

Elementary!

Incorporating BlueMix, Node-RED
and Watson in Domino applications



Welcome! During the next 45 minutes I will show you how you can incorporate BlueMix, Node-RED and Watson in your own applications, either on the web or in the Notes client.

I call this presentation "Elementary" as a homage to Sherlock Holmes and his assistant Dr Watson. Today I will show you how to make IBM's Watson *your* assistant. And it is easier than you may think. **It's Elementary!**



But before we start I want to remind you of our amazing sponsors, without whom we would not be here today. Please visit them in the Exhibitor Showcase and find out what they can help you with. These are all great companies, and I have personally been using products from many of them for years. Most of the remaining companies have products I have been evaluating and recommending to customers.

Take a few minutes to visit them and thank them for making MWLUG possible.



MWLUG 2017
Moving Collaboration Forward

Karl-Henry Martinsson

CEO, Demand Better Solutions, LLC


- In the IT industry for 29 years
- Full time programmer since 1991
- Worked with Notes and Domino since 1994
- Web development since 1995
- IBM Champion since 2014
- Worked as IT journalist for 5 years


IBMCHAMPION 

First a little bit about me. Next month it will be 29 years since I got my first job in the IT industry, and I have been programming most of that time. I started with Lotus Notes about the same time I learned web development, and I am an four-time IBM Champion. In the early 1990's I worked as a journalist, covering the PC industry. We initially used cc:Mail but soon switched to Notes, and I found myself developing applications for internal use. That is how I started with Notes.



MWLUG 2017
Moving Collaboration Forward

**Demand Better Solutions**
Innovative software solutions to real business headaches

- Focus on user experience and interface
 - Make the software easy to use
 - To the user the UI is the application!
- Development for Notes and browser
 - New development, both Notes client and browser
 - Modernization of existing applications
 - Application migration from and to IBM Domino
- Take advantage of modern tools and technology
 - jQuery, Bootstrap, jQuery Mobile, BlueMix, etc

My company is Demand Better Solutions, and our focus is the experience of the user. Any software needs to be easy to use. To the user the interface *is* the application, they don't know or care what's happening in the background or under the hood.

We develop new applications, for the Notes client as well as web applications for the browser, perform modernization of existing applications and migrate application and data from and to Domino.

I prefer to use modern standardized tools like the ones mentioned here. It makes development faster, and therefore less expensive for the client. In addition, the libraries and tools are mostly Open Source, lowering the cost while keeping the quality high.

OK, let's get started!



MWLUG 2017
Moving Collaboration Forward

Agenda

- What is IBM BlueMix?
- What is Node-RED?
- Watson Services
 - Translation (English to Spanish)
 - Text-to-Speech
- Integrate into your own solutions
 - Notes client
 - Web application

This is what I will cover today.

I will give an overview of BlueMix, Node-RED and Watson Services.

Then I will show you how you can access Watsons functionality for translation and text-to speech and integrate them in your own application, both web based and in the Notes client.

Can I see some hands, how many here have tried out BlueMix?

And how many are using BlueMix actively?



What is IBM BlueMix?

- Platform as a Service (PaaS)
- Based on Cloud Foundry
 - Open source, multi cloud platform
- Running on SoftLayer infrastructure
- Supports multiple tools and services
 - Node-RED
 - Watson API

BlueMix is IBM's Platform as a Service. It is based on the open source platform Cloud Foundry. It was conceived in 2009 by a small team at VMWare and released in 2011. In 2015 the Cloud Foundry Foundation was registered as an independent non-profit organization, and the software was transferred to the organization as open source.

BlueMix is running on infrastructure from SoftLayer, a company founded in 2005 and acquired by IBM in 2013.

As you can tell, this is all mature technology.

IBM offers a number of tools and services through BlueMix. We will look at two of them today, Node-RED and Watson.



What is Node-RED?

- “Flow-based programming for IoT”
- Wire together services, APIs and hardware
- One-click deployment
- Built on Node.js
 - Event driven, non-blocking (asynchronous)
 - Light-weight runtime, can run on Raspberry Pi
 - Available as Docker container and on BlueMix

The first of the two services is Node-RED. It is a tool developed by IBM but contributed as open source. It is described as a flow-based programming tool to wire together online services, APIs and hardware as part of the Internet of Things. The deployment of applications, or flows, is very easy: just one click on a button.

The browser-based editor can be used to create Javascript functions, and the light-weight runtime is built on Node.js. This means the runtime can be deployed even on a Raspberry Pi. There is also a Docker container with Node-RED. But we will of course use it on BlueMix.




Node-RED Editor

- Browser based editor
- Click-and-drag of components (nodes)
- Input and output nodes (interface)
 - http, web sockets, email, twitter
- Function nodes
 - switch, split, join, change
- Javascript used to modify payload

As I already mentioned, the editor is browser based. You drag components, or nodes, from a palette into the workspace and connect them together into a flow by dragging lines between the nodes.

There is a number of nodes for input and output, everything from http and web sockets to connections to email and even Twitter. Other nodes are used to control the flow, like switches, and to split and join values.

You can also use Javascript functions to modify the payload sent between the nodes.



MWLUG 2017
Moving Collaboration Forward

Many Available Node

input	output	social	storage	analysis	function	IBM_Watson
inject	debug	e mail	mongodb	sentiment	function	conversation
catch	link	twitter	cloudant	predictive analytics	template	conversation workspace manager
status	mqtt	e mail	dashDB		delay	discovery
link	http response	twitter	mongodb		trigger	discovery query builder
mqtt	websocket		cloudant		comment	comment
http	tcp		dashDB		http request	language translator
websocket	udp				switch	language identify
tcp	mqtt				change	language translator util
mqtt	radio				range	natural language classifier
ibmcloud	ibmcloud				split	Visual Language Understanding
	OpenWhisk				join	personality insights
	play audio				csv	
					html	
					json	
					xml	
					yaml	
					tcp request	
					OpenWhisk	
					rb	
						speech to text
						speech to text custom builder
						text to speech
						text to speech custom builder
						tone analyzer v2
						visual recognition
						visual recognition util
						similarity search
						similarity search util

Here are a some of the available nodes. To the right you see a number of Watson related functionality you have access to through BlueMix. You can see both the language translator and text-to-speech we will use shortly.



IBM Watson Services

- Conversation (bots and virtual agents)
- Document Conversion
- Natural Language Classifier
- Visual Recognition
- Language Translator
- Speech to Text / Text to Speech
- And many more...

IBM offers a number of Watson services on BlueMix. Here are a few of them. The two ones we will use today are the language translator and text to speech.



MWLUG 2017
Moving Collaboration Forward

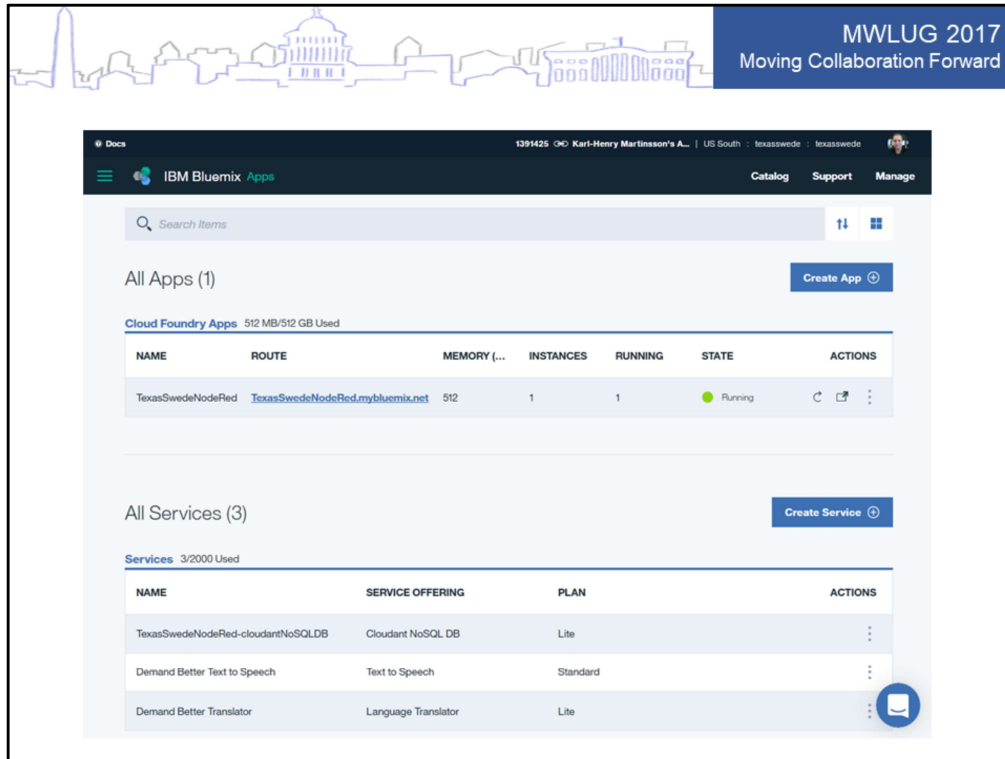
Where Do We Start?

- Create a BlueMix account – free for 30 days
- Setup a Node-RED instance
- Add some Watson Services
 - Language Translator
 - Text to Speech


So let's start. The first thing you do is to setup a BlueMix account, unless you already have one. It is free for 30 days, no credit card needed.

I will not go through all the steps, in the sake of time, but IBM has made it very easy.

After you have access to your BlueMix account, create a Node-RED instance, add the Watson translator and Text-to-Speech services and finally connect them to Node-RED.

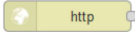
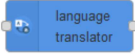





Your Bluemix screen will now look something like this. You see the Node-RED app running at the top, and the installed services below.



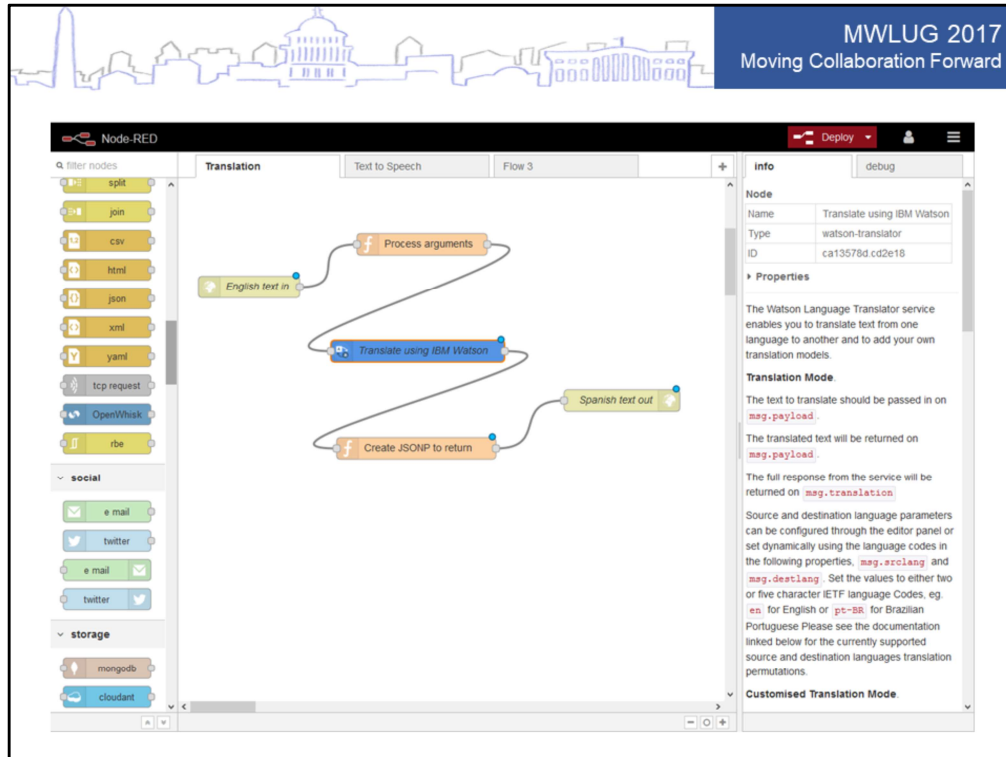
MWLUG 2017
Moving Collaboration Forward

Create Translation Flow


- Create and configure main nodes
 - http input
 - Watson Translate
 - http response
- Nodes to manipulate JSON msg object
 - Process incoming JSON for translation
 - Create JSONP for output
- Connect nodes

The first application will translate a text from English to Spanish. First we create and configure the main nodes: one node to receive http input, the translation node itself and finally a node to return a response. We also need a couple of nodes to manipulate the messages passed between the nodes. The first function will process the incoming JSON so it will match what the translation node requires. The second one will be used to convert the JSON output from the translation node to JSONP, since we will call it from a different domain in a few minutes.

Finally we connect the nodes.



This is what it will look like when the flow is done. Let's take a closer look at the two functions we added.

MWLUG 2017
Moving Collaboration Forward

Process Incoming JSON

```
// Get payload to translate and default is English)
msg.payload = msg.payload || msg.payload.from == '' ? {} : msg.payload.from;
// Pass object to the next node
return msg;


// Get language to translate to (default is Spanish)
if(msg.payload.to === null || msg.payload.to == '') {
  msg.destlang = "es";
} else {
  msg.destlang = msg.payload.to;
}

// Set payload to text to translate
msg.payload = msg.payload.message;
// Pass object to the next node
return msg;
```

For the first function we don't need much code. Only one line is actually needed, to take the message value (which is the text to translate) in the incoming payload and put it in the payload passed to the translation node.

But let's add some additional code that could come in handy later. (click now)

What I do here is to check if the calling application is passing along languages to translate to and from. If there are no values, the default will be from English to Spanish.



MWLUG 2017
Moving Collaboration Forward

Create JSONP to Return

```
// Wrap payload in callback function  
msg.payload = "translated({'translated':'" + msg.payload + '"})";  
  
// Pass object to final node, to return to browser  
return msg;
```

I mentioned earlier that we need to return JSONP (JSON with Padding) to the calling web pages, since it will be located on a different server. I simply wrap the data returned in a Javascript function and return it, to be executed on the receiving side.



MWLUG 2017
Moving Collaboration Forward

Call Flow From Web Page

HTML:

```
<input type="text" id="text" placeholder="Enter text to translate here">
<button id="btnTranslate" class="btn btn-primary">Translate</button>
<div id="translated"></div>
```

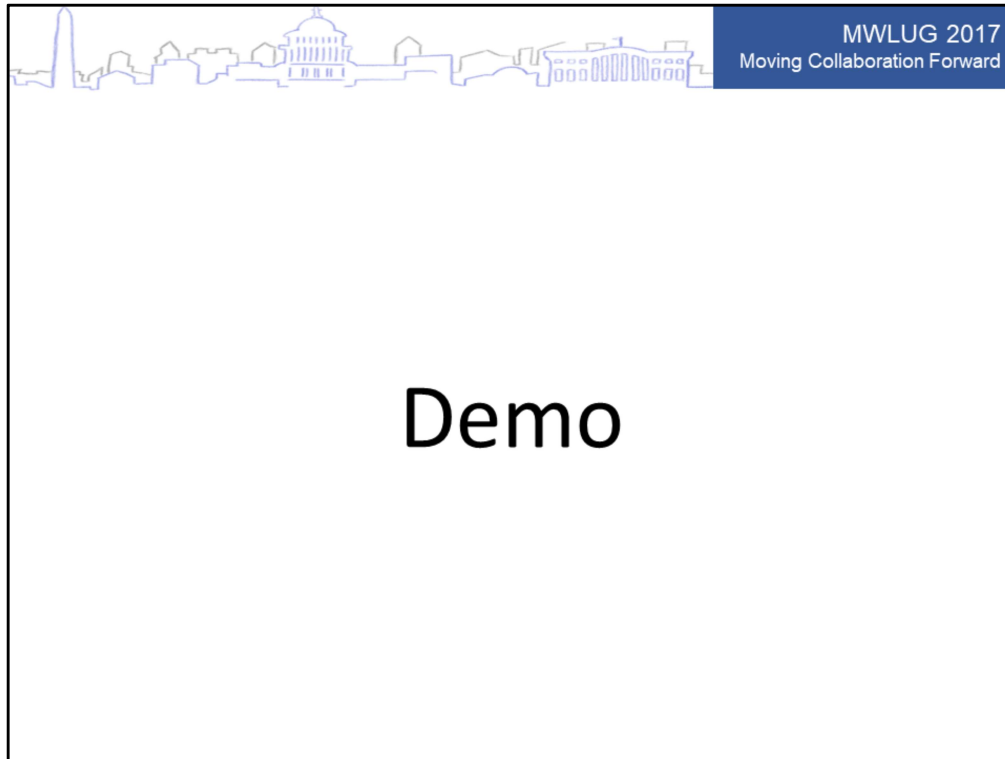
jQuery:

```
$("#btnTranslate").on("click", function (e) {
  e.preventDefault();
  // Store values to pass to flow in a JSON object
  var json = {
    "message": $("#text").val(),
    "from": "en",
    "to": "es"
  };
  // Call Node-RED flow using Ajax
  $.ajax({
    url: "https://texasswedenodened.mybluemix.net/translate",
    dataType: "jsonp",
    data: json
  });
});
// Call-back function for translation
function translated(data) {
  $("#translated").html(data.translated);
}
```


Let's take a look at how we call this flow from a web page. We only need 3 elements: an input field to enter the text to translate, a button to execute the translation and a div element where the translated text will be displayed.

I will be using jQuery to perform the call to my BlueMix instance, but you can of course also use regular Javascript, or some other framework if you like.

What you see here is the code for the click event of the translate button. It will read the value of the input field and store it in a JSON object, together with the from and to languages. In this case we could exclude the language info, as they are the same as the default values, but I am including it for clarity.




Let's take a look at it in action! (*switch to browser*)



MWLUG 2017
Moving Collaboration Forward

Call Flow From Notes

- Simple form
 - Two fields
 - Computed text
 - One button
 - Lotusscript to call URL and parse JSON
- Using my HTTP retrieval class
 - Using MSXML2.ServerXMLHTTP
 - Windows only
 - Code available on my blog




But would it not be nice to get translations in your Notes applications as well? Let's see how that can be done.

We start with a simple form, the same basic design as the web page: a field for the user to enter text to translate into and a button to perform the translation. I also created a hidden field where the translated text will be placed, and a computed for display field to display it to the user without them being able to edit it. The end result will be identical to the web version.

The reason I am not using computed text is that (at least in earlier versions of Notes), computed text would not be printed. So ever since, I use computed-for-display fields instead.

The code behind the button will use Lotusscript to call the URL and parse the JSON returned. I am using my HTTP retrieval class, and since it is implemented using a Windows specific component, this will not work on other platforms than Windows. The code for the class is available on my blog. I will give you the URL at the end.



MWLUG 2017
Moving Collaboration Forward

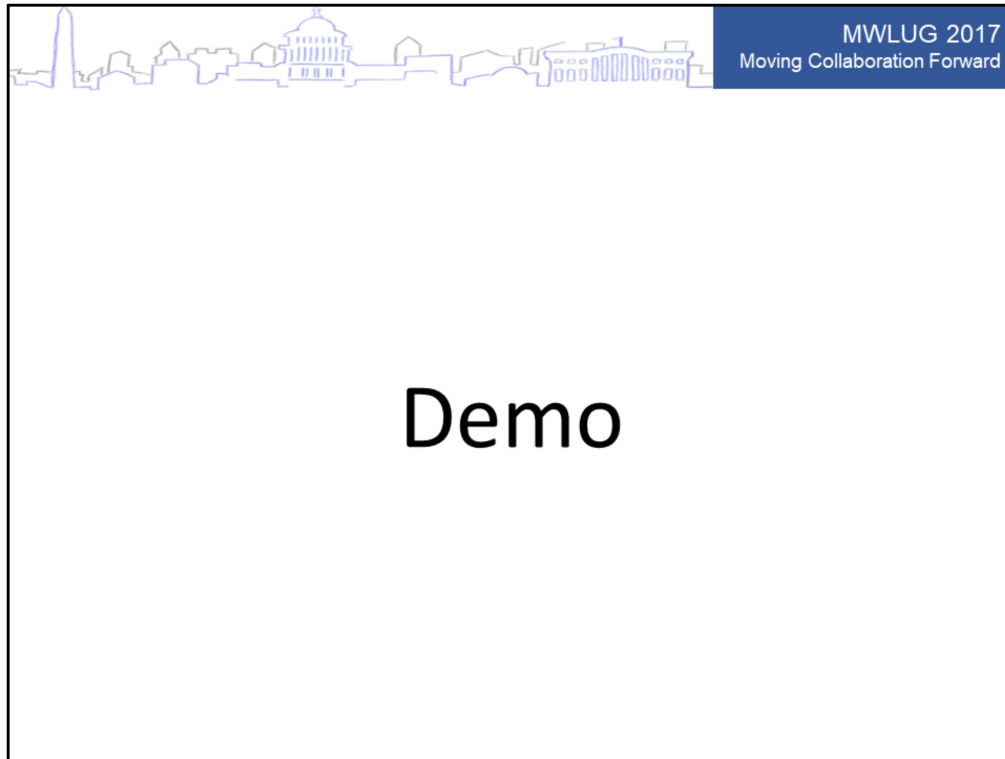
Lotusscript Code

```
Use "class.RemoteHTTP"  
Sub Click(Source As Button)  
  Dim ws As New NotesUIWorkspace  
  Dim uidoc As NotesUIDocument  
  Dim http As New RemoteHTTP()  
  Dim textEnglish As String  
  Dim textSpanish As String  
  Dim url As String  
  Dim response As String  
  Set uidoc = ws.CurrentDocument  
  textEnglish = uidoc.FieldGetText("InputText")  
  url = "https://texasswedenodered.mybluemix.net/translate"  
  url = url + "?message=" & textEnglish  
  url = url + Replace(textEnglish, " ", "+")  
  response = http.GetHTTP(url)  
  textSpanish = parseJSON(response, "translated")  
  Call uidoc.FieldSetText("OutputText", textSpanish)  
  Call uidoc.Refresh()  
End Sub
```


This is the Lotusscript code.

I get the English text reading read the text field. Then I build the URL to call and makes an HTTP call. I am not including the language this time, to show that those arguments are optional.

I read the response from the server and pass it to a function I wrote to parse out the translated text. I then write the translated text into the computed field and refresh the document.



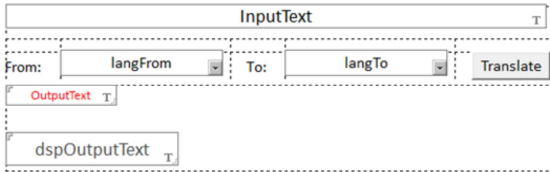
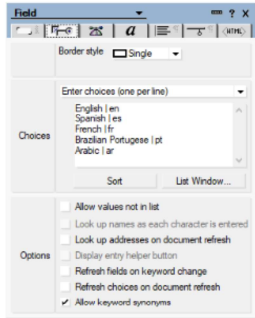
Let's switch to the Notes client and see what it looks like!
(Switch to Notes client, then back to this slide when done)



MWLUG 2017
Moving Collaboration Forward

Improvements

- Select source and destination languages
- Add two dropdowns to form





- Pass languages as arguments
 - Just a few more lines of code

This simple translation application can of course be improved in many ways.

The most obvious one is to add selection of languages to translate from and to. I added two combobox fields, containing the languages to pick from.

Then add a couple of lines of Lotuscript code to the button in the form. It will use the more extended Javascript code in the first node of the translation flow that I showed earlier.



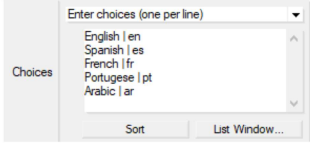
MWLUG 2017
Moving Collaboration Forward

Lotusscript Code

```

Use "class.RemoteHTTP"
Sub Click(Source As Button)
  Dim ws As New NotesUIWorkspace
  Dim uidoc As NotesUIDocument
  Dim http As New RemoteHTTP()
  Dim textEnglish As String
  Dim textSpanish As String
  Dim url As String
  Dim response As String
  Set uidoc = ws.CurrentDocument
  textEnglish = uidoc.FieldGetText("InputText")
  url = "https://texasswedenodered.mybluemix.net/translate"
  url = url + "?from="+uidoc.document.GetItemValue("langFrom")(0)+"&to="+
    uidoc.document.GetItemValue("langTo")(0)+"&message="
  url = url + Replace(textEnglish, " ", "+")
  response = http.GetHTTP(url)
  textSpanish = parseJSON(response, "translated")
  Call uidoc.FieldSetText("OutputText", textSpanish)
  Call uidoc.Refresh()
End Sub

```

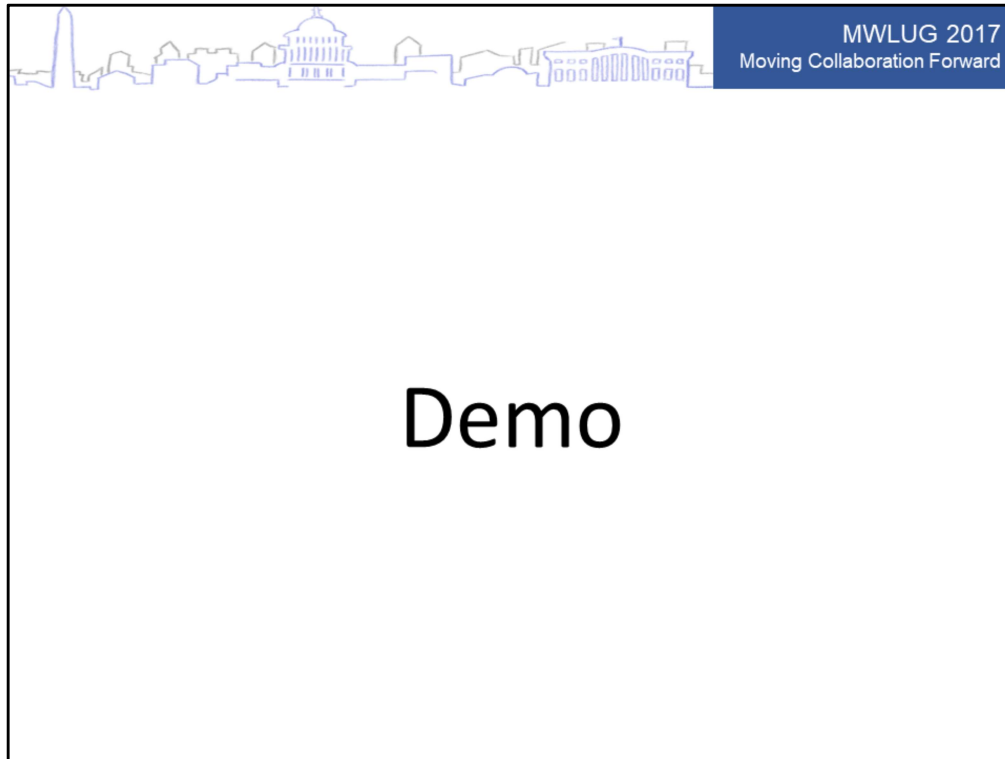


I just had to add four lines of code to read the language value.

Remember that in order to get the alias of the selected language you have to get the backend value. I also had to modify one line to pass the language selection in the URL.

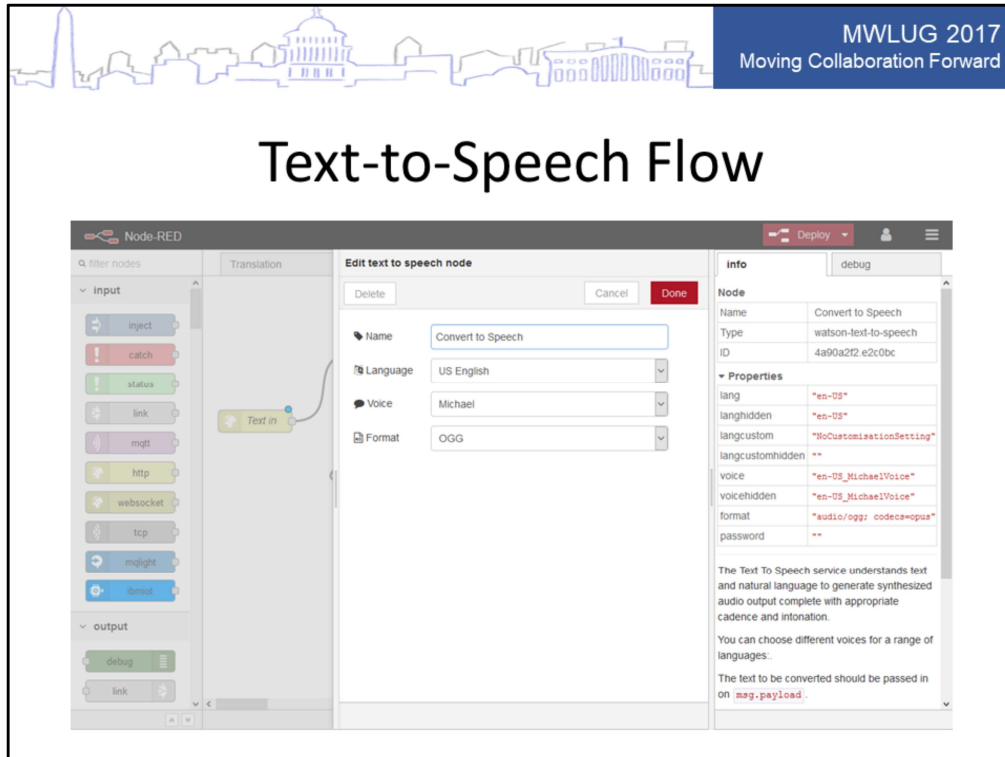
(click)

If you accept that the code is a little bit more compact and harder to read, it can be done without even increasing the number of lines. But we all write code that is easy to read and maintain, right? :-)



Let us see how the new code works.

(switch to browser)




I also promised that I would show how to convert text to speech using Watson.

Here is the flow I created. As you can see, it looks very similar to the previous one. The only difference is that the translation node has been replaced by a text-to-speech node.

(click)

I have set the language, voice and file format to return in the settings for the node. But they can be provided through the API call the same way as I just showed you in the translation application.

Let's look at the code.




MWLUG 2017
Moving Collaboration Forward

Process Incoming Request

```
// Set the payload for the next node to the text we want Watson to speak  
msg.payload = msg.payload.text;  
// Pass object to the next node (text-to-speech)  
return msg;
```

For the first function we don't need much code. Again we just need to modify the payload so it matches what the next node requires. The incoming JSON will contain the text to convert into speech in a value called *text*, and the next node wants it to be in *payload.text*.




MWLUG 2017
Moving Collaboration Forward

Modify Response

```
// Set the payload for the response to the audio file  
// created and passed to us in msg.speech  
msg.payload = msg.speech;  
// Pass the updated object to the HTTP Response node  
return msg;
```

For the first function we don't need much code. Again we just need to modify the payload so it matches what the next node requires. The incoming JSON will contain the text to convert into speech in a value called *text*, and the next node wants it to be in *payload.text*.



MWLUG 2017
Moving Collaboration Forward

Call Node From Web

HTML:

```
<input type="text" id="text" placeholder="Enter text here">
<button id="btnSpeak" class="btn btn-primary"><i class="fa fa-volume-up"></i>Play</button>
<div id="iframeHost"></div>
```

jQuery:

```
$("#btnSpeak").on("click", function(e) {
    e.preventDefault();
    // Remove any existing iframe
    $("#iframeHost").html("");
    // Set the location of Node-RED flow to use
    var baseURL = "https://texaswedenodered.mybluemix.net/speak";
    // Get text to convert to speech
    var text = $("#text").val();
    // Calculate the URL of speech file to insert into iframe
    var url = baseURL+"?text=" + text;
    // Create iframe element in memory
    var iframe = $('<iframe id="audioframe" src="'+url+'" frameborder="1" width="10px"
    height="10px" style="display:none;"></iframe>');
    // Insert iframe element into div on page
    $("#iframeHost").append(iframe);
});
```

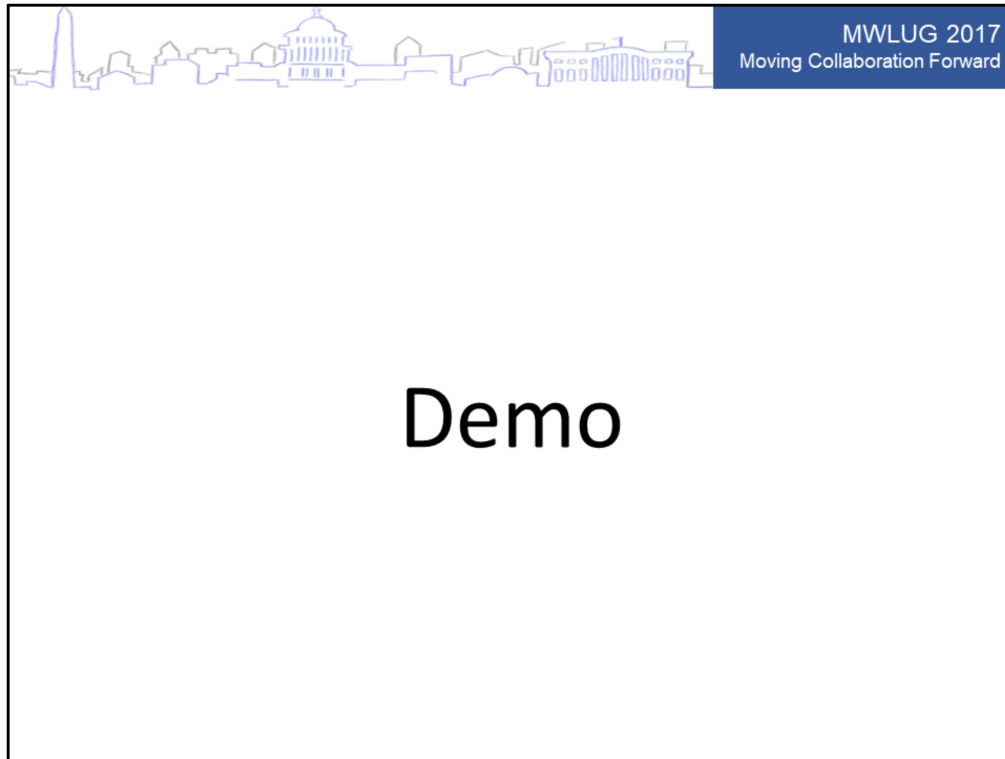
Let's take a look at how we call this flow from a web page. Again I will use only 3 elements. You recognize the input field and button from before.

There is also a div on this page, just like before.

That div is where we will insert an *iframe* pointing to the sound file generated by the URL we will call.

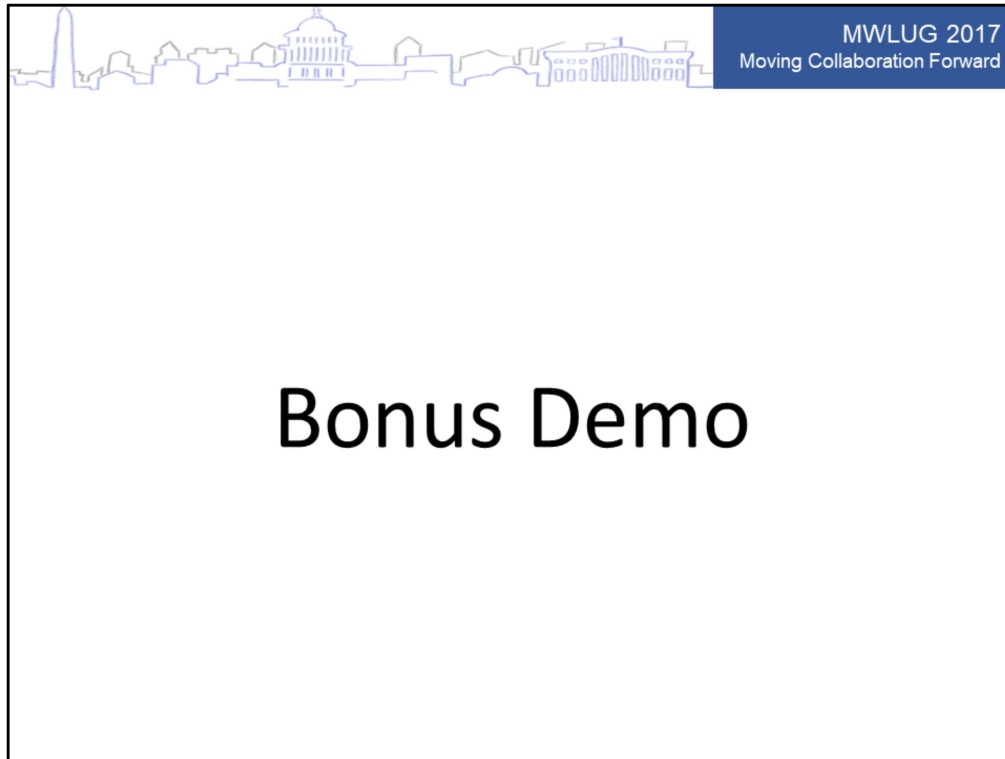
I will again be using jQuery. I create an event for when the button is clicked. In the event I collect the values I need, mainly the text to convert, and build a URL to call.

Next I create an iframe element in memory. An iframe can not communicate with the page it is hosted on. This will allow me to insert contents from other domains, like the sound file from mybluemix.net.



Time to see how it works!


(switch to browser)



Since we have a few minutes left, let's have some fun. Would it not make sense to combine the two examples you just seen into one application? Type in text in English and have it read back in Spanish?

Let's build that from scratch, right here, right now!

(switch to browser)



MWLUG 2017
Moving Collaboration Forward

Conclusion

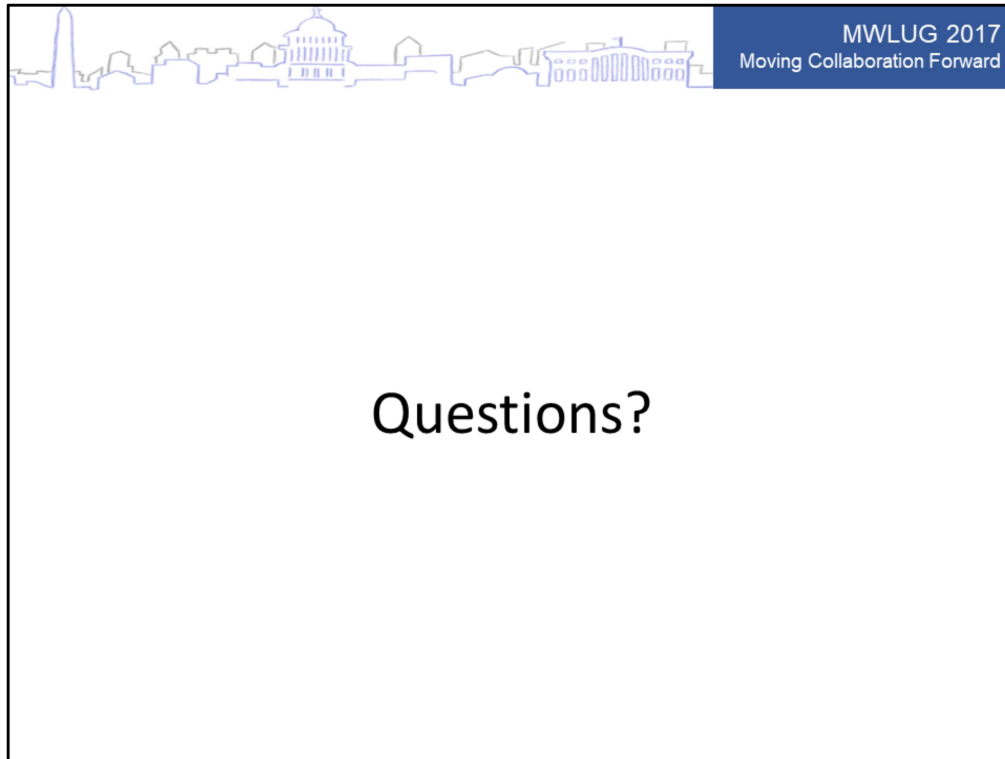
- BlueMix
 - Easy to setup
 - Convenient to use
 - Pay-as-you-go
- Node-RED
 - Click-and-drag of nodes
 - Javascript to manipulate
- Integration in web pages and Notes client

As you have seen, for being an IBM product I have to say BlueMix is very easy to setup and convenient to use. You can get a trial account to BlueMix Premium with 30 days free access, you don't even need to give your credit card. There is also the totally free BlueMix Standard. There you have access to Watson Translation and a number of other services, but not premium services like text-to-speech.

There is really no reason not to take BlueMix for a spin.

I also showed you how you through klick and drag can build flows in Node-RED, and how you use your existing Javascript skills to manipulate the data passed between the nodes.

Node-RED makes it very easy to integrate different BlueMix services into your web applications, but you can also integrate it into Notes client applications.




This concludes my presentation. Any questions?



MWLUG 2017
Moving Collaboration Forward

Thank You

Karl-Henry Martinsson
karl-henry@demandbetersolutions.com
 @texasswede
 @texasswede

<http://www.demandbetersolutions.com>
<http://www.texasswede.com>
<http://blog.texasswede.com>

Thank you for coming. Here is my contact information. The slides and code samples will be uploaded to my blog within a few days.