

# BB10 and WebWorks 201 – Advanced BlackBerry 10 application development using Ripple and the WebWorks SDK

DEV145

@n\_adam\_stanley, @ken\_wallis, @confusement

May 1-3, 2012

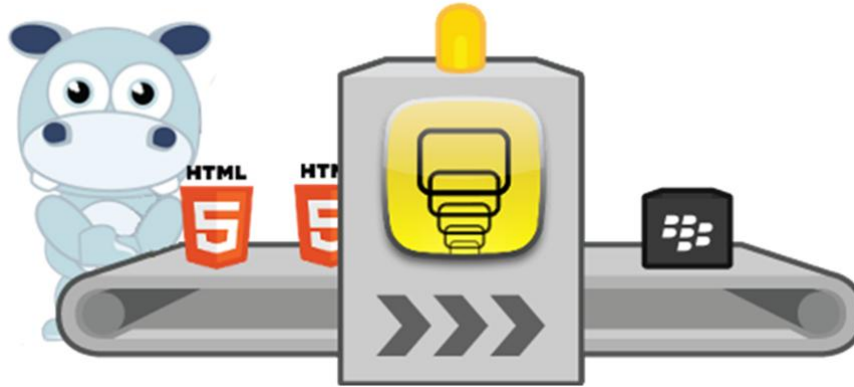
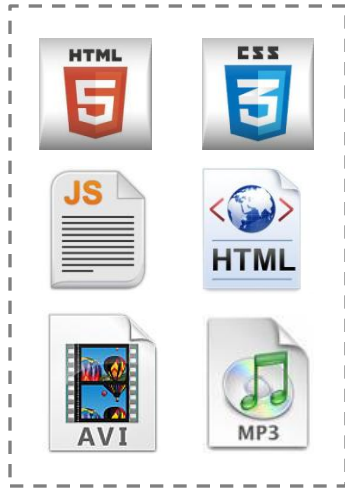
# 1. Intros and demo

What are we going to do today?

[5 mins]

# How to build apps?

**BlackBerry 10 Jam**

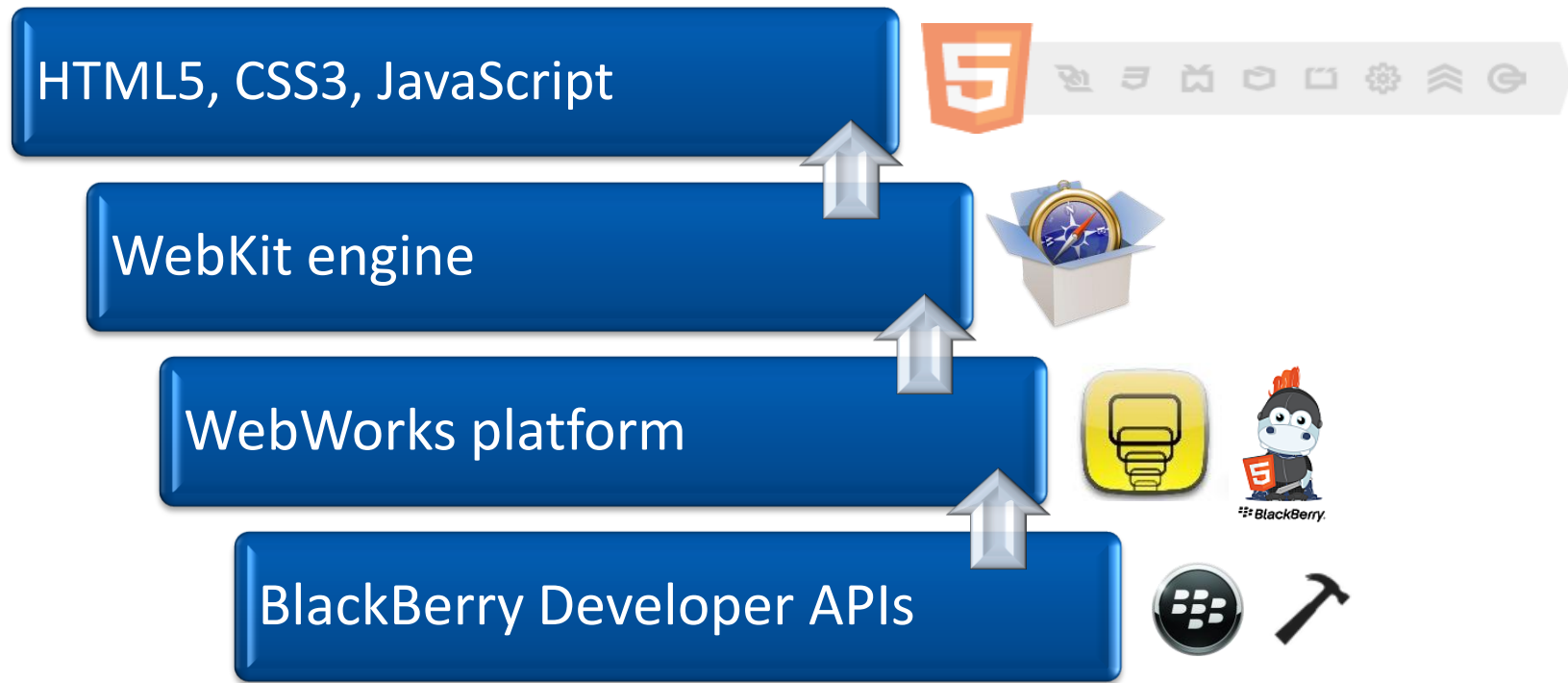


Web Assets

Ripple + WebWorks

BlackBerry Applications

# HTML5 powered by native capabilities BlackBerry 10 Jam



<https://developer.blackberry.com/html5>

# Today's Tasks

- bbUI.js [20 mins]
- WebWorks config.xml [20 mins]
- HTML5 features [30 mins]
- WebWorks APIs [30 mins]

# Time for a demo

 **BlackBerry** 10 Jam



## 2. Setup

Development tools, environment, starter code

[10 mins]

- Assumptions:
  - ▶ You are already familiar with BlackBerry WebWorks
  - ▶ You already have code signing keys
- Must have:
  - ▶ Laptop (Windows XP, 7 or Mac OS)
  - ▶ Local web server
  - ▶ Ripple extension for BB10
  - ▶ WebWorks SDK for BB10



- Skip to the next section **unless** you need to learn how to:
  - ▶ Install Ripple extension
  - ▶ Setup a local Web server
  - ▶ Setup a local WebWorks project
  - ▶ Install WebWorks SDK for BB10
  - ▶ Build using Ripple
  - ▶ Perform code signing
  - ▶ Deploy your application to a simulator
  - ▶ Debug using web inspector

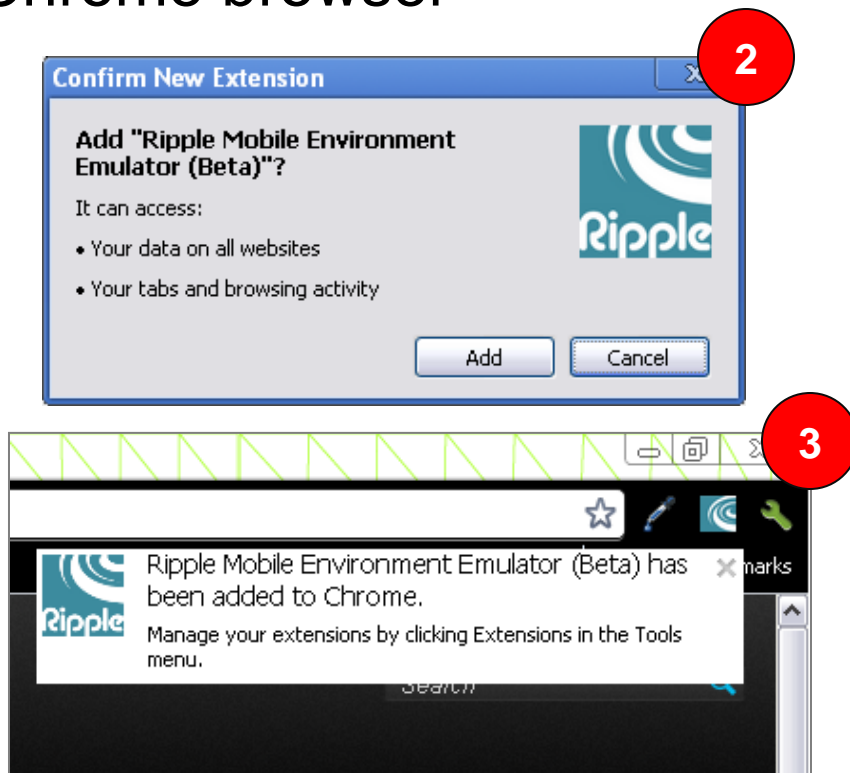
- Download and install the Ripple extension
  - ▶ <http://developer.blackberry.com/html5>
- Recommended installation folder
  - ▶ Windows: C:\program files\Research In Motion\Ripple <version>
  - ▶ Mac: /Applications/Research In Motion/Ripple <version>



- Find **ripple\_ui.crx** file:
  - ▶ Windows: C:\program files\Research In Motion\Ripple <version>
  - ▶ Mac: /Applications/Research In Motion/Ripple <version>
- Launch chrome browser

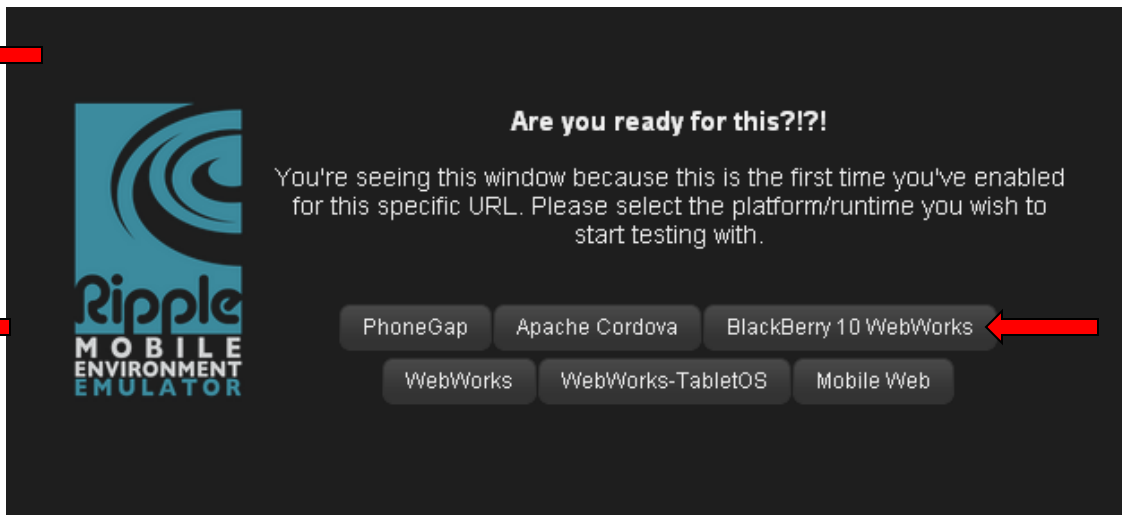
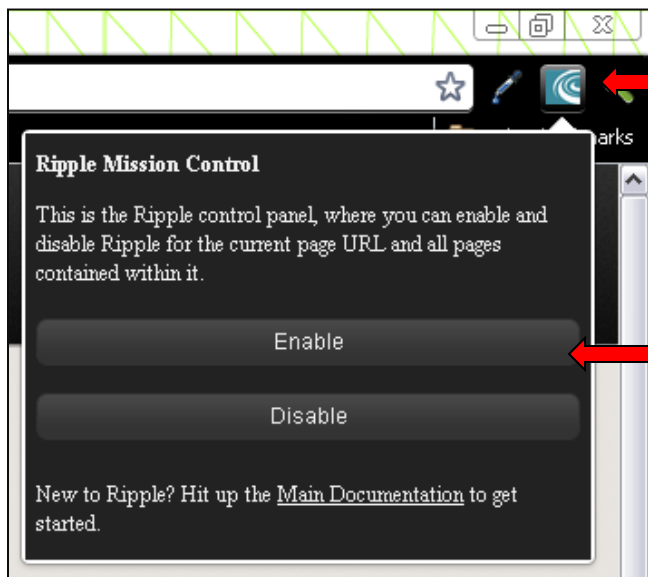
# Install Ripple extension

- Drag `ripple_ui.crx` into the Chrome browser



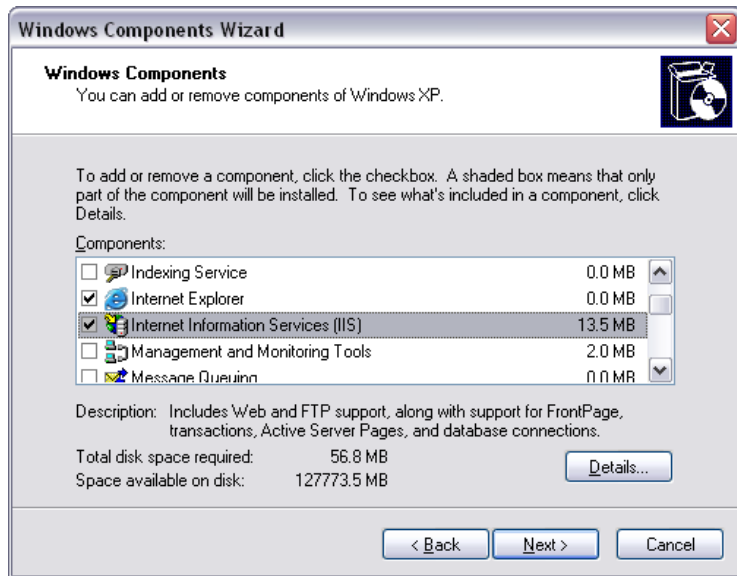
# Install Ripple extension

- Open <http://devblog.blackberry.com> using Chrome browser
- Click on Ripple extension icon → Enable
- Choose 'BlackBerry 10 WebWorks' platform



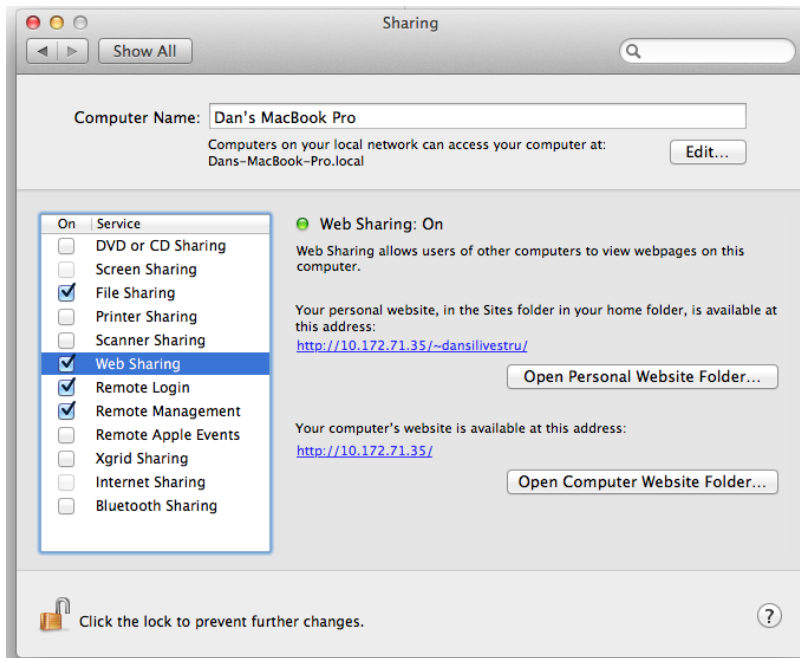
# Setting up a web server

- Windows: Enable Internet Information Services (IIS)
  - ▶ **XP** : Control Panel → Add / remove programs → Windows Components
  - ▶ **Win7**: Control Panel → Programs → Windows Features



# Setting up a web server

- Mac: Enable Web sharing
  - ▶ System Preferences → Sharing → Web Sharing



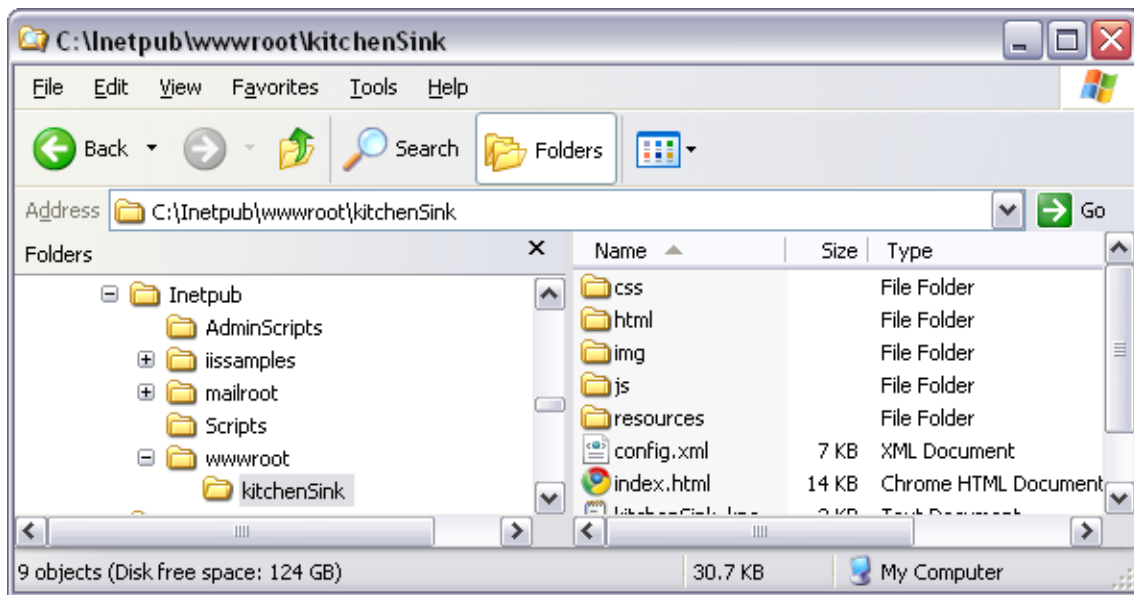
- Sample application for developers
  - ▶ Over 100 samples: “Everything but the kitchen sink”
  - ▶ HTML5
  - ▶ CSS3
  - ▶ WebWorks APIs
  - ▶ BlackBerry web platform capabilities
- Download the source code
  - ▶ <http://github.com/blackberry/WebWorks-Samples>
  - ▶ ZIP archive will contain a kitchenSink folder



# Setup a local WebWorks project

 **BlackBerry** 10 **Jam**

- Extract **kitchenSink** folder into web server “working” folder
  - ▶ Windows default: **C:\inetpub\wwwroot\kitchenSink**
  - ▶ Mac default: **~/Sites/kitchenSink**

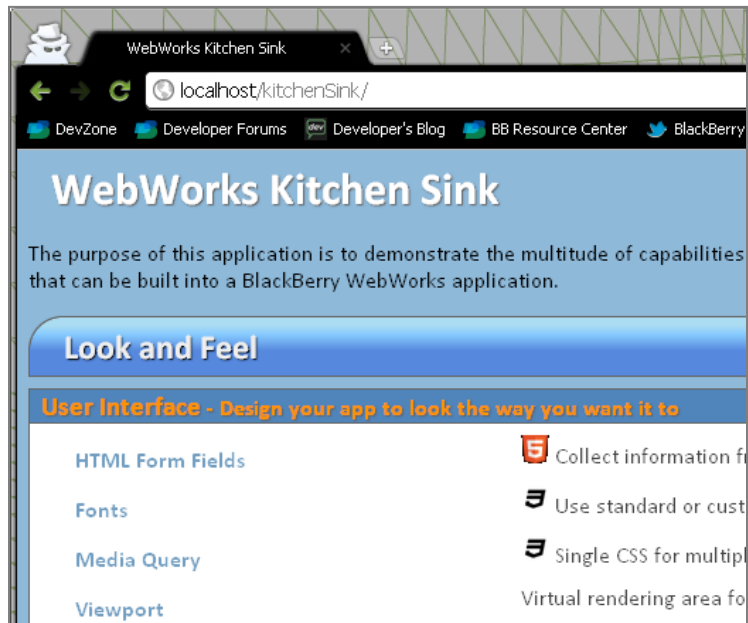


# Setup a local WebWorks project

 **BlackBerry** 10 **Jam**

- Can now load <http://localhost/kitchenSink>

## Browser



## Ripple extension



- Package WebWorks assets into a BlackBerry application
- Required:
  - ▶ BlackBerry WebWorks SDK for BB10
  - ▶ Ripple extension for Chrome
- Optional: BlackBerry 10 simulator
  - ▶ VMware Player (Windows)
  - ▶ VMware Fusion (Mac)

- Install the BlackBerry WebWorks SDK for BB10
  - ▶ <http://developer.blackberry.com/html5/download>

## Package and Distribute

Are you ready to start packaging your BlackBerry WebWorks applications? Choose one of the following SDKs to help you package your application files and distribute your app.

### BlackBerry 10 WebWorks SDK

Want to get started on developing your WebWorks based BlackBerry 10 application? This is the package you want to have.



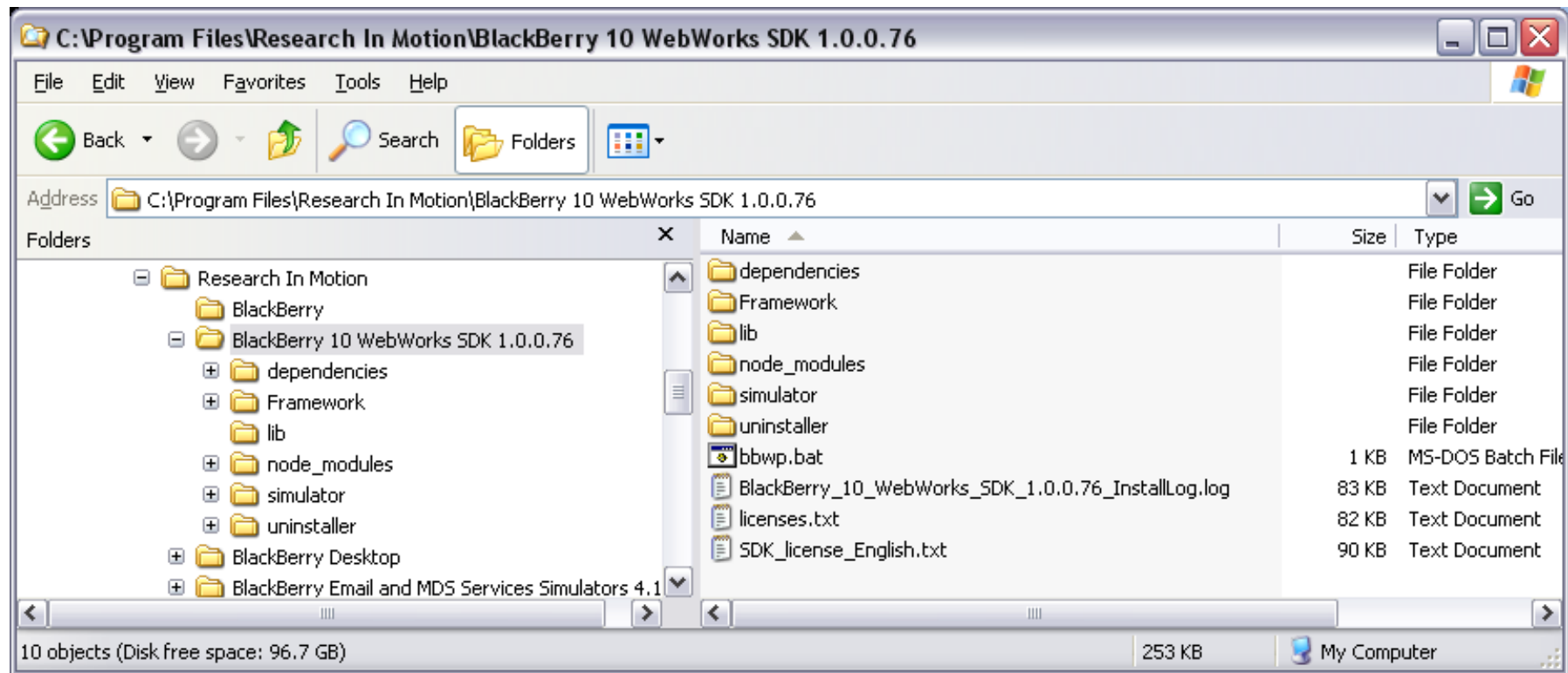
#### BlackBerry 10 WebWorks SDK

1.0.0 beta for Windows (305 Mb)

[For Mac?](#) | [System Requirements](#) | [Previous Versions](#)

# Install WebWorks SDK for BB10

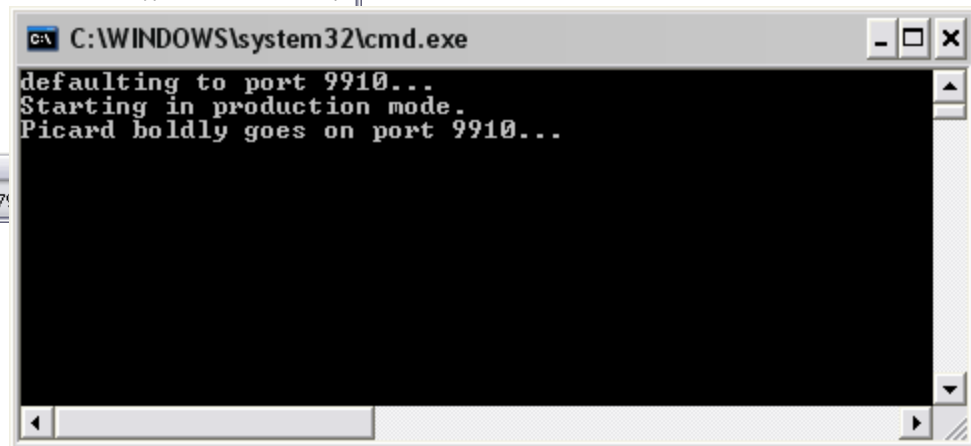
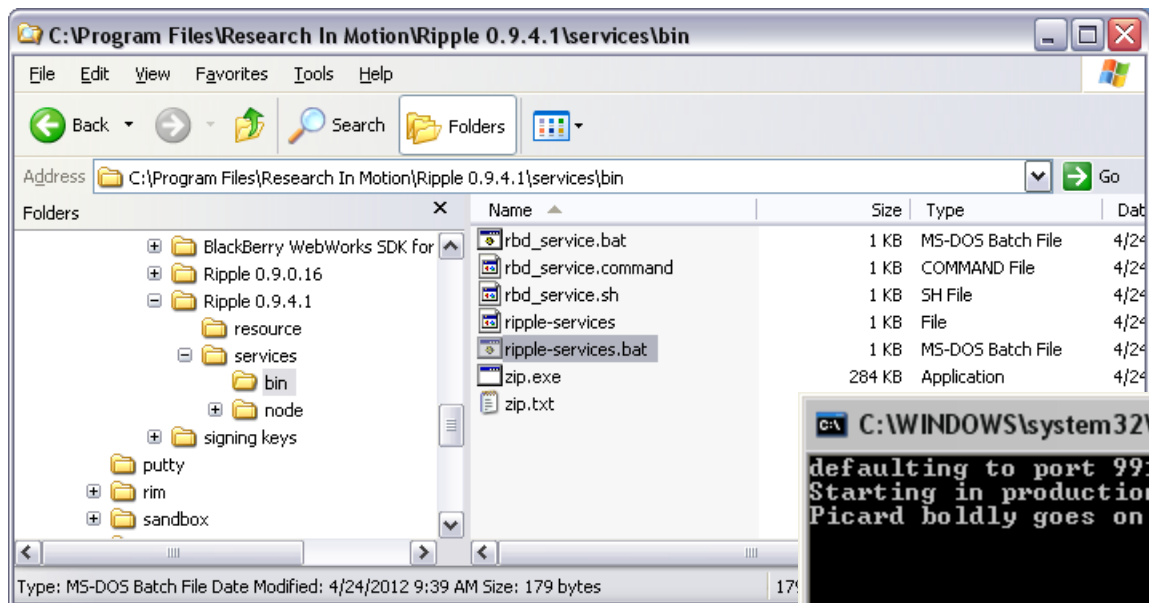
 **BlackBerry** 10 Jam



- Ripple extension can compile, sign and deploy apps!
  - ▶ Start **ripple-services** command line utility
  - ▶ Opens port 9910 for use
- Run services\bin\**ripple-services.bat**:
  - ▶ Windows: C:\program files\Research In Motion\Ripple <version>
  - ▶ Mac: /Applications/Research In Motion/Ripple <version>
- Keep command window open

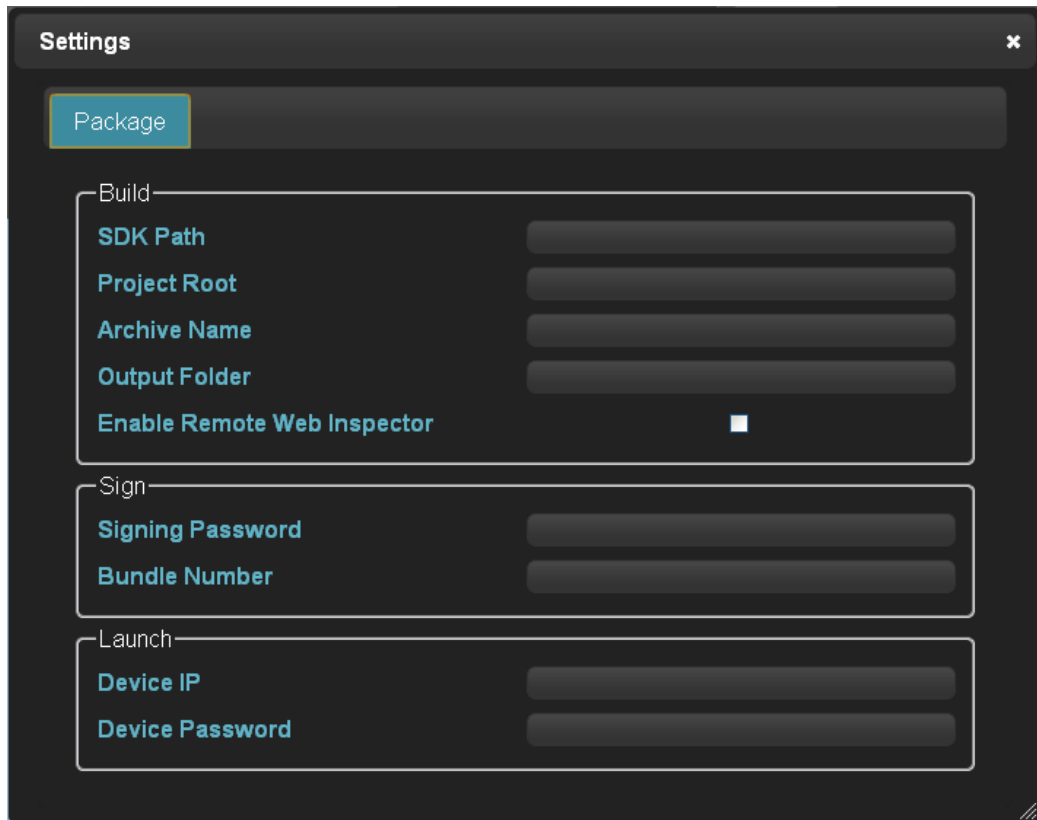
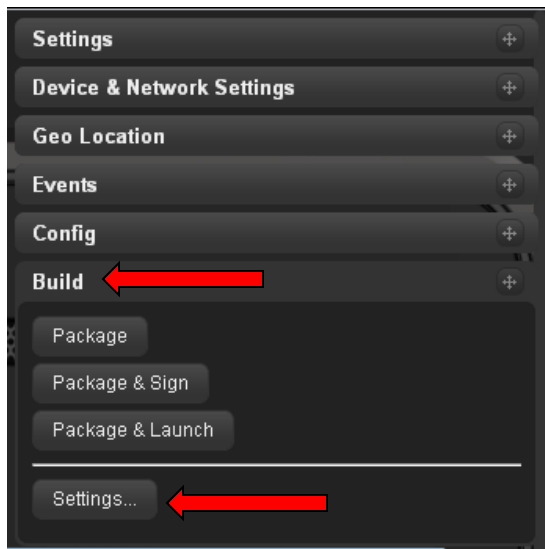
# Build environment setup

 **BlackBerry** 10 Jam



# Build using Ripple

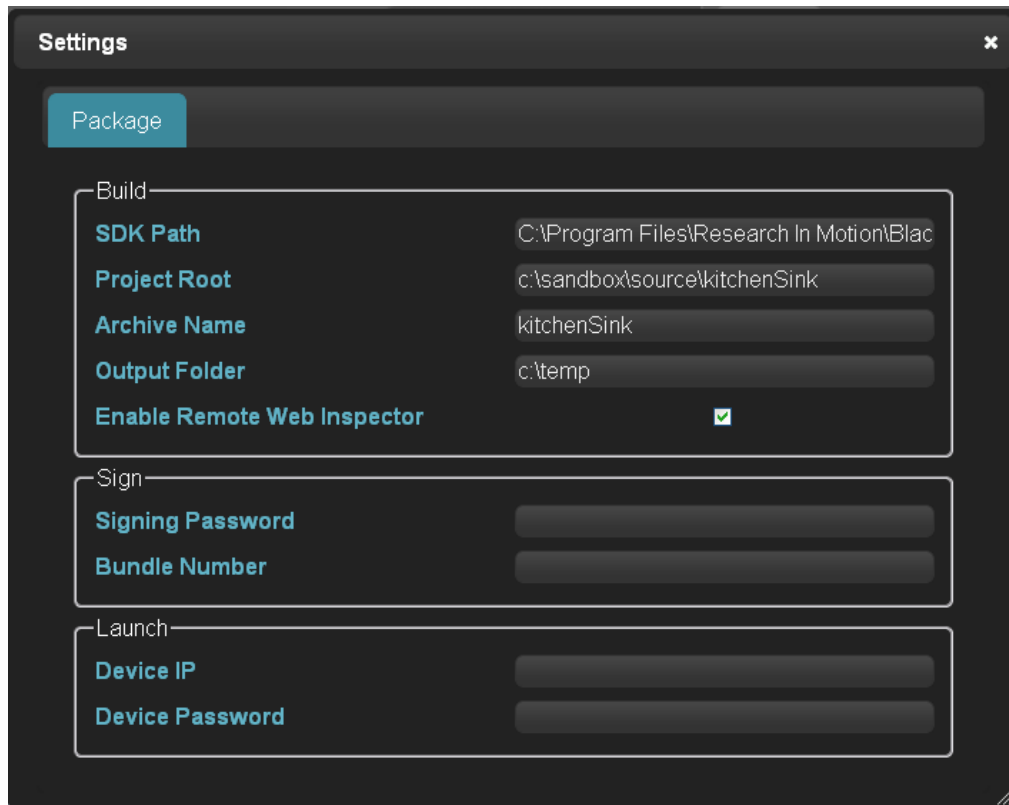
- Open Ripple extension
- Expand Build tab
- Click Settings button





# Build using Ripple

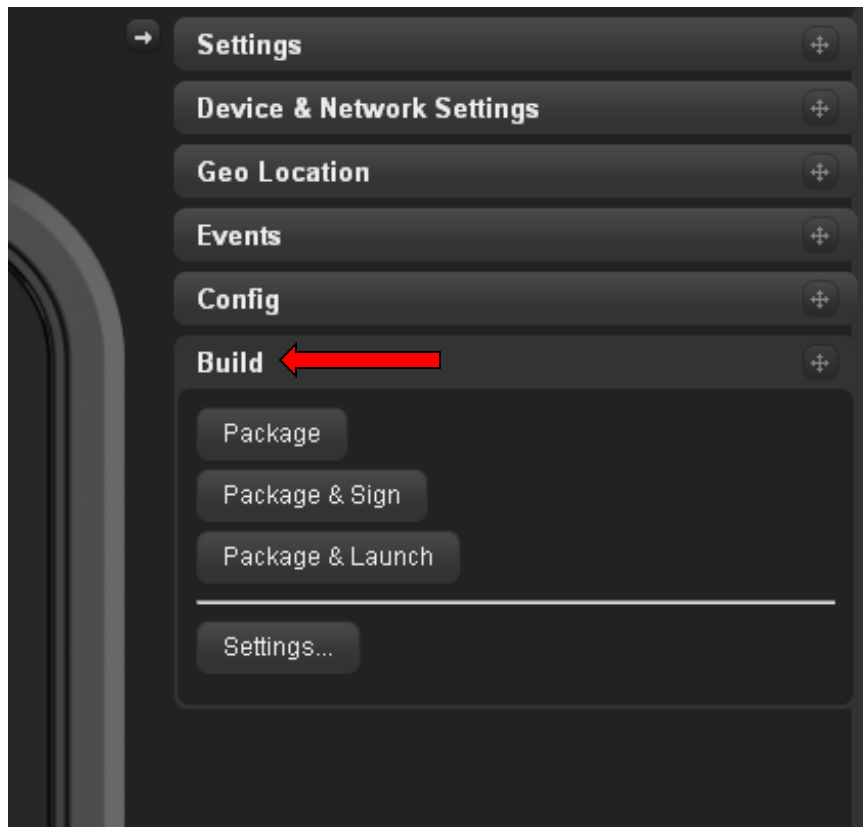
- SDK Path
  - ▶ BBWP installer directory
- Project Root
  - ▶ Application source code
- Archive Name
  - ▶ BAR file name
- Output Folder
  - ▶ Different than project root



The screenshot shows the 'Settings' dialog box for BlackBerry 10 Jam. It has a title bar with 'Settings' and a close button. Below the title bar is a 'Package' tab. The settings are organized into three sections: 'Build', 'Sign', and 'Launch'. The 'Build' section contains: 'SDK Path' (C:\Program Files\Research In Motion\Blac), 'Project Root' (c:\sandbox\source\kitchenSink), 'Archive Name' (kitchenSink), 'Output Folder' (c:\temp), and 'Enable Remote Web Inspector' (checked). The 'Sign' section contains: 'Signing Password' and 'Bundle Number' (both empty). The 'Launch' section contains: 'Device IP' and 'Device Password' (both empty).

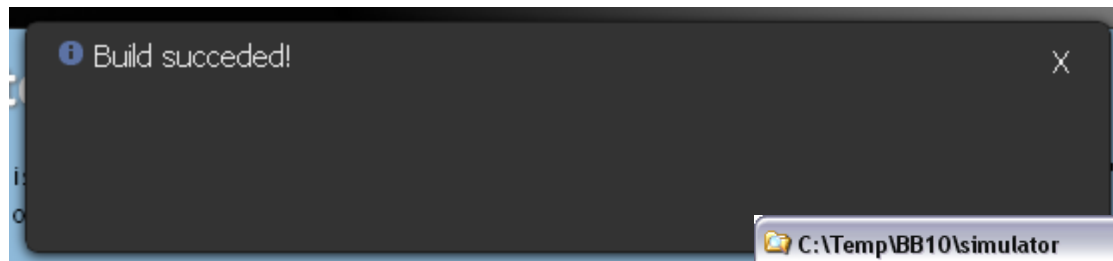
Section	Setting	Value
Build	SDK Path	C:\Program Files\Research In Motion\Blac
	Project Root	c:\sandbox\source\kitchenSink
	Archive Name	kitchenSink
	Output Folder	c:\temp
	Enable Remote Web Inspector	<input checked="" type="checkbox"/>
Sign	Signing Password	
	Bundle Number	
Launch	Device IP	
	Device Password	

- Open Build tab
  - ▶ Package
    - For simulators
  - ▶ Package & Sign
    - For live devices
  - ▶ Package & Launch
    - Deploy to simulator

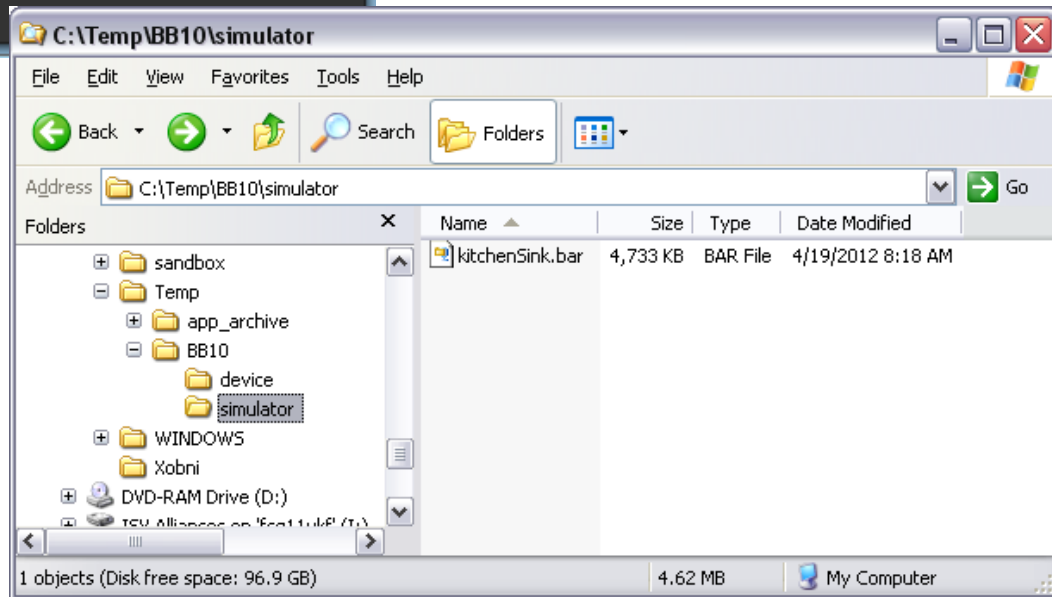


# Build using Ripple

 **BlackBerry** 10 Jam



- Two BAR files created
- Deployment targets:
  - ▶ device
  - ▶ simulator



- BlackBerry 10 Dev Alpha simulator
  - ▶ Load BlackBerry10Simulator.vmx in VMware
  - ▶ C:\Program Files\Research In Motion\BlackBerry 10 WebWorks SDK <version>\simulator
- VMware player is available from:
  - ▶ <http://www.vmware.com/products/player>

- Start simulator
  - ▶ Enable development mode
- Use Ripple to deploy unsigned app to simulator
  - ▶ Enter IP address and Password in settings screen
  - ▶ Select “Package & Launch” option
  - ▶ Deploy to VMWare simulator

# Deployment

 **BlackBerry** 10 Jam

**Settings** ✕

Package

Build

SDK Path

C:\Program Files\Research In Motion\Blac

Project Root

c:\sandbox\source\kitchensink

Archive Name

kitchensink

Output Folder

c:\temp

Enable Remote Web Inspector

☒

Sign

Signing Password

Bundle Number

Launch

Device IP

192.168.1.7

Device Password

.....

→ **Settings** ⊕

Device & Network Settings

⊕

Geo Location

⊕

Events

⊕

Config

⊕

Build

⊕

Package

Package & Sign

Package & Launch

Settings...

- Apps must be signed to run on a live device
  - ▶ Required in order to deploy to BlackBerry App World
- Register for keys
  - ▶ <https://www.blackberry.com/SignedKeys>
- Install keys
  - ▶ <http://bit.ly/JKTsfu>

- Open command prompt and navigate to
  - ▶ C:\Program Files\Research In Motion\BlackBerry 10 WebWorks SDK <version>\dependencies\tools\bin
- Install the keys:

```
blackberry-signer -register -csjpin <csj pin>  
                  -storepass <KeystorePassword> <client-RDK-xxxxxx.csj file>  
                  <client-PBDT-xxxxxx.csj file>
```

- Use Ripple to “Package & Sign”



- Use blackberry-deploy to side-load a signed app
  - ▶ Command line tool found in
    - C:\Program Files\Research In Motion\BlackBerry 10 WebWorks SDK <version>\dependencies\tools\bin
- Deploy to
  - ▶ a live device (app must be signed)
  - ▶ a simulator (app does not have to be signed)

```
blackberry-deploy -installApp -device <Device IP> -package  
<Compiled BAR> -password <Device PWD>
```

```
C:\Program Files\Research In Motion\BlackBerry 10 WebWorks SDK  
1.0.0.76\dependencies\tools\bin>blackberry-deploy -installApp  
-device 192.168.198.134 -package "c:\temp\kitchenSink.bar"  
-password 1234
```

```
Sending Install request...
```

```
Info: Action: Install
```

```
Info: File size: 40731
```

```
Info: Installing ...
```

```
actual_dname::DEV8281a833da63a6b7e2098dae6d0662e1.MjA5OG
```

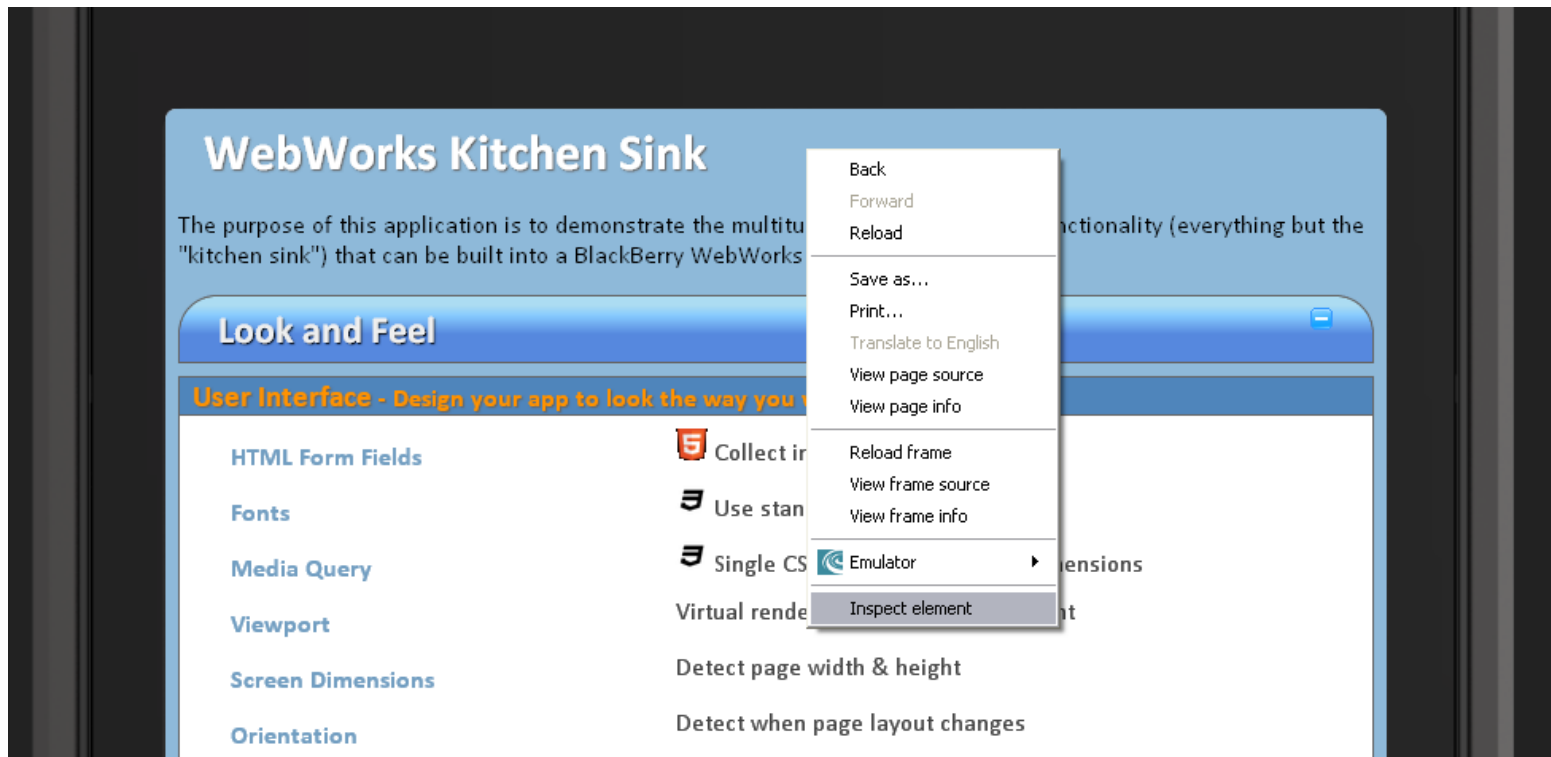
```
RhZTZkMDY2MmUxICAgICA
```

```
actual_id::MjA5OGRhZTZkMDY2MmUxICAgICA
```

```
actual_version::1.0.0.0
```

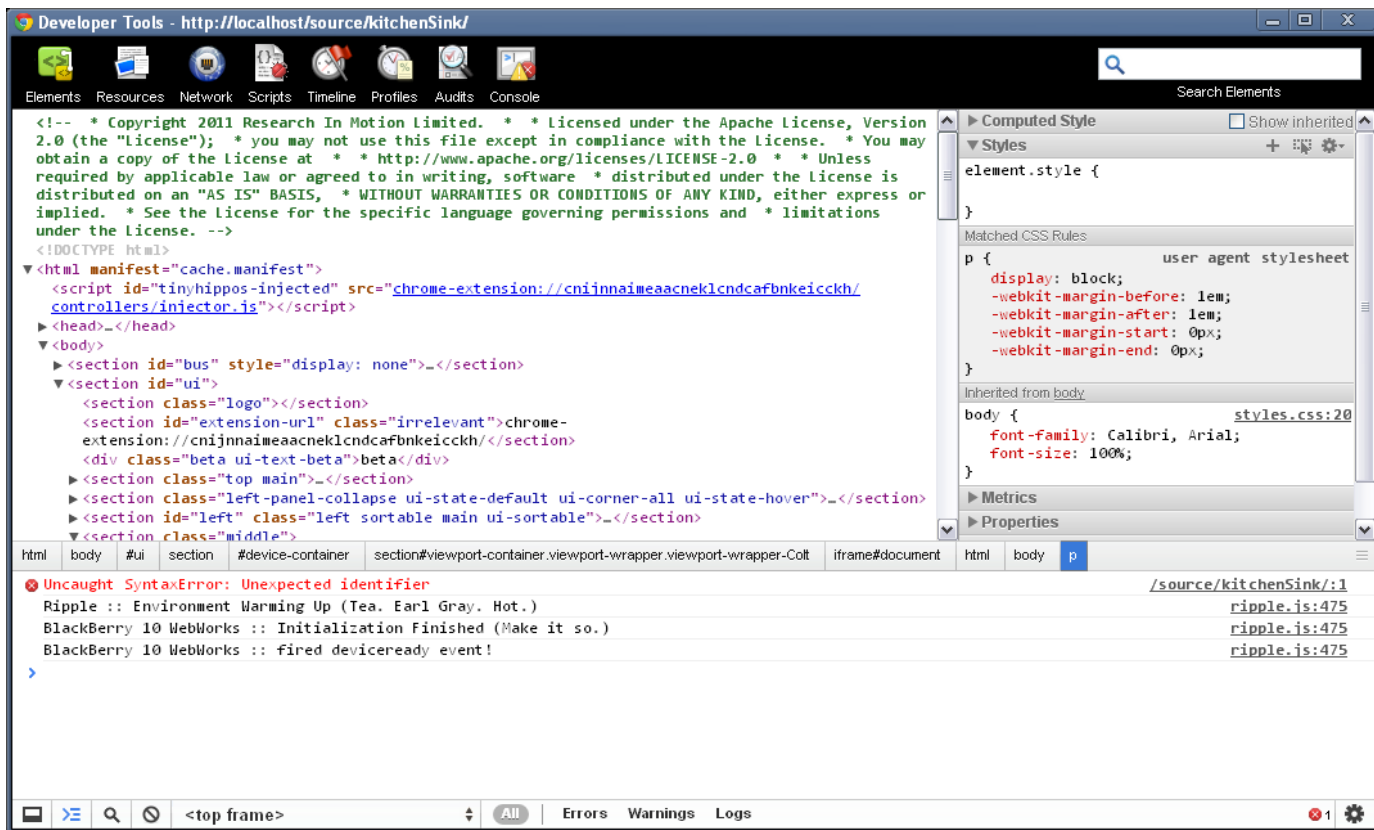
```
result::success
```

- Right click content window → “Inspect Element”



# Web Inspector debugging

 BlackBerry 10 Jam



The screenshot displays the BlackBerry 10 Web Inspector interface. The top toolbar includes icons for Elements, Resources, Network, Scripts, Timeline, Profiles, Audits, and Console. The main content area shows the HTML document structure, with a syntax error highlighted: "Uncaught SyntaxError: Unexpected identifier". The error message is "Uncaught SyntaxError: Unexpected identifier" and the location is "/source/kitchenSink:1". The error details show the following code snippet:

```
<!-- * Copyright 2011 Research In Motion Limited. * Licensed under the Apache License, Version 2.0 (the "License"); * you may not use this file except in compliance with the license. * You may obtain a copy of the license at * http://www.apache.org/licenses/LICENSE-2.0 * Unless required by applicable law or agreed to in writing, software * distributed under the license is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the license for the specific language governing permissions and * limitations under the license. -->
<!DOCTYPE html>
<html manifest="cache.manifest">
  <script id="tinyhippos-injected" src="chrome-extension://cniinnaimeaacneklcndcafbnkeicckh/controllers/injector.js"></script>
  <head>_</head>
  <body>
    <section id="bus" style="display: none;">_</section>
    <section id="ui">
      <section class="logo"></section>
      <section id="extension-url" class="irrelevant">chrome-extension://cniinnaimeaacneklcndcafbnkeicckh/</section>
      <div class="beta ui-text-beta">beta</div>
      <section class="top main">_</section>
      <section class="left-panel-collapse ui-state-default ui-corner-all ui-state-hover">_</section>
      <section id="left" class="left sortable main ui-sortable">_</section>
      <section class="middle">_</section>
    </section>
  </body>
</html>
```

The right-hand pane shows the "Computed Style" for the selected element, displaying the "element.style" and "Matched CSS Rules" (user agent stylesheet) for the "p" element. The "Inherited from body" section shows the "body" style with "font-family: Calibri, Arial;" and "font-size: 100%;". The bottom status bar indicates the current frame is "<top frame>" and shows a total of 1 error.

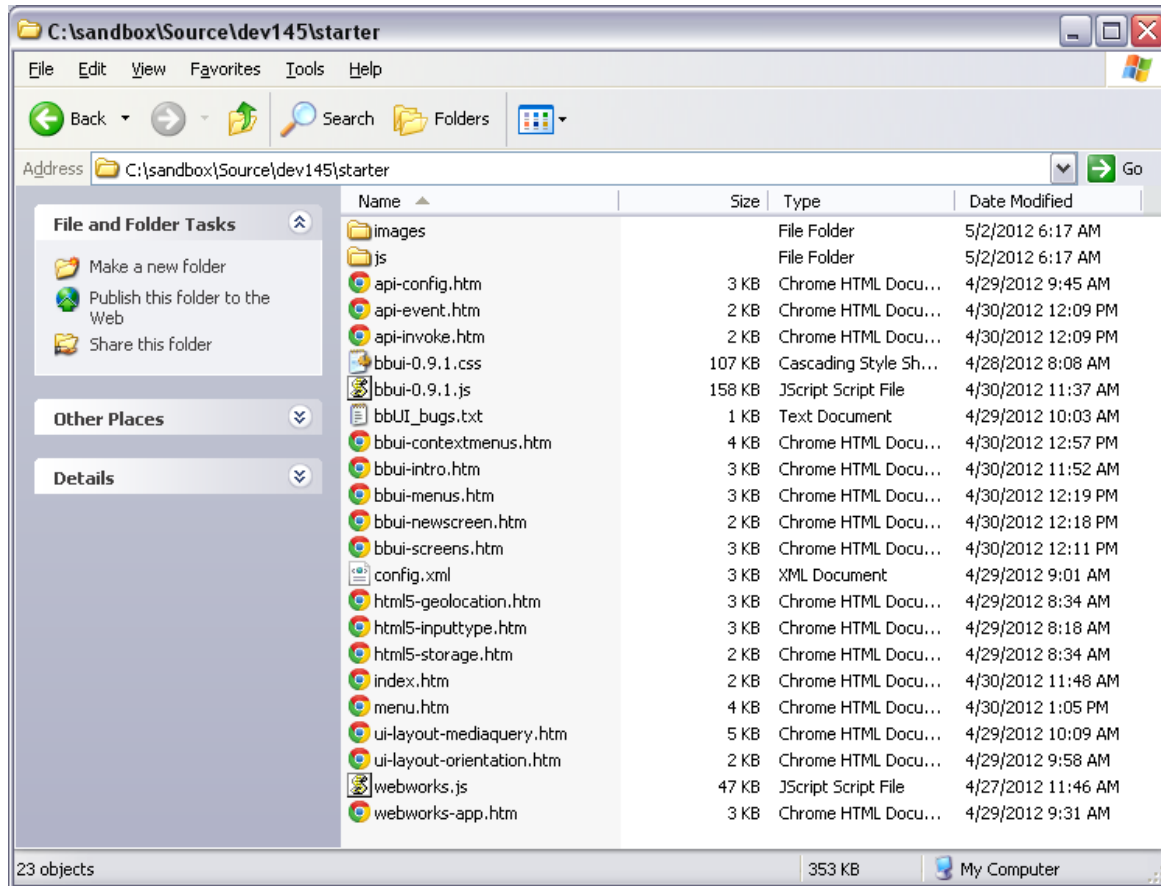
## 3. Getting started

Build, sign and deploy the starter code

[10 mins]

- Create folder dev145 in working folder of local Web server
  - ▶ E.g. c:\inetpub\wwwroot\dev145
- Download starter sample code
  - ▶ <http://github.com/astanley/Presentations/DEV145>
- Extract contents of **starter** from ZIP to dev145 folder

# Setting up the lab

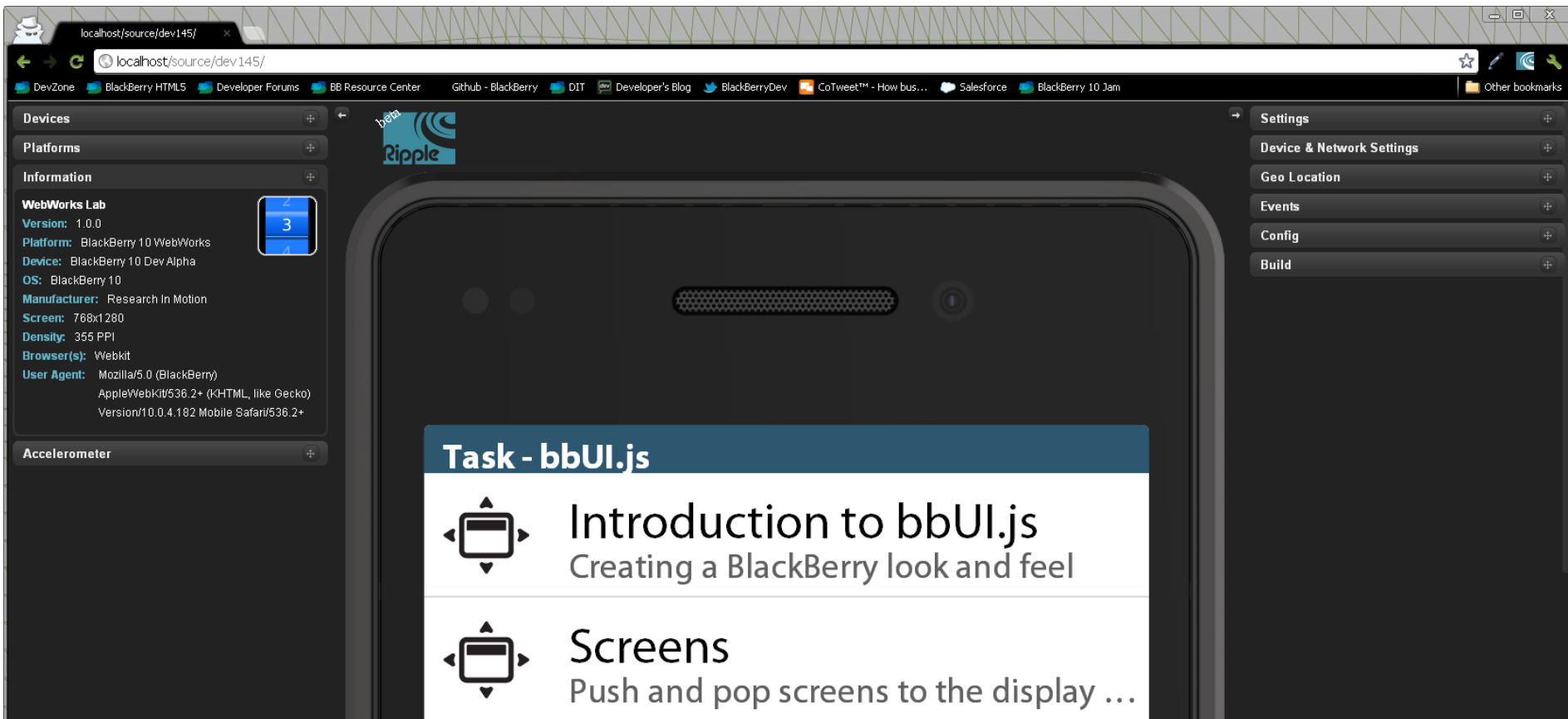


- Open and preview starter app using Ripple
  - ▶ <http://localhost/dev145>
- Build, sign and deploy starter code to live device



# Setting up the lab

 **BlackBerry** 10 Jam



The screenshot shows a web browser interface for the BlackBerry 10 Jam development environment. The browser is displaying a page with a task list titled "Task - bbUI.js". The page content includes two items:

- Introduction to bbUI.js**  
Creating a BlackBerry look and feel
- Screens**  
Push and pop screens to the display ...

The browser's address bar shows the URL `localhost/source/dev145/`. The left sidebar contains a "Devices" panel with a "WebWorks Lab" section, showing details for a BlackBerry 10 Dev Alpha device. The right sidebar contains a "Settings" panel with options like "Device & Network Settings", "Geo Location", "Events", "Config", and "Build".

**WebWorks Lab Information:**

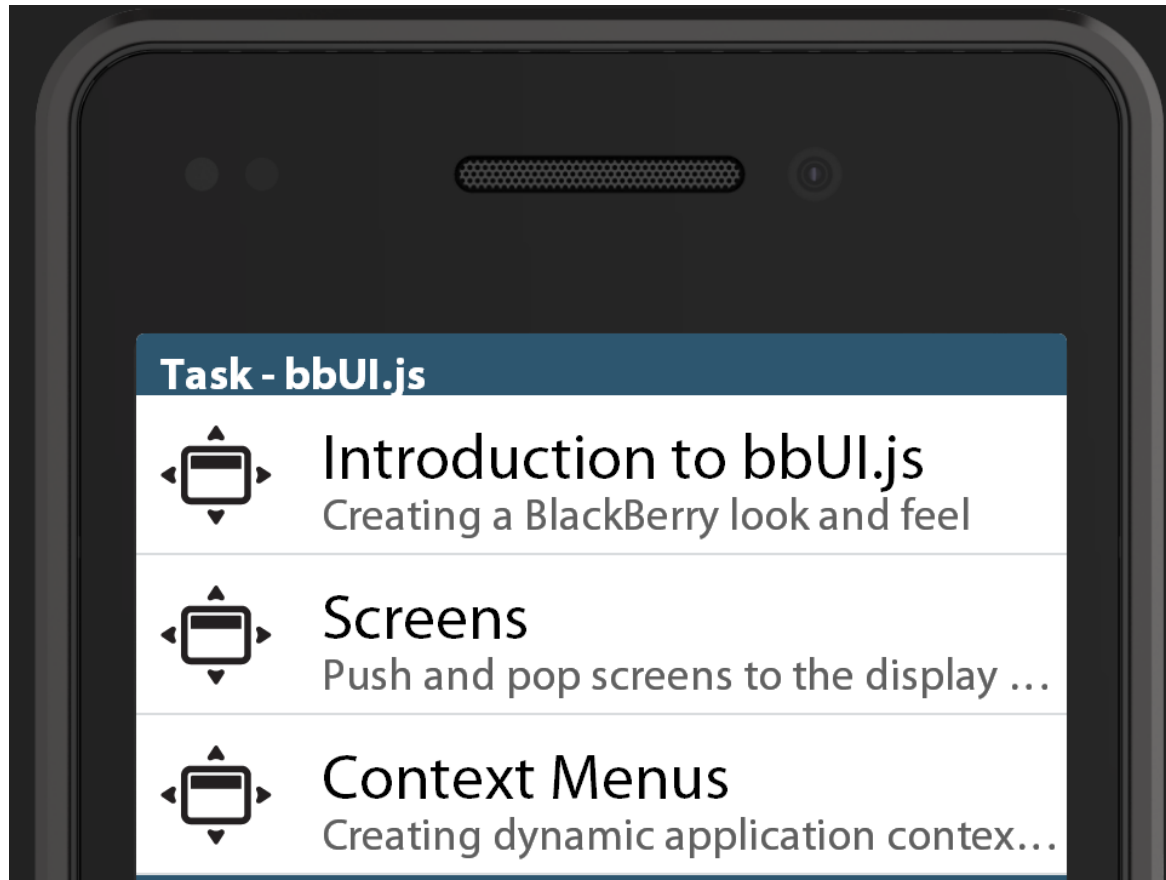
- Version: 1.0.0
- Platform: BlackBerry 10 WebWorks
- Device: BlackBerry 10 Dev Alpha
- OS: BlackBerry 10
- Manufacturer: Research In Motion
- Screen: 768x1280
- Density: 355 PPI
- Browser(s): Webkit
- User Agent: Mozilla/5.0 (BlackBerry; AppleWebKit/536.2+ (KHTML, like Gecko) Version/10.0.4.182 Mobile Safari/536.2+

**Accelerometer:**

## 4. Task: UI – bbUI.js

Creating that BlackBerry look and feel

[20 mins]



- Creating that BlackBerry look and feel with bbUI.js
- Steps:
  - ▶ Explore sample to discover how **bbUI.js** works
  - ▶ Change platforms in Ripple to see changes in UI
  - ▶ Edit **index.htm** and modify bb.init() params object
  - ▶ Change default color of bb10AccentColor to #FF0000
  - ▶ Reload in Ripple to see UI changes
  - ▶ Restore bb10AccentColor to #2D566F

- Push and pop screens to the display stack
- Steps:
  - ▶ Edit **bbui-screens.htm** and assign onclick event to button that pushes **bbui-newscreen.htm** onto the display stack.
  - ▶ Test using Ripple, verify “New Screen” is opened when button selected from **bbui-screens.htm**.
  - ▶ Edit **bbui-newscreen.htm** and add fade effect to <div data-bb-type=“screen”> element.

- Creating dynamic application context menus
- Steps:
  - ▶ Test using Ripple how context menus behave in BB10
  - ▶ Edit **bbui-contextmenu.htm**, adding more menu items to the context menu defined on that page.
  - ▶ Add a second image item that calls peekContextMenu()
  - ▶ Edit **js/bbui-contextmenu.js** and create clickHandler(param)
  - ▶ Assign onclick="clickHandler('ABC') to each context menu

## 5. Task – Config.xml

Application properties, behaviors and security

[20 mins]



## Context Menus

Creating dynamic application contex...

**Task - config.xml**



WebWorks configuration  
document

**Task - HTML5**

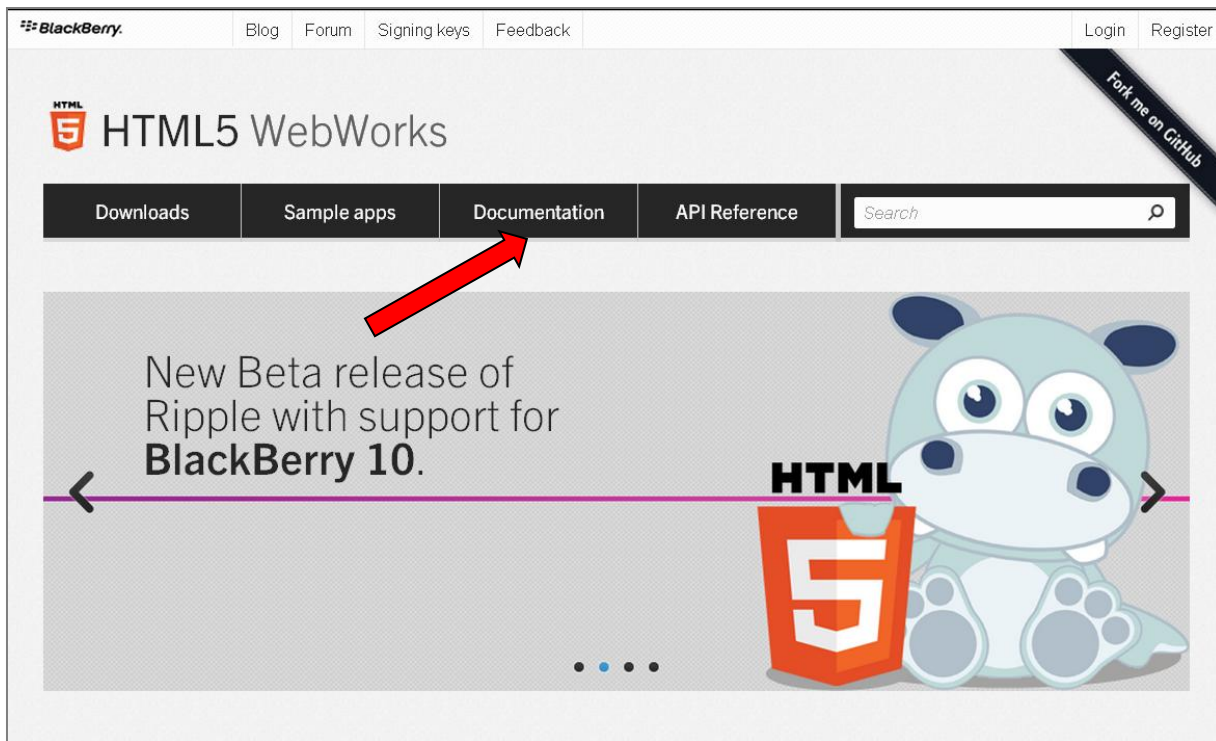


Input types



# BlackBerry 10 config.xml

 **BlackBerry** 10 Jam



<http://developer.blackberry.com/html5>

- Config.xml
  - ▶ Specify application characteristics, behaviors and security
- More info:
  - ▶ “Creating a BlackBerry WebWorks configuration document”
  - ▶ <http://bit.ly/loPqMy>

- Defining application properties
- Steps:
  - ▶ Edit **config.xml**
  - ▶ Modify <name> and <author> elements
  - ▶ Test changes in Ripple (Information tab) or simulator or live device (home screen)
  - ▶ Modify id and version properties of <widget> element
  - ▶ Add <description> element

- Create a home screen icon
- Steps:
  - ▶ Create your own 86 x 86 home screen icon **image.png**
    - <http://www.orison.biz/apps/playbook-icon-maker/>
  - ▶ Save image to root of dev145 webworks project
  - ▶ Modify <icon src="image.png"/> element in **config.xml**
  - ▶ Test changes using Ripple (information panel) or simulator/device (home screen)

- Allowing access to external domains (white listing)
- Steps:
  - ▶ Change `<content src="http://devblog.blackberry.com"/>`
  - ▶ Add `<access uri="http://devblog.blackberry.com"/>`
  - ▶ Test using simulator / device. Verify which content is visible and which has not been loaded.
  - ▶ Add remaining necessary `<access>` elements
  - ▶ Restore `<content src="index.htm"/>` element

## 6. Task – HTML5

Integration with BlackBerry 10 device capabilities

[20 mins]

## Task - HTML5



### Input types

Use input types to target keyboard v...



### Storage

Use localStorage to save data betwe...

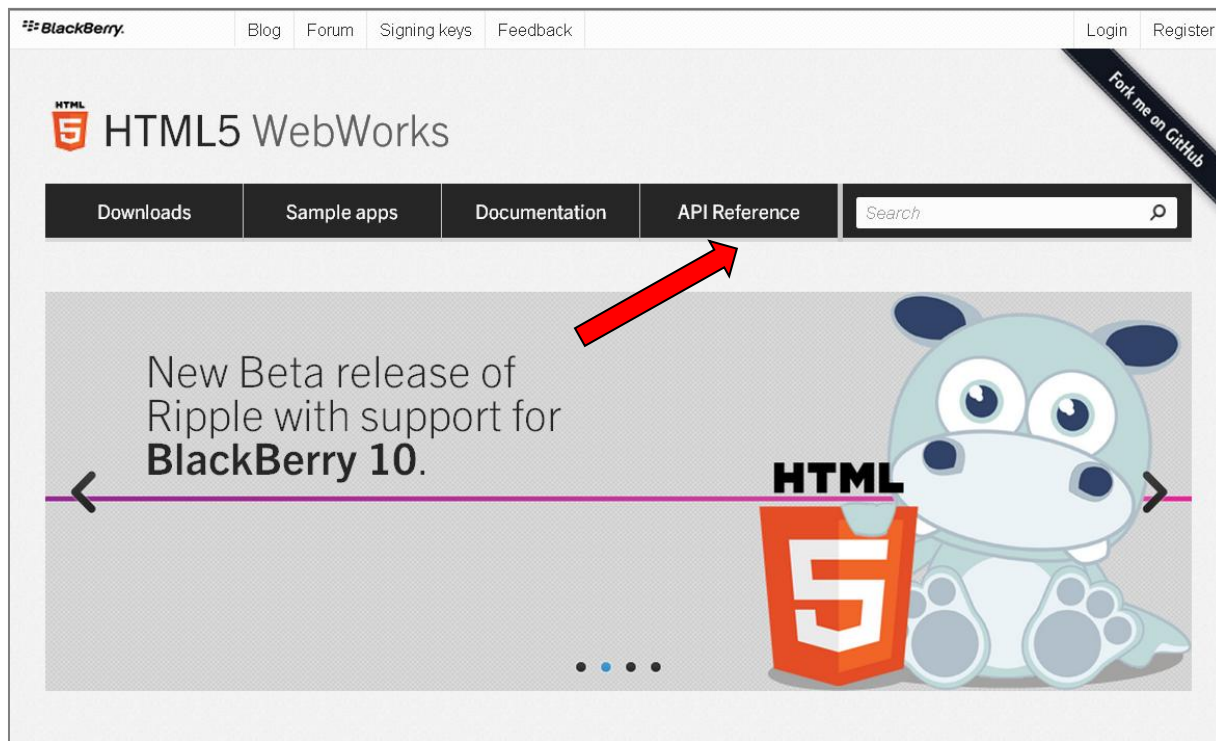


### Geolocation

Use geolocation to retrieve GPS coor...

# BlackBerry 10 HTML5

 **BlackBerry** 10 Jam



<http://developer.blackberry.com/html5/api>



- Use input types to target keyboard variants
- Steps:
  - ▶ Edit **inputtype.html**
  - ▶ Create 5 <input type="text"/> HTML elements
  - ▶ Change value of type property for each to:
    - tel, email, date, time, color
  - ▶ Test using live device or simulator (not Ripple)

- Use localStorage to save data between app restarts
- Steps:
  - ▶ Edit **js/html5-storage.js** and edit saveMessage() method
  - ▶ Save welcome message entered by user to localStorage
  - ▶ View localStorage data using Web Inspector in Ripple
  - ▶ Edit displayMessage() method
  - ▶ Read and display message for data from localStorage
  - ▶ Test using Ripple, change welcome message

- Use geolocation to retrieve GPS coordinates of user
- Steps:
  - ▶ Edit **js/html-geolocation.js**
  - ▶ Call `navigator.geolocation.getCurrentPosition()`
  - ▶ Create `onSuccess` and `onFail` callbacks that display GPS coordinates, or error message, to the page
  - ▶ Add `read_geolocation` permission to **config.xml**
  - ▶ Test changing GPS delay using Ripple geolocation tab

## 7. Task – WebWorks APIs

Integrate with BlackBerry 10 device capabilities

[30 mins]

## Task - WebWorks APIs



**blackberry.app**

Update page title to display applicati...



**blackberry.invoke**

Launch browser to a target Url



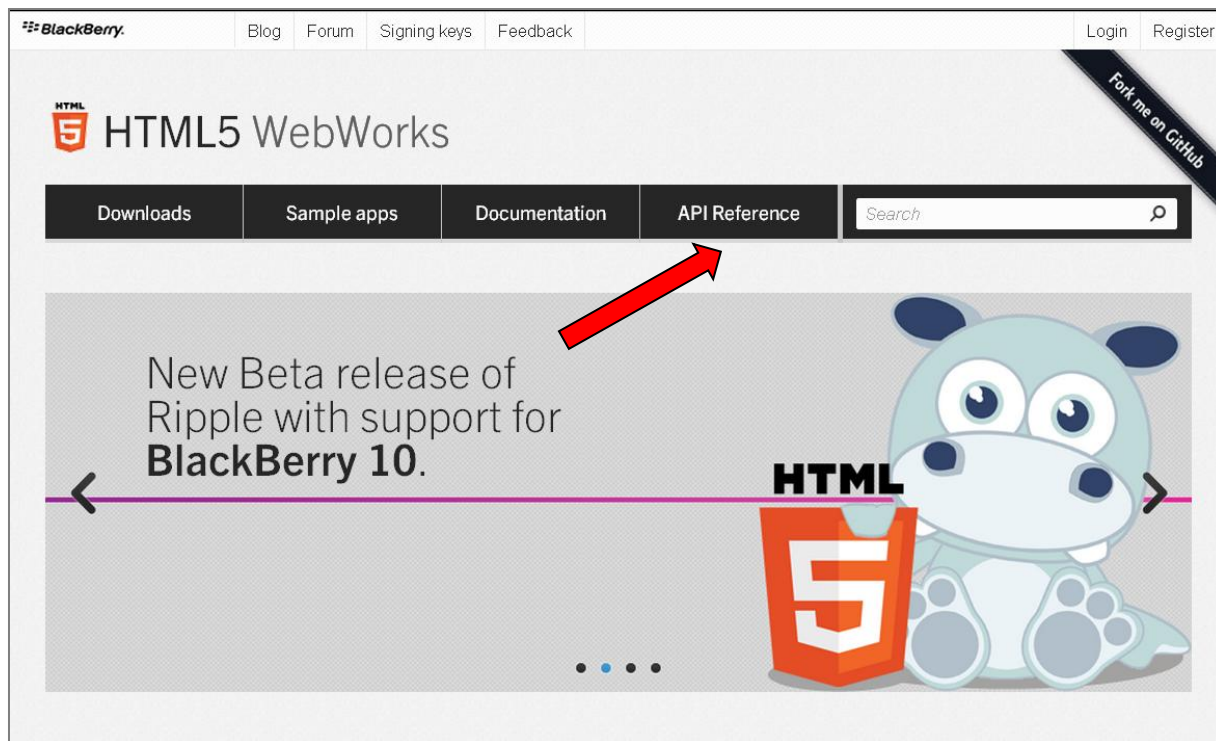
**blackberry.event**

Preserve state when app moved to b...



# BlackBerry 10 WebWorks APIs

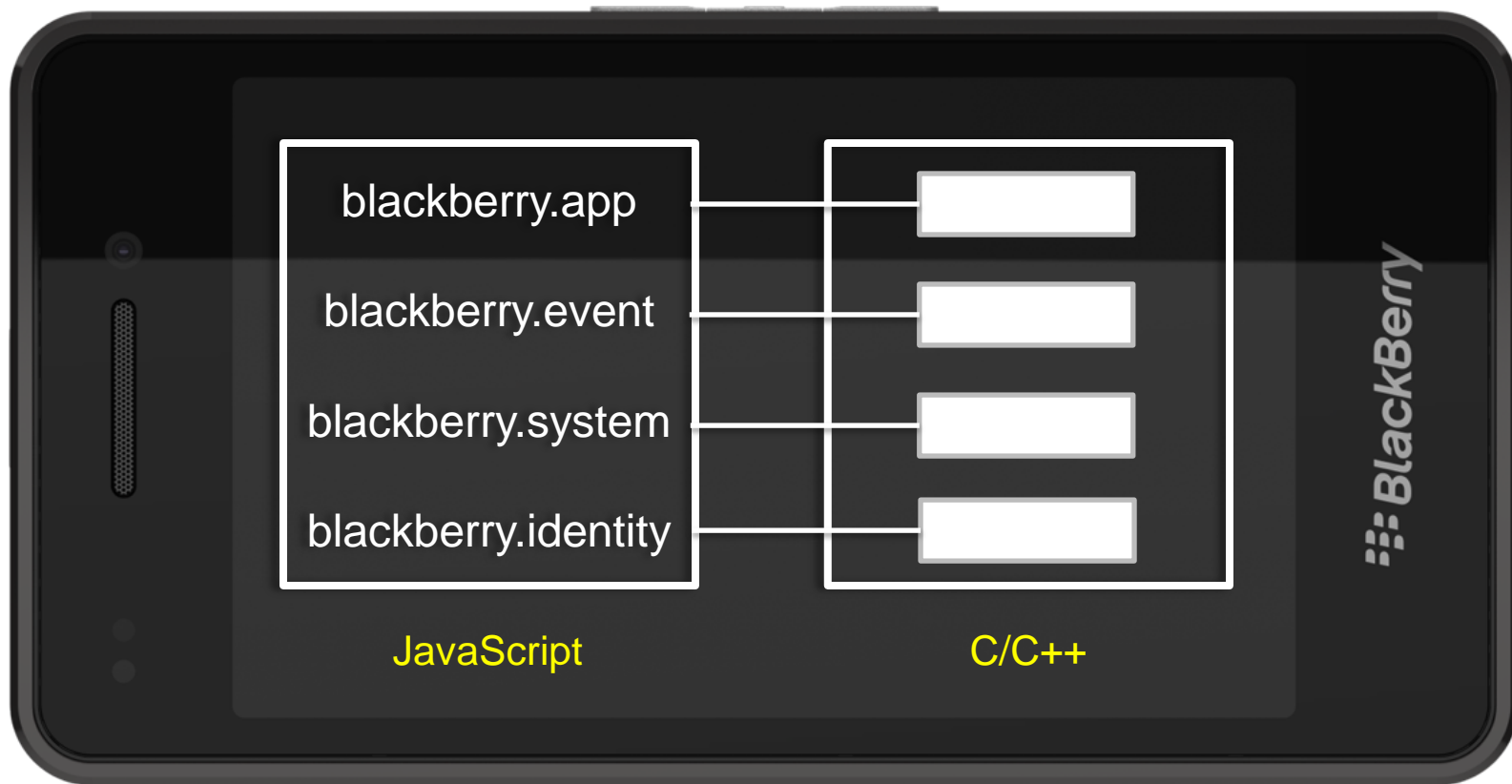
 **BlackBerry** 10 *Jam*



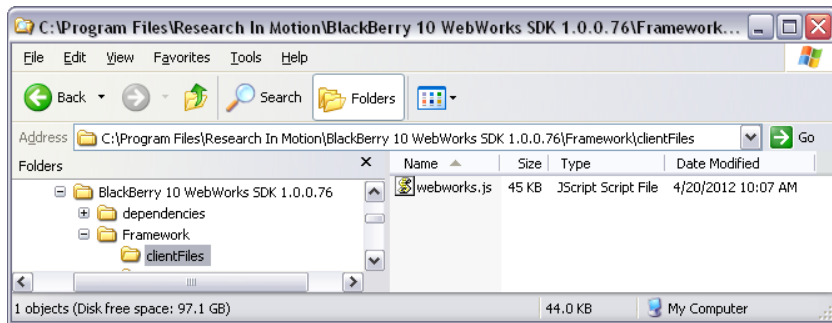
<http://developer.blackberry.com/html5/api>

# BlackBerry 10 WebWorks APIs

 **BlackBerry** 10 Jam



- Add **webworks.js** to your project
  - ▶ Copy from ../BlackBerry 10 WebWorks SDK <version>/framework/clientFiles/



- Add a reference to **webworks.js** in your code
  - ▶ Best practice: Put JS at the end of your HTML page

```
<script src="webworks.js"></script>
```



- Initialize the **webworks.js** framework:
  - ▶ Must create a handler for **webworksready** event
- Only use WebWorks APIs *after* this event has occurred

```
<script>
    function ready() {
        //APIs are now available
    }
    window.addEventListener("load", function(e) {
        document.addEventListener("webworksready", ready);
    });
</script>
```

- Update page title to display application name
- Steps:
  - ▶ Add **webworks.js** to project
  - ▶ Edit **index.htm** and add reference to **webworks.js**
  - ▶ Create event handler for webworksready event
  - ▶ Edit **js/api-app.js**
  - ▶ Read name, author, version properties from **blackberry.app**
  - ▶ Add <feature> to **config.xml** for **blackberry.app**

# Task – WebWorks APIs (Intermediate) **BlackBerry** 10 **Jam**

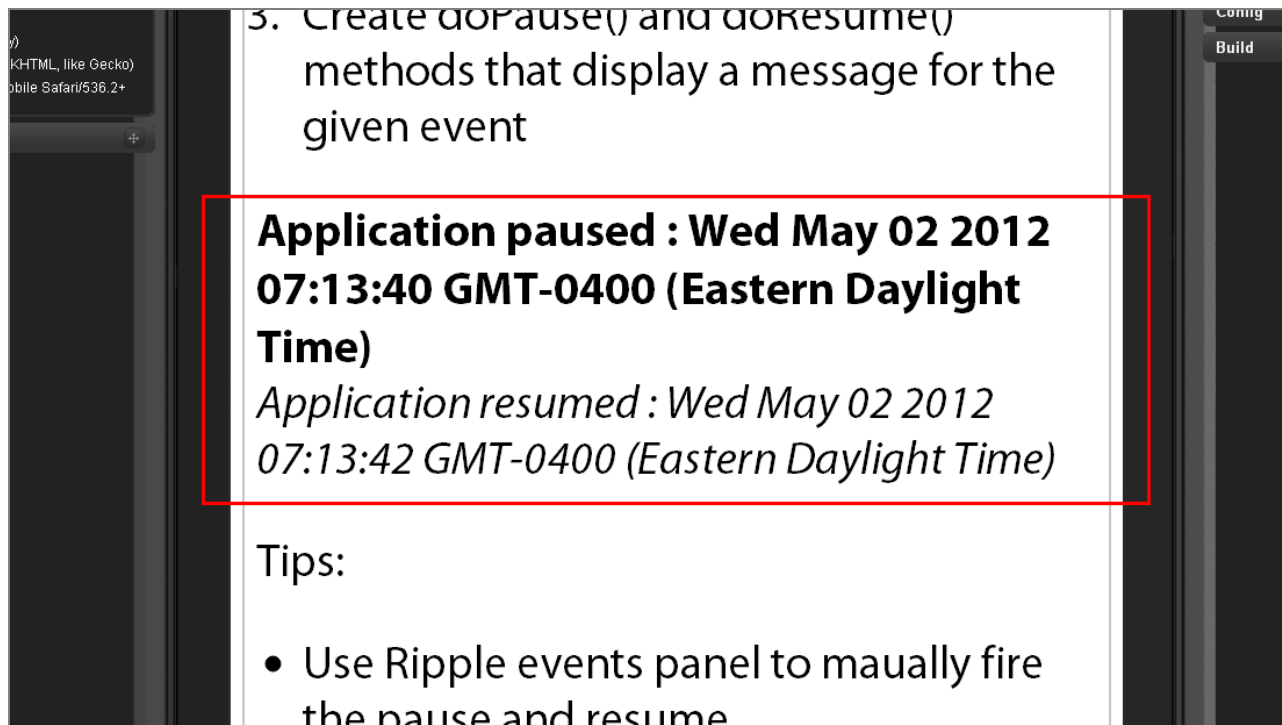
- Launch browser to <http://devblog.blackberry.com>
- Steps:
  - ▶ Edit **config.xml**
  - ▶ Add <feature> elements to **config.xml** for **blackberry.invoke** and **blackberry.invoke.BrowserArguments**
  - ▶ Edit **js/api-invoke.js**
  - ▶ Modify invokeBrowser() JavaScript method
  - ▶ Create and launch BrowserArguments object to target Url

## Task – WebWorks APIs (Advanced) **BlackBerry** 10 **Jam**

- Preserve state when app moved to background
- Steps:
  - ▶ Edit **config.xml**
  - ▶ Add <feature> to **config.xml** for **blackberry.event**
  - ▶ Use **blackberry.event.addEventListener()** to create handlers for pause and resume events
  - ▶ Connect handlers to **pause()** and **resume()** methods

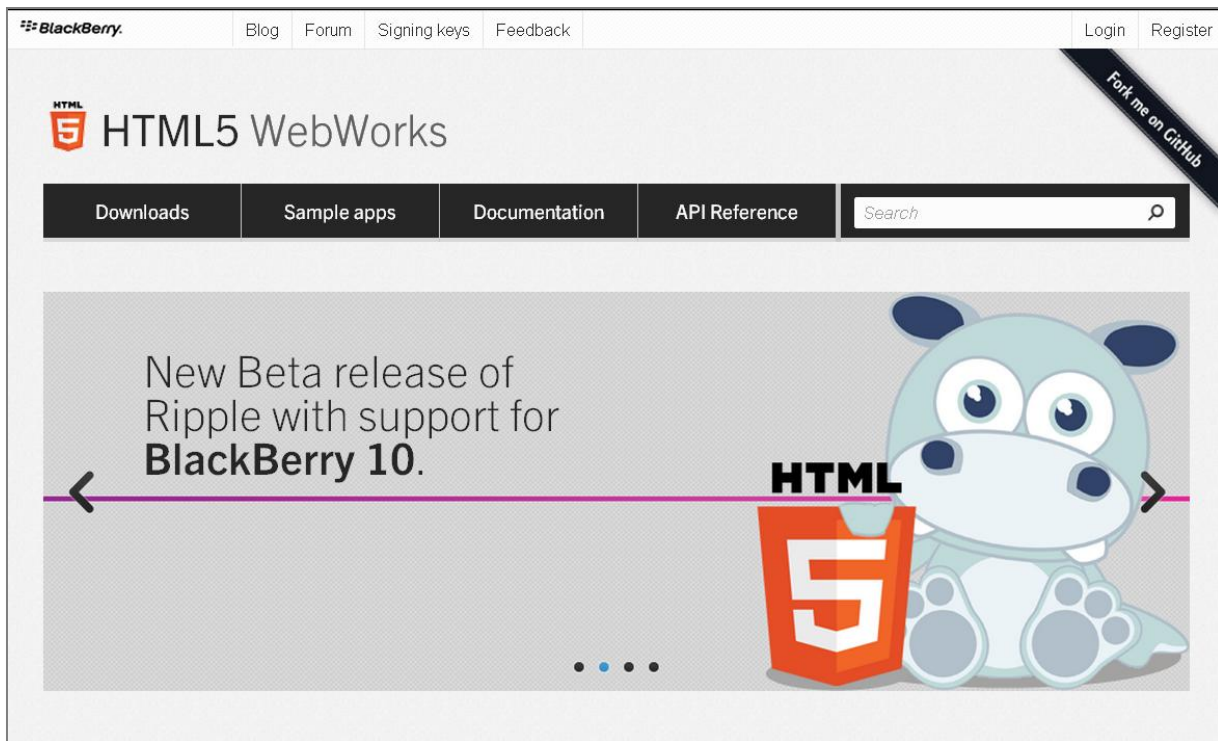
# Task – WebWorks APIs (Advanced) BlackBerry 10 Jam

- Output from pause / resume events:



# For more information

 **BlackBerry** 10 **Jam**



<http://developer.blackberry.com/html5>

# THANK YOU

DEV144

@n\_adam\_stanley, @ken\_wallis, @confusement

May 1-3, 2012