

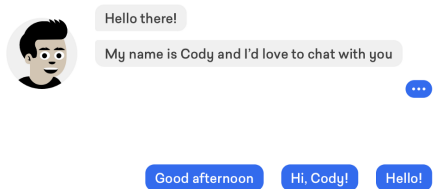
# From Documents to Dialogues

Programming Technology for the Conversational Web

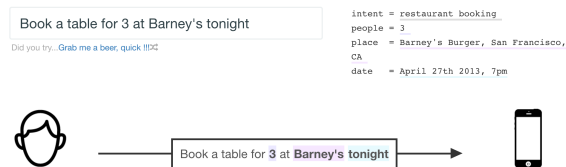
Tom Van Cutsem

ProWeb Workshop @ <Programming>, April 2017

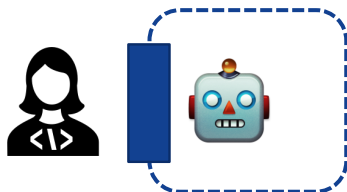
# Talk overview



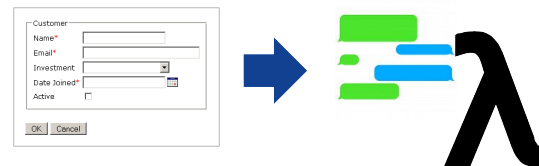
Conversational Web



Natural language is the new UI



Programming bots

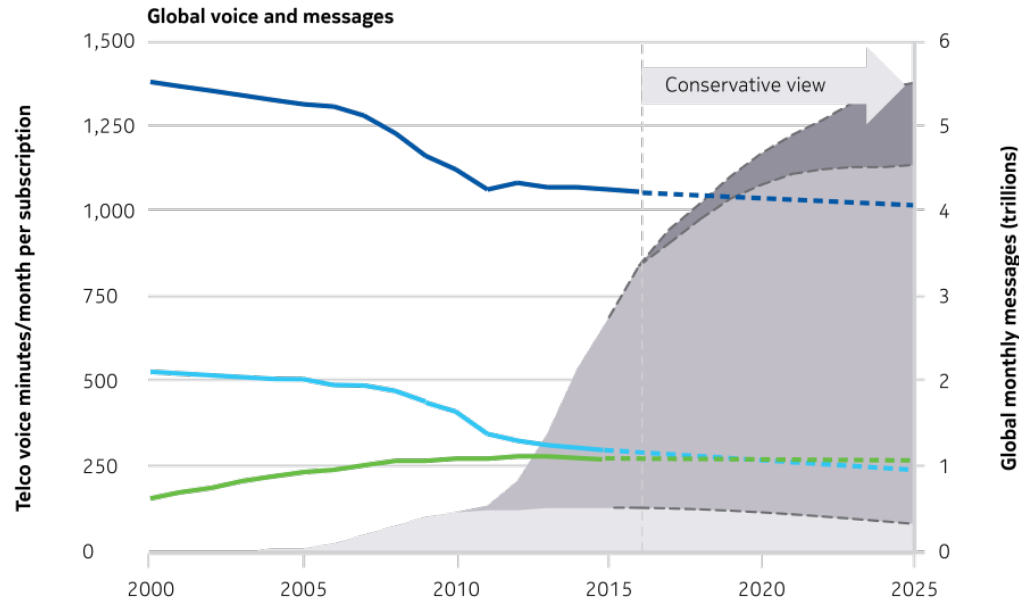


From documents to dialogues:  
New opportunities for PLT

# Conversational Web

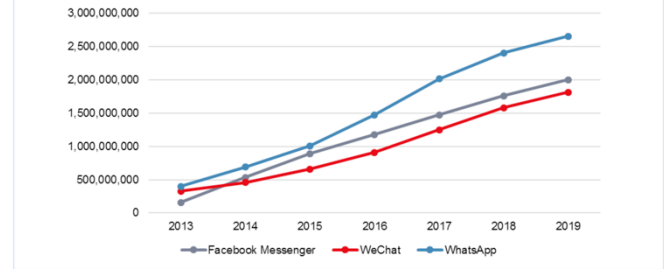
# How did we get here?

## The explosion of messaging and the decline of synchronous voice



(Source: M. Weldon, *The Future X Network*, CRC Press)

Figure 3: Selected chat apps, monthly active users, 2013–19



Source: Ovum OTT Communications Tracker, 4Q16

- Telecom service provider fixed business voice
- Telecom service provider mobile voice
- Telecom service provider fixed residential voice
- SMS
- OTT messages\*
- Telecom service provider advanced messaging (e.g. RCS, etc.)

\*WhatsApp, WeChat, LINE, BlackBerry, Apple, KakaoTalk, Snapchat and Viber Media

# Driven by Millennials

Best Ways for Businesses to Contact Millennials = Social Media & Chat...  
Worst Way = Telephone

## Popularity of Business Contact Channels, by Age

*Which channels are most popular with your age-profiled customers?  
(% of contact centers)*

	% of Centers Reporting Most Popular Contact Channels by Generation				
	Internet / Web Chat	Social Media	Electronic Messaging (e.g., email, SMS)	Smartphone Application	Telephone
<b>Generation Y</b> (born 1981-1999)	24% (1 <sup>st</sup> choice)	24% (1 <sup>st</sup> choice)	21% (3 <sup>rd</sup> choice)	19% (4 <sup>th</sup> choice)	12% (5 <sup>th</sup> choice)
<b>Generation X</b> (born 1961-1980)	21% (3 <sup>rd</sup> choice)	12% (4 <sup>th</sup> choice)	28% (2 <sup>nd</sup> choice)	11% (5 <sup>th</sup> choice)	29% (1 <sup>st</sup> choice)
<b>Baby Boomers</b> (born 1945-1960)	7% (3 <sup>rd</sup> choice)	2% (5 <sup>th</sup> choice)	24% (2 <sup>nd</sup> choice)	3% (4 <sup>th</sup> choice)	64% (1 <sup>st</sup> choice)
<b>Silent Generation</b> (born before 1944)	2% (3 <sup>rd</sup> choice)	1% (4 <sup>th</sup> choice)	6% (2 <sup>nd</sup> choice)	1% (5 <sup>th</sup> choice)	90% (1 <sup>st</sup> choice)



Source: "Global Contact Center Service Tracking Report" Generation Data, 2016.  
N = 751 CC contact centers. Global. Results are based on contact centers that actually have contact center presence. Percentage may not add up to 100% owing to rounding.  
Generation Y is generally defined to be "Millennials".

KPCB INTERNET TRENDS 2016 | PAGE  
107

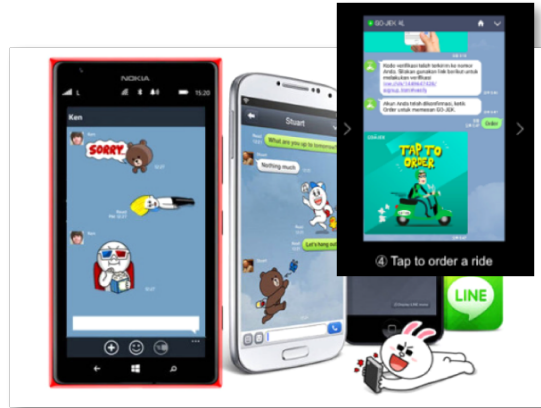
(Source: Mary Meeker's Internet Trends 2016)

# Messaging is easy to integrate with all kinds of services

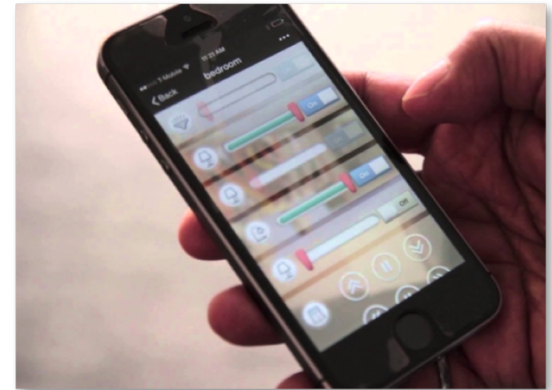
## “Conversational Commerce”



Social commerce



Embed games  
Order taxis



Smart hotel rooms

# Meet Tacobot

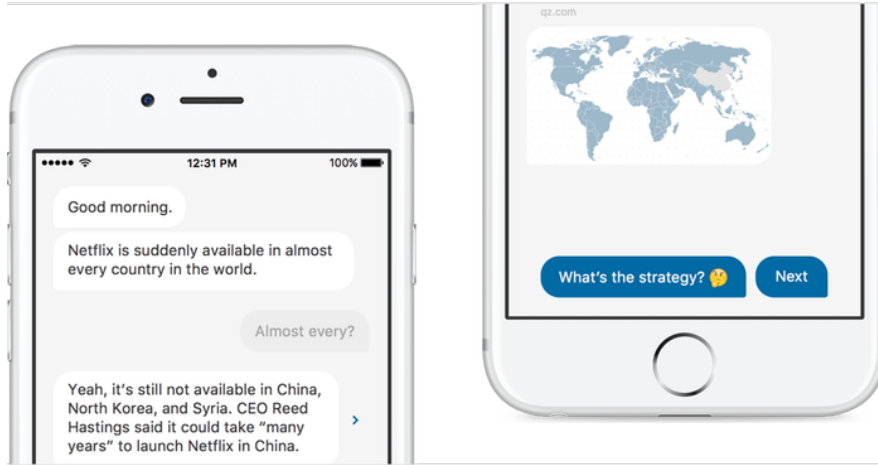
Today

Hey there Martin, I am your TacoBot. I can help you order a meal for you or your team.

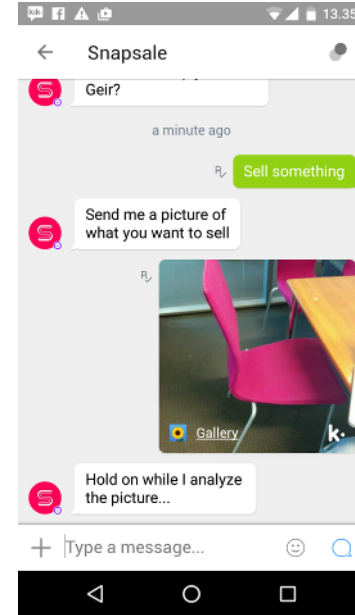
+

😊

# Interesting new use cases



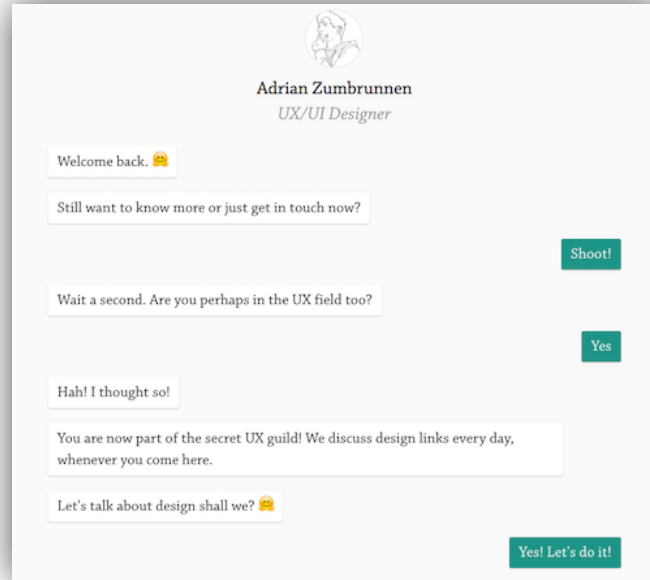
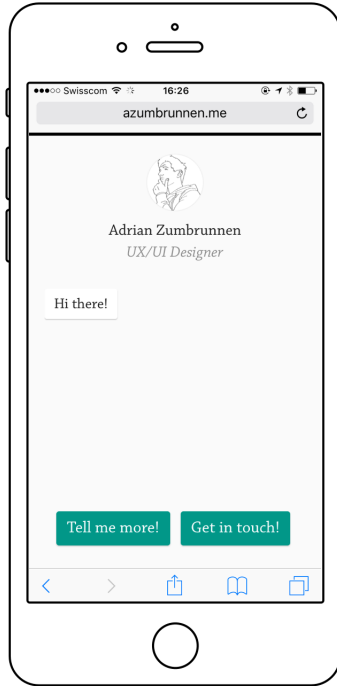
Quartz



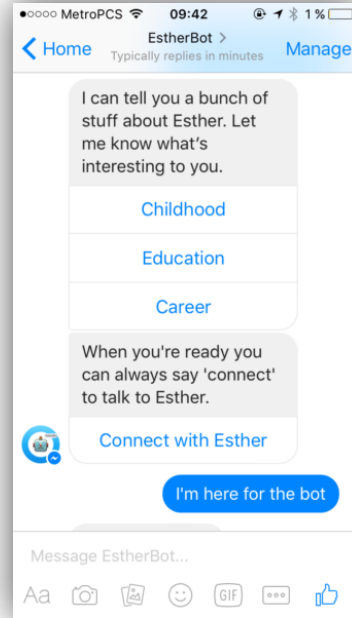
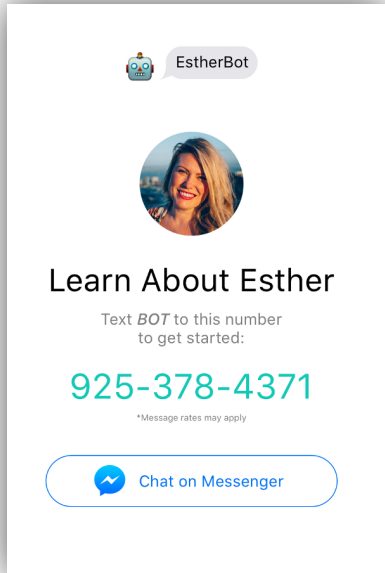
Snapsale



# Interesting new use cases

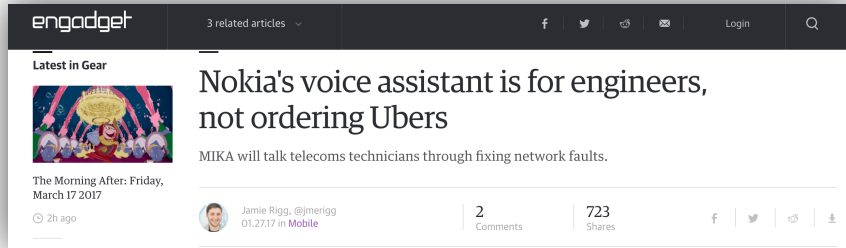


# Interesting new use cases

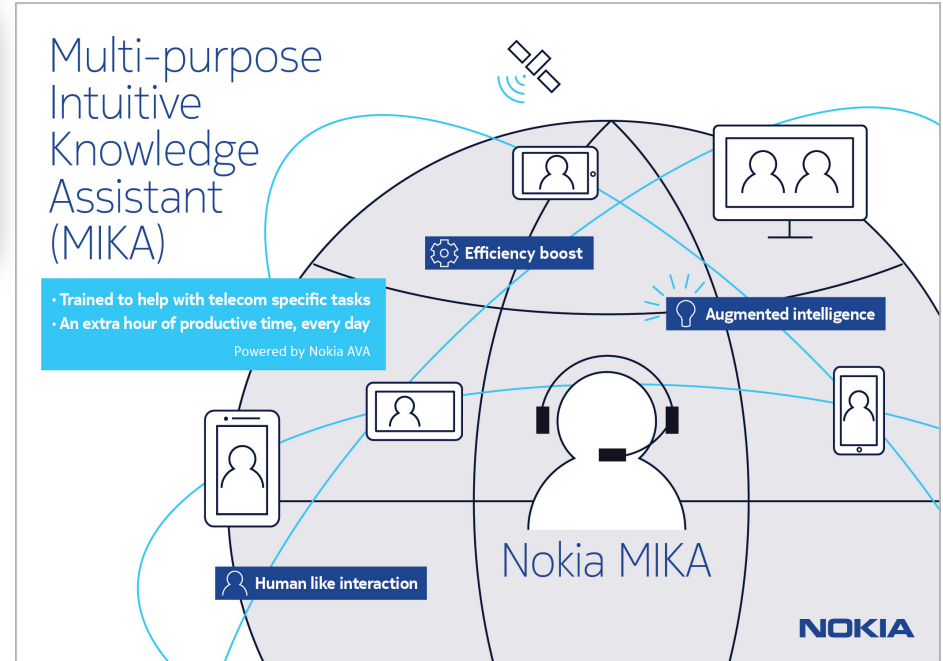


# Bots are not just for consumers

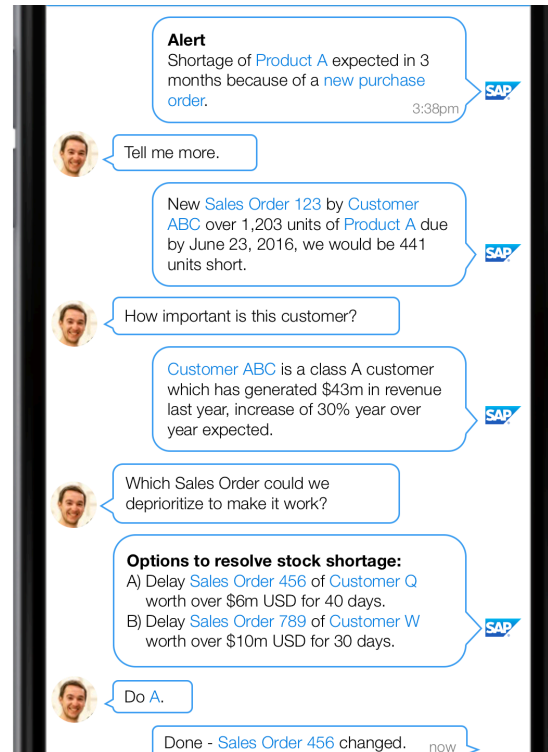
Nokia's MIKA is a digital assistant customized for telco field engineers



"It's unlikely you or I will ever see it in action first-hand, of course, but when your 4G connection unexpectedly dies later this year, MIKA may well be on the case." (Engadget)

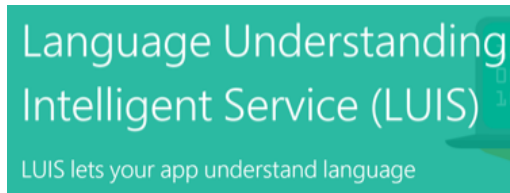


# Chatbots in the Enterprise



(Source: <https://blogs.sap.com/2016/05/18/chatbots-meet-enterprise-software/>)

# Natural Language Understanding has become a commodity



Book a table for 3 at Barney's tonight

Did you try...[Grab me a beer, quick !!!](#)

```
intent = restaurant_booking
people = 3
place = Barney's Burger, San Francisco, CA
date = April 27th 2013, 7pm
```



Book a table for 3 at Barney's tonight



Set the temperature to 70° in my bedroom

Did you try...[Remind me to feed the baby tomorrow at 7am](#)

```
intent = heating_control
temperature = 70°F
where = master_bedroom
```



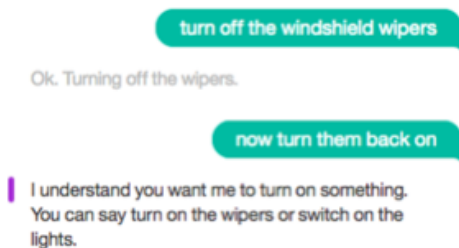
Set the temperature to 70° in my bedroom



(Source: wit.ai)

# Natural language Understanding pitfalls

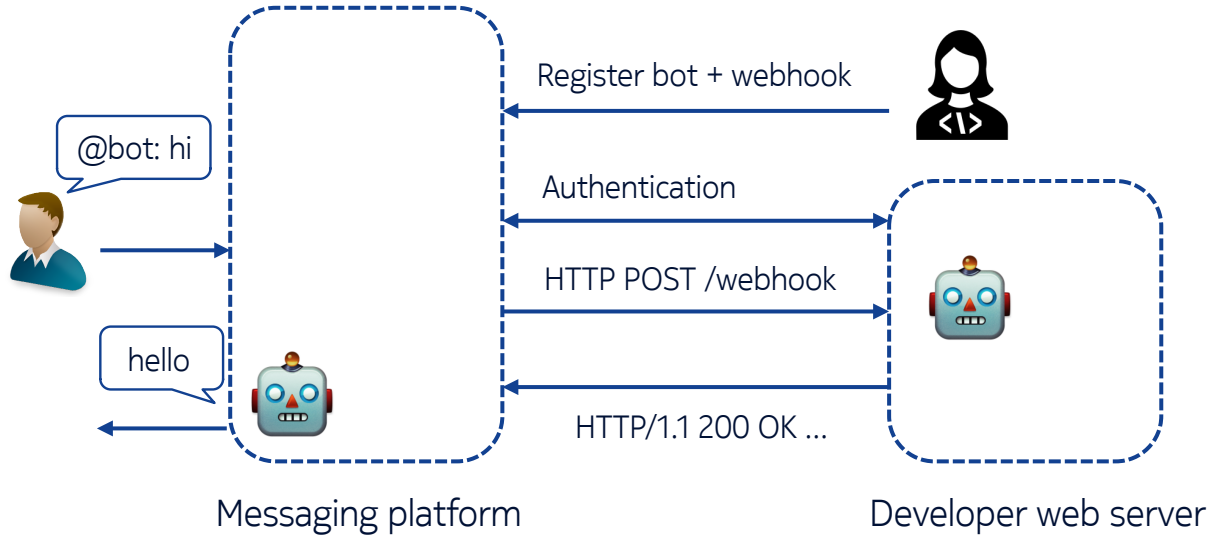
- NLU techniques are getting more powerful, but are far from foolproof.
- Conversation context is key!



(Source: <https://techcrunch.com/2017/02/25/conversational-ai-and-the-road-ahead/>)

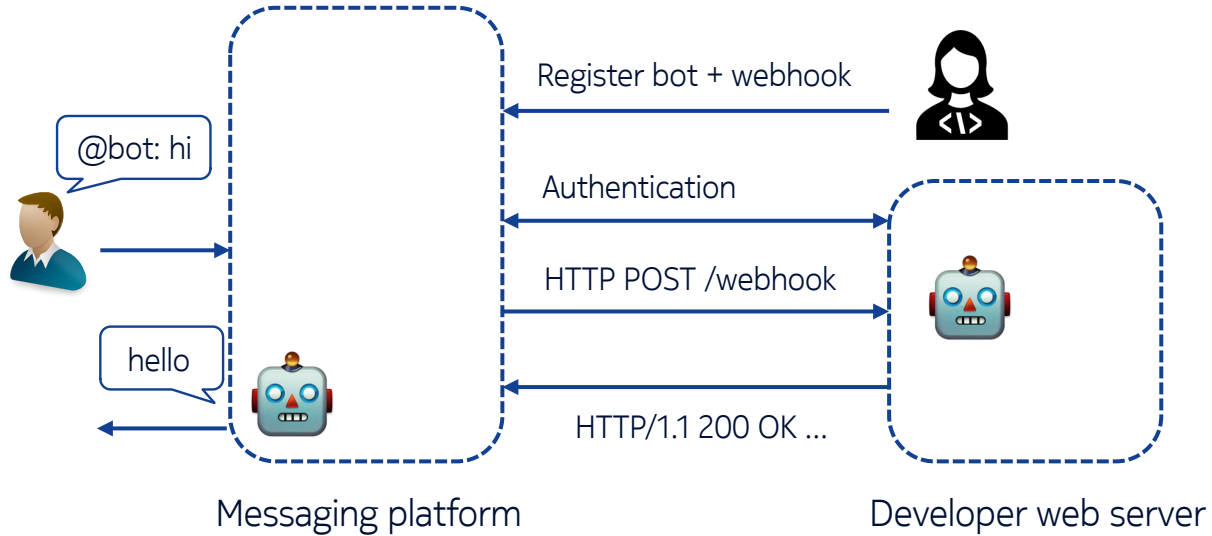
# Programming Chatbots

# Basic Bot Architecture

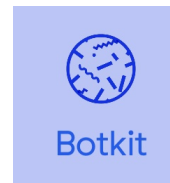
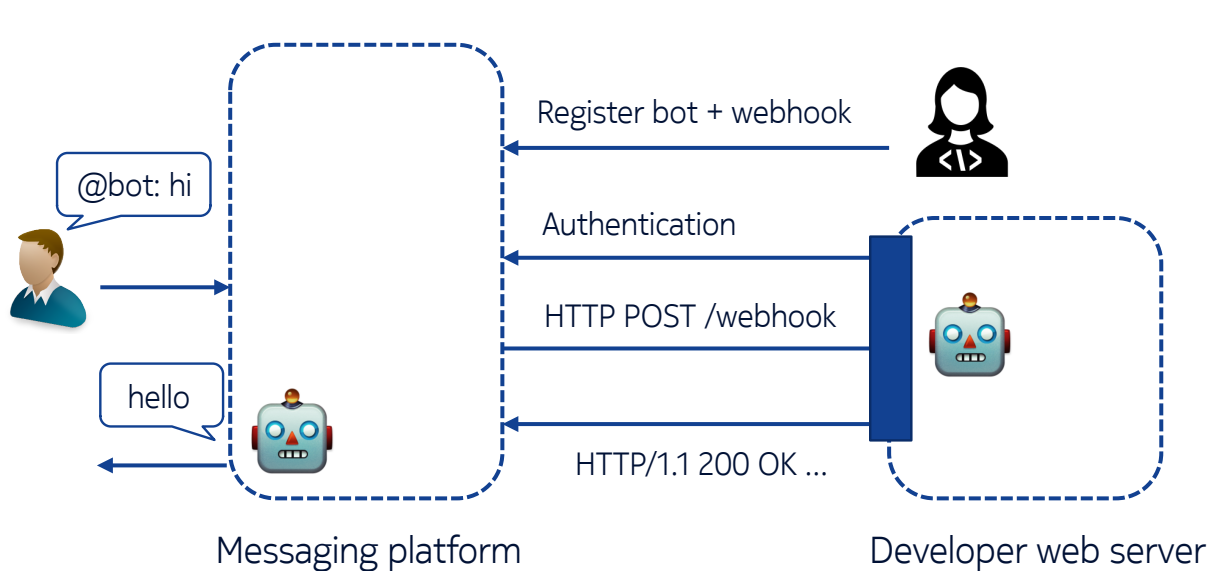




# Basic Bot Architecture



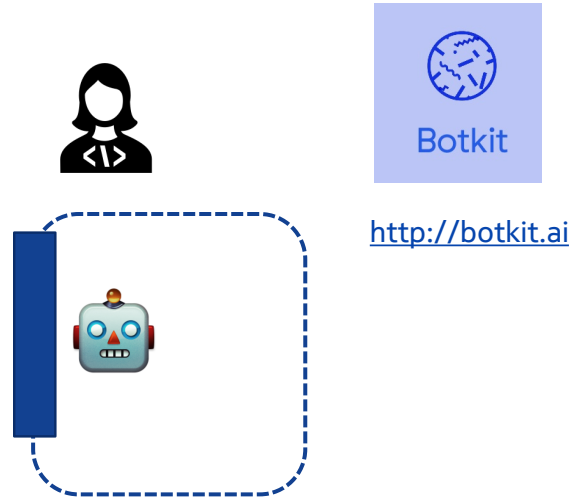
# Bot Frameworks / SDKs



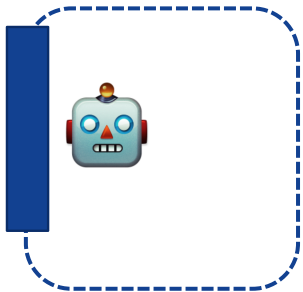
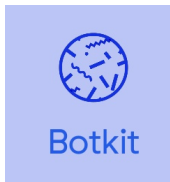
SuperScript



# Bot Frameworks / SDKs



# Bot Frameworks / SDKs: BotKit



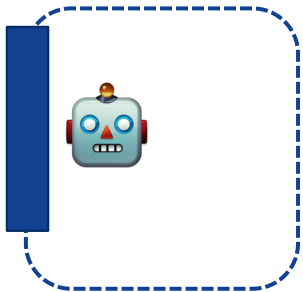
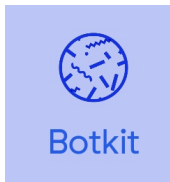
```
var Botkit = require('botkit');

var controller = Botkit.slackbot({...});

// connect the bot to a stream of messages
controller.spawn({
  token: <my_slack_bot_token>,
}).startRTM();

controller.hears(
  ['hello', 'hi'],
  'direct_message,direct_mention,mention',
  function(bot, message) {
    controller.storage.users.get(message.user, function(err, user) {
      if (user && user.name) {
        bot.reply(message, 'hello ' + user.name + '!');
      } else {
        bot.reply(message, 'hello');
      }
    });
  });
```

# Bot Frameworks / SDKs: BotKit



```
var Botkit = require('botkit');

var controller = Botkit.slackbot({...});

// connect the bot to a stream of messages
controller.spawn({
  token: <my_slack_bot_token>,
}).startRTM();

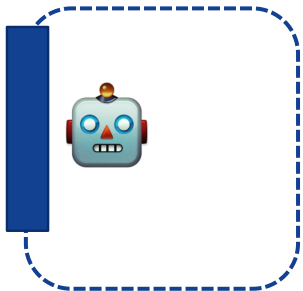
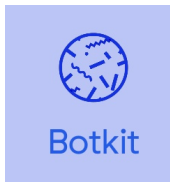
controller.hears(
  ['hello', 'hi'],
  'direct_message,direct_mention,mention',
  function(bot, message) {
    controller.storage.users.get(message.user, function(err, user) {
      if (user && user.name) {
        bot.reply(message, 'hello ' + user.name + '!');
      } else {
        bot.reply(message, 'hello');
      }
    });
  });
```



Config is different  
for each platform

# Bot Frameworks / SDKs: BotKit

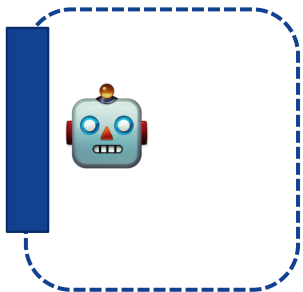
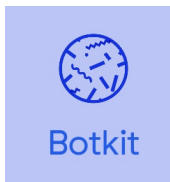
## Matching patterns



```
controller.hears([
  'call me (.*)', 'my name is (.*)'],
  'direct_message,direct_mention,mention',
  function(bot, message) {
    var name = message.match[1];
    controller.storage.users.get(message.user, function(err, user) {
      if (!user) {
        user = { id: message.user };
      }
      user.name = name;
      controller.storage.users.save(user, function(err, id) {
        bot.reply(message,
          'Got it. I will call you ' + user.name + ' from now on.');
```

# Bot Frameworks / SDKs: BotKit

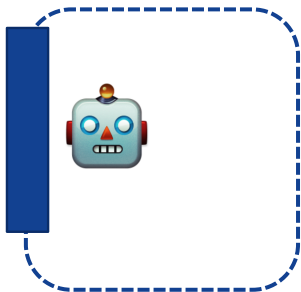
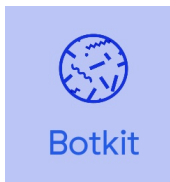
## Questioning the user: conversations



```
controller.hears(  
  ['shutdown'],  
  'direct_message,direct_mention,mention', function(bot, message) {  
    bot.startConversation(message, function(err, convo) {  
      convo.ask('Are you sure you want me to shutdown?', [  
        { pattern: bot.utterances.yes,  
          callback: function(response, convo) {  
            convo.say('Bye!');  
            convo.next();  
            setTimeout(() => process.exit(), 3000);  
          }  
        },  
      ],  
      {  
        pattern: bot.utterances.no,  
        default: true,  
        callback: function(response, convo) {  
          convo.say('*Phew!*');  
          convo.next();  
        }  
      }  
    );  
  });  
});
```

# Bot Frameworks / SDKs: BotKit

## Questioning the user: conversations



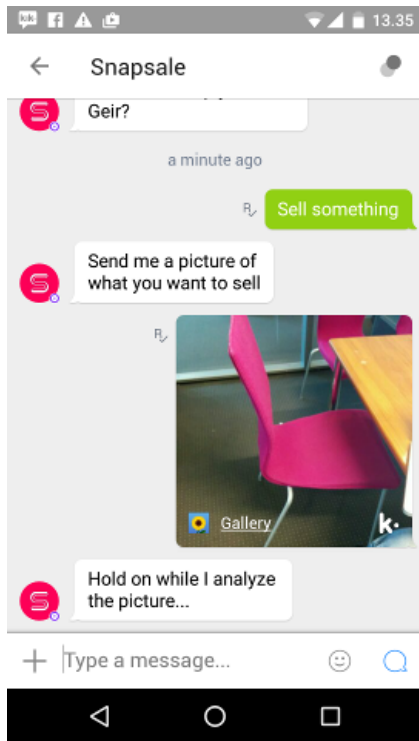
```
controller.hears(  
  ['shutdown'],  
  'direct_message,direct_mention,mention', function(bot, message) {  
    bot.startConversation(message, function(err, convo) {  
      convo.ask('Are you sure you want me to shutdown?', [  
        { pattern: bot.utterances.yes,  
          callback: function(response, convo) {  
            convo.say('Bye!');  
            convo.next();  
            setTimeout(() => process.exit(), 3000);  
          }  
        },  
      ],  
      {  
        pattern: bot.utterances.no,  
        default: true,  
        callback: function(response, convo) {  
          convo.say('*Phew!*');  
          convo.next();  
        }  
      }  
    );  
  });  
});
```



No intelligence: next message from user is treated as response



# Key observation: bots are state machines



## States

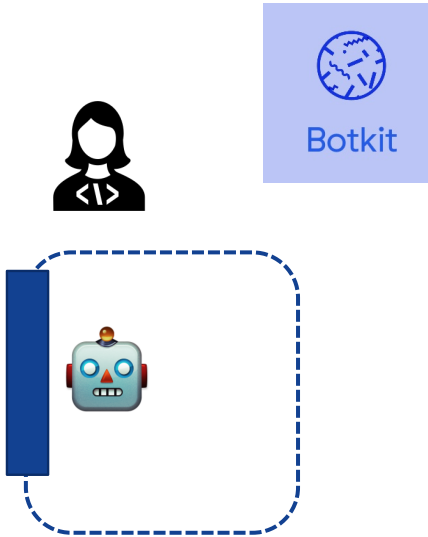
Our bot is basically a big state machine. Each incoming message to the bot triggers the following basic operations:

1. Process the message from the user (input). Could be text or image. The processing that is done depends on the state the user was in before the message came.
2. Perform business logic (e.g listing a for sale ad, or deleting a listing)
3. Decide which state is next. This depends on whether the user input is valid, was it a "Yes" or "No", could we find the address given, etc.
4. Go to next state. This typically involves sending a message to the user with some choices for the new state.

*(Source: Medium.com, Are bots the future of classifieds?, June 2016)*

# BotKit Studio

## Programming conversation threads



Triggers

New Trigger Add

order soup ✖

hangry ✖

soupme ✖

soup ✖

Variables

New Variable Add

soup\_size ✖

selected\_soup ✖

Help

**Threads**

A conversation can have several threads.  
Choose between to view or change the messages in each one.

Here is today's menu:

>"The Special Today is"  
`vars.daily\_special.name`  
`vars.daily\_special.description` Special  
price "\$vars.daily\_special.price"

>"Regular Soups": {#vars.soup\_menu } `name`  
`description` "\$ price"  
{/vars.soup\_menu }

What would you like to order?

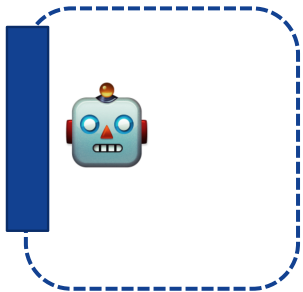
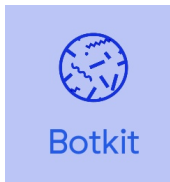
...and then **continue to next message**

And then, **complete this conversation successfully**

Add a line to the script Add Save

# BotKit Studio

## Programming conversation threads



```
controller.hears(['pizzatime'], 'message_received', function(bot,message) {
  var askFlavor = function(err, convo) {
    convo.ask('What flavor of pizza do you want?', function(response, convo) {
      convo.say('Awesome.');
```

```
      askSize(response, convo);
      convo.next();
    });
  };
  var askSize = function(response, convo) {
    convo.ask('What size do you want?', function(response, convo) {
      convo.say('Ok.')
```

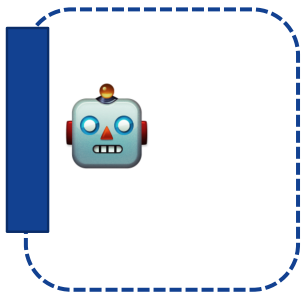
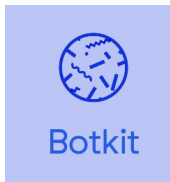
```
      askWhereDeliver(response, convo);
      convo.next();
    });
  };
  var askWhereDeliver = function(response, convo) {
    convo.ask('So where do you want it delivered?', function(response, convo) {
      convo.say('Ok! Good bye.');
```

```
      convo.next();
    });
  };

  bot.startConversation(message, askFlavor);
});
```

# Bot Frameworks / SDKs: BotKit

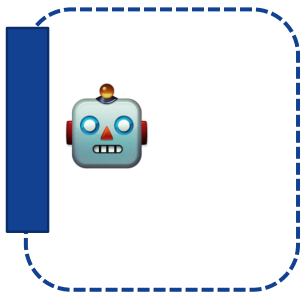
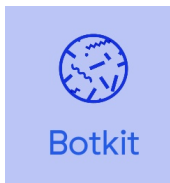
## Custom messages



```
controller.hears(['shirt'], 'message_received', function(bot, message) {  
  bot.reply(message, {  
    attachment: {  
      'type': 'template',  
      'payload': {  
        'template_type': 'generic',  
        'elements': [ {  
          'title': 'Classic White T-Shirt',  
          'image_url': 'http://.../img/item100-thumb.png',  
          'subtitle': 'Soft white cotton t-shirt is back in style',  
          'buttons': [ { 'type': 'web_url',  
                        'url': 'https://...',  
                        'title': 'View Item' },  
                      { 'type': 'web_url',  
                        'url': 'https://...',  
                        'title': 'Buy Item' } ]  
        }, {  
          'title': 'Classic Grey T-Shirt', ...  
        }  
      ] }  
    }  
  });  
});
```

# Bot Frameworks / SDKs: BotKit

## Custom messages



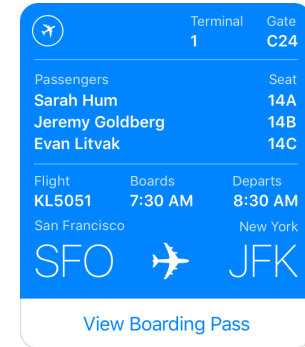
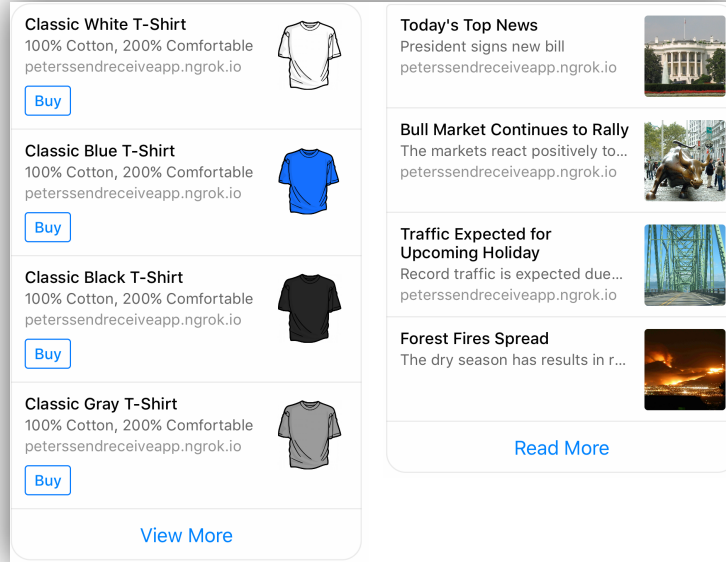
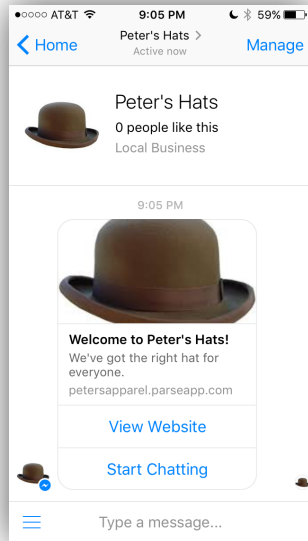
```
controller.hears(['shirt'], 'message_received', function(bot, message) {  
  bot.reply(message, {  
    attachment: {  
      'type': 'template',  
      'payload': {  
        'template_type': 'generic',  
        'elements': [ {  
          'title': 'Classic White T-Shirt',  
          'image_url': 'http://.../img/item100-thumb.png',  
          'subtitle': 'Soft white cotton t-shirt is back in style',  
          'buttons': [ { 'type': 'web_url',  
                        'url': 'https://...',  
                        'title': 'View Item' },  
                      { 'type': 'web_url',  
                        'url': 'https://...',  
                        'title': 'Buy Item' } ]  
        }, {  
          'title': 'Classic Grey T-Shirt', ...  
        }  
      ] }  
    }  
  });  
});
```



Payload of sent/received messages  
is different for each platform

# Custom messages (aka 'cards', 'templates')

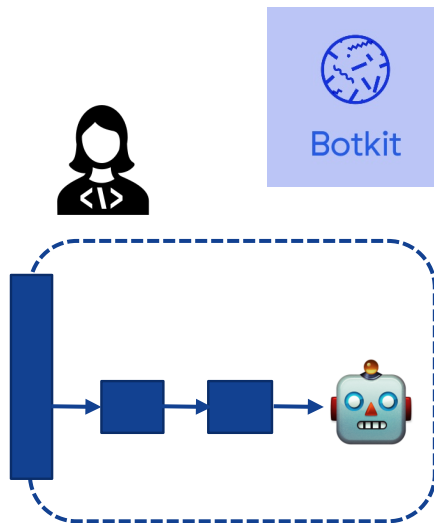
## Example: Facebook Messenger



Platforms offer only limited set of widgets. Picture HTML with `<boardingpass>` tags...

# Bot Frameworks / SDKs: BotKit

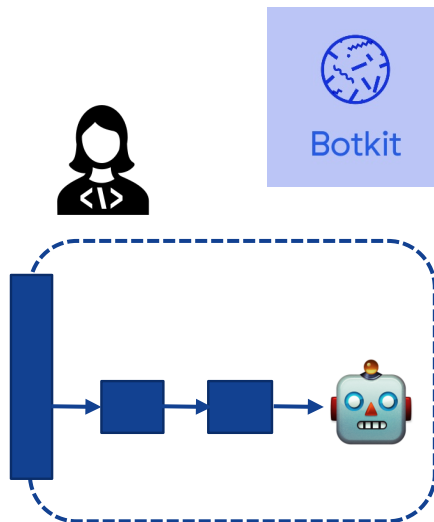
## Middleware



```
controller.middleware.receive.use(function(bot, message, next) {  
  
  // do something...  
  // message.extrainfo = 'foo';  
  next();  
  
});
```

# Bot Frameworks / SDKs: BotKit

## Middleware example: integrating with LUIS



```
controller.middleware.receive.use(luis.middleware.receive(options));

controller.hears(
  '',
  ['direct_message', 'direct_mention', 'mention'],
  luis.middleware.hereIntent,
  function(bot, message) {
    ...
  });
```

start tracking a run



```
{
  "query": "start tracking a run",
  "intents": [
    {
      "intent": "startActivity",
      "score": 0.9999981
    },
    {
      "intent": "stopActivity",
      "score": 1.54796021E-06
    }
  ],
  "entities": [
    {
      "entity": "run",
      "type": "activityType",
      "startIndex": 17,
      "endIndex": 19,
      "score": 0.9391843
    }
  ]
}
```



# Bot Frameworks / SDKs: SuperScript

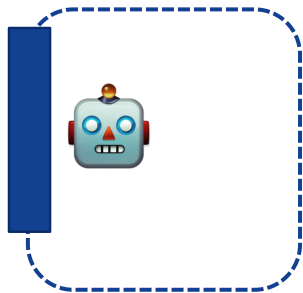
- Create a dialogue using a “script” written in a DSL



SuperScript

```
+ hello from human  
- hi from bot
```

```
+ [hey] hello (nice|mean) bot  
- hello from bot
```



<http://superscriptjs.com/>

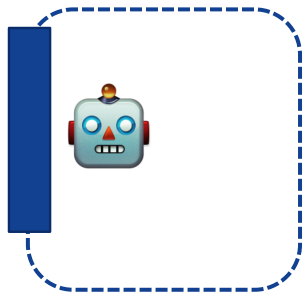
script.ss  $\xrightarrow{\text{npm run parse}}$  script.json

# Bot Frameworks / SDKs: SuperScript

- Create a dialogue using a “script” written in a DSL



SuperScript



<http://superscriptjs.com/>

```
+ conversation
- What is your name?

+ [my name is] *1
% * what is your name
- So your first name is <cap1>?

+ ~yes
% so your first name is *
- Okay good.

+ *
% so your first name is *
- Oh, lets try this again... {@conversation}

+ *
% * what is your name
- I don't get it. {@conversation}

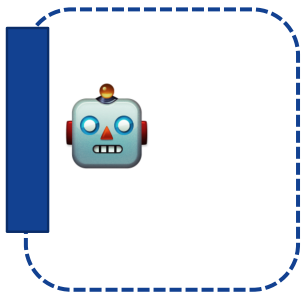
+ *
- Let's talk about something else now. {topic=new_topic}
```

# Bot Frameworks / SDKs: SuperScript

- Create a dialogue using a “script” written in a DSL



SuperScript



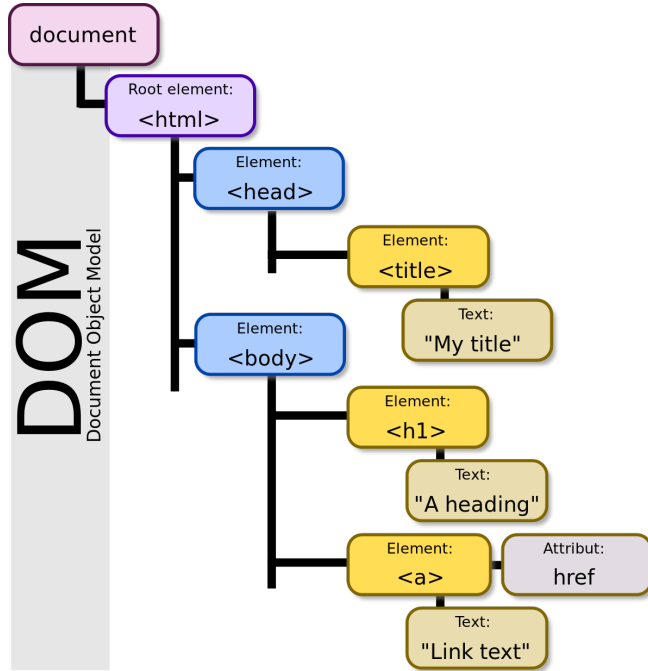
```
exports.getWeather = function(city, cb) {  
  cb(null, "It is probably sunny in " + city);  
}
```

```
+ what is the weather in *1  
- ^getWeather(<cap1>)
```

<http://superscriptjs.com/>

# From Documents to Dialogues

# The Document Object Model

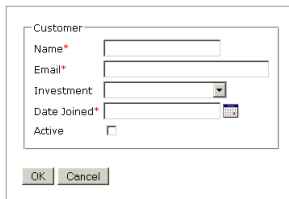


(Source: Wikimedia commons, CC BY-SA 3.0, Birger Eriksson)

A screenshot of a web form titled 'Customer'. It contains several input fields: 'Name' (text input), 'Email' (text input), 'Investment' (dropdown menu), 'Date Joined' (text input with a calendar icon), and 'Active' (checkbox). At the bottom, there are 'OK' and 'Cancel' buttons.

```
document.getElementById('textbox_id').value
```

# The Document Object Model



Imperative WHATWG DOM API ('90s)

Declarative jQuery-style APIs ('00s)

MVC and virtual DOMs ('10s)



Abstraction

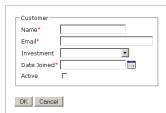
```
document.getElementById('textbox_id').value
```

```
$('#textbox_id').val()
```

```
<button value={this.state.text}/>;
```

# From documents to dialogues

## Documents

A screenshot of a web form titled "Customer". It contains five input fields: "Name", "Email", "Investment" (with a dropdown arrow), "Date Joined" (with a calendar icon), and "Active" (a checkbox). At the bottom are "OK" and "Cancel" buttons.

- DOM = document tree
- Rich ways to present content, limited ways to navigate it
- UX 'design' largely decoupled from underlying model

VS

## Dialogues



- Conversation = state machine
- Rich ways to navigate content, limited ways of presenting it
- UX 'design' still often closely coupled with model

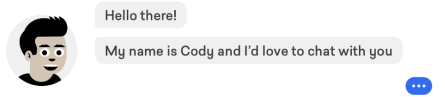
# From documents to dialogues

## Opportunities for PLT $\lambda$

- If a dialogue is a state machine...
- ... what analyses or transformations can we apply to such state machines?
  - Ensure no path leads to a ‘dead end’
  - Ensure no question gets asked twice
  - Ensure conversations don’t get stuck in a loop
  - Merge multiple dialogues into a single consistent conversation
  - ...
- What’s the “virtual DOM” equivalent for dialogues?
  - State machines are precise but very low-level
  - Are there less imperative ways of specifying conversation flow?

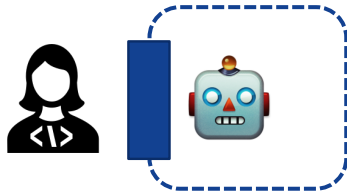


# Talk Summary

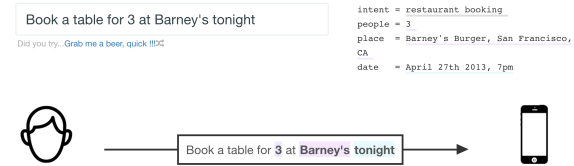


Good afternoon    Hi, Cody!    Hello!

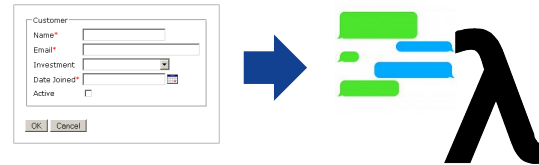
Conversational Web



Programming bots



Natural language is the new UI



New opportunities for PLT

**NOKIA**

# Copyright and confidentiality

The contents of this document are proprietary and confidential property of Nokia. This document is provided subject to confidentiality obligations of the applicable agreement(s).

This document is intended for use of Nokia's customers and collaborators only for the purpose for which this document is submitted by Nokia. No part of this document may be reproduced or made available to the public or to any third party in any form or means without the prior written permission of Nokia. This document is to be used by properly trained professional personnel. Any use of the contents in this document is limited strictly to the use(s) specifically created in the applicable agreement(s) under which the document is submitted. The user of this document may voluntarily provide suggestions, comments or other feedback to Nokia in respect of the contents of this document ("Feedback").

Such Feedback may be used in Nokia products and related specifications or other documentation. Accordingly, if the user of this document gives Nokia Feedback on the contents of this document, Nokia may freely use, disclose, reproduce, license, distribute and otherwise commercialize the feedback in any Nokia product, technology, service, specification or other documentation.

Nokia operates a policy of ongoing development. Nokia reserves the right to make changes and improvements to any of the products and/or services described in this document or withdraw this document at any time without prior notice.

The contents of this document are provided "as is". Except as required by applicable law, no warranties of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose,

are made in relation to the accuracy, reliability or contents of this document. NOKIA SHALL NOT BE RESPONSIBLE IN ANY EVENT FOR ERRORS IN THIS DOCUMENT or for any loss of data or income or any special, incidental, consequential, indirect or direct damages howsoever caused, that might arise from the use of this document or any contents of this document.

This document and the product(s) it describes are protected by copyright according to the applicable laws.

Nokia is a registered trademark of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.