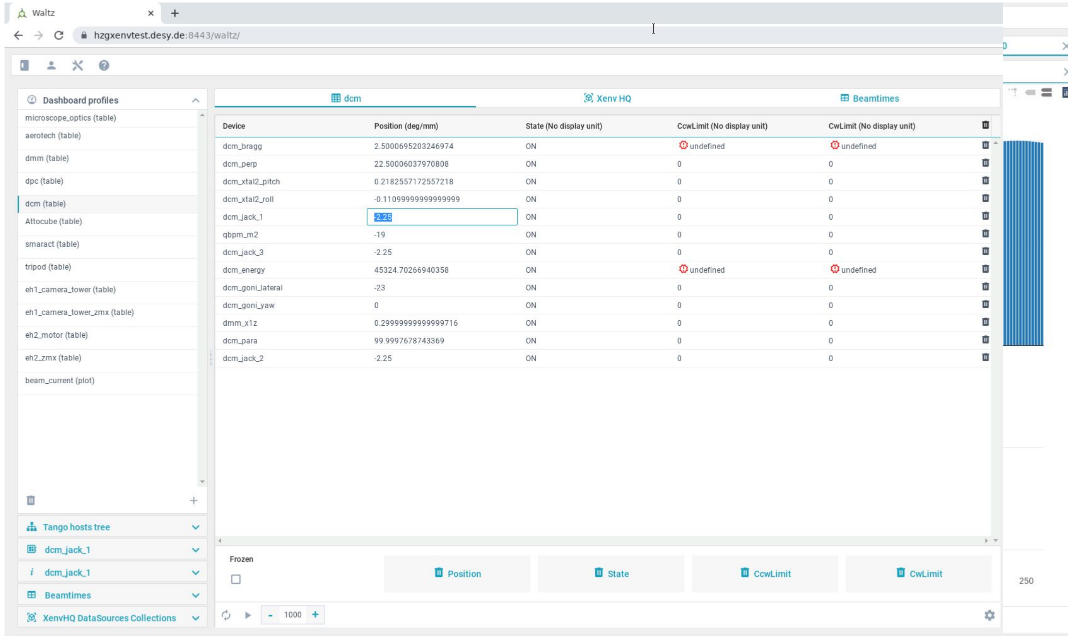
# React-based widgets for Waltz-CS

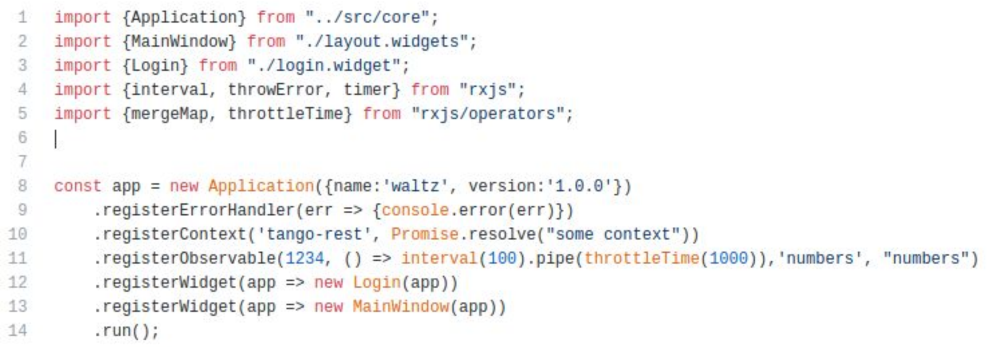
Chernov Vasily, INR RAS

# Waltz is:

1. All-in-one web application like JIVE and ASTOR
    a. ui is oriented for multi device monitoring
    b. control devices
    c. visualize data, save graphs
2. A platform for web based GUIs.
    a. A set of widgets for constructing custom UI
    b. Pluggable architecture
    c. Modern build tools (webpack/rollup)
    d. Middleware connectors to different SCADA (Piazza project)



| Device | Position (deg/mm) | State (No display unit) | CcwLimit (No display unit) | CwLimit (No display unit) |
|---|---|---|---|---|
| dcm_bragg | 2.5000695203246974 | ON | undefined | undefined |
| dcm_perp | 22.50006037970808 | ON | 0 | 0 |
| dcm_xtal2_pitch | 0.2182557172557218 | ON | 0 | 0 |
| dcm_xtal2_roll | -0.11099999999999999 | ON | 0 | 0 |
| dcm_jack_1 | -2.25 | ON | 0 | 0 |
| qbpm_m2 | -19 | ON | 0 | 0 |
| dcm_jack_3 | -2.25 | ON | 0 | 0 |
| dcm_energy | 45324.70266940358 | ON | undefined | undefined |
| dcm_goni_lateral | -23 | ON | 0 | 0 |
| dcm_goni_yaw | 0 | ON | 0 | 0 |
| dmm_x1z | 0.29999999999999716 | ON | 0 | 0 |
| dcm_para | 99.9997678743369 | ON | 0 | 0 |
| dcm_jack_2 | -2.25 | ON | 0 | 0 |



```
import {Application} from "../src/core";
import {MainWindow} from "./layout.widgets";
import {Login} from "./login.widget";
import {interval, throwError, timer} from "rxjs";
import {mergeMap, throttleTime} from "rxjs/operators";


const app = new Application({name:'waltz', version:'1.0.0'})
    .registerErrorHandler(err => {console.error(err)})
    .registerContext('tango-rest', Promise.resolve("some context"))
    .registerObservable(1234, () => interval(100).pipe(throttleTime(1000)),'numbers', "numbers")
    .registerWidget(app => new Login(app))
    .registerWidget(app => new MainWindow(app))
    .run();
```

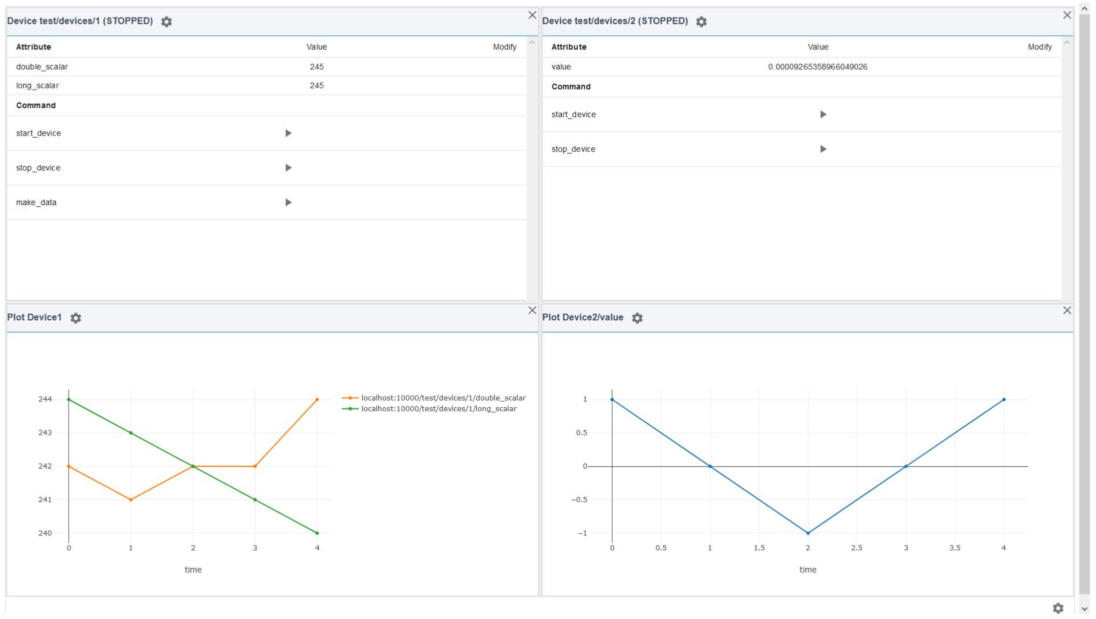
# Waltz GUI has

- Security
- Application + User logs
- Dashboard profiles (per user)
- Table, Plot and List data views
- Drag-n-drop configuration
- Multiple Tango hosts browser
- Search filters
- Tango Manager
- Editable Info panels
- Scripting
- Terminal
- Device filters
- Devices configuration and monitor
- Documentation
- Development platform
- Integration with TINE and EPICS via TANGO
- Unique widgets for unique needs
- Continuous Integration/Continuous Delivery

# Grid Widget

Widget purpose - to build interfaces for multi device monitoring with ability to plot
Widget can:

- provide simple interface tab for device
  - with attributes
  - with clickable void commands
- provide graphs for polled attributes
  - graph can show attributes from different devices
- configure device and graphic tabs
- configure grid geometry and tab color (for better navigation)



Device test/devices/1 (STOPPED)

| Attribute | Value | Modify |
|---|---|---|
| double_scalar | 245 | |
| long_scalar | 245 | |
| **Command** | | |
| start_device | ▶ | |
| stop_device | ▶ | |
| make_data | ▶ | |

Device test/devices/2 (STOPPED)

| Attribute | Value | Modify |
|---|---|---|
| value | 0.00009265358966049026 | |
| **Command** | | |
| start_device | ▶ | |
| stop_device | ▶ | |

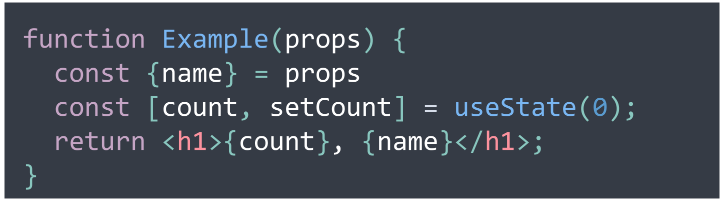
Plot Device1

Plot Device2/value

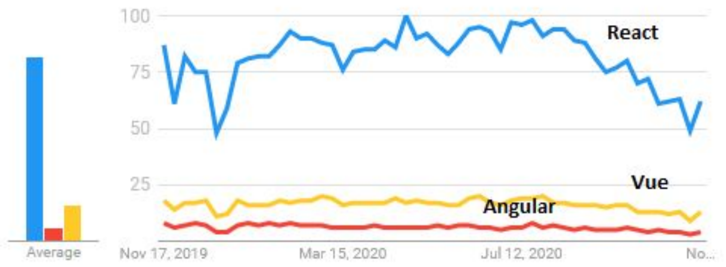
244
243
242
241
240
0
1
2
3
4
time
localhost:10000/test/devices/1/double_scalar
localhost:10000/test/devices/1/long_scalar
1
0.5
0
−0.5
−1
0
0.5
1
1.5
2
2.5
3
3.5
4
time

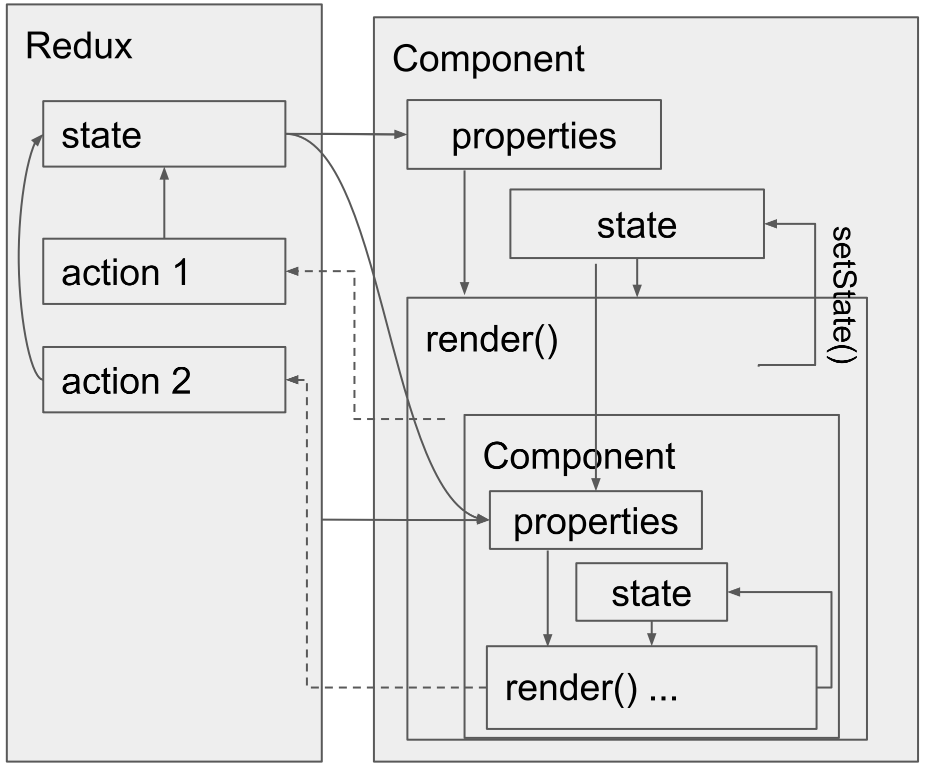
# Quick React overview

- React is the most popular UI framework for JS
- Redux as storage



```
function Example(props) {
  const {name} = props
  const [count, setCount] = useState(0);
  return <h1>{count}, {name}</h1>;
}
```





# Grid Widget for Waltz

**Redux ACTIONS**

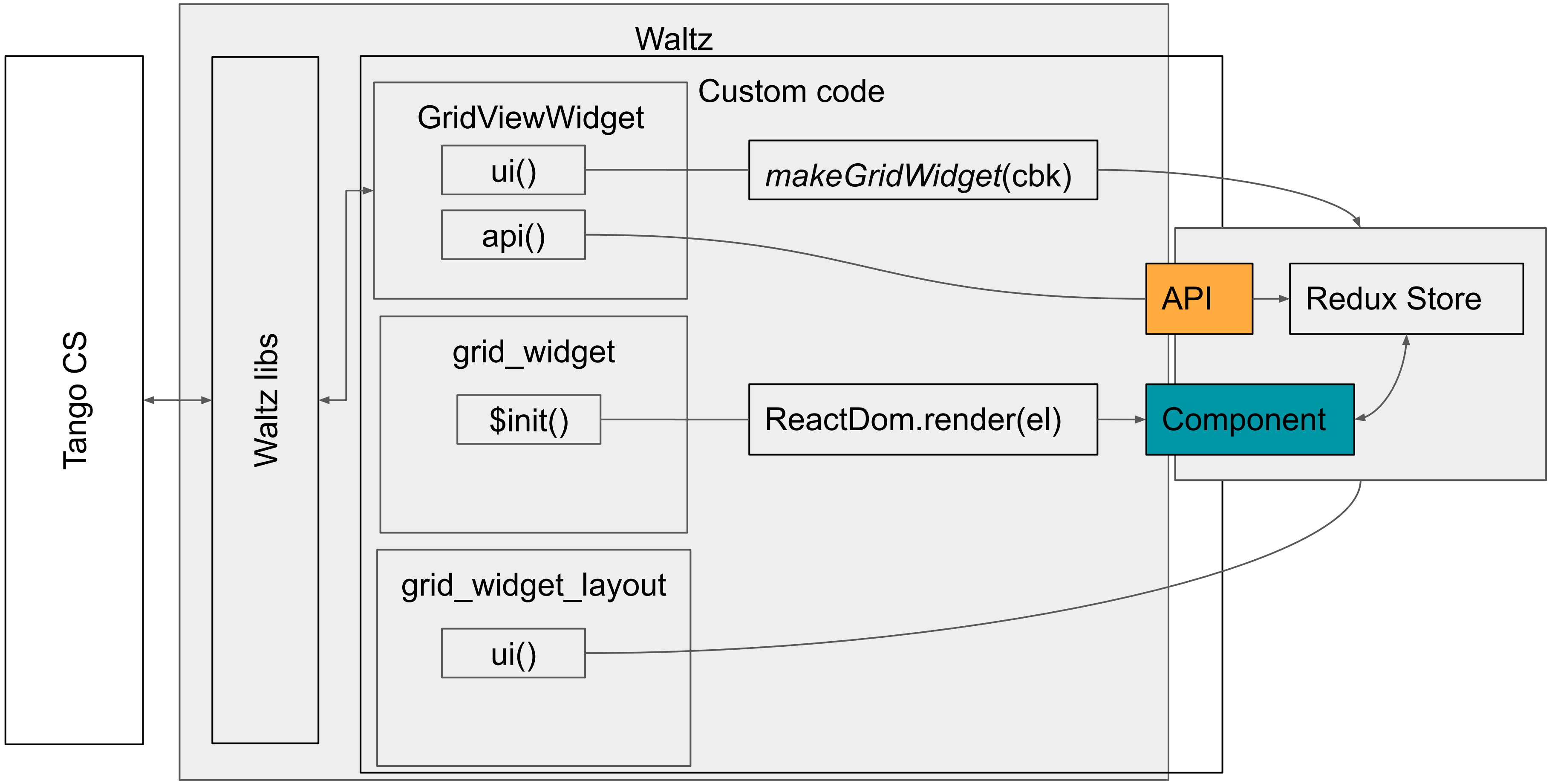
```
setState: (state: GridWidgetStore) => {...}
setDevice: (device: Device) => {...}
removeDevice: (device: Device) => {...}
updateAttributes: (device: Device) => {...}
applyDiff: (diff: GridWidgetStore) => {...}
setGeometry: (
  geom: GridWidgetGeometry) => {...}
setBgColor: (color: string) => {...}
createNewPlot: (plotId: string) => {...}
removePlot: (plot: PlotSettings) => {...}
setPlot: (plot: PlotSettings) => {...}
runCommand: (
  device: DeviceIdentifier,
  name: String,
  cb: CommandCallback) => {...}
```

**Redux STATE**

```
interface GridWidgetStore {
 general?: {
   geometry?:{cols: number,rows: number}
   bgcolor?: string,
   plots?: Array<{id: string, name: string}>
 },
 devices?: Array<{
   name: {host: string,device: string},
   state: string,
   attributes?: Array<{
     name: string, value?: string|number,
     history?: Array<{
       time: number, value: string|number
     }>
   }>, commands?: Array<{name: string}>
 }>
 config?: {
   devices?: Array<{
     name: {host: string,device: string},
     attributes?: Array<{
       name: string, show?: boolean,
       pollingPeriodS?: number,
       displayPlot?: string
     }>,
     commands?: Array<{
       name: string, show?: boolean,
     }>}>}}
```

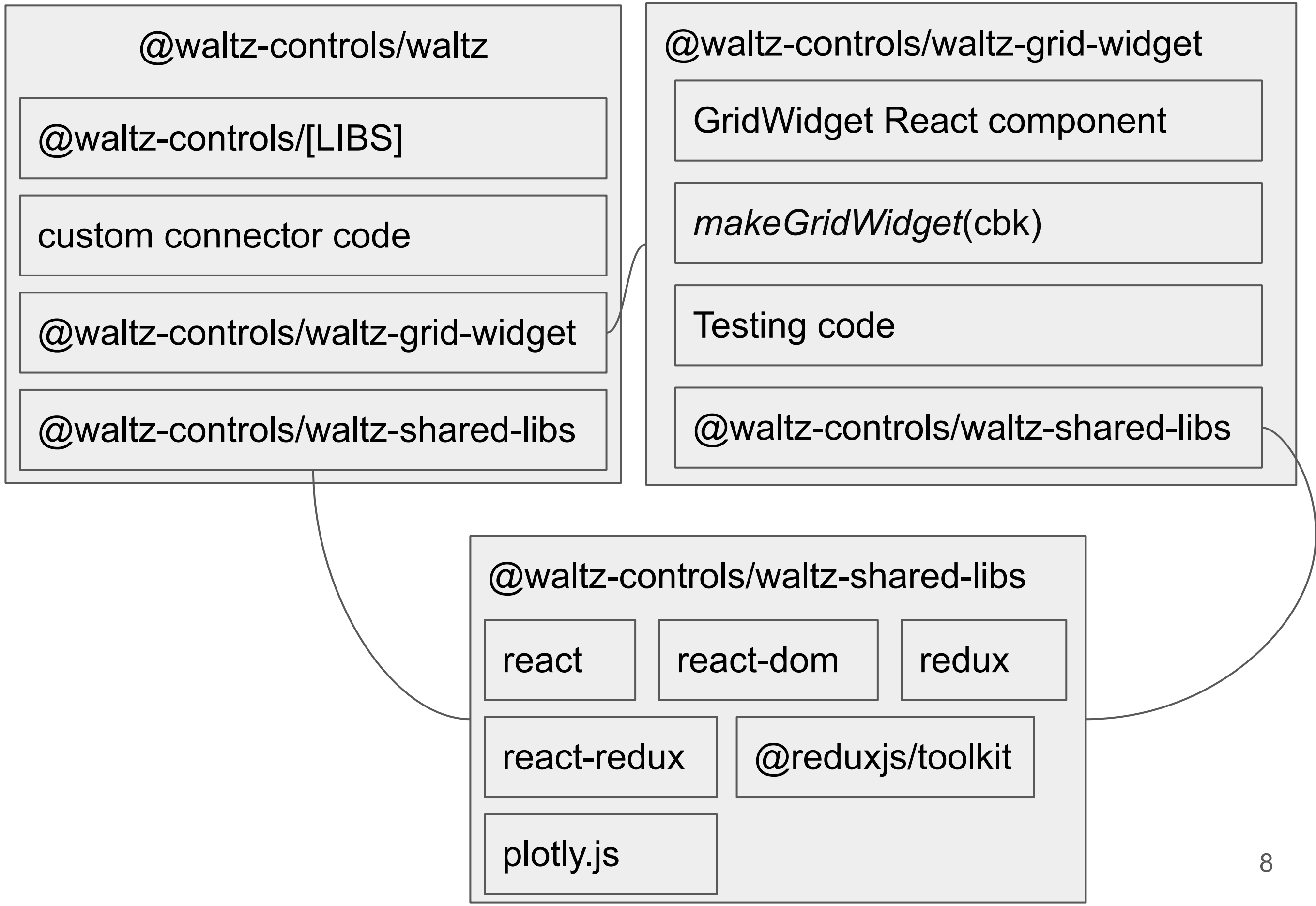
Grid Widget Component

# Waltz Integration: Schema



# Waltz Integration: Project structure

- GridWidget is a standalone React Component
    - Easy to test
- Architecture relies on a shared-libs project since we have to have exact same React/Redux instances along projects
- Custom code that connects Waltz middleware and GridWidget is stored in the main Waltz repository



@waltz-controls/waltz
@waltz-controls/[LIBS]
custom connector code
@waltz-controls/waltz-grid-widget
@waltz-controls/waltz-shared-libs
@waltz-controls/waltz-grid-widget
GridWidget React component
*makeGridWidget*(cbk)
Testing code
@waltz-controls/waltz-shared-libs
@waltz-controls/waltz-shared-libs
react
react-dom
redux
react-redux
@reduxjs/toolkit
plotly.js

## Waltz Integration: Sharing libraries

Shared-libs is based on Webpack DllPlugin technology

webpack.config.js of @waltz-controls/waltz-shared-libs

```
module.exports = {
  context: __dirname,
  entry: {
     vendor: ['react', 'redux', /*...*/ ],
  },
  /*...*/
  plugins: [
   new webpack.DllPlugin({
       path: "../dist/[name]-manifest.json",
       format: true,
       name: "[name]"
   }),
   /*...*/
 ]
}
```

https://webpack.js.org/plugins/dll-plugin/#usage

webpack.config.js of @waltz-controls/waltz

```
module.exports = {
 /*...*/
 plugins: [
   new webpack.DllReferencePlugin({
     context: __dirname,
     manifest: require(
       "@waltz-controls/waltz-shared-libs/" +
       "dist/vendor-manifest.json")}),
   new AddAssetHtmlPlugin({
     filepath: path.resolve(dirname,
       "node_modules/@waltz-controls/" +
       "waltz-shared-libs/dist/vendor.js")}),
   /*...*/
 ]
}
```

package.json of @waltz-controls/waltz

```
{
 /*...*/
 "devDependencies": {
   "@waltz-controls/waltz-shared-libs": "^1.0.2",
   /*...*/
 }
}
```

# Summary

- Waltz is a web version of JIVE and ASTOR
    - is a multi-purpose WebApp
    - is a framework to build custom Web UI’s
    - Future plans is to be interconnection software platform
- Waltz can be successfully extended with a React Components
    - waltz-shared-libs has been added to waltz project to provide necessary React/Redux dependencies
    - to create React Waltz plugin one must connect vendor.js file with webpack DllPlugin mechanism

Thank You!
For your attention