Worldview is a web mapping tool used for education, research, and disaster response. We consume 600+ Imagery products and support time-critical application areas such as wildfire management, air quality measurements, and flood monitoring.
We have come a long way over the past 7 years. Compare this version to current version
This is worldview in 2013, two years later. Talk about what has changed between 2011 and 2013 and how this version differs from the current state of Worldview.
Give a demo of the current state of Worldview. Discuss issues related to building on top of such a large codebase over a long period of time. Discuss our solutions to these issues. (Detailed example below)

We now have a product picker for browsing through all 600+ layers and an events feature that allows you to browse active events and numerous other features.

When a codebase is maintained for 6+ years by numerous developers, it is going to have it’s problems. We have run into the problem of vast amounts of code being put in places in which it was never intended in attempt to get some new functionality to work. Another issue at hand is that things were tightly coupled across different pieces of the application. The team wanted to take a new approach to the structure of the application, but did not have time to break away and focus all our attention on a redo.

As an open source project, a lot of the team’s brainstorming relates to getting more contributors on board with the project. Worldview is a mapping application. What good is Worldview code going to be for anyone unless he/she wants to visualize their own WMS (Web Map Service) or WMTS (Web Map Tile Service) imagery (soon to include vector imagery)? The solution decided upon was to make another repository of small components that could be used across many disciplines. What we found while coding with the anticipation of others using our code for their own projects was that we were forced to write decoupled code.
Discuss the purpose of the worldview components, which is: getting contributors, creating reusable modules that are non-dependent on the worldview app itself, and using modern development tools.
(Video) An example of Worldview’s Imagery Over the Dateline feature being used to show a tropical storm over Fiji.
(VIDEO) Use NASA’s Scientific Visualization Studio example to show how orbit tracks affect the way GIBS near real-time imagery is populated. Explain why our implementation is better than simply wrapping imagery.
Explain how imagery over the dateline was implemented using OpenLayers layer groups.
Explain the different capabilities of the animation widget: date selector, frame rate, increments, and creating GIFs.
Discuss all the pieces of the animation feature that were able to be isolated into reusable components and discuss precache tiles with promise queues.
How to use the worldview-components:

**Installation**

```
npm install worldview-components
```

**Grab what you need**

```
import {DataSelector, Tooltip} from 'worldview-components';
```

Explain how anyone could use one of the worldview-components for their own project.
Talk about future features and how we would like contributors
THAT’S IT. THANK YOU FOR YOUR TIME!

Visit the Application at:
https://worldview.earthdata.nasa.gov

Contribute to Worldview at:
https://github.com/nasa-gibs/worldview

GIVE THANKS.