© 2013 Marty Hall & Yaakov Chaikin

The Google Web Toolkit (GWT): JavaScript Native Interface (JSNI)

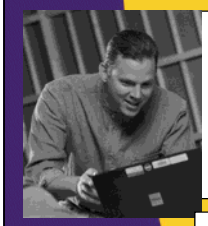
(GWT 2.5 Version)

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/gwt.html

Customized Java EE Training: http://courses.coreservlets.com/
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

© 2013 Marty Hall & Yaakov Chaikin

For live Ajax & GWT training, see training courses at http://courses.coreservlets.com/.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - JSF 2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 7 or 8 programming, custom mix of topics
  - Courses available in any state or country. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate/JPA, GWT, Hadoop, HTML5, RESTful Web Services

Contact hall@coreservlets.com for details

Topics in This Section

- **Calling JavaScript from Java**
  - Format of methods
  - The $wnd and $doc variables
  - Argument types
  - Example: using Scriptaculous effects
- **JavaScript Overlay types**
  - Example: JsCircle class
- **JsArray**
  - Example: JsArray<JsCircle>
- **Calling Java from JavaScript**
  - Format of method types
  - Designating overloaded methods
  - Argument types

© 2013 Marty Hall & Yaakov Chaikin

Overview

Customized Java EE Training: http://courses.coreservlets.com/
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Idea

- **Write Java methods that call JavaScript**
  - Enclose JavaScript code in comments
    - Use $wnd variable to access window object
    - Use $doc variable to access document object
  - Pass primitives, strings, and arrays only
- **Overlay types**
  - You can wrap JSON objects in Java classes
- **JavaScript code can call Java**
  - Use JNI-like format to refer to methods and fields
- **Notes**
  - JSNI can be used only for pure client-side code
    - Method bodies ignored when used in server-side classes
  - JSNI should be used sparingly
    - Most normal JavaScript functionality available in GWT
    - Mostly used to wrap external JavaScript libraries or to process non-RPC return values (e.g., JSON from regular HTTP request)

© 2013 Marty Hall & Yaakov Chaikin

Calling JavaScript from Java

Customized Java EE Training: http://courses.coreservlets.com/
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Basic Syntax

- **Declare method native**
  - Can be private or public, static or instance
- **Use special comments for method body**
  - Start method with /*-{
  - End method with }-*/;
    - private native void foo(...) /*-{ JavaScript-Code }-*/;
- **Argument and return types**
  - String, numeric primitive, boolean
    - Treated normally as argument or return type
  - Array
    - Argument: only used to be passed back to Java
    - Return: only if array came from Java code
  - Object
    - Argument: special syntax for accessing (see later slide)
    - Return: only if Object came from Java code
  - JavaScriptObject
    - Argument: only if JavaScriptObject came from JavaScript code
    - Return: can only be used to pass to another JSNI method

Basic Syntax

Can be public, private, protected, default.
Can be static or instance methods.

Return type can be void, String, or primitive. Array, Object, JavaScriptObject with limits.
Arguments can be same options (except void).

Must be declared native

Methods must start with /*-{

```
private native void alert1(String message) /*-{
  $wnd.alert(message);
}-*/;
```

Methods must end with }-*/;
Normal Java compiler will see
private native void blah(...) /* comment */ ;

Normal JavaScript code here.
Use $wnd to access window object.
Use $doc to access document object.

Note that this is a simplified example. You do not need JSNI to do an alert box. Just call Window.alert in Java. Most standard JavaScript can be done directly in GWT without using JSNI.

Example Overview

- **Goal**
  - Monitor a textfield value and echo result into div
  - If value is “text1” call window.alert
- **Points**
  - Basic JSNI syntax
  - Using the $wnd variable
- **Assumed project setup already done**
  - Clicked “g” to create new Web Application Project
  - Deleted extraneous files as described in last section
  - In auto-generated HTML file, removed everything except script tag and replaced with custom HTML
  - In auto-generated Java class, removed everything except class definition and signature of onModuleLoad

Basic Syntax: Auto-Generated HTML File

```
…
<head><title>JSNI</title>
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/prototype.js"
        type="text/javascript"></script>
<script
  src="./scripts/scriptaculous/scriptaculous.js?load=effects"
  type="text/javascript"></script>
<script src="./scripts/circles.js"
        type="text/javascript"></script>
<script type="text/javascript" language="javascript"
        src="gwtjsni/gwtjsni.nocache.js"></script>
</head>
…
```

Prototype and Script.aculo.us will be used in later examples. JSNI is mainly used to wrap external JavaScript libraries or to process non-RPC server return values. GWT already provides access to most JavaScript functionality, so JSNI is rarely needed for core JavaScript capabilities.

Created automatically by GWT and left unchanged. Except for this line, all of HTML file edited by hand.

Basic Syntax: Auto-Generated HTML File (Continued)

```
…<fieldset>
<legend>Invoking Script.aculo.us Effects</legend>
<h3>Enter "test1", "test2", or "test3".</h3>
<table border="1">
  <tr><th>Input</th>
      <th>Result</th></tr>
  <tr><td id="textfieldID"></td>
      <td id="resultID"></td></tr>
</table><br/>
</fieldset>
<p/>
<legend>Using JavaScript Overlay Types</legend>
<br/>
<span id="circle-button-1"></span>  
<span id="circle-button-2"></span>
<br/><br/>
</fieldset>…
```

The ids match arguments to RootPanel.get in Java code.

Basic Syntax: Auto-Generated Java Class

```
public class GwtJsni implements EntryPoint {
  private TextBox textfield;
  private HTML resultArea;

  public void onModuleLoad() {
    textfield = new TextBox();
    textfield.addKeyUpHandler(new TextfieldHandler());
    resultArea = new HTML("<i>Result will go here</i>");
    RootPanel.get("textfieldID").add(textfield);
    RootPanel.get("resultID").add(resultArea);
    …
  }
```

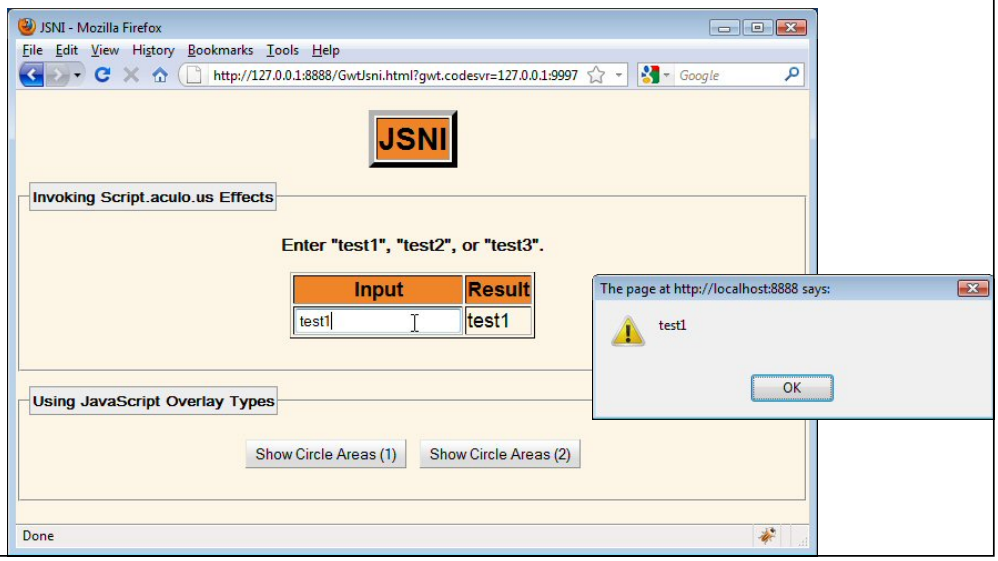
Basic Syntax: Auto-Generated Java Class (Continued)

```
private class TextfieldHandler implements KeyUpHandler {
  public void onKeyUp(KeyUpEvent event) {
    String text = textfield.getText();
    resultArea.setHTML(text);
    if(text.equalsIgnoreCase("test1")) {
      alert1(text);
    } else if(text.equalsIgnoreCase("test2")) {
      highlight1("resultID");
    } else if(text.equalsIgnoreCase("test3")) {
      highlight2("resultID", text);
    }
  }
}

private native void alert1(String message) /*-{
 $wnd.alert(message);
}-*/;
```

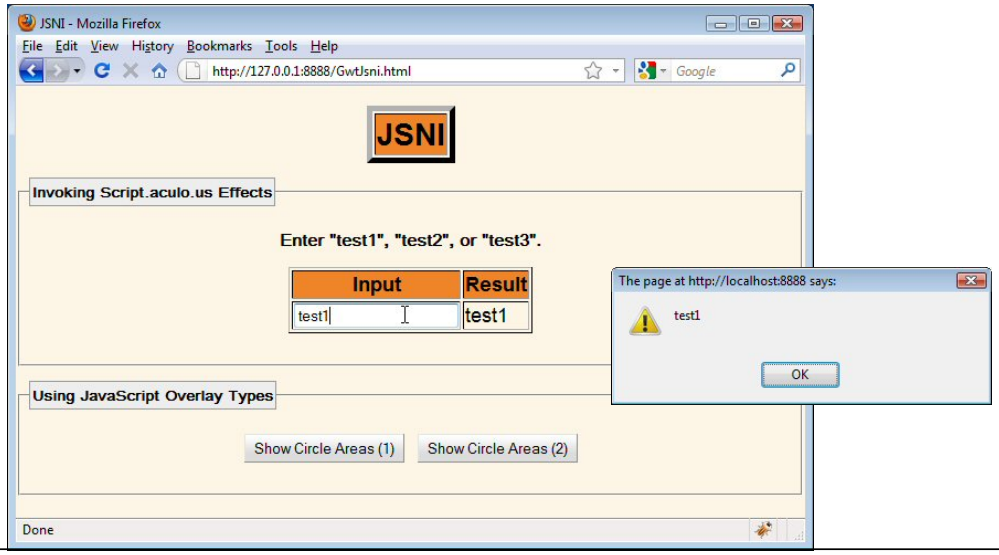

Testing in Development Mode

- R-click project name and Run As → Web Application
  - Or, click project name in Eclipse. Press Run button at top
- Copy/paste URL into browser that has GWT plugin



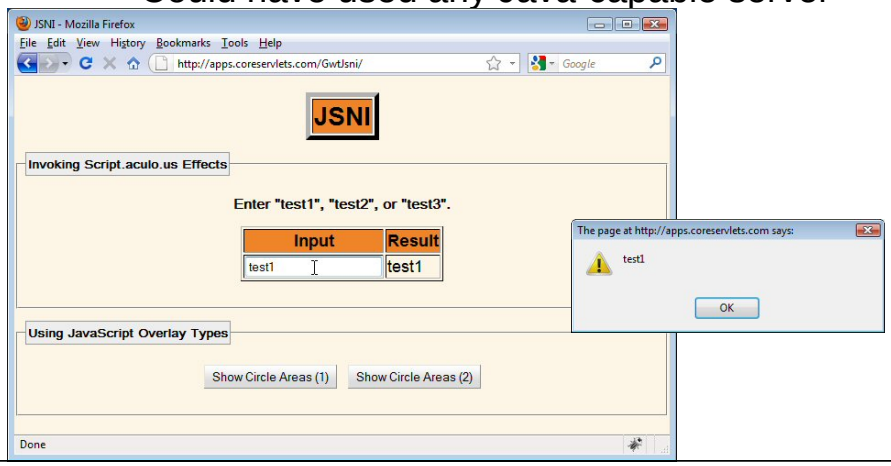
Testing in Production Mode

- R-click project, Google → GWT Compile
  - Or click project and then press red toolbox at top
- Then, change URL by erasing ?gwt.codesvr=…



Testing in Deployed Mode

- **Copied workspace/GwtJsni/war folder**
  - Renamed to GwtJsni to keep similar-looking URL
    - GwtJsni.html is welcome-file, so filename can be omitted
  - Deployed to Tomcat on apps.coreservlets.com
    - Could have used any Java-capable server



© 2013 Marty Hall & Yaakov Chaikin

Using JSNI to Call External JavaScript Libraries

Customized Java EE Training: http://courses.coreservlets.com/
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Using External JavaScript Libraries: Basic Steps

- **Load the library**
  - Put <script> tag in head of HTML page
    - (Or, put <script> tag in *AppName*.gwt.xml file)
- **Define JSNI method**
  - As shown previously
- **Access objects and functions via $wnd**
  - $wnd.someFunction(args)
  - new $wnd.SomeObject(args)
  - var $wnd.newFunction = function(args) { body }
- **Access document via $doc**
  - $doc.title = "New Title";

Example: Using Scriptaculous: HTML

```
…
<head><title>JSNI</title>
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/prototype.js"
        type="text/javascript"></script>
<script
  src="./scripts/scriptaculous/scriptaculous.js?load=effects"
  type="text/javascript"></script>
<script src="./scripts/circles.js"
        type="text/javascript"></script>
<script type="text/javascript" language="javascript"
        src="gwtjsni/gwtjsni.nocache.js"></script>
</head>
…
```

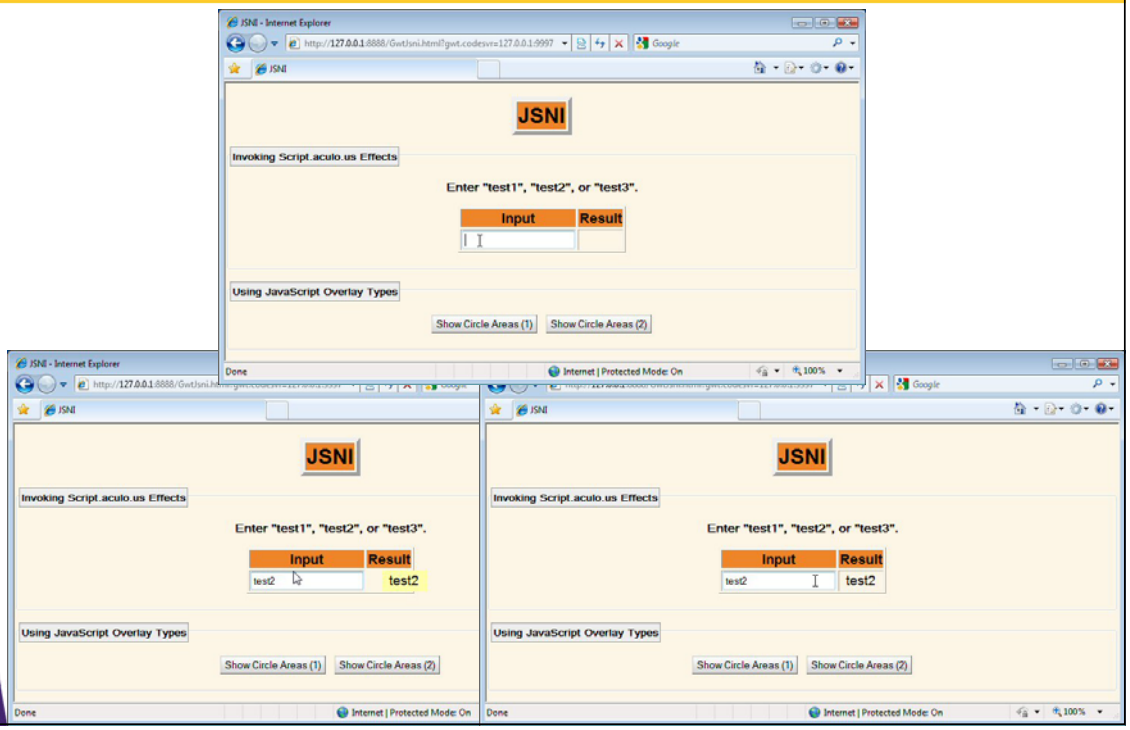
scripts folder installed in "war"
(Same place you put images and css files)

Example: Using Scriptaculous: Java

```
private class TextfieldHandler implements KeyUpHandler {
   public void onKeyUp(KeyUpEvent event) {
     String text = textfield.getText();
     resultArea.setHTML(text);
     if(text.equalsIgnoreCase("test1")) {
       alert1(text);
     } else if(text.equalsIgnoreCase("test2")) {
       highlight1("resultID");
     } else if(text.equalsIgnoreCase("test3")) {
       highlight2("resultID", text);
     }
   }
 }

 private native void highlight1(String id) /*-{
   new $wnd.Effect.Highlight(id);
   new $wnd.Effect.Shake(id);
 }-*/;
```

Example: Using Scriptaculous: Results (Development Mode)



Wrapping Popular JavaScript Frameworks

- **This was just an example**
  - There is already a complete JSNI wrapper for Scriptaculous
- **Wrappers for JavaScript frameworks**
  - Scriptaculous
    - http://gwt.components.googlepages.com/
  - Ext-JS
    - http://www.extjs.com/products/gxt/
    - (This is actually a native GWT re-implementation, not a JSNI wrapper. Still, it makes EXT available in GWT.)
  - Dojo
    - http://code.google.com/p/tatami/
  - SmartClient
    - http://code.google.com/p/smartgwt/

© 2013 Marty Hall & Yaakov Chaikin

JavaScript Overlay Types

Customized Java EE Training: http://courses.coreservlets.com/
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Big Idea

- **Make Java class to wrap a JSON "class"**
  - Extend JavaScriptObject
    - All methods must be final
  - Use native methods for getters and setters that refer to JSON properties.
    - The "this" variable is automatically defined
  - Use regular (but final) methods for derived properties
    - E.g., getArea of Circle might be derived from getRadius
  - You are permitted to implement interfaces
    - New in GWT 2.x
- **Get instance**
  - Must call native method to obtain JSON object
  - Cannot call "new"
- **Use like a normal Java object**

Overlays Example Code: Underlying JSON

```
var circles =
  [ { radius:   1.0 },
    { radius:  10.0 },
    { radius: 100.0 } ];
```


Overlays Example Code: Interface

```
package coreservlets.client;

public interface Circle {
  public double getRadius();
  public void setRadius(double radius);
  public double getArea();
  public String getInfo();
}
```

By having the overlay class (next slide) implement the interface, you make it possible to make methods that support both overlay classes and either regular classes or collections that contain a mixture of both.

Overlays Example Code: Overlay Class

```
…

public class JsCircle extends JavaScriptObject
                      implements Circle {
  // Overlay types must have protected, zero-arg constructors
  protected Circle() {}

  public final native double getRadius() /*-{
    return this.radius;
  }-*/;

  public final native void setRadius(double r) /*-{
    this.radius = r;
  }-*/;

  public final double getArea() {
    return(Math.PI * Math.pow(getRadius(), 2));
  }
```

Real JSON property.

Derived, Java-only property. Must be final

Overlays Example Code: Overlay Class (Continued)

```
  // GWT 2.x supports many Java 6 constructs,
  // but not String.format. So, using explicit
  // String concatenation below.

  public final String getInfo() {
    String info =
      "Radius=" + getRadius() + ", " +
      "Area=" + getArea();
    return(info);
  }
}
```

Overlays Example Code: Getting Instance of Circle

- **Java native method (in GwtJsni.java)**
  - You cannot call "new" on overlay class. You must get instance from JavaScript

```
private native Circle getCircle(int i) /*-{
  return $wnd.circles[i];
}-*/;
```

getCircle really returns JsCircle, but I declare it to return the interface type

- **JavaScript (shown earlier – in circles.js)**

```
var circles =
  [ { radius:   1.0 },
    { radius:  10.0 },
    { radius: 100.0 } ];
```


Overlays Example Code: Button and Listener (in GwtJsni.java)

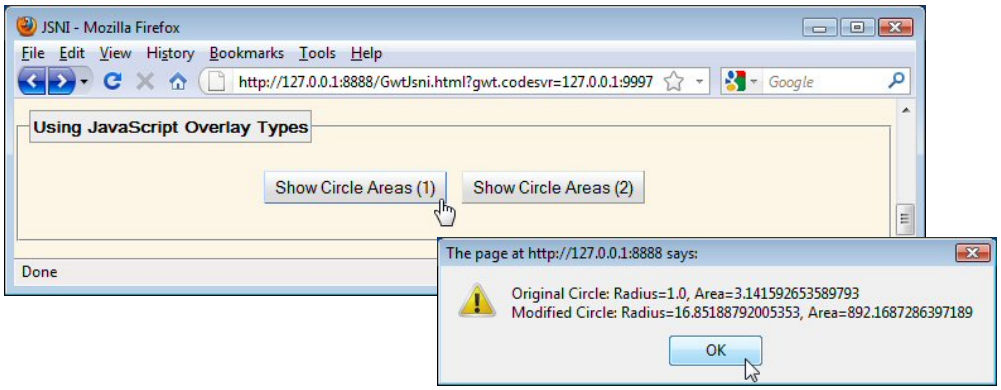
```
public void onModuleLoad() {
   …
   Button circleButton =
     new Button("Show Circle Areas (1)");
   circleButton.addClickHandler(new CircleAreaHandler1());
   RootPanel.get("circle-button-1").add(circleButton);
 }

 private class CircleAreaHandler1 implements ClickHandler {
   public void onClick(ClickEvent event) {
     int numCircles = 3; // Hard-coded length
     for(int i=0; i<numCircles; i++) {
       Circle circle = getCircle(i);
       showCircleAlert(circle);
     }
   }
 }
```

Overlays Example Code: Helper Method (in GwtJsni.java)

```
private void showCircleAlert(Circle circle) {
  String info = "Original Circle: " +
  circle.getInfo() + "\n";
  circle.setRadius(Math.random() * 100);
  info += "Modified Circle: " + circle.getInfo();
  Window.alert(info);
}
```


Overlays Example: Results (Development Mode)



Since the event handler changes the radius of the Circle objects, the above output assumes that this was the first time either button was pressed. On later presses, the "Original Circle" will reflect the radius from the previous invocation.

© 2013 Marty Hall & Yaakov Chaikin

JsArray

Customized Java EE Training: http://courses.coreservlets.com/
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Idea

- **Problem with last example**
  - Number of circles in array was hardcoded into event handler. If JavaScript changed, Java code could easily be out of sync.
  - But, you cannot treat JavaScript array as Java array
- **Solution**
  - You can treat JavaScript array as JsArray<OverlayType>
  - JsArray has length method (not field)
- **Minor deficiencies**
  - JsArray does not implement Iterable
    - So you cannot use for(BlahType b: jsArrayOfBlahs) {…}
  - You must say JsArray<RealOverlayType>
    - Not JsArray<InterfaceType>

Method to Return JsArray

```
private native Circle getCircle(int i) /*-{
  return $wnd.circles[i];
}-*/;
```

This was method used in previous example: return one Circle at a time.

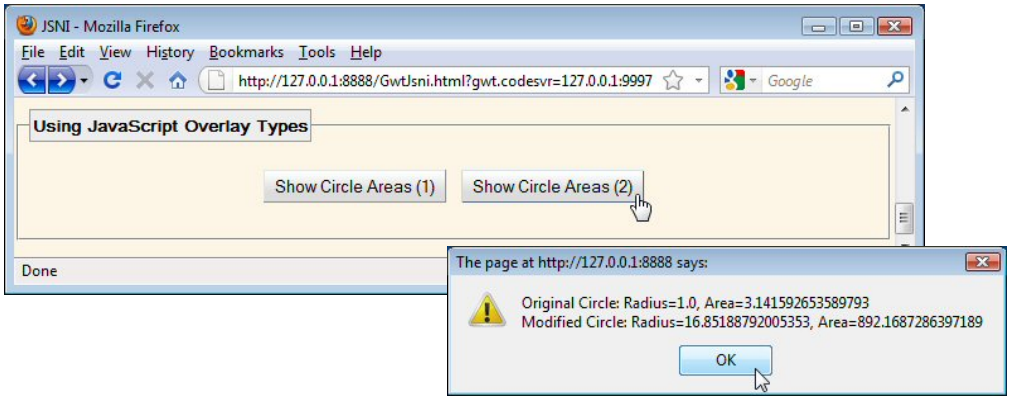
```
private native JsArray<JsCircle> getCircles() /*-{
  return $wnd.circles;
}-*/;
```

This is new and improved version: return entire array.

Event Handler

```
private class CircleAreaHandler2
        implements ClickHandler {
   public void onClick(ClickEvent event) {
     JsArray<JsCircle> circles = getCircles();
     for(int i=0; i<circles.length(); i++) {
       Circle circle = circles.get(i);
       showCircleAlert(circle);
     }
   }
 }
```

Results (Development Mode)



Again, since the event handler changes the radius of the Circle objects, the above output assumes that this was the first time either button was pressed. On later presses, the "Original Circle" will reflect the radius from the previous invocation.

© 2013 Marty Hall & Yaakov Chaikin

Calling Java from JavaScript Native Methods

Customized Java EE Training: http://courses.coreservlets.com/
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Calling Java from JavaScript: Overview

- **Follows Java Native Interface (JNI) format**
  - Summary here. Details: see JNI reference or GWT docs.
- **Format for static methods**
  - @className::methodName(paramSignature)(args)
    - className: fully qualified name
    - methodName: normal name
    - paramSignature: JNI parameter signature (see next page)
    - args: argument names
- **Format for instance methods**
  - this.@className::methodName(paramSignature)(args)
  - obj.@className::methodName(paramSignature)(args)
    - obj must be passed in from Java
- **Format for field access**
  - obj.@className::fieldName

Parameter Signatures

- **Separated by semicolons, not commas**
- **Special format for types (copied from JNI)**

Real Type	Param Sig
boolean	Z
int	I
double	D
String	Ljava/lang/String
BlahObject	Lpackage1/package2/BlahObject
blah[]	[sig

- **Example**
  - Real method (class is test.client.SomeClass)
    - public double foo(double d, String s) { ...}
  - JSNI call
    - var value =
      this.@test.client.SomeClass::foo(D; Ljava/lang/String)(2.5, "hi");

Calling Java from JavaScript: Example

```
private class TextfieldHandler implements KeyUpHandler {
   public void onKeyUp(KeyUpEvent event) {
     String text = textfield.getText();
     resultArea.setHTML(text);
     if(text.equalsIgnoreCase("test1")) {
       alert1(text);
     } else if(text.equalsIgnoreCase("test2")) {
       highlight1("resultID");
     } else if(text.equalsIgnoreCase("test3")) {
       highlight2("resultID", text);
     }
   }
 }
```


Calling Java from JavaScript: Example (Continued)

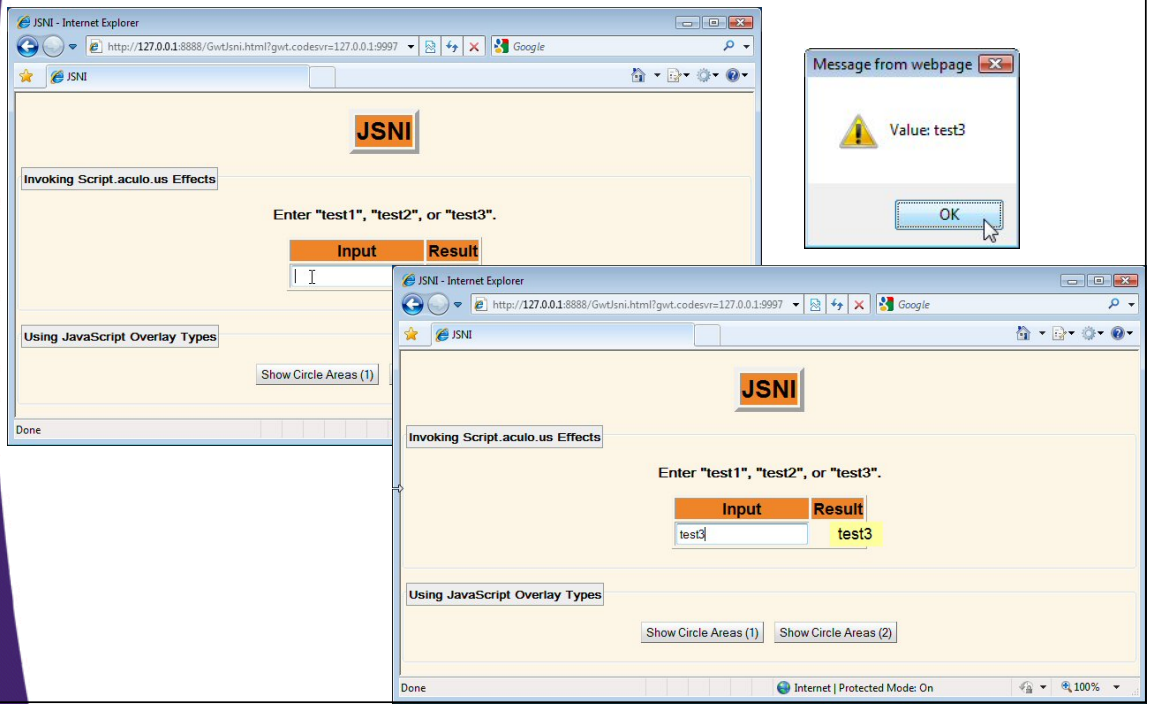
```
private double randomTime(double n) {
  return(Math.random() * n);
}

private void alert2(String text) {
  Window.alert("Value: " + text);
}

private native void highlight2(String id, String text) /*-{
  var time =
    this.@coreservlets.client.GwtJsniApp::randomTime(D)(10);
  this.@coreservlets.client.GwtJsniApp::alert2
    (Ljava/lang/String;)(text);
  new $wnd.Effect.Highlight(id, { duration: time});
  new $wnd.Effect.Shake(id, { duration: time});
}-*/;
```

You call Java from JavaScript native methods, not from regular JavaScript functions.

Calling Java from JavaScript: Example: Results (Devel. Mode)



© 2013 Marty Hall & Yaakov Chaikin

Wrap-Up

Customized Java EE Training: http://courses.coreservlets.com/
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Summary

- **Approach**
  - Declare final native
  - Enclose body in /*-{ … }-*/;
  - Use $wnd to access window object
  - Use JNI format to call Java from JavaScript
- **Example**

```
private native void alert1(String message) /*-{
  $wnd.alert(message);
}-*/;
```

- **Purposes**
  - Mostly for wrapping existing JavaScript libraries or handling non-RPC JSON server response data
    - Most standard JS tasks can be done directly in GWT

© 2013 Marty Hall & Yaakov Chaikin

Questions?

JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training.

Customized Java EE Training: http://courses.coreservlets.com/
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.