

# 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

rqt\_console\_Console - rqt

Console

Displaying 19 messages

#	Message	Severity	Node	Stamp	Topics	Location
#19	[Info] RapidPBD editor ready.	Info	/rapid_pbd...	13:17:24.81...	/message_s...	/var/ros/rw...
#18	[Warn] The root link base_footprint has an inertia specified in the URDF, but KDL does not support a root lin...	Warn	/rapid_pbd...	13:17:24.80...	/message_s...	/tmp/binar...
#17	[Warn] The root link base_footprint has an inertia specified in the URDF, but KDL does not support a root lin...	Warn	/rapid_pbd...	13:17:24.66...	/message_s...	/tmp/binar...
#16	[Info] RapidPBD program executor ready.	Info	/rapid_pbd...	13:17:24.53...	/head_traj_...	/var/ros/rw...
#15	[Info] Querying content in a futher 0 datacentres	Info	/message_s...	13:17:24.15...	/message_s...	message_st...
#14	[Info] waitForService: Service [/message_store/update] is now available.	Info	/rapid_pbd...	13:17:24.18...	/message_s...	/tmp/binar...
#13	[Info] waitForService: Service [/message_store/insert] is now available.	Info	/rapid_pbd...	13:17:24.17...	/message_s...	/tmp/binar...
#12	[Info] waitForService: Service [/message_store/update] has not been advertised, waiting...	Info	/rapid_pbd...	13:17:24.16...	/message_s...	/tmp/binar...
#11	[Info] waitForService: Service [/message_store/insert] is now available.	Info	/rapid_pbd...	13:17:24.15...	/message_s...	/tmp/binar...
#10	[Info] waitForService: Service [/message_store/insert] has not been advertised, waiting...	Info	/rapid_pbd...	13:17:23.79...	/message_s...	/tmp/binar...
#9	[Info] waitForService: Service [/message_store/insert] has not been advertised, waiting...	Info	/rapid_pbd...	13:17:23.79...	/message_s...	/tmp/binar...
#8	[Info] [Client 1] Subscribed to rapid_pbd/editor_events	Info	/rosbridge_...	13:17:22.91...	/rapid_pbd...	protocol.py...
#7	[Info] [Client 1] Subscribed to rapid_pbd/editor_events	Info	/rosbridge_...	13:17:22.89...	/rapid_pbd...	protocol.py...
#6	[Info] [Client 1] Subscribed to rapid_pbd/editor_events	Info	/rosbridge_...	13:17:22.88...	/rapid_pbd...	protocol.py...
#5	[Info] [Client 1] Subscribed to rapid_pbd/program_list	Info	/rosbridge_...	13:17:22.81...	/rapid_pbd...	protocol.py...

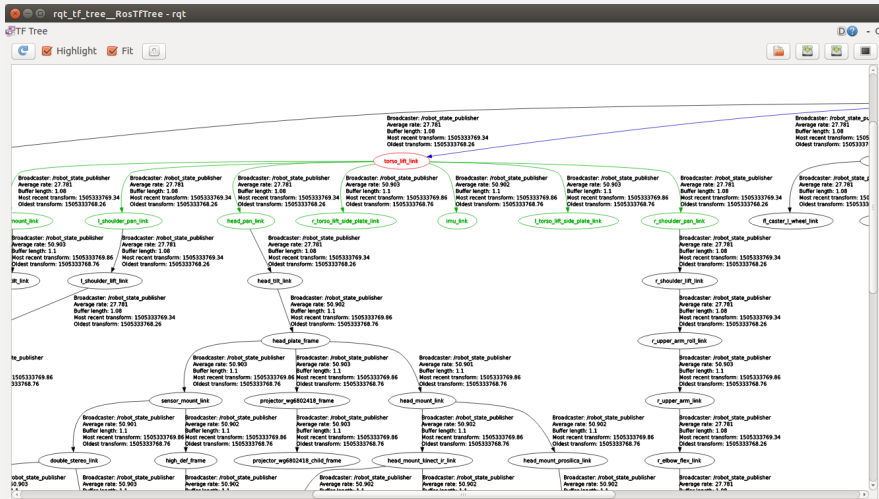
Exclude Messages...

...with severities: Debug Info Warn Error Fatal

...from node: /message\_store /rapid\_pbd/editor\_node /rapid\_pbd/program\_executor /rapid\_pbd/surface\_segmentation\_node /rosbridge\_websocket

Highlight Messages...

...containing:  Regex



custom\_landmarks.rviz\* - RViz

Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point

Displays

- Global Options
- Fixed Frame
- Background Color
- Frame Rate
- Global Status: Ok
- Fixed Frame
- Grid
- Scene
- Landmark
- Alignment
- Output
- InteractiveMark...
- PointCloud2
- DepthCloud
- RobotModel

base\_link

48; 48; 48

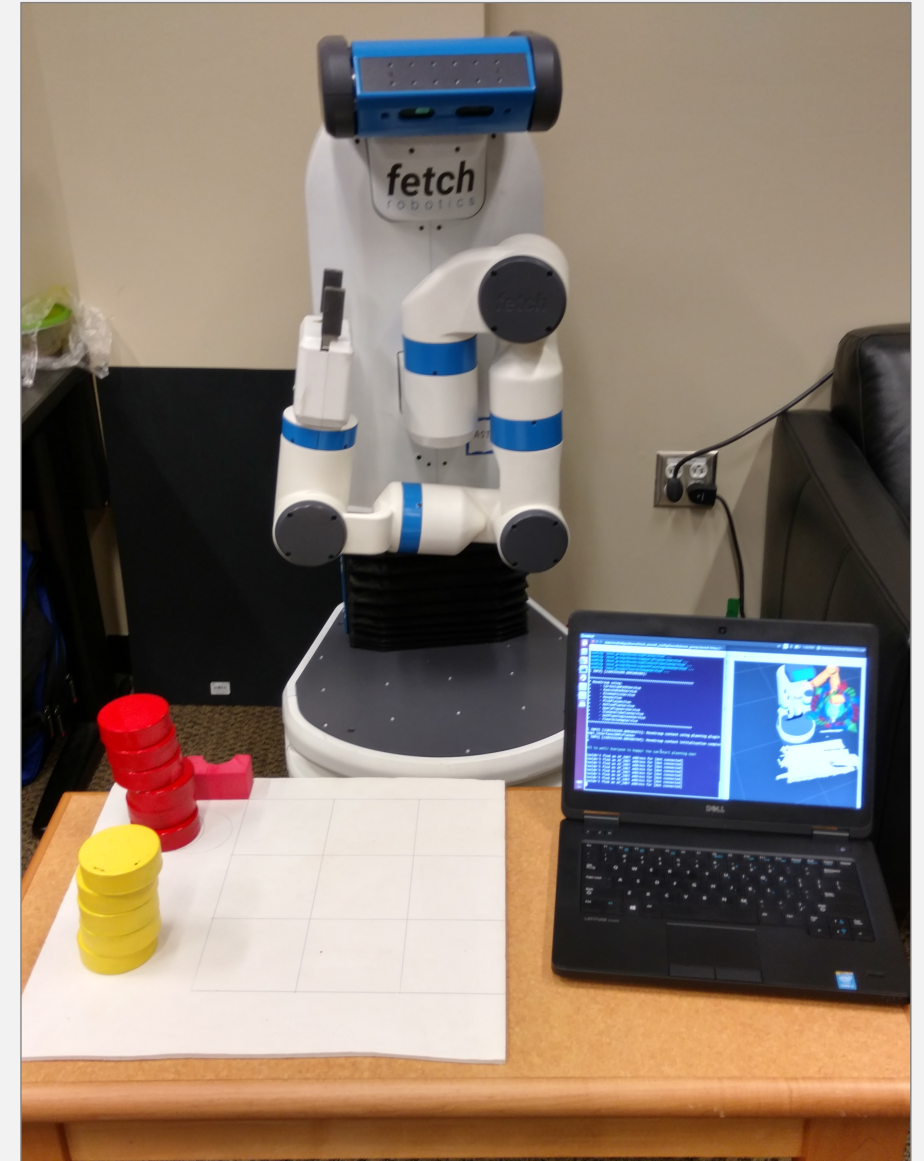
30

Add Duplicate Remove Rename

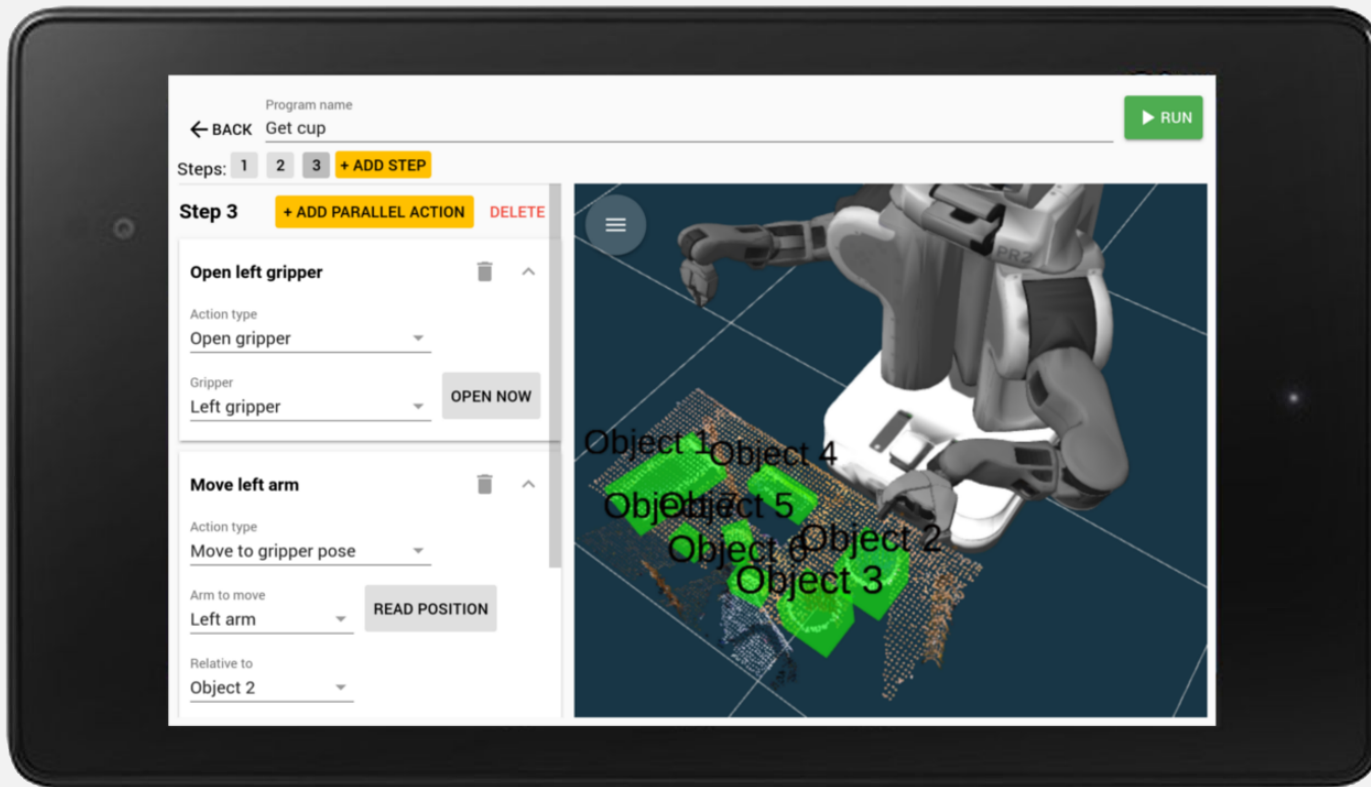
Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.

30 fps

# 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://wiki.ros.org/ros_explorer))

The image shows a terminal window on the left and a web browser window on the right. The terminal displays the command `roslaunch ros_explorer ros_explorer.launch` and its output, including the ROS\_MASTER\_URI and the start of the ROS core service. The browser window shows the ROS.org website with the ROS Explorer page. The page includes a navigation menu with links for Documentation, Browse Software, News, and Download. The main content area lists various ROS resources such as Distributions, Packages, Core Libraries, Common Tools, Robots/Hardware, Robots, Sensors, and Motors. A sidebar on the right provides additional navigation options like Wiki, Page, and User.

```
Terminal
/home/jstn/catkin_ws_indigo/src/ros_explorer/launch/ros_explorer.launch
[catkin olivaw ~]$ roslaunch ros_explorer ros_explorer.launch

/opt/ros/indigo
rosapi (rosapi/rosapi_
rosbridge_websocket (r
ROS_MASTER_URI=http://loca
core service [/rosout] fou
process[rosbridge_websocke
process[rosapi-2]: started
registered capabilities (c
- rosbridge_library.capab
- rosbridge_library.capab
- rosbridge_library.capab
- rosbridge_library.capab
- <class 'rosbridge libra
- rosbridge_library.capab
- rosbridge_library.capab
- rosbridge_library.capab
9879.071000 /rosbridge_web
d on port 9090
9880.496000 /rosbridge_web
9889.651000 /rosbridge_web
tal.
```

ROS.org  
About | Support | Discussion Forum | Service Status | Q&A answers.ros.org  
Search:  Submit

Documentation Browse Software News Download

## Documentation

ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license.

Available Translations: German | Spanish | French | Italian | Japanese | Korean | Brazilian Portuguese | Portuguese | Russian | Thai | Turkish | 简体中文 | Ukrainian | Vietnamese

ROS:

- Install**  
Install ROS on your machine.
- Getting Started**  
Learn about various concepts, client libraries, and technical overview of ROS.
- Tutorials**  
Step-by-step instructions for learning ROS hands-on
- Contribute**  
How to get involved with the ROS community, such as submitting your own repository.
- Support**  
What to do if something doesn't work as expected.

Software:

- Distributions**  
View the different release Distributions for ROS.
- ⊕ Packages**  
Search the 2000+ software libraries available for ROS.
- Core Libraries**  
APIs by language and topic.
- Common Tools**  
Common tools for developing and debugging ROS software.

Robots/Hardware:

- Robots**  
Robots that you can use with ROS.
- Sensors**  
Sensor drivers for ROS.
- Motors**  
Motor controller drivers for ROS.

Wiki

- Distributions
- ROS/Installation
- ROS/Tutorials
- RecentChanges
- Documentation

Page

- Edit (Text)
- Edit (GUI)
- Info
- Subscribe
- Add Link
- Attachments
- More Actions: ▾

User

- Justin Huang
- Settings
- Logout

<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)

# ANATOMY OF A WEB COMPONENT

```
<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");

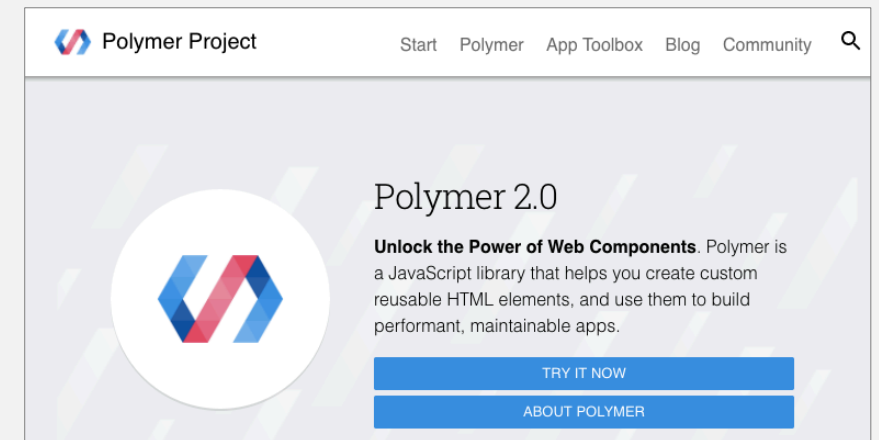
  // Add event listener
  websocket.addEventListener("connection", onConnected);

  // 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

```
<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>
```

[← Back to dashboard](#)

WebSocket URL

ws://localhost:9090

CONNECT

**Nodes**

Loading...

**Topics**

Loading...

**Services**

Loading...

**Parameters**

CREATE

Loading...

**Disconnected from websocket server**

WebSocket URL

ws://localhost:9090

RETRY

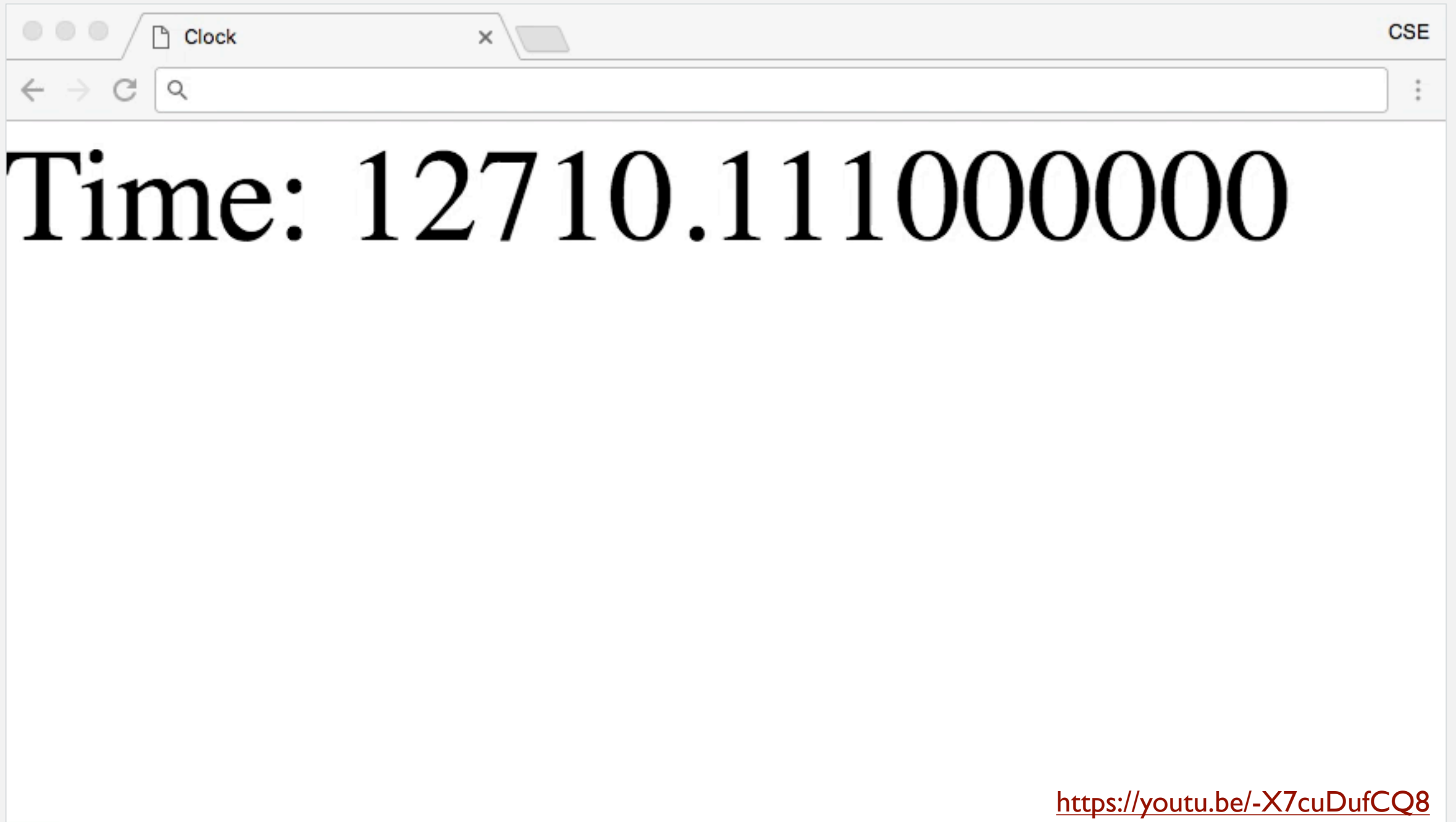


## DATA BINDING: SECOND EXAMPLE

```
<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.clock.secs}}.{{time.clock.nsecs}}</p>
```



<https://youtu.be/-X7cuDufCQ8>

# ROS WEB COMPONENTS


- A collection of ROS-related web components
- Search for “ROS” at <https://www.webcomponents.org/>

The screenshot shows the webcomponents.org website with a search bar containing 'ros'. The search results are displayed under the heading 'Results for "ros"'. There is one collection listed: 'ros-element-collection' with 8 items, described as 'A collection of ROS elements for Polymer'. The collection is attributed to 'jstnhuang'.

The screenshot shows a list of 8 ROS-related web components. Each entry includes a name, a brief description, a star rating, and a download icon.


Element Name	Description	Stars	Download
ros-rviz	A Polymer element for ROS visualization.	3	2
ros-websocket	A Polymer element wrapping roslibjs's ROS object.	0	0
ros-service	An element for calling ROS services using roslibjs.	0	0
ros-param	Polymer element for using ROS Parameters	0	0
ros-joint-states	Polymer element that subscribes to a ROS joint states topic.	0	0
ros-topic	A Polymer element for publishing and subscribing to topics using roslibjs.	0	0
ros-action-client	A Polymer element for the roslibjs ActionClient.	0	0
ros-log	Adds robot logs to the website	0	0


Click on an element to see documentation, sample code, and demos

by [jstnhuang](#)

★ 0 🔗 0 👁️ 0 ! 0

LICENSED UNDER [BSD-3-CLAUSE](#),  
LAST UPDATED 4 MONTHS AGO, &  
INSTALLED VIA [BOWER](#) +

 [View on GitHub](#)

 [Star on GitHub](#)  
Requires authentication

[Overview](#)

### Elements

`<ros-topic>`

[Demo](#) DEMO

## Dependencies

[polymer](#) 1.9 - 2  
[ros-socket](#) ^3.0.0

# 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:

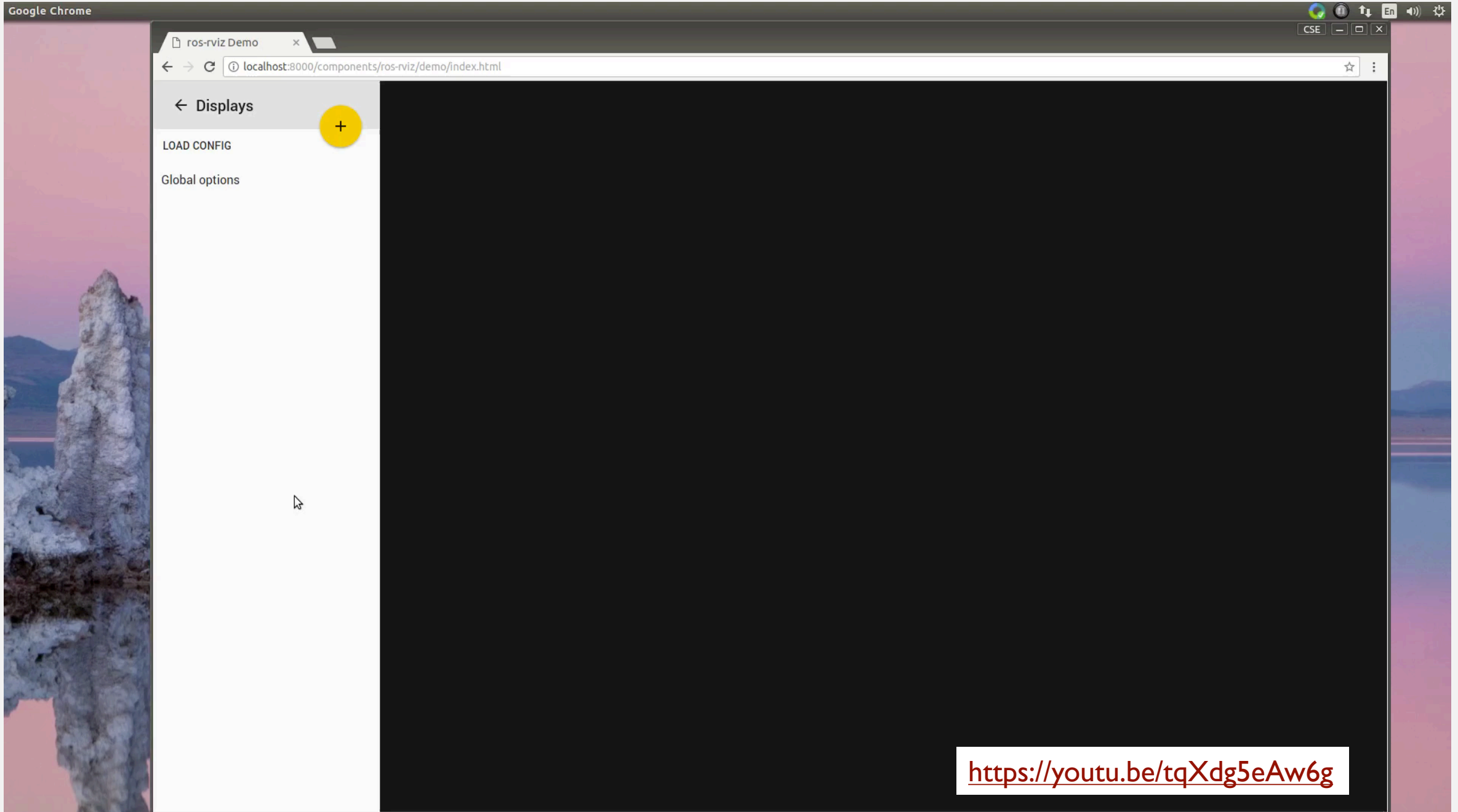
```
<ros-socket ros="{ros}"></ros-socket>
<ros-topic
  auto
  id="topic"
  on-message="handleMessage"
  topic="/clock"
  ros="{ros}"
  msg-type="rosgraph_msgs/Clock"
></ros-topic>
```

### 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.



# <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

The screenshot displays the rapid\_pbd web interface in a browser window. The browser address bar shows the URL `localhost:8081/#/program/59b9aaf1a406545924bbb47`. The interface includes a navigation bar with a 'Program name' field containing 'Get cup' and a 'RUN' button. Below this, there are step indicators for 1 through 5, with a '+ ADD STEP' button. The first step is expanded, showing a 'Move right arm' action. The action type is 'Move to gripper pose', and the arm to move is 'Right arm'. The relative reference is 'Robot torso'. The coordinates for the move are: x: 0.31901481816816396, y: -0.7151745009881394, and z: -0.04790878406968724. A 3D visualization of a robot is shown on a grid floor. The right side of the interface features a 'Console' panel with a 'top' dropdown and a 'Filter' button. The bottom of the browser window shows a 'Console' tab.

<https://youtu.be/BvIORSzbw8w>

# REACTIVE DATABASE PATTERN

- Publish state using latched publisher, republish when state changes

```
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);  
}
```

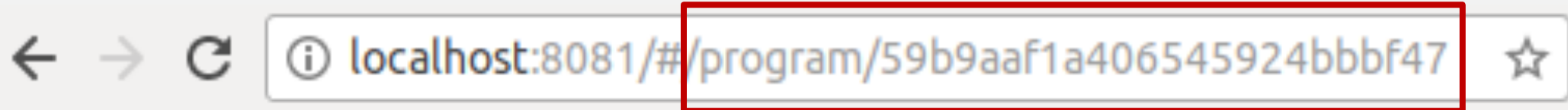
← Do database update

← 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



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

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

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

Subscribe to /program/59b9aaf...

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

Bind data and pass it to program element

rapid-pbd x CSE - □ ×

localhost:8081/#/ ☆ ⋮

+ CREATE NEW PROGRAM

Name	Actions
Get cup	OPEN DELETE

↗

rapid-pbd x CSE - □ ×

localhost:8081/#/ ☆ ⋮

+ CREATE NEW PROGRAM

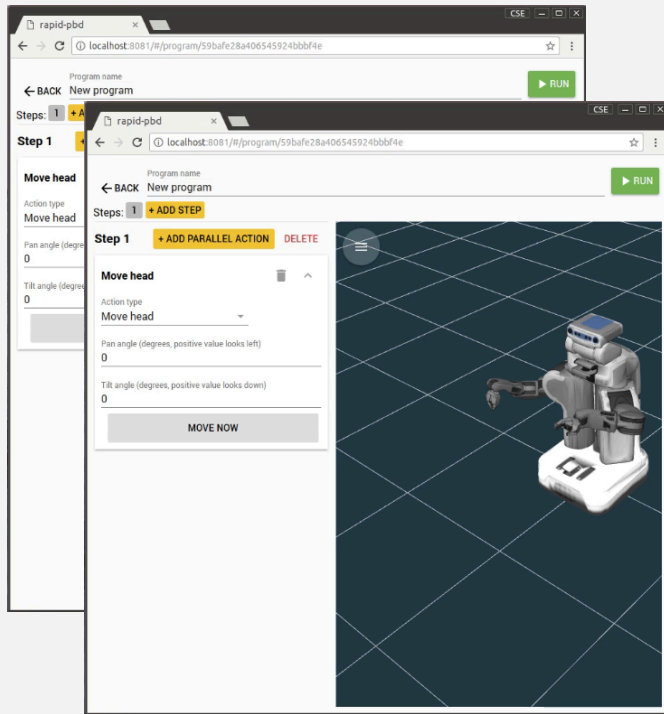
Name	Actions
Get cup	OPEN DELETE

<https://youtu.be/KINBqsMW7iY>

## 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

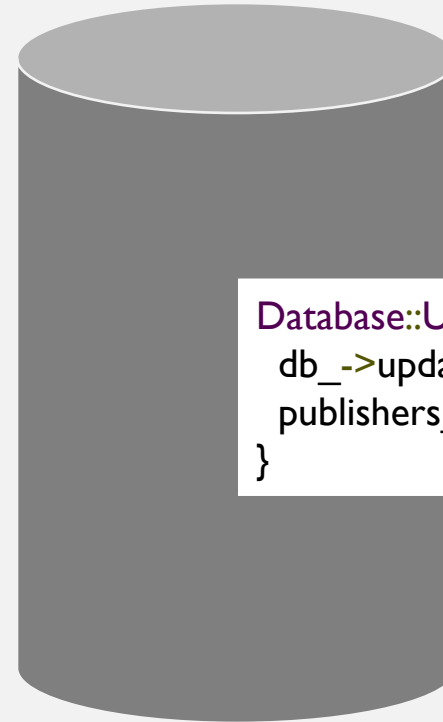
type: Add step  
program\_id: 59b9aaf...  
step: { ... }



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

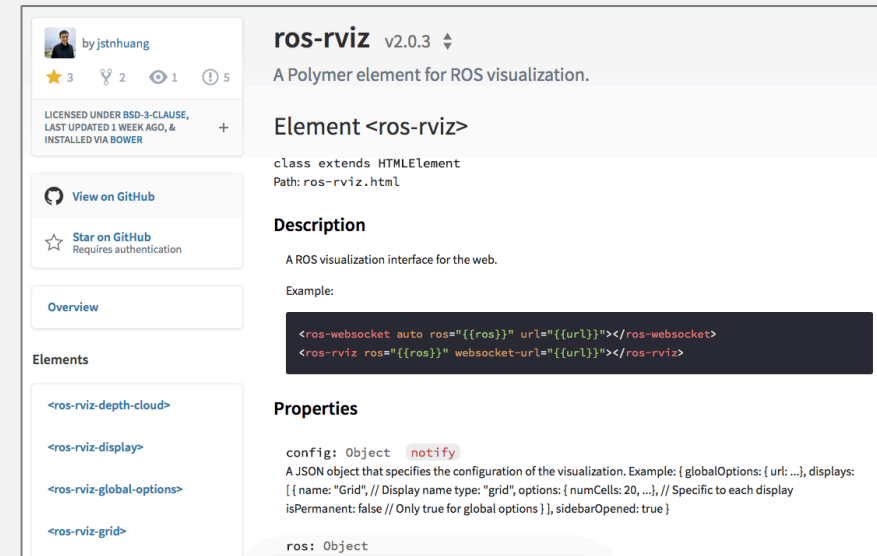
- Bower will download projects at specified GitHub URLs, as well as its dependencies, to the `bower_components/` folder
- `--save` adds the entry to `bower.json`
- Future developers just run “`bower update`” to download/update components

```
> bower install --save jstnhuang/ros-websocket
```

```
> bower install --save jstnhuang/ros-rviz
```

# FIND DOCUMENTATION

- Find documentation on [webcomponents.org](https://www.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/>



The screenshot shows the webcomponents.org page for the `ros-rviz` element. The page header includes the author's name "by jstnhuang", a star count of 3, a fork count of 2, a watch count of 1, and a notification count of 5. Below the header, there are links to "View on GitHub" and "Star on GitHub" (which requires authentication). The main content area is titled "ros-rviz v2.0.3" and describes it as a "Polymer element for ROS visualization." The "Element <ros-rviz>" section shows the class name "class extends HTMLElement" and the file path "Path: ros-rviz.html". The "Description" section states it is a "ROS visualization interface for the web" and provides an example of its usage in HTML: `<ros-websocket auto ros="{{ros}}" url="{{url}}"></ros-websocket>` and `<ros-rviz ros="{{ros}}" websocket-url="{{url}}"></ros-rviz>`. The "Properties" section lists a "config" property of type "Object" with a "notify" event, and a "ros" property of type "Object".

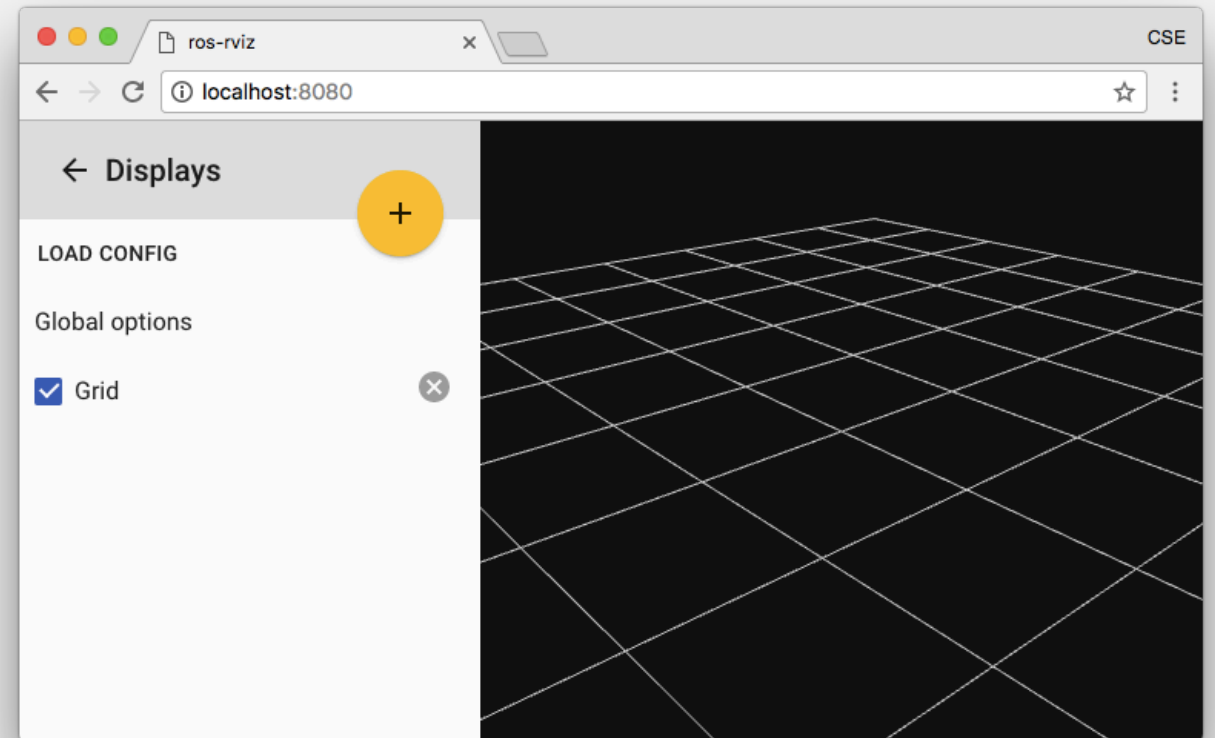
# CREATE INDEX.HTML

```
<!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

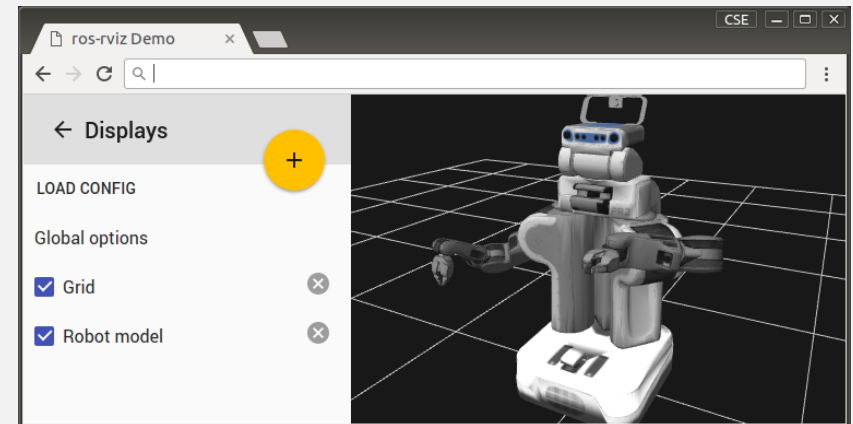
```
> cd my_project  
> python -m SimpleHTTPServer 8080 .
```



# 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](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/> and <https://custom-elements-everywhere.com>

# REFERENCE

## Learn more about web components

---

Find web components

<https://www.webcomponents.org>

---

Polymer library

<https://www.polymer-project.org>

## ROS components and web applications

---

ROS web components

<https://www.webcomponents.org/search/ros>

---

ROS Explorer

[https://wiki.ros.org/ros\\_explorer](https://wiki.ros.org/ros_explorer)

---

Rapid PbD

[https://github.com/jstnhuang/rapid\\_pbd](https://github.com/jstnhuang/rapid_pbd)

---

Codelt!

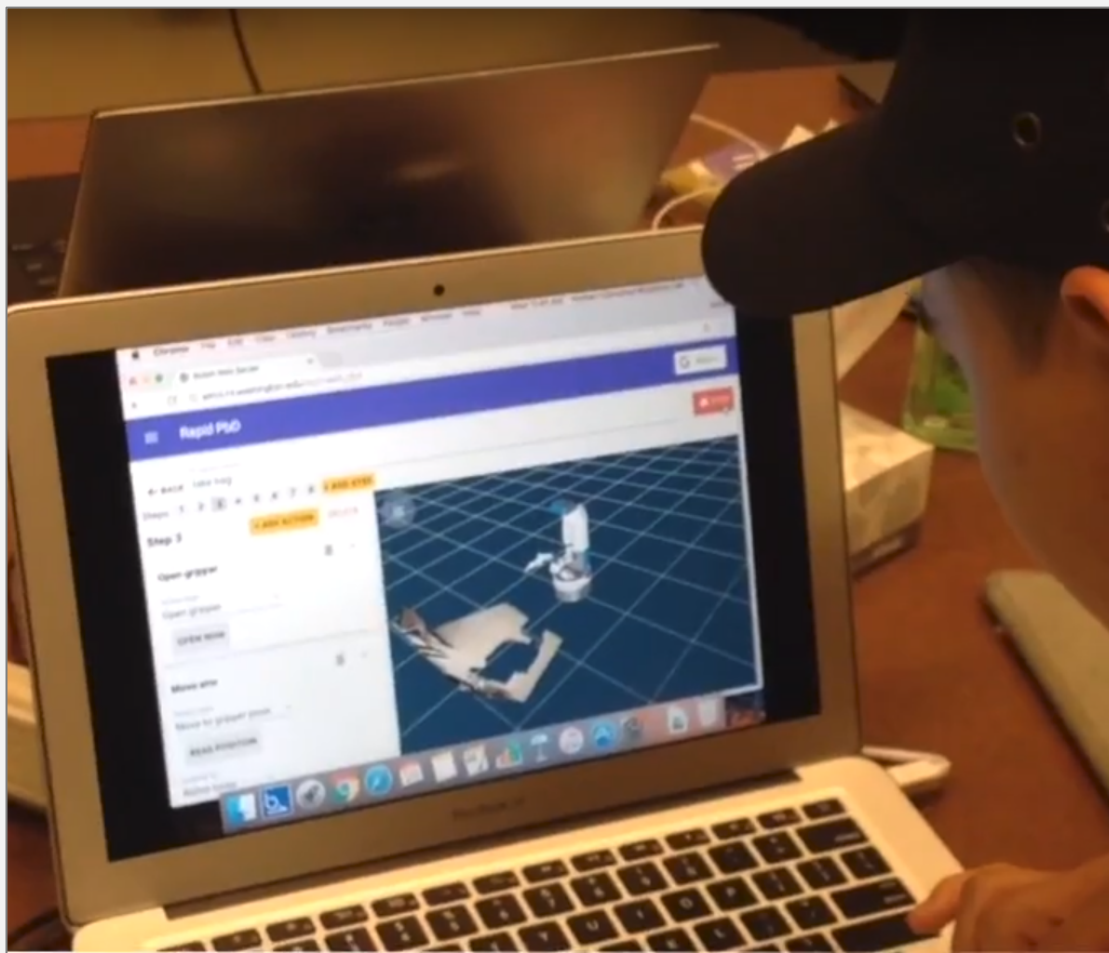
[https://github.com/hcrlab/code\\_it](https://github.com/hcrlab/code_it)

---

Robot Web Server

<https://github.com/hcrlab/rws>

# 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

