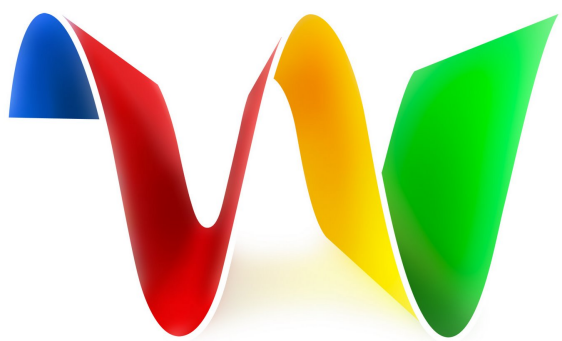

Google™
10

Making smart and scalable Wave robots

David Mark Byttow
Marcel Prasetya
5-20-10



View live notes and ask questions about this session on Google Wave!

http://bit.ly/robots-io2010

Google 10

What are we going to cover?

- Robot API Overview
  - How does it work?
  - What's new in V2?
- Super-powered Robots
  - New features and best practices
  - Deep dive by example
- Examples
  - Disassembling unique robots
- Robot Liberation

Wave in 2 minutes or less

Google 10 I/O

Google Wave - Get stuff done with groups of people

Simple Sharing Model

Reply Anywhere

Live Editing

Custom Extensions

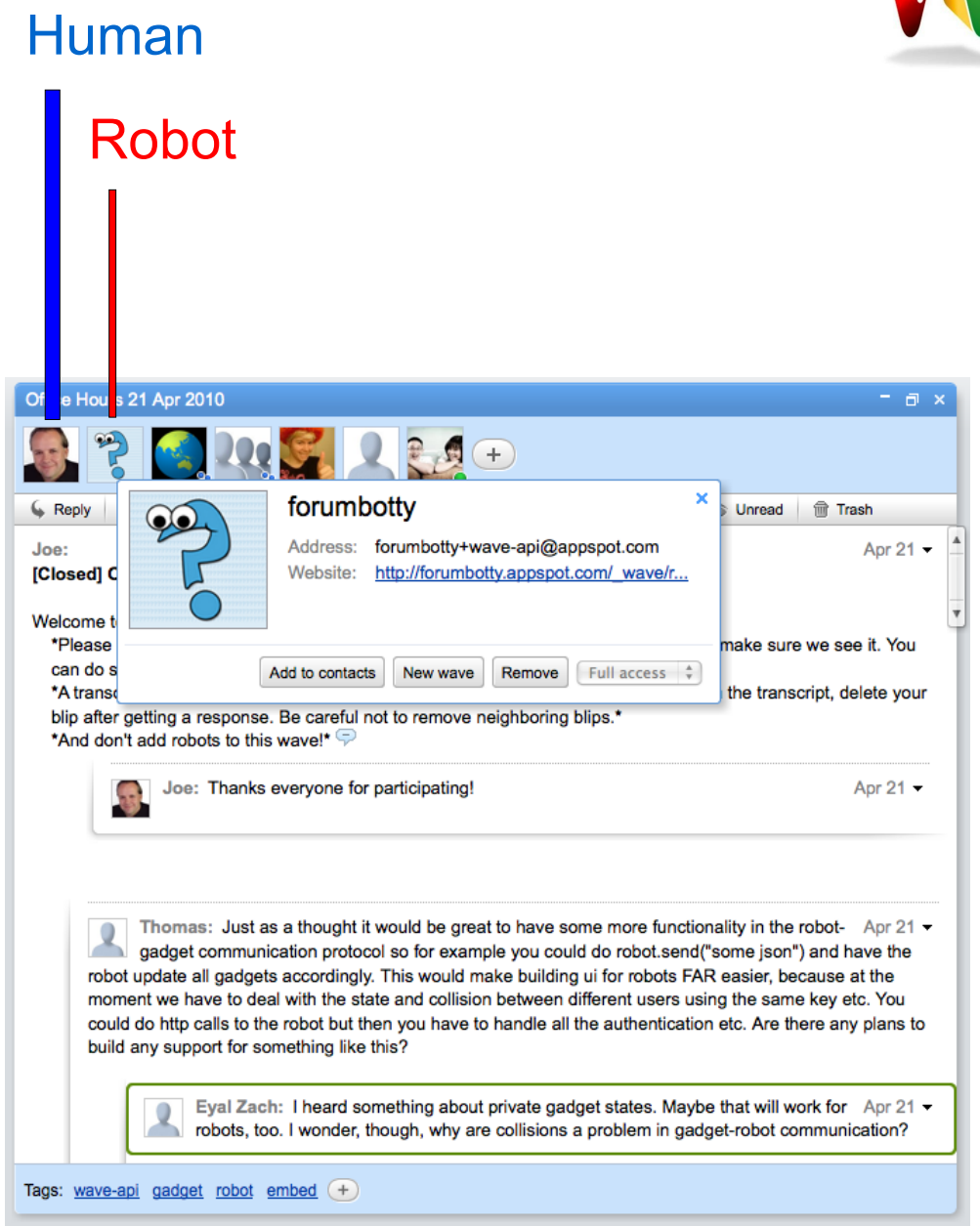
A wave is equal parts conversation and document

Google I/O 10

Robot API Overview

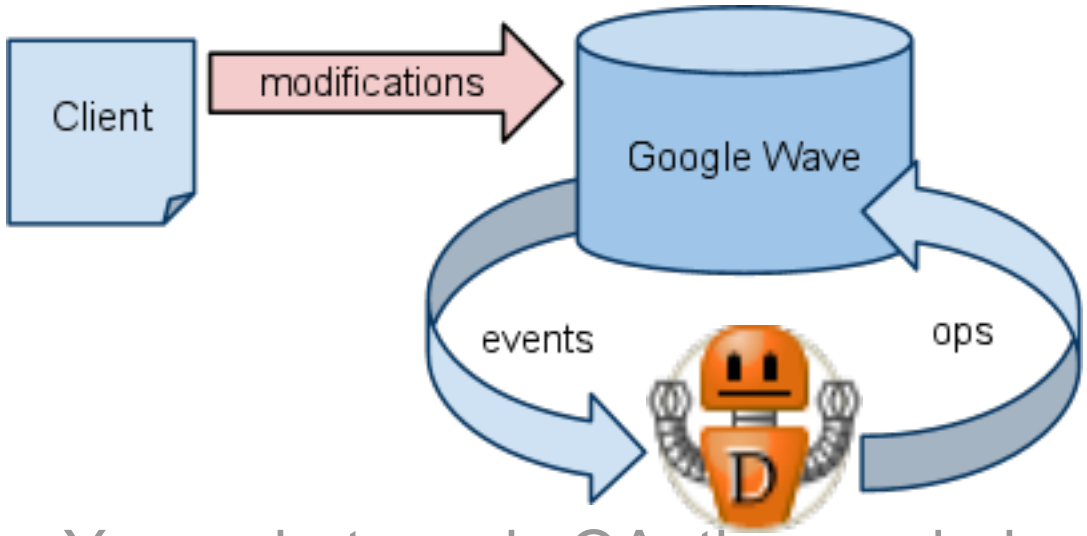
What's a Robot again?

- Automated participants using an HTTP-based, JSON protocol to listen to events and respond with operations
- In the Wave world, robots and humans have equal rights.

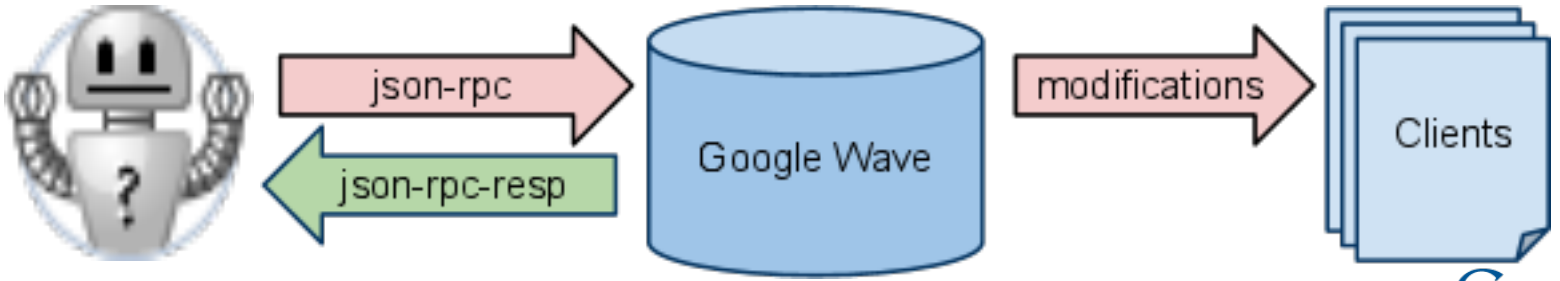


Robot Protocol - Overview

- **Passive** - Google Wave sends events for any wave your robot is a participant of and interested in

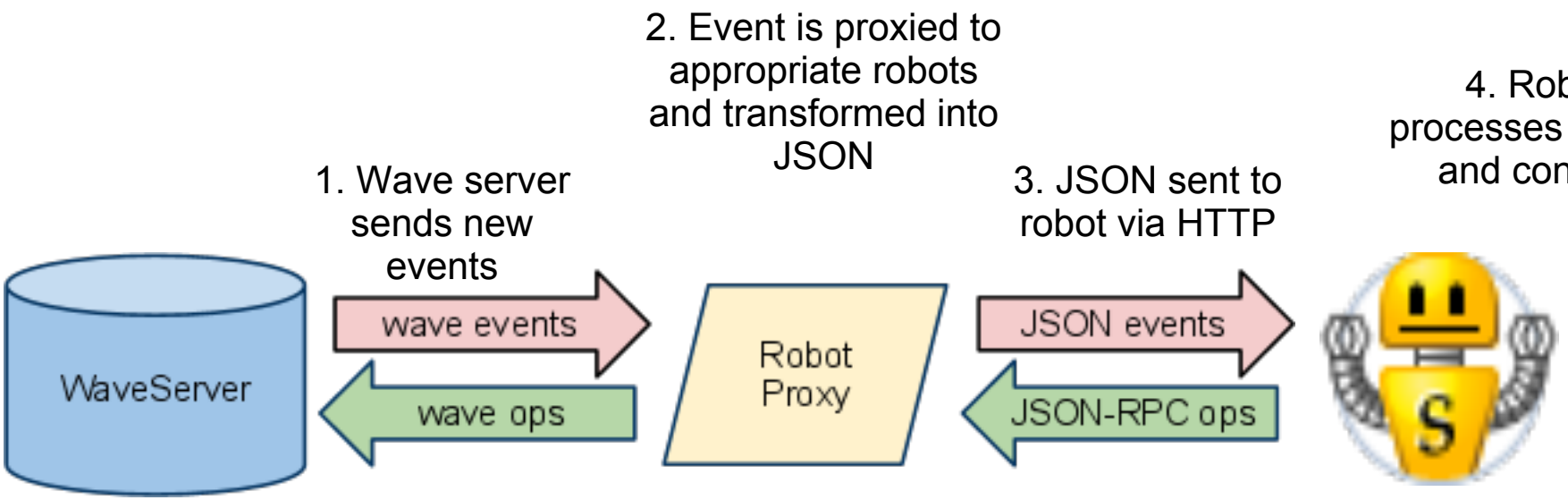


- **Active** - Your robot sends OAuth-encoded requests to Google Wave and receives responses based on the requests



Robot Protocol - Events

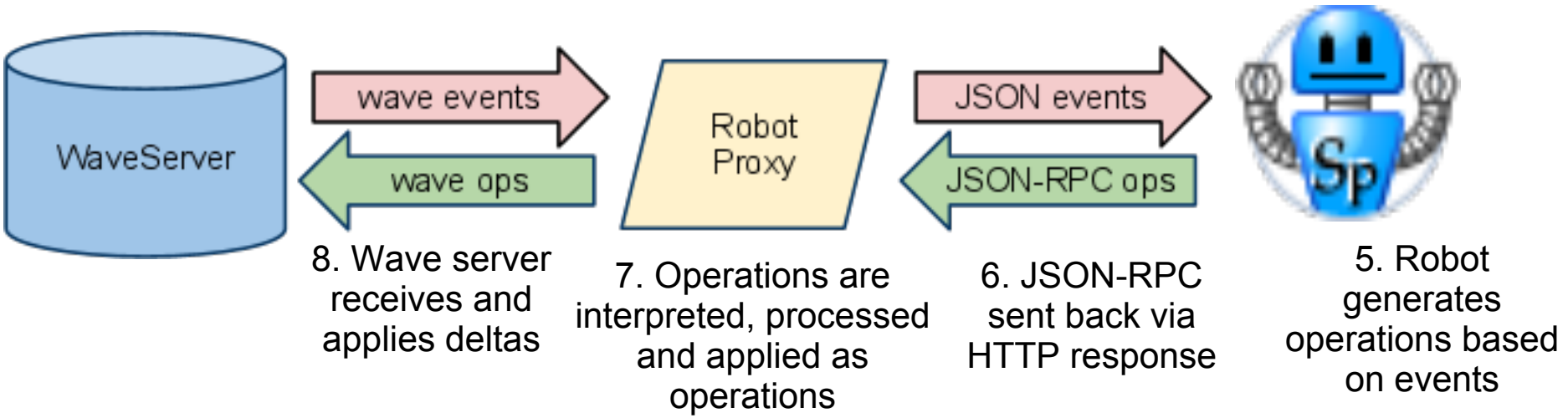
- Robots receive events from the Google Wave server when waves that they are participants of are modified
  - Incoming event bundle is in JSON format



- Event JSON example: http://code.google.com/apis/wave/extensions/robots/protocol.html#MessageBundles

Robot Protocol - Operations

- Robots receive operations from the Google Wave server to modify waves
  - Outgoing event bundle is in JSON-RPC format, and applied in order



- Example operation JSON:

http://code.google.com/apis/wave/extensions/robots/protocol.html#Operations

Client Libraries

- Open-sourced Python and Java client libraries
- Client libraries
  - Handle event and operation bundle serialization to and from JSON into objects
  - Abstract the Wave model into classes
  - Enables developers to focus on the functionality instead of the implementation details of the JSON wire protocol
- What about PHP, Ruby, Perl, C#, Lisp, etc?
  - We started with App Engine as the platform for robots, so we developed client libraries only for the App Engine runtime environments (Python and Java)... *more on that later.*

Client Libraries - "Hello World" in Python

```
from waveapi import events
from waveapi import robot
from waveapi import appengine_robot_runner

def OnWaveletSelfAdded(event, wavelet):
  wavelet.reply("\nHi everybody! I'm a Python robot!")

def OnWaveletParticipantsChanged(event, wavelet):
  for newParticipant in event.participants_added:
    wavelet.reply("\nHi : " + newParticipant)

if __name__ == '__main__':
  myRobot = robot.Robot('Examply',
      image_url='http://example.appspot.com/icon.png',
      profile_url='http://example.appspot.com/')

  myRobot.register_handler(events.WaveletParticipantsChanged,
      OnWaveletParticipantsChanged)
  myRobot.register_handler(events.WaveletSelfAdded,
      OnWaveletSelfAdded)
  appengine_robot_runner.run(myRobot)
```

- Say hello when added to the wave
- Say hello to new participants
- Robot profile setup
- Robot event listeners

Robot API V2 - What's new?

- **More robust operations**
  Versatile operations that run both on client and server to reduce client-side complexity and snapshot synchronization
- **Bandwidth Control**
  Robots have more flexibility in controlling the frequency and payload size of incoming event requests
- **Profile semantics**
  Robots can override their own profile images and names (e.g. to act on behalf of users)

- **Data Liberation**
  Robots may search across and fetch wavelets that they have access to
- ... and more

Robot API Best Practices

Safely search and replace content with **document.modify** operation

- In V1, most of our operations were index and range based, which result in a fragile API
- Example on replacing all occurrences of "foo" with "bar" in V1

```
TextView textView = blip.getDocument();
String toFind = "foo"; int index;
while ((index = textView.getText().indexOf(toFind)) != -1) {
  Range range = new Range(index, index + toFind.length());
  textView.replace(range, "bar");
}
```

- In V2, we introduce the document.modify operation that can perform actions without using index as the reference

```
blip.all("foo").replace("bar");
```

Safely search and replace content with **document.modify** operation (continued)

```
blip.all("foo").replace("bar");
```

selector *action*

- *Selector* restricts or target the operation. *Action* modifies the matched regions

Selector
all
first
at
range

Action
insert
insertAfter
replace
delete
annotate

```
blip.all("Google").annotate("link/manual", "http://www.google.com");
blip.all(ElementType.GADGET).replace("[blocked]");
blip.first(ElementType.IMAGE).insertAfter("(tm)");
blip.range(6, 12).delete();
```

Being frugal with context and filter

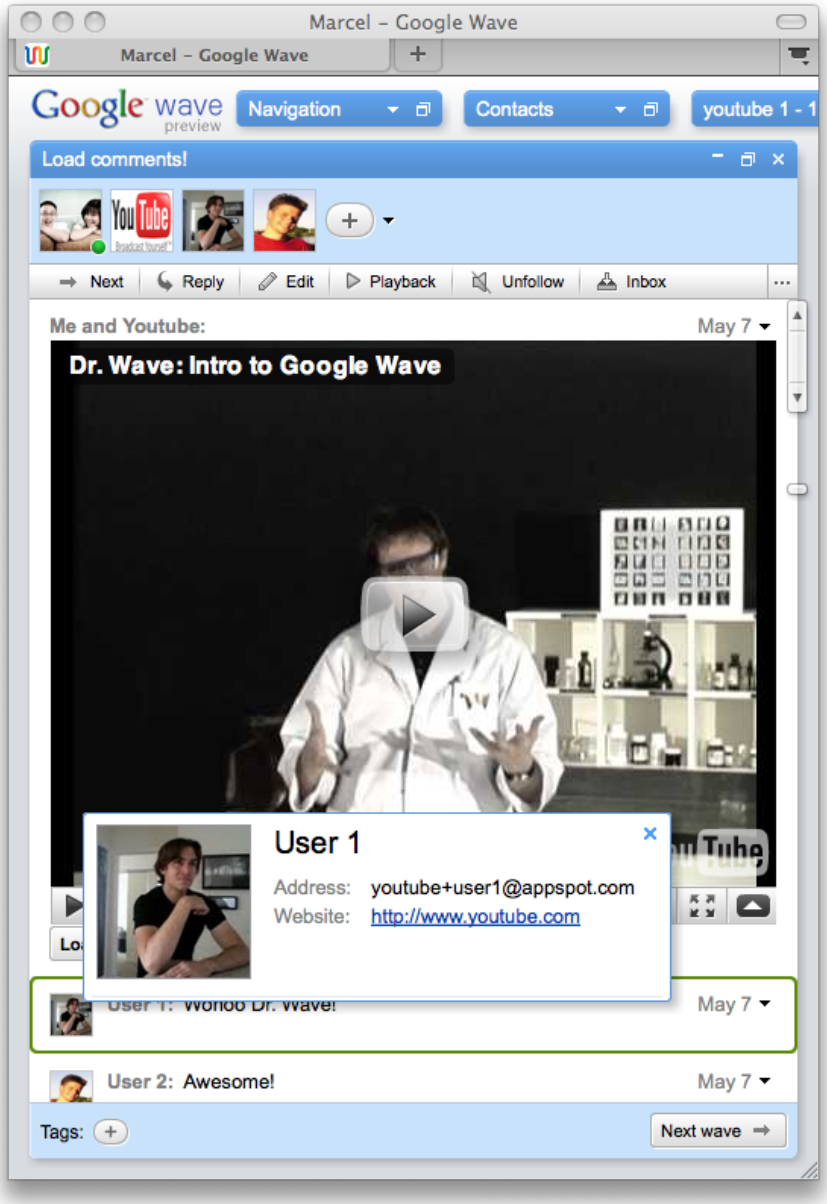
- *Context* specifies which blips should be included in the event bundle. Available context: *ROOT*, *PARENT*, SELF, *SIBLINGS*, *CHILDREN*, and *ALL*
- *Filter* tells the server to only send event which properties match the given filter
- If your robot is subscribed to a chatty event, setting the proper context and filters could help to reduce the bandwidth and CPU consumption for your robot

```
@Capability(contexts = {Context.PARENT, Context.SELF},
        filter = "\\[\\[Google\\]\\]")
public void onDocumentChanged(DocumentChangedEvent e) {
  ...
}
```

Robot participant namespacing with proxying-for

- A single robot proxies/represents many participants with *Proxying-For*, specified in the participant id (<robotid>+**<proxyid>**@appspot.com)
- Robot that acts as a gateway for another service can use this feature to send operations on behalf of the service users
- A Youtube robot youtube@appspot.com can send replies to the wave on behalf of user1 and the reply will be authored by youtube+user1@appspot.com

Robot participant namespacing with proxying-for (continued)




```
// Get comments from Youtube service.
String vidId = "YiGdUmvPRy8";
List<Comment> comments = getComments(vidId);

// Post the comments to the wave.
for (Comment comment : comments) {
  // Setup a proxied wavelet.
  String user = comment.getUser();
  Wavelet proxied = wavelet.proxyFor(user);

  // Make the reply.
  proxiedWavelet.reply(comment.getText());
}
```

Google I/O 10

Actively pushing data into Wave

- In V1, robots can only send operations in response to events. With the *Active API*, robot can send operations outside the event loop
- A gateway robot can use Active API to update a wave if there's an event on an external service

```
public void cronHandler() {
  // Setup OAuth.
  robot.setupOAuth(Credentials.CONS_KEY, Credentials.CONS_SECRET);

  // Create the stub wavelet.
  Wavelet wavelet = robot.blindWavelet(waveId, waveletId);
  wavelet.setRobotAddress("youtube@appspot.com");

  // Post the new comments to wave.
  for (Comment comment : getComments(videoId)) {
    wavelet.proxyFor(comment.getUser()).reply(comment.getText());
  }

  // Submit the pending operations.
  String url = "https://www-opensocial.googleusercontent.com/api/rpc";
  robot.submit(wavelet, url);
}
```

Robot Examples - Disassemble!

Monty python™

- Evaluates and executes Python code
- Outputs the result directly in wave
- Bootstraps the Python API itself allowing for operations to be generated directly in Wave
- Features used:
  - WaveletSelfAdded and BlipSubmitted events
  - Document.modify to replace content
- Source code

Google I/O 10

Mr Ray

- Allows non-wave users to participant in conversations on Wave.
- Syncs waves with the robot and creates a usable, alternative client for non-Wave users to participate in.
- **1st place** in Mashable's Google Wave API Contest!
- Features used:
  - Proxy-For, used to represent non-wave users
  - Gadget for control flow communication

Google 10 I/O

Ferry

- Exports waves to Google Docs
- Syncs changes to wave with created document
- Features used:
  - Fetch wavelet, to retrieve wave contents
  - Gadget element, as a control panel
  - OAuth, for Google Docs access

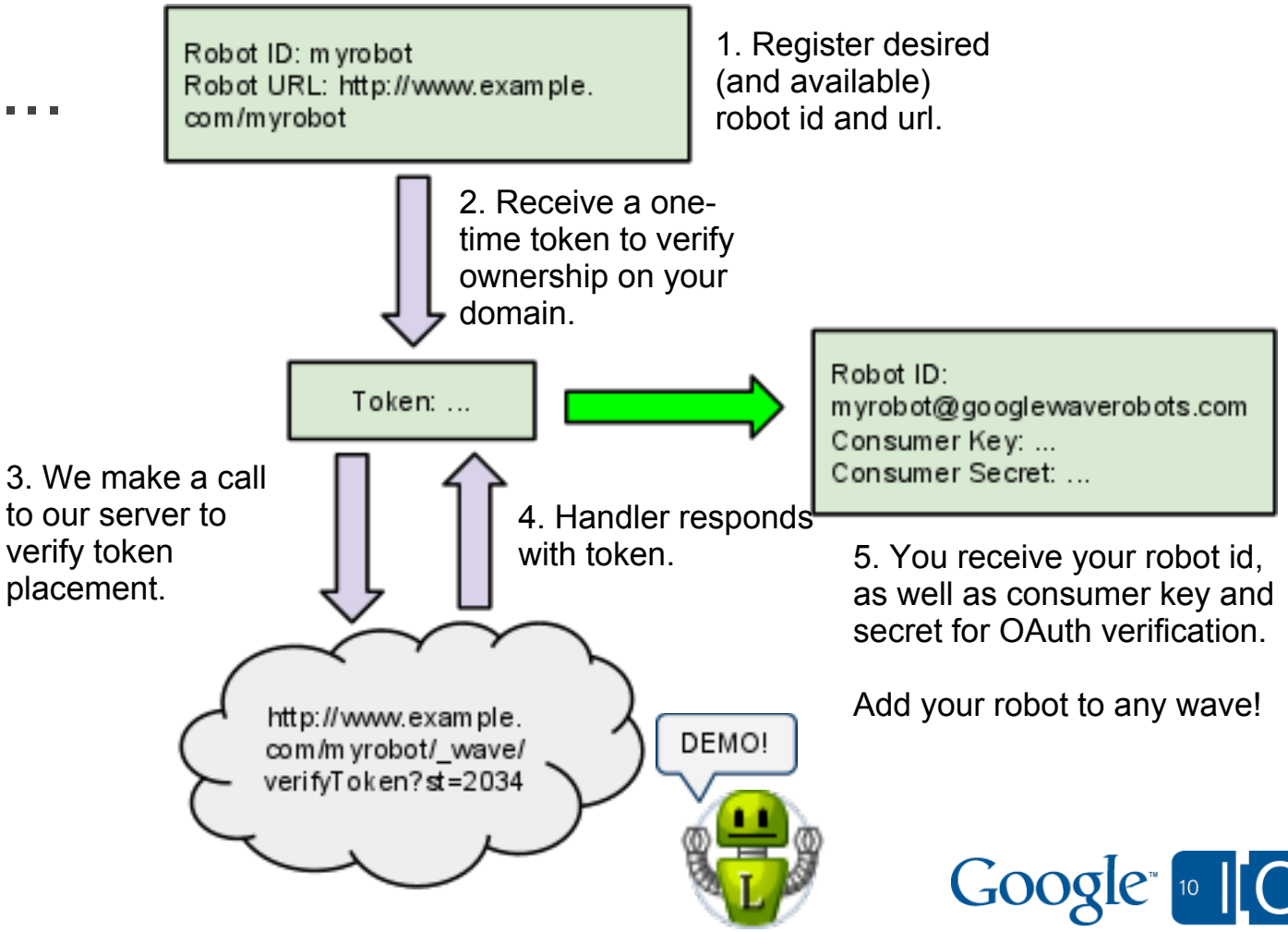
Google 10 I/O

Robot Liberation!

Robot Liberation - Viva la Revolution!

- We now support running robots on your own domain
- Yes, you can still run your robots on App Engine

How it works...



Liberated Robot Example - GoBot

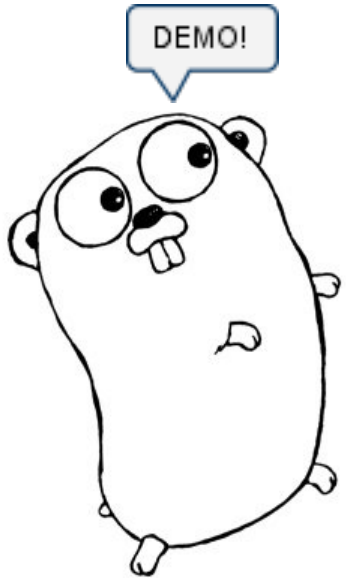
- Using the new, open-source Go Programming Language developed at Google.

```
package main

import (
  "http"
  "waveapi"
)

func handlerFunc(e *waveapi.Event, w *waveapi.Wavelet) {
  w.Reply("Hello World!")
}

func main() {
  r := waveapi.NewRobot(
    "Hello robot",
    "http://exmaple.com/avatar.png",
    "http://example.com/profile.html",
    "")
  r.RegisterHandler(waveapi.E_WaveletSelfAdded, handlerFunc)
  http.ListenAndServe(":8080", r)
}
```



Google 10 I/O

Robots can speak any language... with help

- Javascript
- Lisp
- Perl
- php
- Ruby
- C#
- Objective-C
- coldfusion
-

Get started today

http://code.google.com/apis/wave

Thank you for listening!

Questions?

http://bit.ly/robots-io2010

Google 10


Google™
10