Using vtk.js as an ES6 dependency

This guide illustrates how to build an application using vtk.js as a dependency using a modern toolsets such as Webpack, NPM.

(#Creation-of-the-Project-structure)Creation of the Project structure

```
$ mkdir MyWebProject
$ cd MyWebProject
$ npm init

  This utility will walk you through creating a package.json file.
  It only covers the most common items, and tries to guess sensible defaults.

  See `npm help json` for definitive documentation on these fields
  and exactly what they do.

  Use `npm install <pkg> --save` afterwards to install a package and
  save it as a dependency in the package.json file.

  Press ^C at any time to quit.
  name: (MyWebProject) your-project-name
  version: (1.0.0) 0.0.1
  description: vtk.js application
  entry point: (index.js) src/index.js
  test command:
  git repository:
  keywords: Web visualization using VTK
  author:
  license: (ISC) BSD-2-Clause
  About to write to /.../MyWebProject/package.json:

  {
    "name": "your-project-name",
    "version": "0.0.1",
    "description": "vtk.js application",
    "main": "src/index.js",
    "scripts": {
      "test": "echo \"Error: no test specified\" && exit 1"
    },
    "keywords": [
      "Web",
      "visualization",
      "using",
      "VTK"
    ],
    "author": "",
    "license": "BSD-2-Clause"
  }


  Is this ok? (yes)
```

Then install and save your dependencies

```
$ npm install vtk.js --save
$ npm install kw-web-suite --save-dev
```

(#Webpack-config)Webpack config

webpack.config.js

```
var path = require('path');
var webpack = require('webpack');
var vtkRules = require('vtk.js/Utilities/config/dependency.js').webpack.v2.rules;

var entry = path.join(__dirname, './src/index.js');
const sourcePath = path.join(__dirname, './src');
const outputPath = path.join(__dirname, './dist');

module.exports = {
  entry,
  output: {
    path: outputPath,
    filename: 'MyWebApp.js',
  },
  module: {
    rules: [
        { test: entry, loader: "expose-loader?MyWebApp" },
        { test: /\.html$/, loader: 'html-loader' },
    ].concat(vtkRules),
  },
  resolve: {
    modules: [
      path.resolve(__dirname, 'node_modules'),
```

```
      sourcePath,
    ],
  },
};
```

(#package-json)package.json

You should extend the generated **package.json** file with the following set of scripts.

package.json

```
{
  [...],
  "scripts": {
    "build": "webpack --progress --colors --mode development",
    "build:release": "webpack --progress --colors --mode production",
    "start": "webpack-dev-server --content-base ./dist",

    "commit": "git cz",
    "semantic-release": "semantic-release"
  }
}
```

(#Application)Application

Here is an example of a vtk.js application similar to an example (https://kitware.github.io/vtk-js/examples/ConeSource.html) available within the source code.

src/index.js

```
import vtkFullScreenRenderWindow from 'vtk.js/Sources/Rendering/Misc/FullScreenRenderWindow';

import vtkActor           from 'vtk.js/Sources/Rendering/Core/Actor';
import vtkCalculator      from 'vtk.js/Sources/Filters/General/Calculator';
import vtkConeSource      from 'vtk.js/Sources/Filters/Sources/ConeSource';
import vtkMapper          from 'vtk.js/Sources/Rendering/Core/Mapper';
import { AttributeTypes } from 'vtk.js/Sources/Common/DataModel/DataSetAttributes/Constants';
import { FieldDataTypes } from 'vtk.js/Sources/Common/DataModel/DataSet/Constants';

import controlPanel from './controller.html';

// ----------------------------------------------------------------------------
// Standard rendering code setup
// ----------------------------------------------------------------------------

const fullScreenRenderer = vtkFullScreenRenderWindow.newInstance();
const renderer = fullScreenRenderer.getRenderer();
const renderWindow = fullScreenRenderer.getRenderWindow();

// ----------------------------------------------------------------------------
// Example code
// ----------------------------------------------------------------------------

const coneSource = vtkConeSource.newInstance({ height: 1.0 });
const filter = vtkCalculator.newInstance();

filter.setInputConnection(coneSource.getOutputPort());
filter.setFormula({
  getArrays: inputDataSets => ({
    input: [],
    output: [
      { location: FieldDataTypes.CELL, name: 'Random', dataType: 'Float32Array', attribute: AttributeTypes.SCALARS },
    ],
  }),
  evaluate: (arraysIn, arraysOut) => {
    const [scalars] = arraysOut.map(d => d.getData());
    for (let i = 0; i < scalars.length; i++) {
      scalars[i] = Math.random();
    }
  },
});

const mapper = vtkMapper.newInstance();
mapper.setInputConnection(filter.getOutputPort());

const actor = vtkActor.newInstance();
actor.setMapper(mapper);

renderer.addActor(actor);
renderer.resetCamera();
renderWindow.render();

// ----------------------------------------------------------
// UI control handling
// ----------------------------------------------------------

fullScreenRenderer.addController(controlPanel);
const representationSelector = document.querySelector('.representations');
```

```
const resolutionChange = document.querySelector('.resolution');

representationSelector.addEventListener('change', (e) => {
  const newRepValue = Number(e.target.value);
  actor.getProperty().setRepresentation(newRepValue);
  renderWindow.render();
});

resolutionChange.addEventListener('input', (e) => {
  const resolution = Number(e.target.value);
  coneSource.setResolution(resolution);
  renderWindow.render();
});
```

The control template:

src/controller.html

```
<table>
  <tr>
    <td>
      <select class='representations' style="width: 100%">
        <option value='0'>Points</option>
        <option value='1'>Wireframe</option>
        <option value='2' selected>Surface</option>
      </select>
    </td>
  </tr>
  <tr>
    <td>
      <input class='resolution' type='range' min='4' max='80' value='6' />
    </td>
  </tr>
</table>
```

The main web page loading the generated application.

dist/index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1">
  </head>
  <body>
    <script type="text/javascript" src="MyWebApp.js"></script>
  </body>
</html>
```

(#Build-the-application)Build the application

```
$ npm run build
```

(#Start-a-dev-WebServer)Start a dev WebServer

```
$ npm start
```

Open your browser at `http://localhost:8080/`