REACTIVE WEB INTERFACES WITH POLYMER AND ROS

Justin Huang and Maya Cakmak

Paul G. Allen School, University of Washington

September 21-22, 2017
GREAT INTERFACES, DIFFICULT SETUP
TETHERED TO COMPUTERS
WHY THE WEB

- Cross platform, cross device
- Huge community of UI devs
- App-like features:
  - Add to Home Screen
  - Push Notifications
  - Loads even when offline
Example: ROS Explorer (https://wiki.ros.org/ros_explorer)

https://youtu.be/rz7NWRXNwu8
OUTLINE

• Web components

• ROS web components

• Examples
WEB COMPONENTS

• New HTML standard

• “Build your own HTML element”

• Works in all browsers (natively or with polyfill)
<head>
<!-- Web components polyfill -->
<script src="path/to/webcomponents-loader.js"></script>

<!-- Import web component to use -->
<link rel="import" href="path/to/ros-websocket.html"/>
</head>

<body>
<!-- Add component to document -->
<ros-websocket id="websocket"></ros-websocket>
</body>
<ros-websocket id="websocket"></ros-websocket>

<script>
    var websocket = document.getElementById("websocket");

    // Add event listener
    websocket.addEventListener("connection", onConnected);

    // Get / set properties
    console.log(websocket.url);
    websocket.url = "ws://demo.robotwebtools.org:9090";

    // Call methods
    websocket.connect();
</script>
POLYMER

• JavaScript library for creating web components

• Provides declarative syntax and data binding, other conveniences

• https://www.polymer-project.org

• Not needed to use web components
DATA BINDING WITH POLYMER

```html
<ros-websocket url="{{url}}"></ros-websocket>

<paper-dialog modal id="disconnected">
  <h2>Disconnected from websocket server</h2>
  <paper-input label="Websocket URL" value="{{url}}">
    ...
  </paper-input>
</paper-dialog>
```
Disconnected from websocket server

Websocket URL
ws://localhost:9090

RETRY
DATA BINDING: SECOND EXAMPLE

```xml
<ros-websocket auto ros="{{ros}}"></ros-websocket>

<ros-topic auto ros="{{ros}}"
  topic="/clock" msg-type="rosgraph_msgs/Clock"
  last-message="{{time}}"></ros-topic>

<p>Time: {{time.clocksecs}}.{{time.clock.nsecs}}</p>
```
ROS WEB COMPONENTS

- A collection of ROS-related web components
- Search for “ROS” at https://www.webcomponents.org/
Click on an element to see documentation, sample code, and demos

**ros-topic** v1.0.0

A Polymer element for publishing and subscribing to topics using roslibjs.

**Element <ros-topic>**

class extends HTMLElement
Path: ros-topic.html

**Description**

An element for publishing and subscribing to topics using roslibjs.

**Example:**

```
<ros-websocket ros="{{ros}}"></ros-websocket>
<ros-topic
  auto
  id="topic"
  on-message="handleMessage"
  topic="/clock"
  ros="{{ros}}"
  msg-type=""rosgate_msgs/Clock"
>"</ros-topic>
```

**Subscribing to a topic**

Simple specify the topic name and msgType. If **auto** is set, then the element will automatically subscribe and fire **message** events.

**Dependencies**

polymer 1.9 - 2
ros-websocket ^3.0.0
<ros-rviz>: like RViz, but an HTML element

https://youtu.be/tqXdg5eAw6g
BUILDING BLOCKS FOR WEB APPS

- Using web components like `<ros-rviz>`
- Reactive database pattern
- User event pattern
<ros-rviz> embedded inside rapid_pbd, a programming by demonstration system

https://youtu.be/Bv10RSzbw8w
• Publish state using latched publisher, republish when state changes

```cpp
Database::Update(const string& id, const Program& program) {
    db_->_updateID(id, program);
    if (publishers_.find(id) == publishers_.end()) {
        int queue_size = 1;
        bool latched = true;
        publishers_[id] = nh_.advertise<Program>(
            "program/" + id, queue_size, latched);
    } 
    publishers_[id].publish(program);
}
```

Do database update
Create latched publisher to topic /program/59b9aa.. if it doesn’t already exist
Publish updated data
- Web clients get data by subscribing to a topic
- Can use data binding from URL all the way to view

```xml
<app-route route="route"
    pattern="/program/:id"
    data="{{routeData}}">
</app-route>

<ros-topic auto ros="{{ros}}"
    topic="program/{{routeData.id}}" msg-type="pbd_msgs/Program"
    last-message="{{program}}">
</ros-topic>

<pbd-program program="{{program}}"></pbd-program>
```

Parse URL and set routeData.id to 59b9aaf1a4065...

Subscribe to /program/59b9aaf...

Bind data and pass it to program element
USER EVENT PATTERN

- Frontend publishes events to server to modify data
- Server interprets event, modifies data, and republishes it
- Easy to record frontend interactions with rosbag
- Downside: frontend is sluggish if server connection is slow
Browsers

UserEvent.msg

- type: Add step
- program_id: 59b9aa...
- step: { ... }

Modify program
Save to DB
Republish program

Server

Database::Update(id, program) {
    db_->updateID(id, program);
    publishers_[id].publish(program);
}
QUICKSTART
INSTALL NODE.JS

• Install Node.js if you don’t already have it.

• Recommended: install Node using NVM (Node Version Manager)

> curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.4/install.sh | bash
> source ~/.bashrc
> nvm install node
INSTALL BOWER AND POLYMER

• Bower is a package manager for frontend projects

> npm install -g bower polymer-cli
CREATE A NEW PROJECT

> mkdir my_project
> cd my_project
> bower init
(Answer questions)
**DOWNLOAD WEB COMPONENTS**

- Bower will download projects at specified GitHub URLs, as well as its dependencies, to the `bower_components/` folder.
- `--save` adds the entry to `bower.json`.
- Future developers just run “bower update” to download/update components.

> bower install --save jstnhuang/ros-websocket
> bower install --save jstnhuang/ros-rviz
• Find documentation on webcomponents.org

• https://www.webcomponents.org/element/jstnhuang/ros-rviz/elements/ros-rviz

• https://www.webcomponents.org/element/jstnhuang/ros-websocket/elements/ros-websocket

• Learn more about Polymer at https://www.polymer-project.org/2.0/start/
<!doctype html>
<html>
<head>
<title>ros-rviz</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, minimum-scale=1.0, initial-scale=1.0, user-scalable=yes">
<script src="bower_components/webcomponentsjs/webcomponents-loader.js"></script>
<link rel="import" href="bower_components/ros-websocket/ros-websocket.html" />
<link rel="import" href="bower_components/ros-rviz/ros-rviz.html" />
<link rel="import" href="bower_components/polymer/lib/elements/dom-bind.html" />
<style>
html, body {
  height: 100%;
  margin: 0;
}
</style>
</head>
<body>
<dom-bind>
<template is="dom-bind">
<ros-websocket auto id="websocket" ros="{{ros}}" url="{{url}}"></ros-websocket>
<ros-rviz ros="{{ros}}" websocket-url="{{url}}"></ros-rviz>
</template>
</dom-bind>
</body>
</html>
SERVE THE WEBPAGE

- At this point, you should see `<ros-rviz>` and be able to add a grid
- Adding more displays will require backend support

> cd my_project
> python -m SimpleHTTPServer 8080
VISUALIZE A URDF

• Set up and run a mesh file server according to https://github.com/hcrlab/wiki/blob/master/web_development/serving_urdf.md

• Run rosbridge_server and tf2_web_republisher as shown below

```
roslaunch rosbridge_server rosbridge_websocket.launch
rosrun tf2_web_republisher tf2_web_republisher
```
FOR REACT USERS

• React data-binding sets *attributes* rather than *properties*. To set object/array properties or listen to events, get a ref to the element.

• See [https://robdodson.me/interoperable-custom-elements/](https://robdodson.me/interoperable-custom-elements/) and [https://custom-elements-everywhere.com](https://custom-elements-everywhere.com)
## Reference

Learn more about web components

<table>
<thead>
<tr>
<th>Find web components</th>
<th><a href="https://www.webcomponents.org">https://www.webcomponents.org</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Polymer library</td>
<td><a href="https://www.polymer-project.org">https://www.polymer-project.org</a></td>
</tr>
</tbody>
</table>

**ROS components and web applications**

<table>
<thead>
<tr>
<th>ROS web components</th>
<th><a href="https://www.webcomponents.org/search/ros">https://www.webcomponents.org/search/ros</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>ROS Explorer</td>
<td><a href="https://wiki.ros.org/ros_explorer">https://wiki.ros.org/ros_explorer</a></td>
</tr>
<tr>
<td>Rapid PbD</td>
<td><a href="https://github.com/jstnhuang/rapid_pbd">https://github.com/jstnhuang/rapid_pbd</a></td>
</tr>
<tr>
<td>Codelt!</td>
<td><a href="https://github.com/hcrlab/code_it">https://github.com/hcrlab/code_it</a></td>
</tr>
<tr>
<td>Robot Web Server</td>
<td><a href="https://github.com/hcrlab/rws">https://github.com/hcrlab/rws</a></td>
</tr>
</tbody>
</table>
WEB INTERFACES IN THE WILD
REACTIVE WEB INTERFACES WITH POLYMER AND ROS

Justin Huang and Maya Cakmak
Paul G. Allen School, University of Washington
September 21-22, 2017