ParaView Glance is an open-source web app for in-browser 3D visualization

- up to medium-size data
- interactive manipulation of pre-computed polygons
  - volumetric images, molecular structures, geometric objects, point clouds
- written in JavaScript and vtk.js + can be further customized with vtk.js and ParaViewWeb for custom web and desktop apps
- source and installation instructions https://github.com/kitware/paraview-glance

---

1. Create a visualization with several layers, make **all layers visible in the pipeline**
2. Many options in File → Export Scene... ⇒ save as VTKJS to your laptop
3. Open https://kitware.github.io/paraview-glance/app
4. Also running the app on an Arbutus VM http://206.12.92.61:9999
5. Drag the newly saved file to the dropzone on the website
6. Interact with individual layers in 3D: rotate and zoom, change visibility, representation, variable, colourmap, opacity

Automatically load a visualisation into Glance

https://discourse.paraview.org/t/customise-pv-glance/2831

- Use syntax `GLANCEAPPURL?name=FILENAME&url=FILEURL`
- E.g. using ParaView Glance website
  `https://kitware.github.io/paraview-glance/app?name=sineEnvelope.vtkjs&url=https://raw.githubusercontent.com/razoumov/publish/master/data/sineEnvelope.vtkjs`
  - shortened to `https://bit.ly/2KtPWNf`
- Using the app on the Arbutus VM
  `http://206.12.92.61:9999?name=sineEnvelope.vtkjs&url=https://raw.githubusercontent.com/razoumov/publish/master/data/sineEnvelope.vtkjs`
  - shortened to `https://bit.ly/3eZDfIh`
- You can parse long strings with JavaScript (forward two slides)

Automatically load multiple files into Glance

- Use syntax `GLANCEAPPURL?name=[FILENAME1,FILENAME2]&url=[FILEURL1,FILEURL2]`
- Using ParaView Glance website
  `https://kitware.github.io/paraview-glance/app?name=[sineEnvelope.vtkjs,secondclip.vtkjs]&url=[https://raw.githubusercontent.com/razoumov/publish/master/data/sineEnvelope.vtkjs,https://raw.githubusercontent.com/razoumov/publish/master/data/secondclip.vtkjs]`
  - Shortened to `https://bit.ly/3asYGOq`
- On the Arbutus VM `http://206.12.92.61:9999?name=[sineEnvelope.vtkjs,secondclip.vtkjs]&url=[https://raw.githubusercontent.com/razoumov/publish/master/data/sineEnvelope.vtkjs,https://raw.githubusercontent.com/razoumov/publish/master/data/secondclip.vtkjs]`
  - shortened to `https://bit.ly/2VJBJSN`

Embed your vis into a website with an iframe (`embed.html`)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sine envelope function</title>
  </head>
  <body>

    <h1>3D sine envelope function</h1>

    <script>
      var app = "https://kitware.github.io/paraview-glance/app";
      var datadir = "https://raw.githubusercontent.com/razoumov/publish/master/data/";
      var file = "sineEnvelope.vtkjs";
      document.write("<iframe src='" + app + "?name=" + file + "&url=" +
                     datadir + file + "' id='iframe' width='1100' height='900'></iframe>");
    </script>

    <p>More stuff in here</p>

  </body>
</html>
```

- JavaScript here only to parse long strings

Multiple iframes (`double.html`)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sine envelope function</title>
  </head>
  <body>
    <h1>3D sine envelope function</h1>

    <script>
      var app = "https://kitware.github.io/paraview-glance/app";
      var datadir = "https://raw.githubusercontent.com/razoumov/publish/master/data/";
      var file1 = "sineEnvelope.vtkjs";
      document.write("<iframe src='" + app + "?name=" + file1 + "&url=" +
                     datadir + file1 + "' id='iframe' width='550' height='900'></iframe>");
      var file2 = "secondclip.vtkjs";
      document.write("<iframe src='" + app + "?name=" + file2 + "&url=" +
                     datadir + file2 + "' id='iframe' width='550' height='900'></iframe>");
    </script>

    <p>More stuff in here</p>
  </body>
</html>
```

- JavaScript here only to parse long strings

Build ParaView Glance on your own machine

```
$ git clone https://github.com/Kitware/paraview-glance.git glance
$ cd glance
$ git tag -l      # show tags (releases)
$ git checkout tags/v4.9.0 -b v4.9.0   # latest 4.9.4 did not work for me

$ npm install      # install the dependencies into ./node_modules
$ npm run build    # build the package
$ unset HOST       # required on my Mac
$ npm run dev      # start the dev server, wait ~30-60 seconds until bundle finished
$ open http://localhost:9999      # open the app

$ npm run build:release    # final bundle and assets to dist/
$ open dist/index.html     # if opened this way, the sample gallery data won't load

$ cp /path/to/sineEnvelope.vtkjs dist/
```

1. Type `start 2` on presenter's laptop to start local ParaView Glance dev server
2. Click on either:
   - `http://localhost:9999` ☞ click on any vis in the gallery
   - `http://localhost:9999?name=sineEnvelope.vtkjs&url=http://localhost:9999/sineEnvelope.vtkjs` (will automatically load your dataset)

Hide the landing page

1. `cp dist/index.html dist/noLandingPage.html`
2. Edit `dist/noLandingPage.html`:
   - add '`glanceInstance.showApp();`' before before loading the dataset ('`glanceInstance.processURLArgs();`')
3. `unset HOST && npm run dev  # wait until bundle finished`
4. `http://localhost:9999/noLandingPage.html?name=sineEnvelope.vtkjs&url=http://localhost:9999/sineEnvelope.vtkjs`

Real scientific visualization

1. `cp /path/to/initialTimeScene.vtkjs dist/`
2. `unset HOST && npm run dev    # wait until bundle finished`
3. `http://localhost:9999/noLandingPage.html?name=initialTimeScene.vtkjs&url=http://localhost:9999/initialTimeScene.vtkjs`

Summary

Questions?

- ParaViewWeb JavaScript library
  - requires a ParaView server
  - the most complete PVW app is Visualizer: most of ParaView Qt desktop application features within a web browser
  - can develop your own apps
- vtk.js JavaScript library
  - no server $\Rightarrow$ up to medium-size data
  - follows the general design principles of VTK
  - not all VTK classes implemented
- ParaView Glance open-source web app for in-browser 3D visualization
  - no server $\Rightarrow$ up to medium-size data
  - server support in future versions
  - the easiest, no programming required to use the base app
  - ideal for sharing pre-built 3D scenes via the web