intel Software

Node.js* Client & Web Bridge Ready for ROS* 2.0

Minggang Wang
September 30 2018

Agenda

- Who We Are
- Why Use Node.js & Web in Robotics
- Thinking in “ROS 2.0 + Web”
- What We Have Done for “ROS 2.0 + Web”
- The Design of `rclnodejs` & `ros2-web-bridge`
- List of Features
- Performance Comparison: Node.js, C++ & Python
- Video Demo
- Intel ❤ Robot
- Contacts & Resource Links

Who we are

- Intel Open Source Technology Center (OTC) is home to the core of Intel's open source development efforts.
- We’re from OTC Web Team; we do web technology in client, edge, cloud, IoT, W3C standard, Robotics & etc., to keep web open, secure, rich-featured and performant.

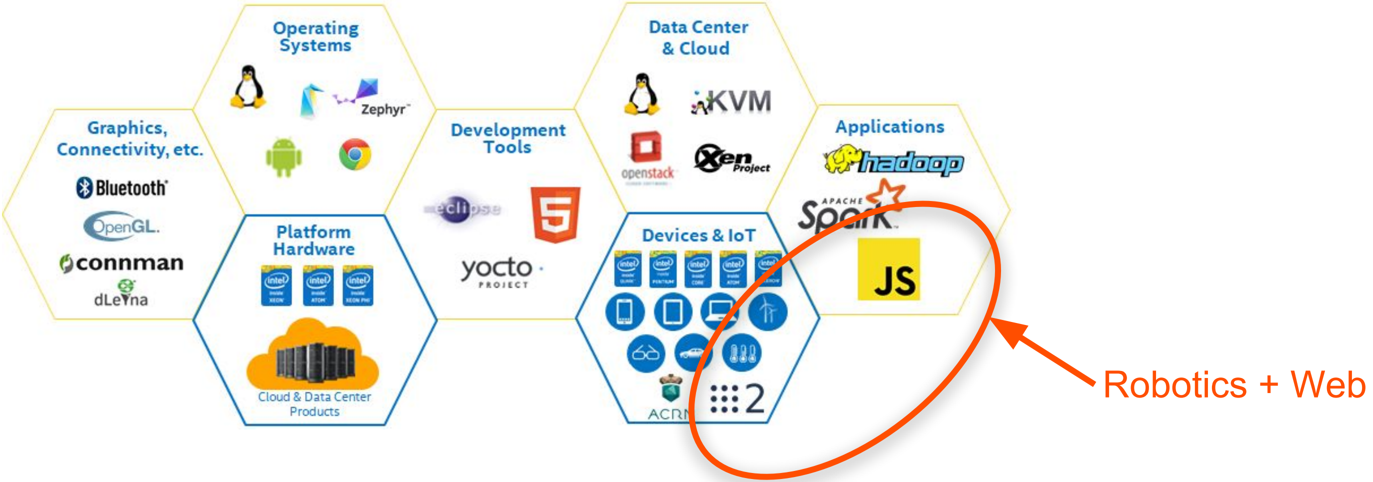
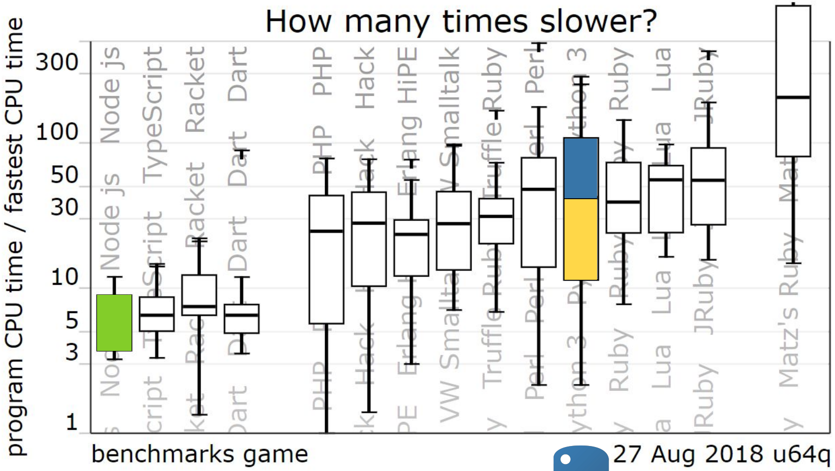


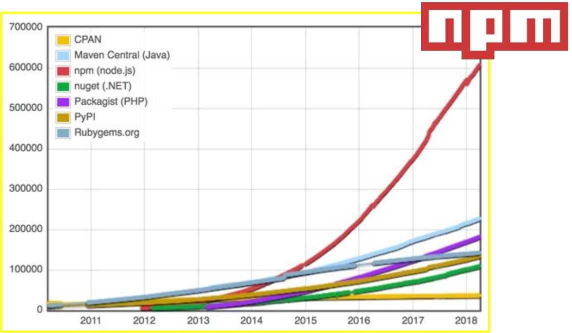
Figure: What Intel OTC does

Why Use Node.js & Web in Robotics

- High-performance (JIT), faster than Python
  - Do more on same robot control board
- Strong ecosystem/community
  - The most popular language on Github*
  - Largest package system in the world
- Easy deployment & debugging
- Naturally for web interface



Benchmark Link

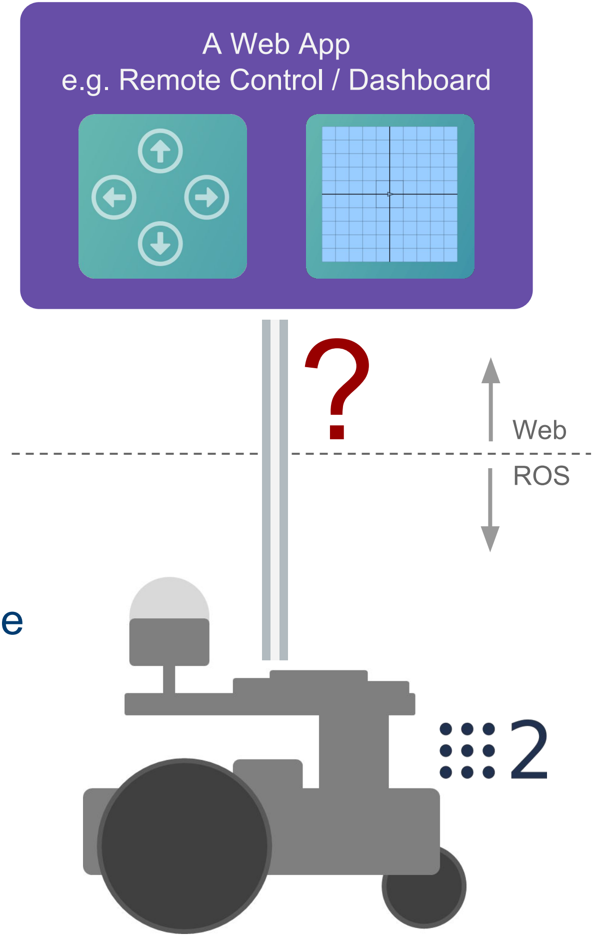


NPM* is the largest package system in the world (>600k)

Thinking in “ROS 2.0 + Web”

- Web is best choice for remote control & dashboard
  - e.g. status inspection, supervised motion control, posture visualization, video streaming & etc.
  - Available anywhere, easy to embed, tons of resources & etc.
- How to bring ROS into the web?
  - RWT* can bring ROS 1.0 APIs into a web browser
  - Nothing for ROS 2.0 back in Mid’17, so we did one
  - But is it the best way to expose all ROS API in web? e.g. service
- Another approach: Node.js web server, is flexible & effective
  - ROS API exposed in server; only business logic in web - RaaS
  - Don’t be scared, web server is just a few lines in Node.js
  - Same skill set for both frontend & backend, easy debugging

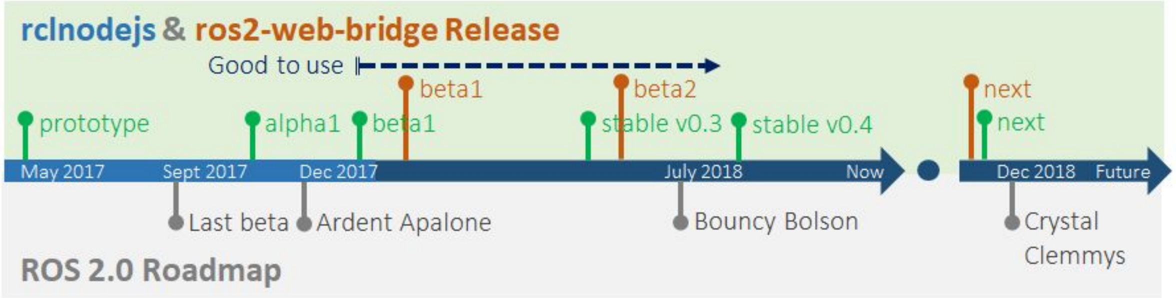
ROS + Web = Better Robot... **But How**?

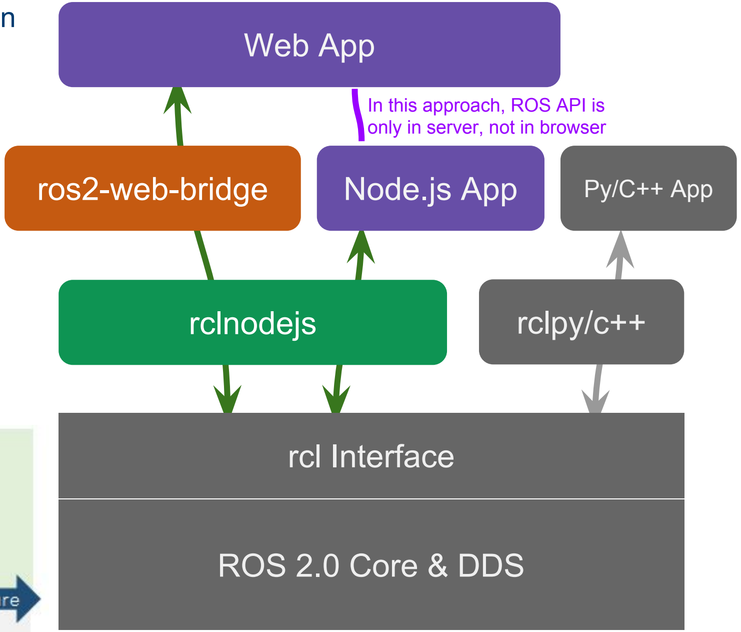


What We Have Done for “ROS 2.0 + Web”

2 packages. Both hosted in GitHub RWT thanks to Jihoon

- rclnodejs (github repo)
  It’s a Node.js client of ROS 2.0. It provides fast, easy & powerful JavaScript API of ROS 2.0
- ros2-web-bridge (github repo)
  Make it possible to call ROS 2.0 API in a web page. It’s compatible with `roslibjs*`





We’ve done the green and orange, and made the purples possible

The Design of rclnodejs (the ROS 2.0 Node.js API)

Principles and philosophy

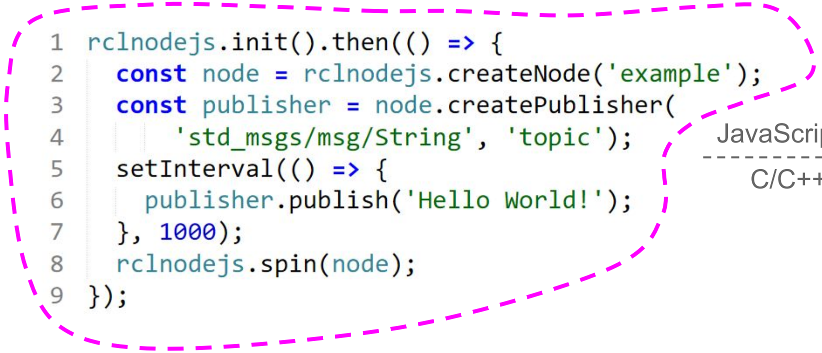
- A thin wrapper to rcl -- same mindset, fast & easier to adapt change
- Event-driven, non-blocking (promise/event) -- advantage of async IO, very simple & efficient
- Able to use new ROS message without recompiling -- everything's on the fly
- User-friendly debugging -- easy to figure out what's wrong
- Embrace ES6* -- most recent cool features of JavaScript language

Legend

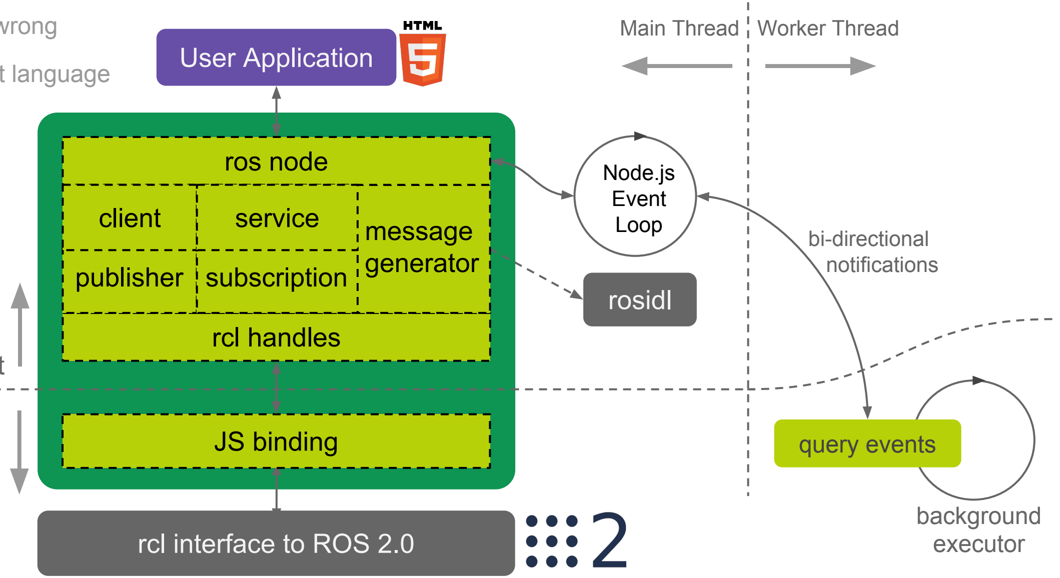
⬛ ROS 2

🟩 rclnodejs

As a result, user can write ROS app easily & effectively.



```
rclnodejs.init().then(() => {
  const node = rclnodejs.createNode('example');
  const publisher = node.createPublisher(
      'std_msgs/msg/String', 'topic');
  setInterval(() => {
    publisher.publish('Hello World!');
  }, 1000);
  rclnodejs.spin(node);
});
```

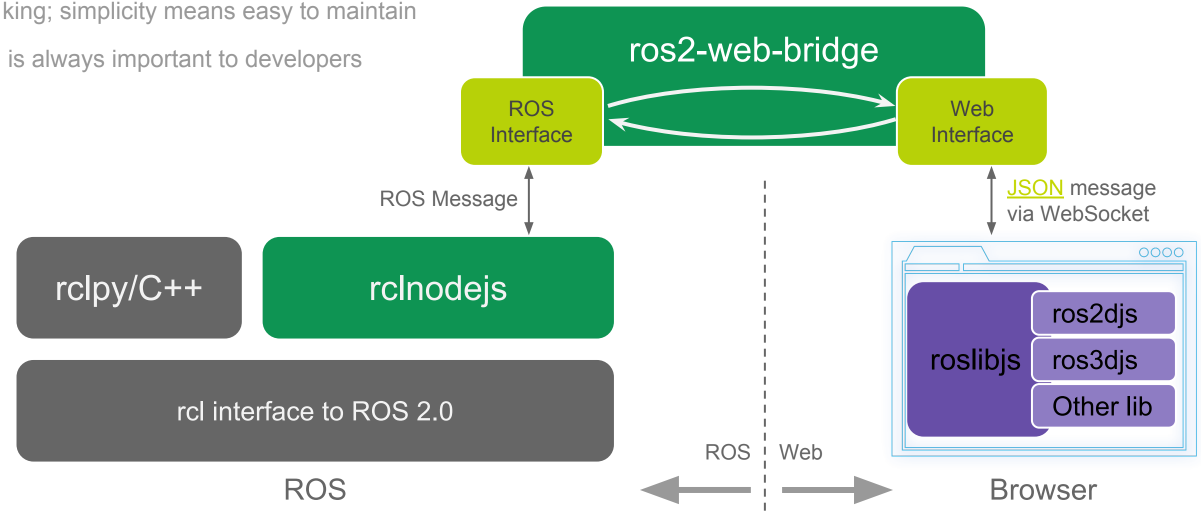


ros2-web-bridge Design (Bring ROS in Browser)

Principles and philosophy

- Meet user’s expectation, be compatible with ROS 1.0 bridge (rosbridge_suite)
    - Protocol compatible with the existing protocol of JSON messages (ROS 1.0)
    - Existing Web Tools can be directly used, e.g. 2D/3D visualization
- Keep it fast and simple -- speed is the king; simplicity means easy to maintain
- User-friendly debugging -- debugging is always important to developers

As a result, RWT ROS 1.0 components are transparently compatible with ROS 2.0



List of Features

rclnodejs

- ROS node -- create/destroy ROS nodes
- Publisher/Subscription -- send/receive ROS message
- Client/Service -- write client/service of ROS request
- QoS support -- configure network QoS policy
- Timer -- periodical notification/callback
- Time/Time Source -- different type of clocks
- Actionlib w/ RethinkRobotics* -- preemptable task management
- Message Gen (idl) -- dynamic generation on the fly
- Validation utilities -- check if it meets rules
- Logging -- easier debugging

ros2-web-bridge

- Publisher/Subscription -- send/receive msg in browser
- Client/Service -- write client/service of ROS request in browser
- Status message support -- figure out what's going on

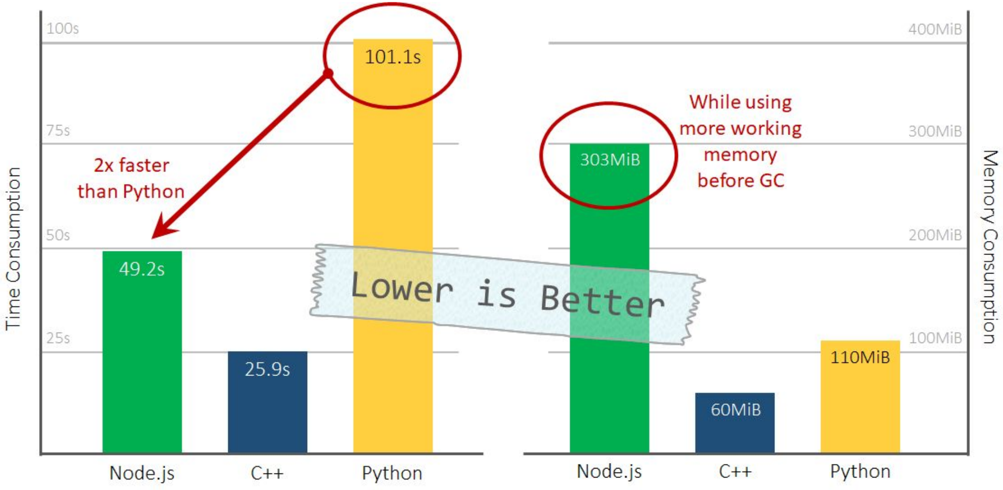
Performance Comparison: Node.js, C++ & Python

Test case: to publish a ROS message, measure the time and memory consumption

- When runcount increases, the trends tend to stabilize
- Same trends were also observed on other types of tests
- Both trends match the common sense

Conclusion: Node.js is times faster than Python, but consumes more memory in runtime.

* Don’t forget to run Python with -O



i5-4250 Haswell/4G RAM/Ubuntu 16.04

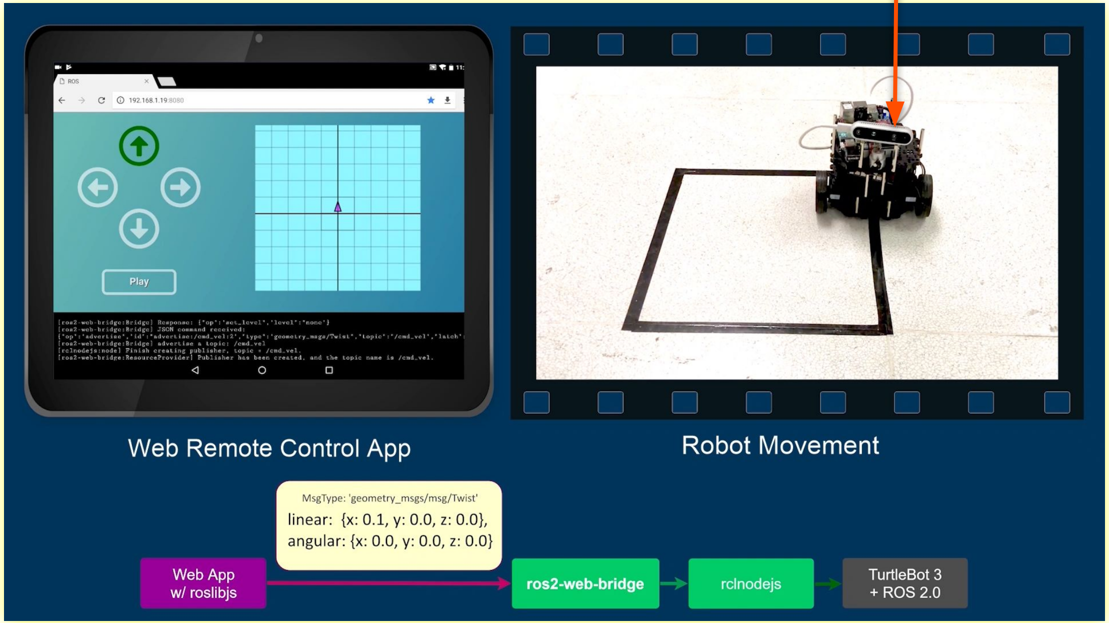
Video Demo (URL)

- Turtlebot* 3 + ROS 2.0
  - Intel® RealSense™ camera
  - Up Board* with Intel Atom®
- A web app as remote control
  - roslibjs + ros2-web-bridge
  - Easy to build powerful UI
  - Running everywhere
- Source code: github
- ROS 2.0 Message Type:
  `geometry_msgs/msg/Twist`

Intel® RealSense™ Depth Camera D415
A compact camera designed to bring depth sensing to more devices:

- Depth FOV: 69.4x42.5x77
- Active IR stereo rolling shutter
- Up to 90 FPS RGB
- Range 0.3-10M+
- Includes ROS 2.0 Wrapper

For more info, please visit https://realsense.intel.com



```
[ros2-web-bridge:Bridge] Response: {"op":'set_level','level':"none'}
[ros2-web-bridge:Bridge] JSON command received:
{"op":'advertise','id':"advertise:/cmd_vel:2','type':'geometry_msgs/Twist","topic":"/cmd_vel','latch':
[ros2-web-bridge:Bridge] advertise a topic: /cmd_vel
[rclnodejs:node] Finish creating publisher, topic = /cmd_vel.
[ros2-web-bridge:ResourceProvider] Publisher has been created, and the topic name is /cmd_vel.
```

Web Remote Control App

Robot Movement

MsgType: 'geometry_msgs/msg/Twist'
linear: {x: 0.1, y: 0.0, z: 0.0},
angular: {x: 0.0, y: 0.0, z: 0.0}

Web App w/ roslibjs → ros2-web-bridge → rclnodejs → TurtleBot 3 + ROS 2.0

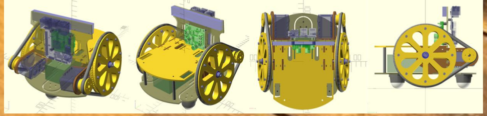
Intel ❤ Robot (Intel's Contribution to Robotics)

- AI/ML/CV Software for ROS 2.0
  - ○ Object detection/segmentation/tracking/velocity estimation & etc.
  - ○ A ROS service to support Intel® OpenVINO™ - the Open Visual Inference & Neural Network Optimization Toolkit.
  - ○ A bridge to connect ROS 2.0 & OpenCV*.
- Movidius^{TM} NCS: dedicated AI hardware by Intel®
  - ○ A ROS service/publisher for object classification and detection
  - ○ Support multiple CNN models of Caffe* and Tensorflow*
- RealSense^{TM} depth camera: perceive the world in 3D
  - ○ Up to 10 meter range, up to 90 fps
  - ○ Realtime 1080p RGB video + 720p depth video
  - ○ Integrated publisher to ROS 2.0, visualized by ROS rviz*
- Better Manipulation with Better ROS MoveIt*
  - ○ Hand-eye calibration
  - ○ Grasp planner (with accelerated grasp detection)
- Redesign of ROS 2.0 Navigation

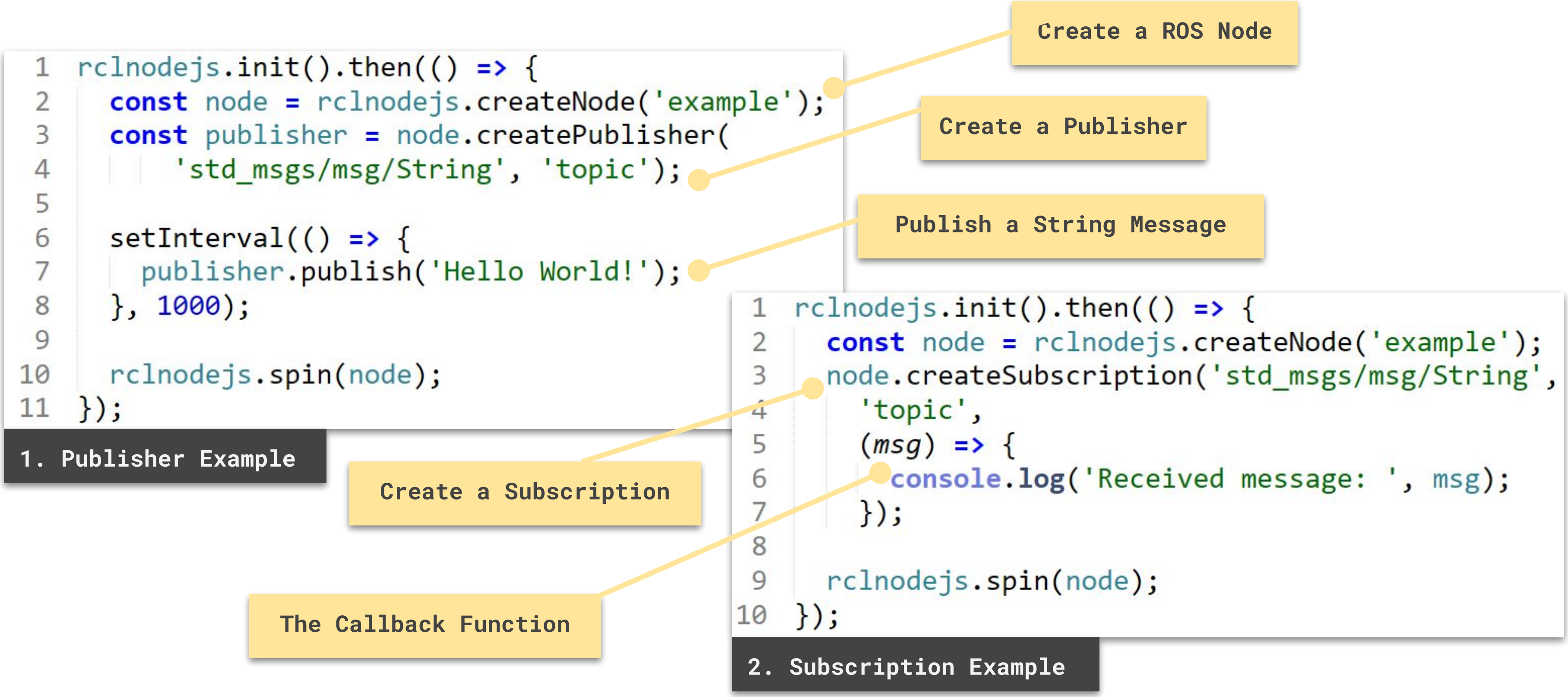
Intel® SAWR robot, both software & hardware are opensource. Simple, inexpensive, built on desktop Ubuntu + ROS, for teaching & learning.

- Opensource chassis or Turtlebot 3
- SLAM capability
- Intel$^{(R)}$ RealSense^{TM} depth camera

SAWR = Simple Autonomous Wheeled Robot



Code Example: Publisher/Subscription



```
rclnodejs.init().then(() => {
  const node = rclnodejs.createNode('example');
  const publisher = node.createPublisher(
      'std_msgs/msg/String', 'topic');

  setInterval(() => {
    publisher.publish('Hello World!');
  }, 1000);

  rclnodejs.spin(node);
});
```

1. Publisher Example

```
rclnodejs.init().then(() => {
  const node = rclnodejs.createNode('example');
  node.createSubscription('std_msgs/msg/String',
    'topic',
    (msg) => {
      console.log('Received message: ', msg);
    });

  rclnodejs.spin(node);
});
```

2. Subscription Example

Create a ROS Node

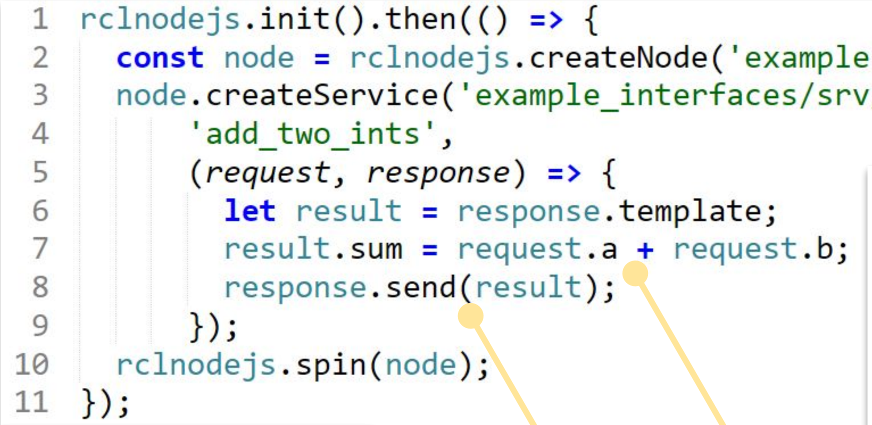
Create a Publisher

Publish a String Message

Create a Subscription

The Callback Function

Code Example: Service/Client



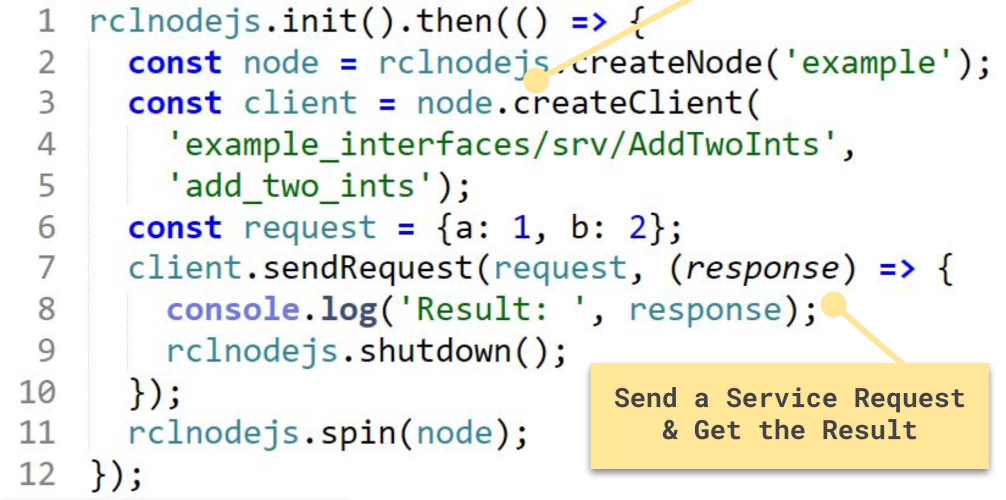
```
rclnodejs.init().then(() => {
  const node = rclnodejs.createNode('example');
  node.createService('example_interfaces/srv/AddTwoInts',
      'add_two_ints',
      (request, response) => {
        let result = response.template;
        result.sum = request.a + request.b;
        response.send(result);
      });
  rclnodejs.spin(node);
});
```

3. Service Example

Create a Service

Get Requested Data

Send Result to Client



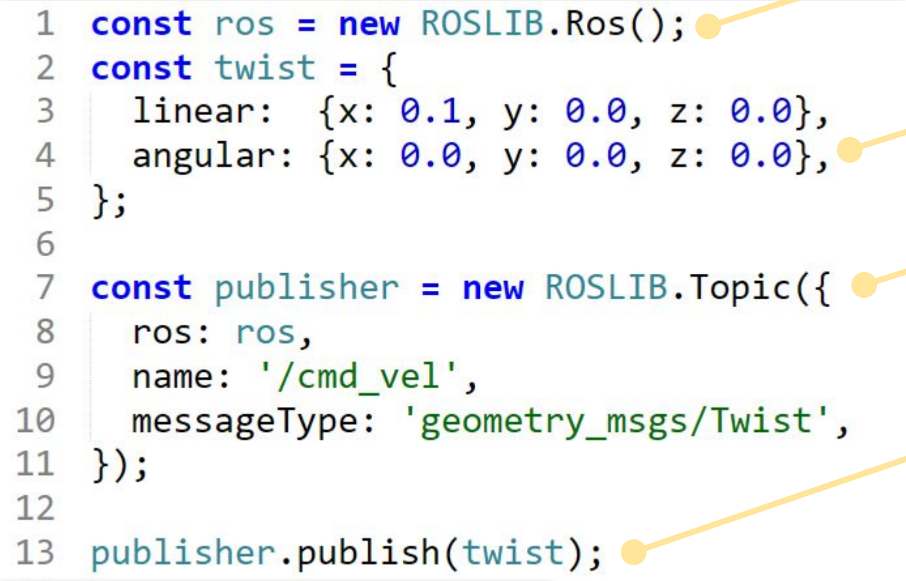
```
rclnodejs.init().then(() => {
  const node = rclnodejs.createNode('example');
  const client = node.createClient(
    'example_interfaces/srv/AddTwoInts',
    'add_two_ints');
  const request = {a: 1, b: 2};
  client.sendRequest(request, (response) => {
    console.log('Result: ', response);
    rclnodejs.shutdown();
  });
  rclnodejs.spin(node);
});
```

4. Client Example

Create a Client

Send a Service Request
& Get the Result

Code Example: ROS in Web Browser



```javascript
const ros = new ROSLIB.Ros();
const twist = {
  linear:  {x: 0.1, y: 0.0, z: 0.0},
  angular: {x: 0.0, y: 0.0, z: 0.0},
};

const publisher = new ROSLIB.Topic({
  ros: ros,
  name: '/cmd_vel',
  messageType: 'geometry_msgs/Twist',
});

publisher.publish(twist);
```

5. ROS in Web Browser Example

A New roslibjs Instance

Define a Twist Message

Create a Publisher in Browser

Publish the Twist Message.

This message will be sent to ROS 2.0 via ros2-web-bridge.

Contacts & Resource Links

Contacts: Minggang Wang

email: minggang.wang@intel.com

Useful links:

- rclnodejs: github, npm
- ros2-web-bridge: github, npm
- Intel ROS 2.0 projects: wiki (also 1.0)
- Robot Web Tools: libs/widgets/systems/etc.
- rosnodejs by RethinkRobotics* for ROS 1.0

The developer/QA team

- Minggang Wang
- Kenny Yuan
- Wanming Lin
- Yi Han
- Zhong Qiu

Questions...

intel®

Software

Legal Notices and Disclaimers

•Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

•Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

•Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/performance.

•Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

•This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

•No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
•Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

•All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

•Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.