Outline of Visualization Code Walk-through:

Front end: Code is in lsst-camera-visualization/frontend/src/js

- Startup
    - **client.js**
- User interaction
    - Command input
        - **Parsing: components/Terminal.js**
    - Displays
        - **React/UI Elements: components/*.js**
        - **util/*.js**
- Firefly interface
    - **util/firefly.js - Interface to all Firefly functions**
    - **components/Viewer.js and util/viewer.js - Our frontend interface to Firefly**
    - **Regions: util/region.js**
- Back end communication
    - Commands
        - **commands/commandDispatcher.js: Dispatches a command, does not dispatch backend tasks; each command will do this on its own.**
        - **Command implementations (average_pixel, create_box, etc): commands/*.js**
    - Information returned
        - **Each backend task uses Firefly.LaunchTask (see util/firefly.js) to start a backend process. The function returns a promise with the backend result in a JavaScript object.**
    - Configuration (*e.g.*, boundaries)
- Display latest
    - **src/js/util/viewer.js** - updateviewer branch

Notes:

- Frontend uses the Redux design pattern for state/data management, so things are spread out between multiple files.
- https://github.com/lsst-camera-visualization/frontend#developer-notes for file system structure
- We use webpack to transpile the code into a minified ES5 file (client.min.js) that runs in the browser

For development:

    Npm run dev
    localhost:8081/dev.html

Back end: Code is in lsst-camera-visualization/backend/src

To setup the python path:

https://github.com/lsst-camera-visualization/lsst_firefly#on-local-machine

- Communication
    - All python backend commands launched by dispatcher.py (firefly needs to be configured to launch external tasks). It converts the JavaScript object (JSON) into python objects for the subsequent scripts and converts results (in python objects) back into JSON format.
        - **Dispatcher.py**
    - Python dependency
        - Numpy
        - Scipy
        - astropy
- Launching
    - **Tasks.py**
        - The frontend converts command parameters into JSON object. **Tasks.py** in the backend invokes corresponding tasks based on the command.
- Image specification
    - Currently the demo displays a CCD-level image. Link to the default image is hardcoded. But user can load other FITS images by using **load_image**.
    - Ideally we would like the header to include information on5 the level (i.e. CCD, raft, focal plane) and the overscan.
- Algorithms
    - Average_pixel
        - **task_scripts/averagePixel.py**
    - Histogram
        - Graph - **task_scripts/graphPixel.py**
        - Graph_projection - **task_scripts/graphProj.py**
        - Graph_noise (Table_noise?) - **task_scripts/graph_noise.py**
    - Show_boundary
        - **task_scripts/boundary.py** - Extract header information into json object and pass it to frontend
    - Hot_pixel
        - **task_scripts/hotPixel.py** - Highlight pixels that exceed specified threshold.
    - Noise
        - **task_scripts/noise.py** - Calculate the second moment
    - Other commands
        - Integration with LSST Stack.