Mobile RPG

with PhoneGap

Presented by

Scott Klement

http://www.profoundlogic.com

Marriage is like my mobile phone contract. When you first signed up you were able to use all of the services you wanted, as much as you desired...no limit and no extra cost. A few months later and you no longer use many of the services because you just can't be bothered to pay the price!

The Agenda

Agenda for this session:

1. The what/why of PhoneGap
   - Web vs. Native vs. Hybrid
   - Utilizing Device Features
   - What is Cordova
2. Review/Discussion of Developing
   - Using the CLI
   - PhoneGap Environment and Docs
   - Hello World Example
   - Web programming review/discussion
3. Writing the RPG Code
   - Communicating with IBM i
   - Handling offline state
   - Example
4. PhoneGap Plugins
   - beyond what a browser could do
   - Barcode scanner example

Agenda Note: Self-Learning

My goal for this session:

… is not to teach you "all you need to know".

- Learn why PhoneGap valuable
- Learn the gist of things
- Be able to research/learn the rest on your own

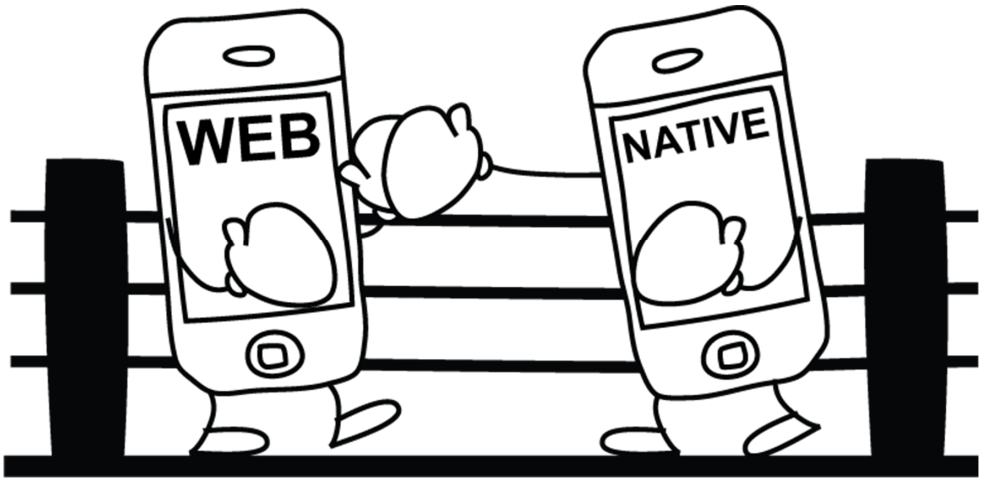
I do not like to learn a lot of stuff at once!

- Learn a little bit, then try it
- Then learn more, and try that
- Hands-on learning
- Figure it out when you need it.

My goal is to give you a good foundation so that you can do this yourselves.

Users Want Apps! (89% of Time)

HTML Browser VS. Native



Need a Middle Ground?

Browser

Runs on the server

Pros:
- No need to code in native device languages
- Displays in any mobile Web browser
- Cost-effective

Cons:
- No native capabilities (Camera, etc)
- Does not behave like "an app"

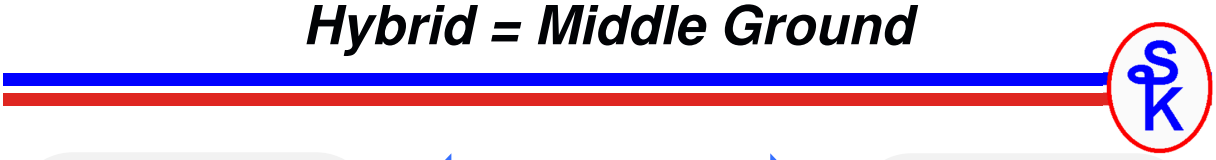
Native

Runs on the device

Pros:
- Adds native device look/features to apps
- Adds more functionality

Cons:
- Need to know mobile languages/outsource
- Separate development effort

Hybrid = Middle Ground



Browser

Runs on the server

Pros:
- No need to code in native device languages
- Displays in any mobile Web browser
- Cost-effective

Cons:
- No native capabilities (Camera, etc)
- Does not behave like "an app"

Hybrid

Wraps Web (HTML/CSS/JS) apps in Native Code

Pros:
- All pros of web
- Can behave like an app
- Native device features

Cons:
- Extra steps to package/distribute (vs. web – same as native.)

Native

Runs on the device

Pros:
- Adds native device look/features to apps
- Adds more functionality

Cons:
- Need to know mobile languages/outsource
- Separate development effort

What is PhoneGap?

Makes web code into an app (i.e. "hybrid")

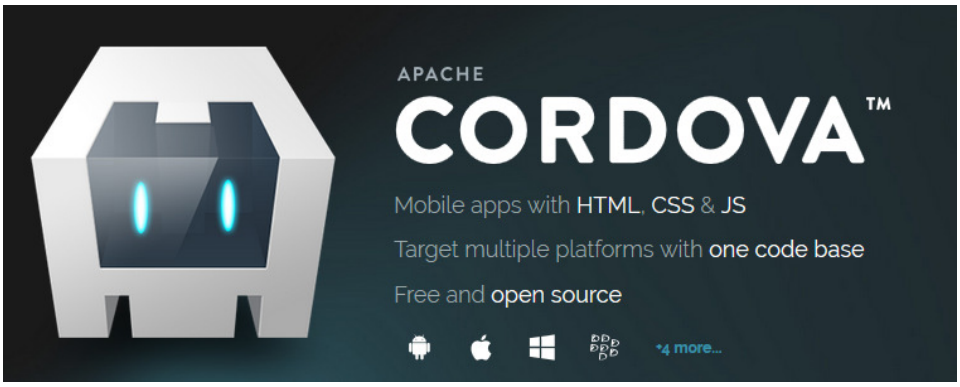
- A packaged up version of Apache Cordova
- Provides desktop coding environment (GUI)
- and/or command-line interface (CLI)
- a developer app to test on devices

You code HTML, CSS and JavaScript. Run the PhoneGap commands to make it into an app.

Plugins provide a way to do things a browser could not

- Filesystem on device
- Hardware features
- Camera, GPS, Accelerometer, Vibration, etc

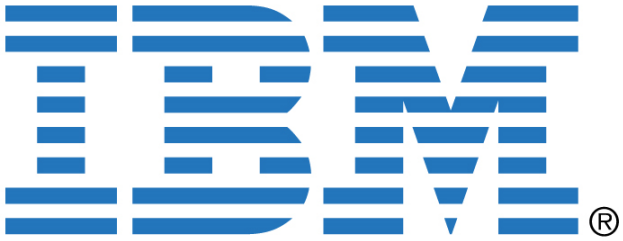
What is Cordova?



This is the core of PhoneGap

- Converts web tech into an app
- Provides the plugins (or environment for 3rd party plugins)
- PhoneGap repackages Cordova, and adds developer tools
- Usually can use the terms "PhoneGap" and "Cordova" interchangably

What About RPG?



RPG is *the best* language for your business rules

- …but does not run on mobile (directly)
- …but can output web application
- …or can communicate with web application
- *Therefore will work with PhoneGap!*

PhoneGap: Process Overview

Scott has only learned the command-line interface (CLI) so will use that.

Here is the basic process we will use

- Use PhoneGap to generate the code for an app
- Modify the web code to suit your needs
- Add code to call RPG program on server
- Write RPG program to provide needed business data
- Test on your PC
- Build into an app
- Deploy onto the device
- Deploy to App Store / Play Store, etc **

** I will not cover the last step today. Instead, you can follow the steps in the documentation.

PhoneGap: Some Details

PhoneGap's web site: http://phonegap.com
- Go here to get info to download/install
- CLI interface uses Node.js and npm
- Docs are also on phonegap.com (click "CLI" or "Desktop App")
- Additional (better?) docs found on http://cordova.apache.org

Develop the App On your PC
- …but communicate with IBM i / RPG for business data
- Use web browser (Firefox or Chrome recommended) to test/debug

To test and deploy on device
- Generate a "real app" (phonegap build)
- Deploy to device via USB cable for testing (phonegap run)
- Deploy via App Store / Play / etc for production users to install

Recommended Environment

On Windows:
- Install Git-Bash
  - you'll want git if you use any special plugins
  - the bash command-line is much better than DOS
- Mac doesn't need this – it already has git and bash

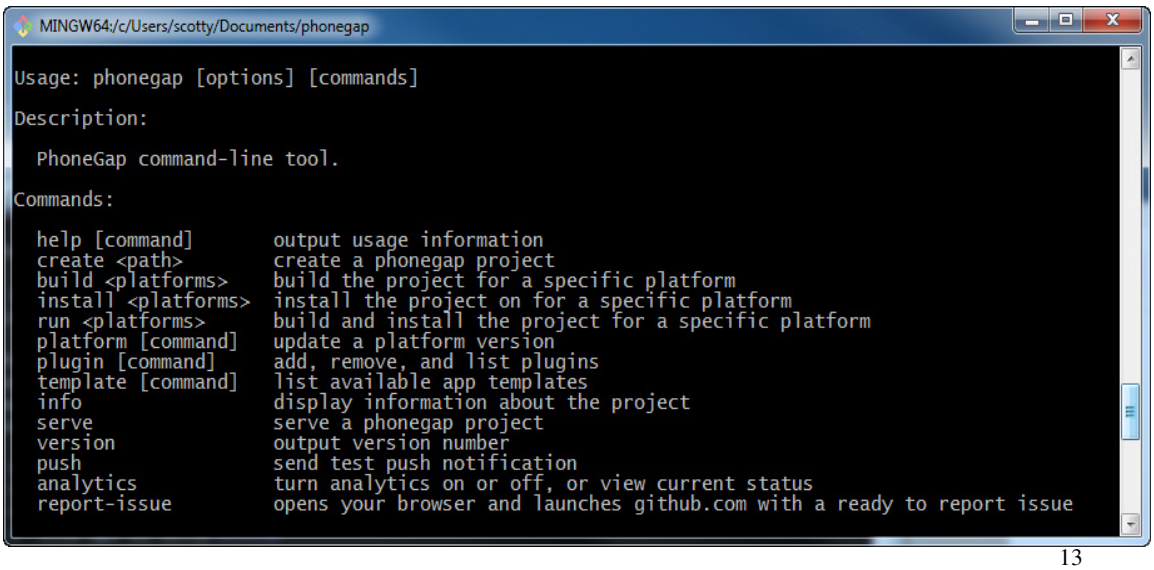
Install Node.js
- PhoneGap is written in Node, and needs Node.js to run

Edit Code with RDi
- Use "Local Files".
- Has good support for HTML, JavaScript and CSS
- Notepad++ is another option if you don't own RDi

PhoneGap: The CLI

Scott uses the Command-Line Interface (CLI) *-- nerd!*

- You can use Desktop App if you prefer
- I learned CLI, and that's what I know… so is what I'll show you



PhoneGap Create

`phonegap create directory-name [OPTIONS]`

Creates an app, including a skeleton, potentially with sample code.

`Options` (see more with "`phonegap create help`")
`--template` = template to use
`--name` = name of app (instead of "hello world")

EXAMPLE:

```
cd Documents/phonegap
phonegap create ScottsApp --name "Scott's App"
```

Notice:
- config.xml file contains plugins and build info
- www directory has web code that you can change.

PhoneGap Serve

```
phonegap serve [OPTIONS]
```

Sets up your PC as a simple web server so you can try the app.

- use this to test/debug the app on your PC
- with PhoneGap app on device for testing on the device
- much faster than generating/installing an app for each test
- can automatically reload app when source changes

EXAMPLE:

```
cd Documents/phonegap/ScottsApp
phonegap serve --port 3000
```

Then:

- Point browser at localhost:3000
- Use browser's developer tools
- Developer tools often have responsive design tools
- Point app at *your-pc-name*:3000

PhoneGap Build/Run

```
phonegap platform add <platform>
```

- "platform add" tells phonegaps which platform(s) you will build in your environment (ios, android, windows, and more…)
- it will verify installation of the needed software for the platforms

```
phonegap build [optional-plaform]
```

- "build" will create the app for the platforms you've added
- You can give the platform as a parameter to just build one platform, if omitted all platforms are built.

```
phonegap run <platform> [--device]
```

- "run" will run the app for the platform you've selected
- With --device, it will run on the device using USB cable
- Without --device, it will open an emulator for the platform

NOTE: Each platform will require software for that platform. See the docs for details.

Hello World Demonstration

The default template in phonegap is a simple hello world application.

- demonstrates an HTML/CSS page with an image and a message
- uses JavaScript to detect when the device is ready
- changes message when device is ready

Scott will demonstrate:
- generating the app (phonegap create)
- quick look at config.xml
- quick look at structure
- testing the app (phonegap serve)
- debugging an app (using browser)
- changing the app
- build app for android

Web Technology Review

PhoneGap Apps:
- built using web technology (HTML, CSS and JavaScript)
- then, generated into an app
- can be generated for any platform (iOS, Android, Blackberry, Windows)

However:
- Many RPG developers are not familiar with Web development
- It is too much to teach here!
- But, a quick review may be helpful?

But, I also need to keep this short
- Concentrate on PhoneGap rather than web development!
- …and how to interact with RPG programs.

HTML Code Review

<TAG-NAME> text data </TAG-NAME>
- TAG-NAME is the name of the HTML tag – and denotes the start
- /TAG-NAME is the end of the (most recent) tag with that name
- data in between can be nested tags or text data to be displayed

<TAG-NAME attr1="value" attr2="value">
- tags can have attribures (attr1, attr2 are examples)
- these look like variables with string values assigned after the tag name

Escaping Characters / Entities
- Entities are used to escape chars, or provide special chars
- Entities start with ampersand (the & symbol) and end with semicolon
- Examples: < > &

HTML Code Structure

```
<!DOCTYPE html>
<html>
  <head>
... title, meta tags, link for stylesheet(s), inline stylesheets, inline javascript functions ...
  </head>
  <body>
... tags for data to show, script tags that produce data to show ...
  </body>
</html>
```

- DOCTYPE identifies this as HTML5
- html starts/ends the html document
- head is "header" information
- body is "body of the document" (info displayed to user)

Common HTML Tags

- doctype, html, head, body (as shown)
- title = title of page (shown as browser tab name)
- meta = options for how the browser/app behaves
- script = associates the html file with JavaScript code
- link = associate the html file with CSS (style sheet) information
- div = page division or "section of page". Often designates a style.
- p = paragraph of text
- img = image/picture
- table = formats data in a tabular fashion
- form = contains fill-in form data
- input = tag for form inputs (text fields, buttons, dropdowns, etc)

Teaching all of HTML is too much for this talk. But there is lots of information on this on the web. Here's one good place:
https://www.w3schools.com/html/

HTML Code Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Scott's App</title>
    <meta name="format-detection" content="telephone=no" />
    <script type="text/javascript" src="cordova.js"></script>
    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <script type="text/javascript" src="js/app.js"></script>
  </head>
  <body>
    <div class="text-content">
       <p id="waitMsg">RPG Content Will Load Here</p>
    </div>
  </body>
</html>
```

CSS Review (1 of 2)

- CSS = Cascading Style Sheets
- Styles "cascade"
  - Inner tags override parent tags
  - more specific settings override more general (tag / class / id / inline)
  - If multiple files, they override in the order linked in
- Defines the specific look of the page elements
- For example, setting fonts, colors, margins, and positions of items

```
body {
  font-size: 16px;
  color: #000000;  /* html code for black */
}

.text-content {
  font-size: 14px;
}

#waitMsg {
  color: #d0d0d0;  /* a shade of gray */
}
```

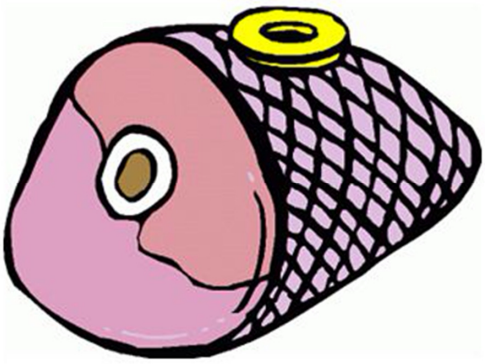

CSS Review (2 of 2)

```
div { /* applies to all div tags */ }

.my-class { /* applies to all elements with class="my-class" */ }
  font-size: 14px;
}

#my-id { /* applies to an element with id="my-id" */ }

div.my-class { /* applies to div tags with class="my-class" */ }
```

- These are "selectors"
- Selector with "dot" means class name
- Selector with "hash" means id
- Without dot or hash is the html tag name
- Tag+dot or Tag+hash means it applies only to tags with that class/id

Learn more here: https://www.w3schools.com/css/

What JavaScript is **NOT**



JAVASCRIPT IS NOT JAVA!!

"JavaScript is to Java what Hamster is to Ham"

They have similar names, but that's about it.

JavaScript Basics

```
var x = "value";
var y = 123;

x = 5;
y = "5";

if ( x == y )  /* attempts to convert types */
if ( x === y )  /* only matches when same type */
```

- Full programming language
- Interpreted (not compiled)
- loosely typed
- variable types are determined by what is assigned to the variable
- data type can change when a new value is assigned
- has everything you'd expect, if, while, concatenation, math operations, etc
- can communicate back to IBM i / RPG program
- can manipulate/change the screen on-the-fly
- code can run when a particular thing occurs

JavaScript Functions

```
function myFunction(elemid, message) {
  var content = document.getElementById(elemid);
  content.innerHTML = message;
}
myFunction("waitMsg", "Done waiting!");

var func2 = function(message) {
  var content = document.getElementById("waitMsg");
  content.innerHTML = message;
}
func2("Done waiting!");

function repeat(num, func) {
  for (var i=0; i<num; i++) {
     func(i);
  }
}
repeat(5, func2);
```

- Similar to subprocedures
- Defined with "function" keyword.
- Can be defined on-the-fly and assigned to a variable
- can be called from the variable
- can be passed as parameters and called via the parameter

JavaScript Events

```
function myFunc() {
  var content = document.getElementById("waitMsg");
  content.innerHTML = "Device is ready!";
}

document.addEventListener("deviceready", myFunc, false);
```

Mobile-specific events:

- deviceready = device is ready to use
- pause = app is paused (user switched to another app)
- resume = app has resumed from pause event
- backbutton, menubutton, searchbutton = special buttons on some devices

HTML also has a plethora of standard events:

- when an element is clicked/tapped
- when a key is pressed on the keyboard
- when data has loaded over the network
- etc.

Check Out W3 Schools

The focus of this session is PhoneGap, not web development.
…but, web development is crucial to PhoneGap!
…and many (most?) RPG developers aren't familiar with it!

Unfortunately, I don't have time to teach web development in this session.

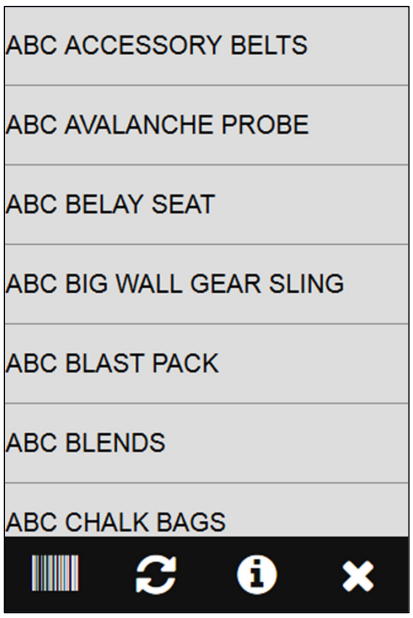
Check out the tutorials on http://w3schools.com

Important: Once you understand the basics, you don't need every detail!
- When you need something, find an example on the web!
- Use frameworks that do a lot of the work for you!

You will find this much more productive.

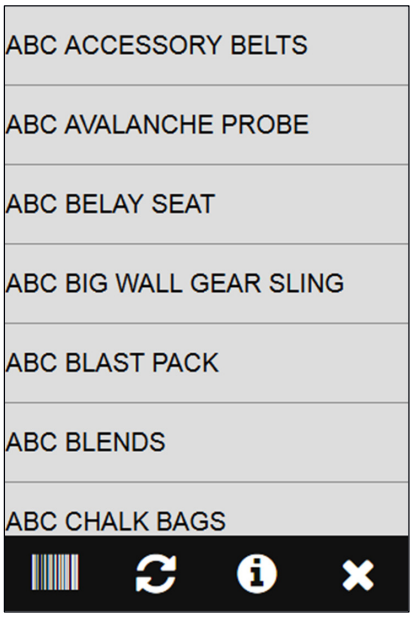
For example, look at the "HowTo" on http://w3schools.com
Also check out tutorials on jQuery mobile, BootStrap, Font Awesome, etc.

How Does It Work?



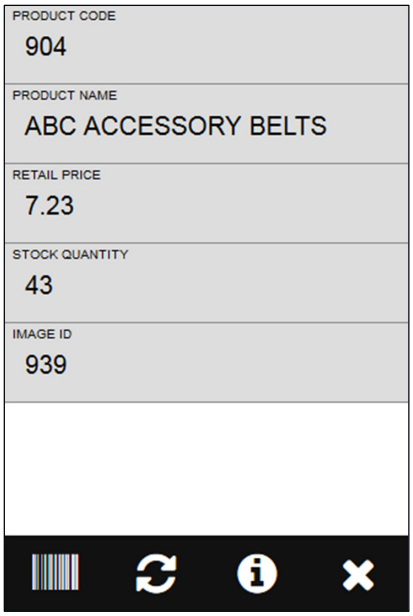
- Lists products from IBM i "PRODP" table
- refresh button will call RPG program to get data from PRODP
- X button will clear the data
- Info button tells about the app
- barcode button uses the camera to scan a barcode
- You can tap a product to view details

Demo App: Product Listing Screen



- Two divs "middle" and "bottom" contain the list and menu, respectively.
- "middle" contains an HTML table listing the products
- "bottom" icon bar uses the "font awesome" open source tool for the icons
- CSS sets the colors, fonts, spacing, etc.
- When you tap things, they call JavaScript routines.

Demo App: Product Details Screen



Shows details about a product

Works the same as the listing screen (Same divs, table, icon bar, etc)

This time the "X" icon will go back to the listing screen.

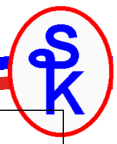
The other icons do the same thing as the listing screen does

HTML Code (1 of 2)

```
<head>
  <link rel="stylesheet" href="style.css"/>
  <link rel="stylesheet" href="font-awesome/css/font-awesome.min.css"/>
  <script type="text/javascript" src="code.js"></script>
  <script type="text/javascript" src="cordova.js"></script>
  <script type="text/javascript">
     beginSetup();
  </script>
</head>
```

- The <link> and <script> tags load the stylesheets and scripts, respectively
- These relative links point to the data in the phonegap project "www" directory
- The "cordova.js" file does not need to exist, it will be added by PhoneGap
- beginSetup() is called after all javascript is loaded

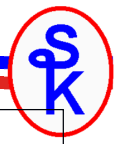
HTML Code (2 of 2)



```
<body>

  <div id="middle">
  </div>

  <div id="bottom" class="navbar icon-bar">
    <a href="#" onclick="scanItem()"><i class="fa fa-barcode"></i></a>
    <a href="#" onclick="refreshList()"><i class="fa fa-refresh"></i></a>
    <a href="#" onclick="appInfo()"><i class="fa fa-info-circle"></i></a>
    <a href="#" onclick="closeItem()"><i class="fa fa-close"></i></a>
  </div>

</body>
```

- the "middle" div is built dynamically by the JavaScript code
- the "icon bar" calls JavaScript routines when the icons are clicked (tapped)
- the CSS classes for the icons come from Font Awesome

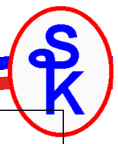
Communicating With RPG Using AJAX



```
function ajaxCall(method, url, data, handler) {

  var ajax = new XMLHttpRequest();
  ajax.open(method, url, true);
  ajax.setRequestHeader("content-type", "text/plain");

  ajax.onreadystatechange = function() {
    if (ajax.readyState == 4 && ajax.status == 200) {
      handler(ajax.responseText);
    }
  }

  ajax.send(data);
}
```

- This calls a URL with the HTTP network protocol
- A callback routine (handler) is run with the results when the arrive
- I've omitted error-handling for brevity
- The code in its entirety is available on my web site

Running the AJAX Routine



```
function refreshList() {

  ajaxCall("GET", "http://my-ibmi/webservices/prodinfo/json/list", null,
    function(data) {

      var list = JSON.parse(data);
      if (!list.success) {
        alert("Error: " + list.errMsg);
        return;
      }

      localStorage.setItem("product-list", data);
      loadList();
  });
}
```

- JSON.parse() parses JSON into a JavaScript array/object
- localStorage lets us store the data, it now can be used when off-line
- localStorage works in the browser, too (whereas a file plugin would not)

Format Of the Data Exchanged

```
{
  "success": true,
  "errMsg": "",
  "item": [
  {
    "code": "1234",
    "name": "Product Name",
    "price": 99.99,
    "stock": 1234,
    "image": "1234"
  },
  {  .. another product.. },
  {  .. another product.. },
  { etc }
}
```

```
var list = JSON.parse(data);
if (!list.success) {
  alert("Error: " + list.errMsg);
  return;
}

for (var i=0; i<list.item.length; i++) {
  table += "<tr>"
        +  "<td "onclick=\"viewItem('"
        +         list.item[i].code
        +       "')\">"
        +  list.item[i].name
        + "</td>"
        + "</tr>";
 }
```

- The loadList() routine (excerpt on right) builds the HTML table
- This table is inserted into the "middle" div

The RPG Program Overview

- Using IBM HTTP Server (powered by Apache)
- *(You could also use a web services server if you prefer)*
- Gets list of products from the PRODP database table (physical file)
- Generates JSON data using string concatenation
- Writes JSON back to the app by calling the IBM-supplied QtmhWrStout() API
- Your RPG is a regular RPG program – it can do any logic that it wants to.
- Learn more about web services to learn about all different ways of doing this.

URL Tells Apache What to Call

These Apache directives allow the URL to run an RPG program

```
ScriptAlias /webservices/prodinfo /qsys.lib/SKWEBSRV.LIB/PROD001R.PGM

<Directory /qsys.lib/SKWEBSRV.lib>
   Require all granted
</Directory>
```

- ScriptAlias maps URL into a program object to call
- Require all granted gives users access to use this library
- This runs under profile QTMHHTP1 (but this is configurable)

```
http://my-ibmi/webservices/prodinfo/json/list
```

- Apache sees the /webservices/prodinfo and calls SKWEBSRV/PROD001R

Running the AJAX Routine

```
exec SQL declare C1 cursor for
    select PRID, PNAME, PPRICE, PIMG, PSTOCK
      from PRODP order by PNAME;

  exec SQL open C1;
  exec SQL fetch next from C1 into :C1;

  jsonData = '{ +
                "success": true, +
                "errMsg": "No errors occurred", +
                "item": [';

  firstItem = *on;
```

Starts the JSON document

```
{
  "success": true,
  "errMsg": "message here",
  "item": [
... items will go here ...
  ]
}
```

- I removed the error handling to make this easier to digest
- code is continued on next slide.

Running the AJAX Routine

```
dow %subst(sqlstt:1:2) = '00' or %subst(sqlstt:1:2) = '01';

   if not firstItem;
      jsonData += ',';
   endif;

   firstItem = *off;

   jsonData += '{ +
                 "code": "'  + %char(c1.prid)   + '", +
                 "name": "'  + %trim(c1.pname)  + '", +
                 "price": "' + %char(c1.pprice) + '", +
                 "stock": "' + %char(c1.pstock) + '", +
                 "image": "' + %char(c1.pimg)   + '" +
               }';

   exec SQL fetch next from C1 into :C1;
enddo;

exec SQL close C1;

jsonData += ']}';
```

Puts a comma between each product

```
{
   "code": "1234",
   "name": "Product Name",
   "price": 99.99,
   "stock": 1234,
   "image": "1234"
}
```

Ends the JSON document

Sending The Results Back

```
dcl-s jsonData varchar(100000);
dcl-s headers varchar(500);
  .
  .
headers = 'Status: 200 OK' + CRLF
        + 'Content-Type: text/plain' + CRLF
        + CRLF;

QtmhWrStout( headers: %len(headers): ignore );
  .
  .
// code to create JSON is here ...
  .
  .
QtmhWrStout( jsonData: %len(jsonData): ignore );
```

The QtmhWrStout() API writes data back to Apache (which then sends it to the device)

Headers tell Apache what we're doing

- **Status = the HTTP status code (200=Success)**
- **a line with just CRLF ends the headers**

jsonData is the variable I put my JSON code into. It must be sent after the headers.

What Are Plugins?

Remember the limitations of using a web browser:

- Data only works when connected to network *(already solved, above)*
- *it cannot access the features of the device*

Plugins are native device code that we can run from JavaScript.

- access to the file system (open/read/write files on the device's disk)
- access to other hardware resources
  - camera – take pictures, scan barcodes, etc
  - GPS – coordinates of where the user is located
  - alerts / badges / push notifications – even when app isn't active
  - SMS (text messages) – we can send them!
  - accelerometer – detects when user moves the device around
  - vibrations – makes the phone vibrate

Read more about them on the Cordova site:
http://cordova.apache.org/docs/en/latest/

PhoneGap CLI For Plugins

The PhoneGap CLI allows you to work with the plugins in your project:

- `phonegap plugin list`

  Views the plugins currently in your project. The core plugins are added automatically, but you can see them by running the "plugin list" command.

- `phonegap plugin add <plugin>`

  Adds an additional plugin into the list. The plugin can reference a directory on your PC, a URL to a git project on the web, or the name of a core plugin.

- `phonegap plugin remove <plugin name>`

  Removes a plugin from your project.

For example, I want the cordova barcode scanner, but don't want the splash screen, so:

```
phonegap plugin add phonegap-plugin-barcodescanner
Phonegap plugin add cordova-plugin-whitelist
```

Barcode Scanner Plugin

Different plugins work differently.
- some provide their own JavaScript events
- most provide JavaScript routines you can call

View the documentation for your plugin for details.

The BarcodeScanner plugin docs are here (need to scroll down)
https://github.com/phonegap/phonegap-plugin-barcodescanner

This plugin defines a cordova.plugins.barcodeScanner.scan() API.

It has three parameters:
- function called when scanned successfully
- function called when an error occurs
- (optional) JavaScript object with options you can configure

These functions are passed information about what happened.

JavaScript to Call Scanner

```
function scanItem() {

  var scanner = null;
  if (cordova && cordova.plugins && cordova.plugins.barcodeScanner) {
    scanner = cordova.plugins.barcodeScanner;
  }
  if (!scanner) {
    alert("Barcode scanner plugin not found!");
    return;
  }

  scanner.scan( scanGood, scanError);

}
```

This code will:
- check if the barcode scanner plugin is there.
- if not there, show an error
- if there, call the scan() API to scan a barcode

Results Are Passed to Functions

```
function scanGood(result) {

  if (result.cancelled) {
    return;
  }

  viewItem(result.text);
}
```

result has 3 fields:

- **format = barcode type (not used here)**
- **text = data read from barcode**
- **cancelled = set to '1' if user cancelled the scanner**

The viewItem() routine (routine not shown) is passed the barcode. It will then find that item and view it on the display.

```
function scanError(msg) {

  alert("Scan failed: " + msg);

}
```

errors are passed the error message as a parameter. You can simply show it to the user.

This Presentation

You can download a PDF copy of this presentation and the sample code that I used from

http://www.scottklement.com/presentations/

Thank you!