Introduction to itk.js

Matthew McCormick
Alexis Girault

bit.ly/itkjsKitwareBiomedical2019

Goals

Overview

itk.js combines Emscripten and ITK to enable high-performance spatial analysis in a JavaScript runtime environment.

The project provides tools to a) build C/C++ code to JavaScript (asm.js) and WebAssembly, b) bridge local filesystems, native JavaScript data structures, and traditional file formats, c) transfer data efficiently in and out of the Emscripten runtime, and d) asynchronously execute processing pipelines in a background thread. *itk.js* can be used to execute ITK, VTK or arbitrary C++ codes in the browser or on a workstation / server with Node.js.

Last updated: 2019-02-05

NEXT >

http://insightsoftwareconsortium.github.io/itk-js/docs/

Kitware

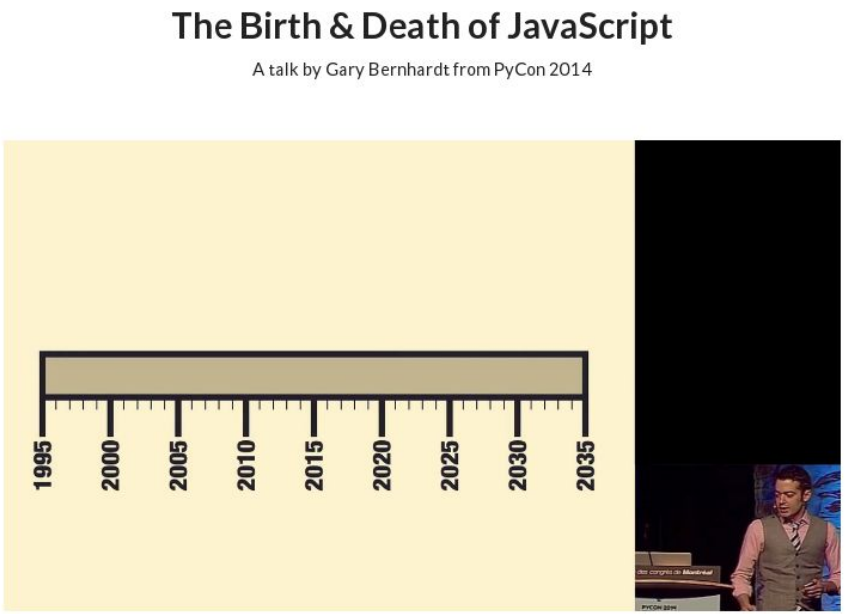
itk.js and vtk.js

- **vtk.js** is a web browser visualization library that complements image analysis with **itk.js**
- For an Introduction to vtk.js, see

bit.ly/vtkjsKitwareBiomedical2019

Kitware

Towards an Internet, JavaScript World: Timeline



1995 - Brendan Eich write JavaScript at Netscape

2008 - ECMAScript 5

2012 - Emscripten initial release

2014 - DCMTK compiled to Emscripten

2015 - ITK compiled with Emscripten, itk.js started

2015 - ECMAScript 6

2015 - WebAssembly Community Group

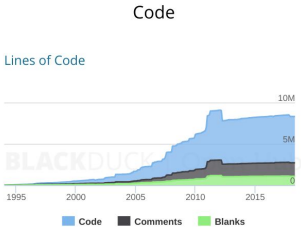
2017 March - WebAssembly cross-browser consensus

2018 January VTK compiled with Emscripten

Kitware

How do we use ITK and VTK in modern platforms?

- Move to **The Web**
- How do we move to The Web?
- Bring ITK and VTK's *7.3 million lines* of **C++ code** developed over *24 years by over 800 people* **to the browser**



In a Nutshell, Visualization Toolkit...

... has had 67,314 commits made by 463 contributors
representing 5,579,537 lines of code

... is mostly written in C++
with an average number of source code comments

... has a well established, mature codebase
maintained by a very large development team
with stable Y-O-Y commits

... took an estimated 1,705 years of effort (COCOMO model)
starting with its first commit in January, 1994
ending with its most recent commit 1 day ago

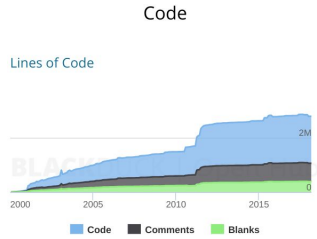
In a Nutshell, Insight Toolkit...

... has had 52,698 commits made by 335 contributors
representing 1,733,715 lines of code

... is mostly written in C++
with a well-commented source code

... has a well established, mature codebase
maintained by a very large development team
with stable Y-O-Y commits

... took an estimated 497 years of effort (COCOMO model)
starting with its first commit in January, 2000
ending with its most recent commit about 2 months ago



Kitware

What is Emscripten?

- **Emscripten** is a toolchain for compiling to **asm.js** and **WebAssembly**, built using LLVM, that lets you run C and C++ on the web at near-native speed, without plugins.
- **asm.js** is an extraordinarily optimizable, low-level subset of JavaScript that is compatible with all web browsers with JavaScript support.
- **WebAssembly** is a binary instruction format for a stack-based virtual machine. It is a portable target that enables deployment on the web for client and server applications.

Kitware

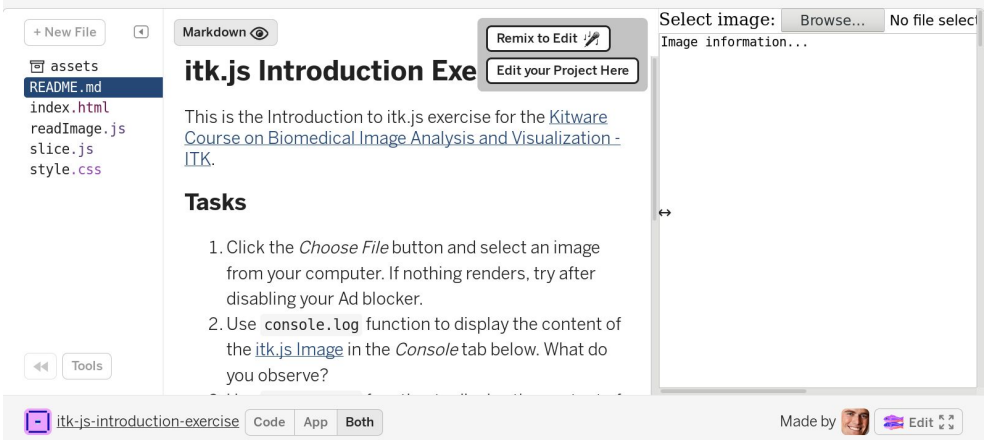
What is itk.js?

- **itk.js** combines Emscripten, ITK, and VTK to enable high-performance spatial analysis in a JavaScript runtime environment.
- **itk.js** provides tools to
  - Build C/C++ code to JavaScript (asm.js) and WebAssembly
  - Bridge local filesystems, native JavaScript data structures, and traditional file formats
    - Special support for vtk.js / itk.js JavaScript *Image*, *Mesh*, and *PolyData* data structures
  - Transfer data efficiently in and out of the Emscripten runtime, and
  - Asynchronously execute processing pipelines in a background thread.
  - Execute arbitrary C++ processing pipelines configured with CMake

Kitware

Exercise: Read a local image file

- itk.js supports many scientific image file formats.
- Exercise: read an image with itk.js, examine it, and view it with vtk.js.
  - Project: https://glitch.com/~itk-js-introduction-exercise
  - App: https://itk-js-introduction-exercise.glitch.me
  - Code: https://glitch.com/edit/#!/itk-js-introduction-exercise

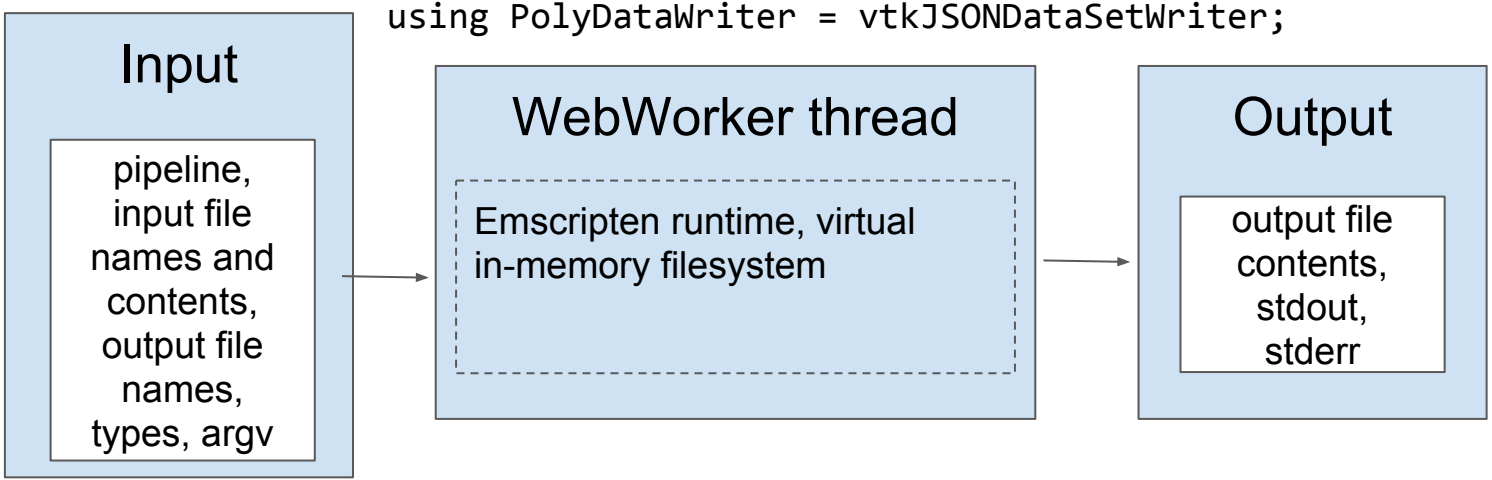


Kitware

What is itk.js, technically?

ProcessingPipeline.cxx:

```
int main(int argc, char * argv[]) {
[...]
using ImageReader = itk::ImageFileReader< ImageType >;
using MeshReader = itk::MeshFileReader< MeshType >;
using ITKVTKConverter = itk::ImageToVTKImageFilter< ImageType >;
using PolyDataWriter = vtkJSONDataSetWriter;
```



input and output file type support

- Text
- Binary
- itk/Image
- itk/Mesh
- vtkPolyData

Kitware

Advice

- **Use itk.js for processing / analysis code in the browser**
  - Better performance
  - Re-use existing code base, development and testing tools
  - Use with vanilla C++, ITK, VTK, and CMake-based library pipelines
- **Use itk.js for ingesting data**
  - 2D / 3D / ND image file formats
  - Geometry file formats
  - DICOM
- **Couple itk.js for processing with vtk.js for visualization**
  - vtk.js *ITKHelper* for using itk.js images in vtk.js
  - itk.js is adding support for generation of vtk.js PolyData

Kitware

When is it a good idea to use Emscripten / itk.js?

- The analysis algorithms are written in **C++**.
- Client-side processing helps with **interactivity**.
- Client-side processing helps with **scalability**.
- HTML/CSS/JavaScript/WebAssembly sites help with **sustainability** and **reproducibility**.

Kitware

When is it a bad idea to use Emscripten / itk.js?

- Computation takes minutes or longer.
  - Use Girder Worker, WSLink, etc.
- The algorithm is written in Python and cannot be converted to C++.
  - * Currently
- The client must operate on gigabytes of data.
  - * Emscripten can help use advanced compression algorithms like Zstandard
- A relatively small static executable cannot be generated

Kitware

What clients are supported?

- Node.js JavaScript runtime
- All major mobile and desktop web browsers
- An ES5 fallback is used when WebAssembly is not available
- Requires WebWorker support, i.e. Edge, Firefox, Chrome, Safari, iOS Safari, Chrome for Android, Samsung Internet
    - Promise polyfill required for Internet Explorer 11

Kitware

How much data can be processed in the browser?

- Only limited by the system's available RAM
  - Gigabytes of data have been processed

How much maintenance burden is required to keep the Emscripten alive?

- Extremely little -- this is the advantage to re-using C++ codebases.
- The same code can be developed and tested on a native OS.
- Most fixes are for correct compiler feature detection for third party libraries.

Kitware

How much JavaScript is required to use it?

- It requires about five lines to execute a pipeline.
- Additional configuration is usually required for WebPack, etc.
- The JavaScript must be able to work with the inputs / outputs
  - ArrayBuffer's
  - File's
  - Blob's
  - JSON
  - PNG, JPEG, BYU, OBJ, ... image and geometry file formats
  - vtk.js PolyData
  - itk.js Image
  - itk.js Mesh

Kitware

Does it work with GPUs?

- No, the only current avenue to GPU computing in the browser is WebGL / shaders, but this is not well suited for general compute.
- A WebGPU standard is in discussion stages.

What is threading support like?

- pthread / C++11 thread support is coming.
- itk.js will allow enabling thread support for browser clients that support it with a graceful single-threaded fallback for clients that do not.

Kitware

Future Steps

- vtk.js PolyData support maturity
- Documentation
  - Try out the Hello World: http://insightsoftwareconsortium.github.io/itk-js/examples/hello_world_node.html
- *npm init [...]* support
  - Help get a project started quickly
- Multi-threaded WebAssembly support
- Extended module API binding / TypeScript

Learn more and get involved:
http://insightsoftwareconsortium.github.io/itk-js/docs/

Enjoy ITK!

Kitware