
Google™
11
I/O

Using GWT and Eclipse to Build Great Mobile Web Apps

Chris Ramsdale
Product Manager, GWT and Google Plugin for Eclipse

Feedback: http://goo.gl/mn6Y4
Twitter: #io2011 #gwt

Why Are We Here?

The Problems

- Mobile devices
  - ...have smaller screens
  - ...can have different orientations
  - ...have slower processors
  - ...can be disconnected

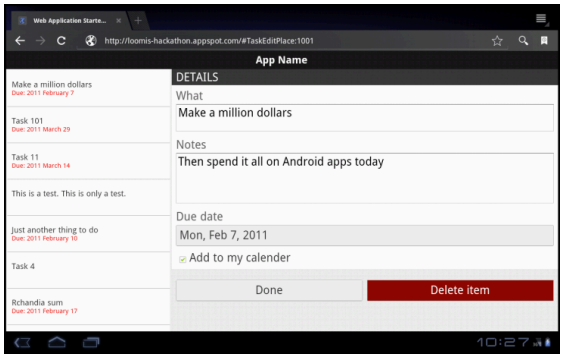
The Solution

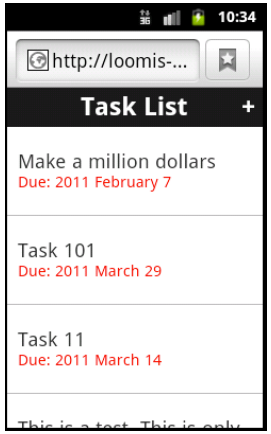
- Use GWT and Google's Plugin for Eclipse (GPE) to build great mobile web apps for iPhone and Android

- And helps you
  - ...build great mobile UIs for any form factor
  - ...optimize the user experience
  - ...make it snappy
  - ...work without a connection

Example App: Cloud Tasks

- Web based task mgmt app
- UIs for Android, iPhone, iPad, and desktop browsers
- Orientation change handling
- Offline support
- Source code available at the end





Google I/O

- Build great UIs for any form factor
- Optimize the user experience
- Make it snappy
- Work without a connection

Build Great Mobile UIs For Any Form Factor

- UI code like you expect it
  - Declarative UI layout with UiBinder
- Speed up UI development with WYSIWYG tools
  - Using GWT Designer to build great UIs

Build Great Mobile UIs For Any Form Factor

UI code like you expect it

```
<ui:UiBinder>
    <ui:style>
        .addButton {
            color: white;
            font-size: 18pt;
            background: none;
            border: none;
            text-align: right;
            font-weight: bold;
        }
    </ui:style>

    ...

</ui:UiBinder>
```

Build Great Mobile UIs For Any Form Factor

UI code like you expect it

```
<ui:UiBinder>
    <ui:style>
        .addButton {
            color: white;
            font-size: 18pt;
            background: none;
            border: none;
            text-align: right;
            font-weight: bold;
        }
    </ui:style>

    ...

</ui:UiBinder>
```

Build Great Mobile UIs For Any Form Factor

UI code like you expect it

```
<ui:UiBinder>
    <g:DockLayoutPanel>

        <!-- Header -->
        <g:north size="32">
          <g:HTMLPanel></g:HTMLPanel>
        </g:north>

        <g:center>
            <g:DockLayoutPanel>
                <!-- Task List. -->
                <g:west size="30">
                    <g:SimpleLayoutPanel addStyleNames="{style.taskList}"/>
                </g:west>

                <!-- Content. -->
                <g:center>
                    <g:DeckLayoutPanel></g:DeckLayoutPanel>
                </g:center>
            </g:DockLayoutPanel>
        </g:center>
    </g:DockLayoutPanel>
</ui:UiBinder>
```

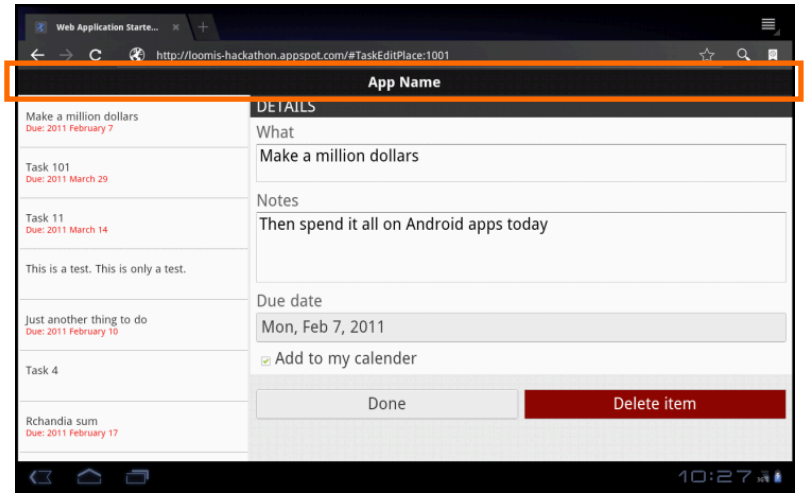
Build Great Mobile UIs For Any Form Factor

UI code like you expect it

```
<ui:UiBinder>
    <g:DockLayoutPanel>

        <!-- Header -->
        <g:north size="32">
          <g:HTMLPanel></g:HTMLPanel>
        </g:north>

        <g:center>
            <g:DockLayoutPanel>
                <!-- Task List. -->
                <g:west size="30">
                    <g:SimpleLayoutPanel addStyleNames="{style.taskList}"/>
                </g:west>

                <!-- Content. -->
                <g:center>
                    <g:DeckLayoutPanel></g:DeckLayoutPanel>
                </g:center>
            </g:DockLayoutPanel>
        </g:center>
    </g:DockLayoutPanel>
</ui:UiBinder>
```

Build Great Mobile UIs For Any Form Factor

UI code like you expect it



Google I/O

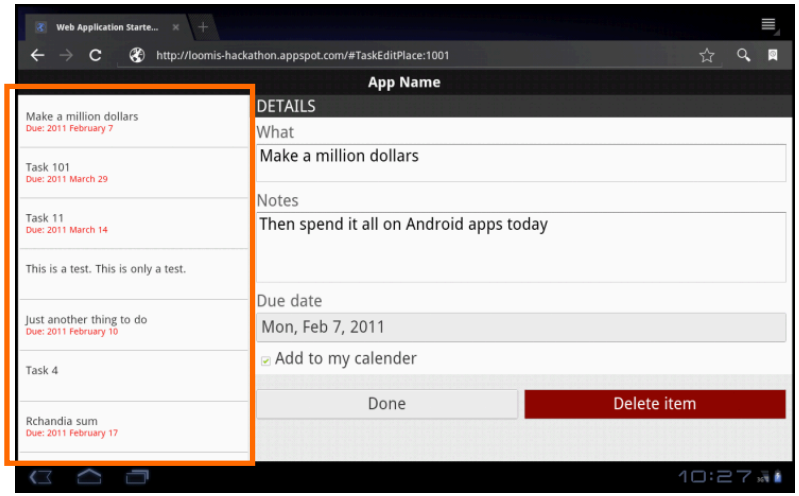
Build Great Mobile UIs For Any Form Factor

UI code like you expect it

```
<ui:UiBinder>
    <g:DockLayoutPanel>

        <!-- Header -->
        <g:north size="32">
          <g:HTMLPanel></g:HTMLPanel>
        </g:north>

        <g:center>
            <g:DockLayoutPanel>
                <!-- Task List. -->
                <g:west size="30">
                    <g:SimpleLayoutPanel addStyleNames="{style.taskList}"/>
                </g:west>

                <!-- Content. -->
                <g:center>
                    <g:DeckLayoutPanel></g:DeckLayoutPanel>
                </g:center>
            </g:DockLayoutPanel>
        </g:center>
    </g:DockLayoutPanel>
</ui:UiBinder>
```

Build Great Mobile UIs For Any Form Factor

UI code like you expect it



Google I/O 11

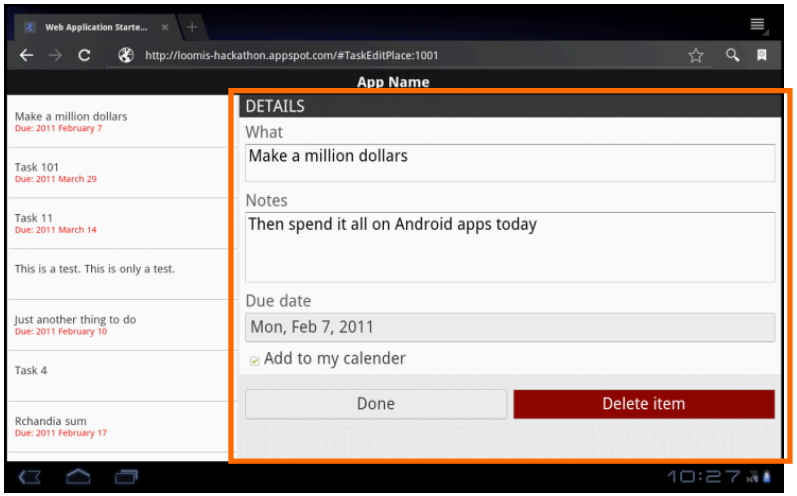
Build Great Mobile UIs For Any Form Factor

UI code like you expect it

```
<ui:UiBinder>
    <g:DockLayoutPanel>

        <!-- Header -->
        <g:north size="32">
          <g:HTMLPanel></g:HTMLPanel>
        </g:north>

        <g:center>
            <g:DockLayoutPanel>
                <!-- Task List. -->
                <g:west size="30">
                    <g:SimpleLayoutPanel addStyleNames="{style.taskList}"/>
                </g:west>

                <!-- Content. -->
                <g:center>
                    <g:DeckLayoutPanel></g:DeckLayoutPanel>
                </g:center>
            </g:DockLayoutPanel>
        </g:center>
    </g:DockLayoutPanel>
</ui:UiBinder>
```

Build Great Mobile UIs For Any Form Factor

UI code like you expect it



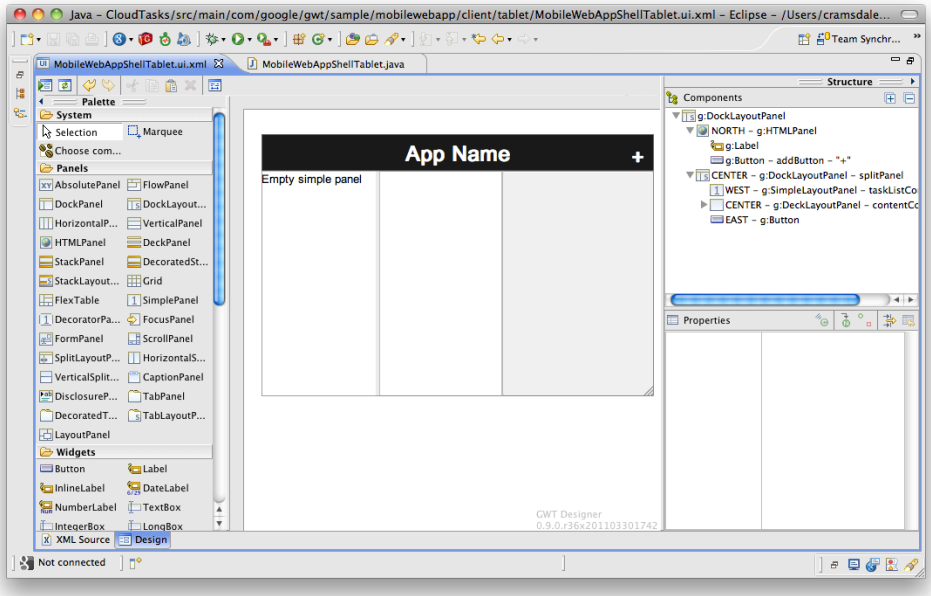
Google I/O

Build Great Mobile UIs For Any Form Factor

Speed up UI development with WYSIWYG tools

Build Great Mobile UIs For Any Form Factor

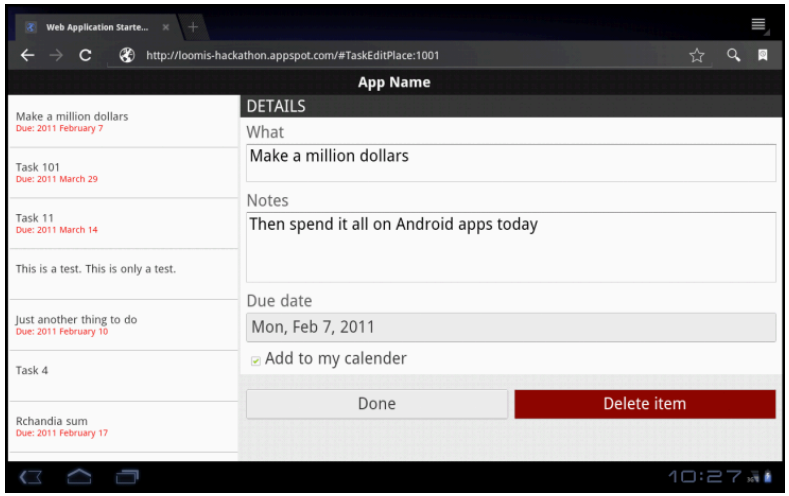
Speed up UI development with WYSIWYG tools



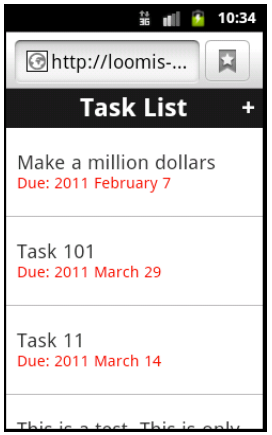
Google I/O

Build Great Mobile UIs For Any Form Factor

Device-specific, declarative UIs



XML ListView**Tablet**.ui.xml



XML ListView**Phone**.ui.xml

Google I/O

- Build great UIs for any form factor
- Optimize the user experience
- Make it snappy
- Work without a connection

Optimize the User Experience

- Desktops, tablets, and phones...oh my!
  - Deferred binding to sort them out
- Landscape vs. portrait
  - Capture and respond to orientation changes
- Icing on the cake
  - Maximize code reuse by using the Model-View-Presenter (MVP) design pattern

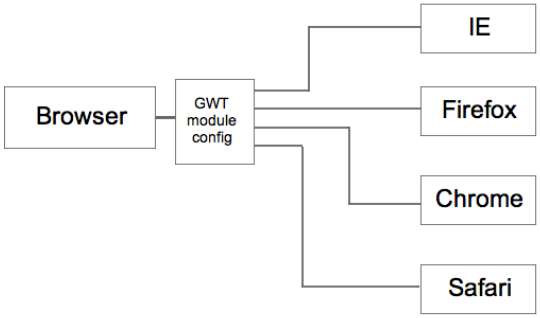
Optimize the User Experience

Select the right UI using deferred binding

Deferred what?

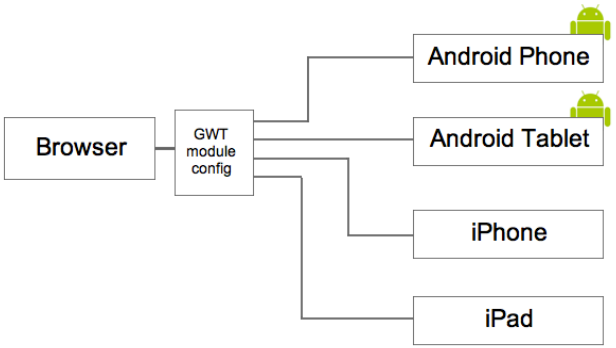
Optimize the User Experience

Deferred binding - optimize for the browser



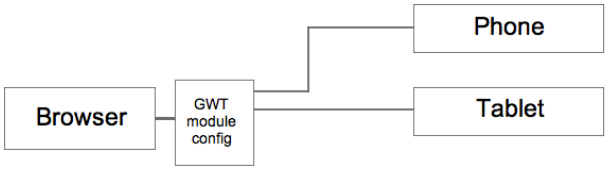
Optimize the User Experience

Select the right UI using deferred binding



Optimize the User Experience

Select the right UI using deferred binding



Optimize the User Experience

Select the right UI using deferred binding

CloudTasks.gwt.xml (GWT app config)

```
<!-- Use ViewFactoryImplMobile for mobile form factor. -->
<replace-with class="com.google.mobilecalendar.client.ViewFactoryImplMobile">
  <when-type-is class="com.google.mobilecalendar.client.ViewFactory"/>
  <when-property-is name="formfactor" value="mobile"/>
</replace-with>
```

Optimize the User Experience

Select the right UI using deferred binding

CloudTasks.gwt.xml (GWT app config)

```
<!-- Use ViewFactoryImplMobile for mobile form factor. -->
<replace-with class="com.google.mobilecalendar.client.ViewFactoryImplMobile">
  <when-type-is class="com.google.mobilecalendar.client.ViewFactory"/>
  <when-property-is name="formfactor" value="mobile"/>
</replace-with>
```

Optimize the User Experience

Select the right UI using deferred binding

CloudTasks.gwt.xml (GWT app config)

```
<!-- Use ViewFactoryImplMobile for tablet form factor. -->
<replace-with class="com.google.mobilecalendar.client.ViewFactoryImplTablet">
  <when-type-is class="com.google.mobilecalendar.client.ViewFactory"/>
  <when-property-is name="formfactor" value="tablet"/>
</replace-with>
```

Optimize the User Experience

Select the right UI using deferred binding

CloudTasks.gwt.xml (GWT app config)

```
<!-- Use ViewFactoryImplMobile for tablet form factor. -->
<replace-with class="com.google.mobilecalendar.client.ViewFactoryImplTablet">
  <when-type-is class="com.google.mobilecalendar.client.ViewFactory"/>
  <when-property-is name="formfactor" value="tablet"/>
</replace-with>
```

Optimize the User Experience

Select the right UI using deferred binding

FormFactor.gwt.xml

```
<property-provider name="formfactor">
  <![CDATA[
    var ua = navigator.userAgent.toLowerCase();
    if (ua.indexOf("ipad") != -1) {
      return "tablet";
    } else if (ua.indexOf("iphone") != -1 || ) {
      return "mobile";
    } else if (ua.indexOf("android") != -1) {
      // Overly complex hueristic for determining Android form factor
      ...
    }
  ]]>
</property-provider>
```

Optimize the User Experience

Select the right UI using deferred binding

FormFactor.gwt.xml

```
<property-provider name="formfactor">
  <![CDATA[
    var ua = navigator.userAgent.toLowerCase();
    if (ua.indexOf("ipad") != -1) {
      return "tablet";
    } else if (ua.indexOf("iphone") != -1 || ) {
      return "mobile";
    } else if (ua.indexOf("android") != -1) {
      // Overly complex hueristic for determining Android form factor
      ...
    }
  ]]>
</property-provider>
```

Optimize the User Experience

Select the right UI using deferred binding

FormFactor.gwt.xml

```
<property-provider name="formfactor">
  <![CDATA[
    var ua = navigator.userAgent.toLowerCase();
    if (ua.indexOf("ipad") != -1) {
      return "tablet";
    } else if (ua.indexOf("iphone") != -1 || ) {
      return "mobile";
    } else if (ua.indexOf("android") != -1) {
      // Overly complex hueristic for determining Android form factor
      ...
    }
  ]]>
</property-provider>
```

Optimize the User Experience

Select the right UI using deferred binding

FormFactor.gwt.xml

```
<property-provider name="formfactor">
  <![CDATA[
    var ua = navigator.userAgent.toLowerCase();
    if (ua.indexOf("ipad") != -1) {
      return "tablet";
    } else if (ua.indexOf("iphone") != -1 || ) {
      return "mobile";
    } else if (ua.indexOf("android") != -1) {
      // Overly complex heuristic for determining Android form factor
      ...
    }
  ]]>
</property-provider>
```

Optimize the User Experience

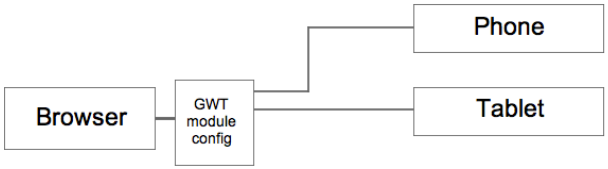
Select the right UI using deferred binding

CloudTasks.gwt.xml (GWT app config)

```
<!-- Use ViewFactoryImplMobile for tablet form factor. -->
<replace-with class="com.google.mobilecalendar.client.ViewFactoryImplTablet">
  <when-type-is class="com.google.mobilecalendar.client.ViewFactory"/>
  <when-property-is name="formfactor" value="tablet"/>
</replace-with>
```

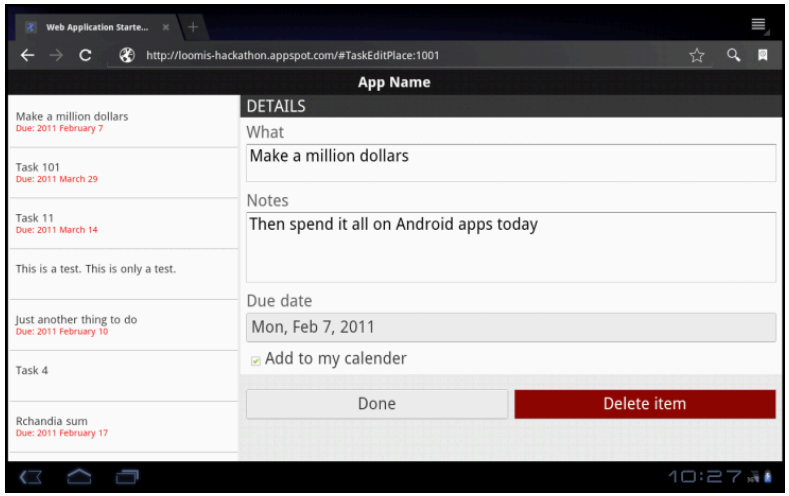
Optimize the User Experience

Select the right UI using deferred binding

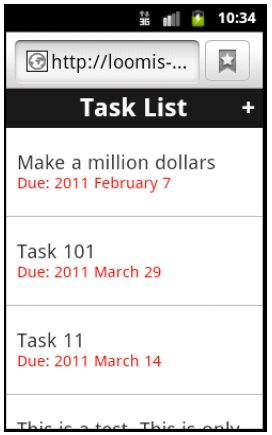


Build Great Mobile UIs For Any Form Factor

Device-specific, declarative UIs



XML ListView**Tablet**.ui.xml

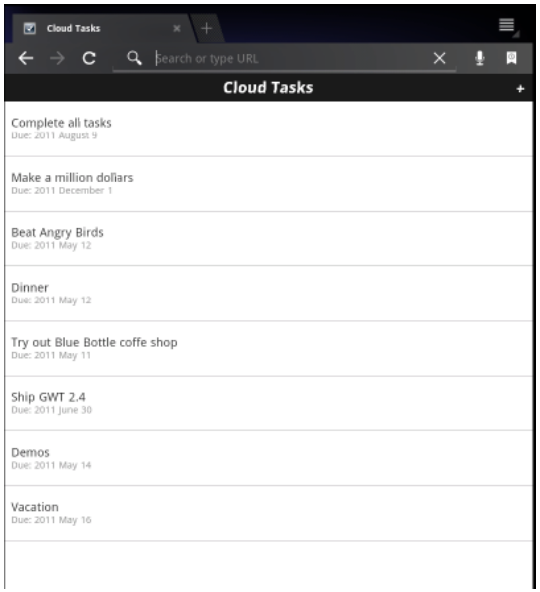


XML ListView**Phone**.ui.xml

Google I/O

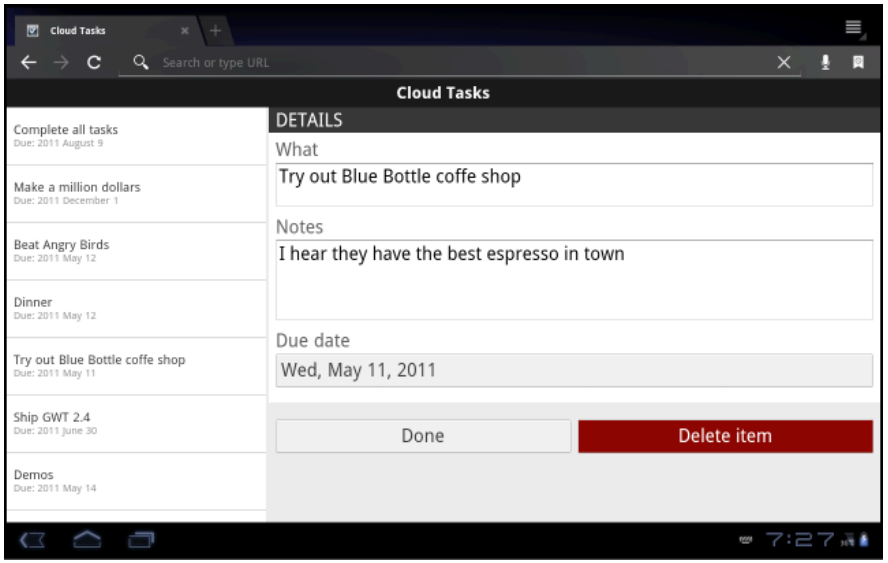
Optimize the User Experience

Capture and respond to orientation changes



Optimize the User Experience

Capture and respond to orientation changes



Google I/O 11

Optimize the User Experience

Capture and respond to orientation changes

Optimize the User Experience

Capture and respond to orientation changes

```
/*
 * Listen for changes in the window size so we can adjust the
 * orientation of the app. This will also catch orientation changes
 * on mobile devices.
 */
windowResizeHandler = Window.addResizeHandler(new ResizeHandler() {
  public void onResize(ResizeEvent event) {
    if (isOrientationPortrait != calculateOrientationPortrait()) {
      isOrientationPortrait = !isOrientationPortrait;
      adjustOrientation(isOrientationPortrait);
    }
  }
});
```

Optimize the User Experience

Capture and respond to orientation changes

```
/*
 * Listen for changes in the window size so we can adjust the
 * orientation of the app. This will also catch orientation changes
 * on mobile devices.
 */
windowResizeHandler = Window.addResizeHandler(new ResizeHandler() {
  public void onResize(ResizeEvent event) {
    if (isOrientationPortrait != calculateOrientationPortrait()) {
      isOrientationPortrait = !isOrientationPortrait;
      adjustOrientation(isOrientationPortrait);
    }
  }
});
```

Optimize the User Experience

Capture and respond to orientation changes

```
private static boolean calculateOrientationPortrait() {
    return Window.getClientHeight() > Window.getClientWidth();
}
```

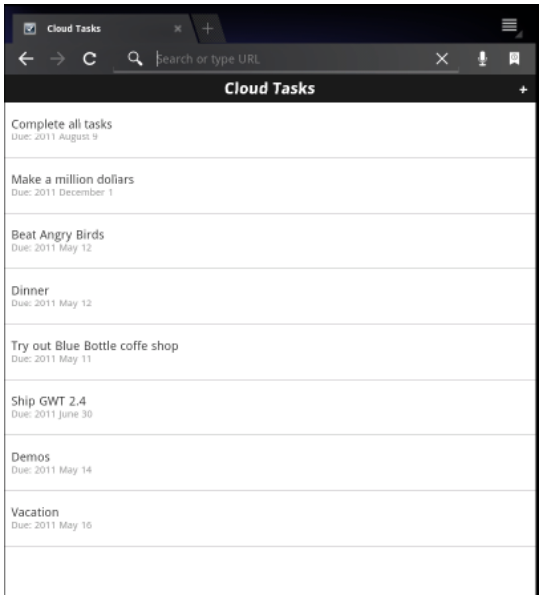
Optimize the User Experience

Capture and respond to orientation changes

```
/*
 * Listen for changes in the window size so we can adjust the
 * orientation of the app. This will also catch orientation changes
 * on mobile devices.
 */
windowResizeHandler = Window.addResizeHandler(new ResizeHandler() {
  public void onResize(ResizeEvent event) {
    if (isOrientationPortrait != calculateOrientationPortrait()) {
      isOrientationPortrait = !isOrientationPortrait;
      adjustOrientation(isOrientationPortrait);
    }
  }
});
```

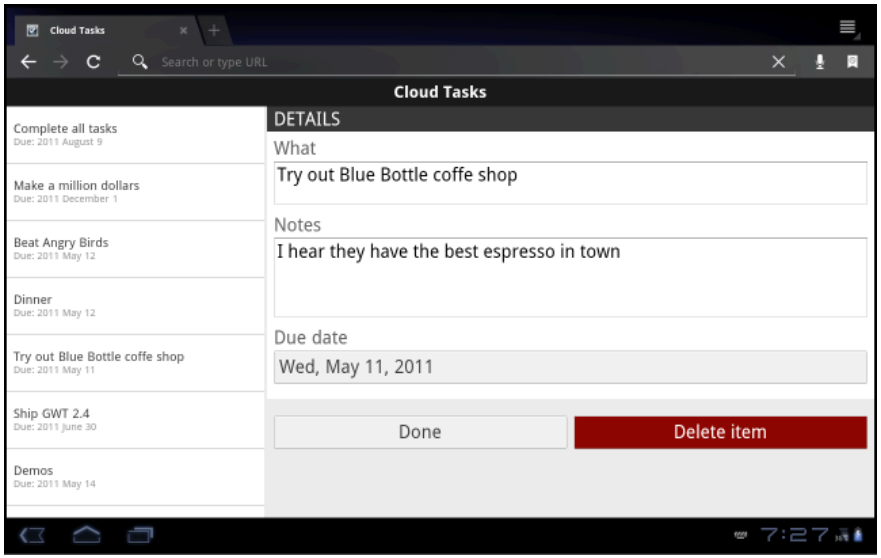
Optimize the User Experience

Capture and respond to orientation changes



Optimize the User Experience

Capture and respond to orientation changes



Google I/O 11

Optimize the User Experience

Maximize code reuse using the MVP pattern

Optimize the User Experience

Maximize code reuse using the MVP pattern

- MVP recap
  - Model-View-Presenter design pattern
  - Similar to MVC - less logic in the View
  - Business logic goes are part of Model/Presenter
  - Keep Views as simple as possible
  - Faster test cycles: run tests as vanilla JRE tests

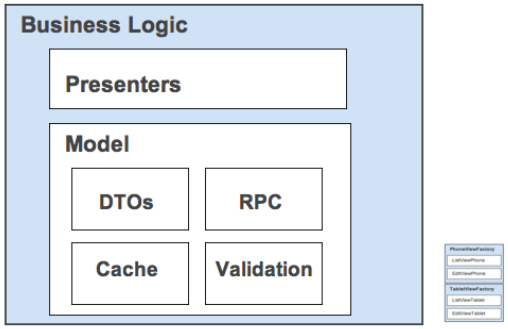
Optimize the User Experience

Maximize code reuse using the MVP pattern

- Tablet vs Phone
  - Business logic is the same
  - Views change
  - Ideal design pattern for supporting multiple form factors

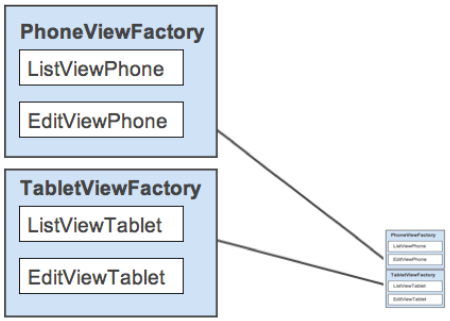
Optimize the User Experience

Maximize code reuse using the MVP pattern



Optimize the User Experience

Maximize code reuse using the MVP pattern



Optimize the User Experience

Select the right UI using deferred binding

CloudTasks.gwt.xml (GWT module config)

```
<!-- Use ViewFactoryImplTablet for tablet form factor. -->
<replace-with class="com.google.mobilecalendar.client.ViewFactoryImplTablet">
  <when-type-is class="com.google.mobilecalendar.client.ViewFactory"/>
  <when-property-is name="formfactor" value="tablet"/>
</replace-with>
```

- Build great UIs for any form factor
- Optimize the user experience
- Make it snappy
- Work without a connection

Make it Snappy

- Increase responsiveness
  - **Problem**: HTTP requests are expensive over 3G connections
- Minimize startup time
  - **Problem**: Mobile processing power still trails desktops/laptops
  - **Problem**: Parsing large amounts of Javascript can be a large source of latency

Make it Snappy

- Increase responsiveness
  - **Solution**: Use GWT's Client Bundle to batch resource fetches
- Minimize startup time
  - **Solution**: Use Code Splitting to grab only the code you need
  - **Solution**: Remove unused code, reduce overall code size using GWT compiler optimizations

Make it Snappy

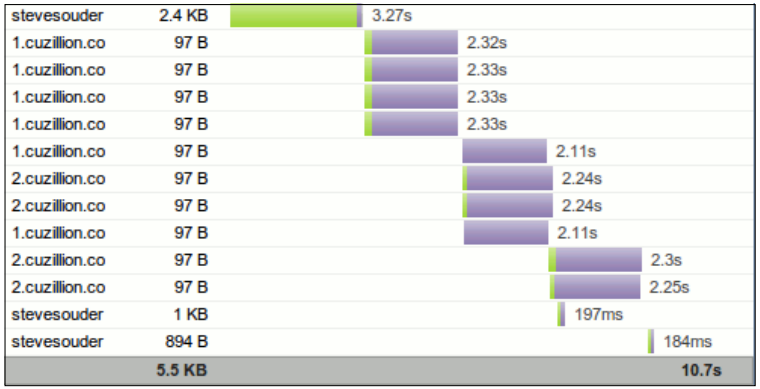
Increase responsiveness by bundling resource fetches

Demo

Make it Snappy

Increase responsiveness by bundling resource fetches

- Even worse, mobile browsers limit the number of concurrent requests
- Ex http request waterfall chart for Android - 4 concurrent TCP connections



http://calendar.perfplanet.com/2010/mobile-performance-analysis-using-pcapperf/

Make it Snappy

Increase responsiveness by bundling resources

```
public interface Resources extends ClientBundle {
    @Source("image0.gif")
    public ImageResource image0();

    @Source("image1.gif")
    public ImageResource image1();

    ...
}
```

Make it Snappy

Increase responsiveness by deferring resource fetching

```
void onShowImagesButtonClicked() {
    GWT.runAsync(new RunAsyncCallback() {
        public void onSuccess() {
            showImagesDialog();
        }
    }
}
```

Make it Snappy

Increase responsiveness by deferring resource fetching

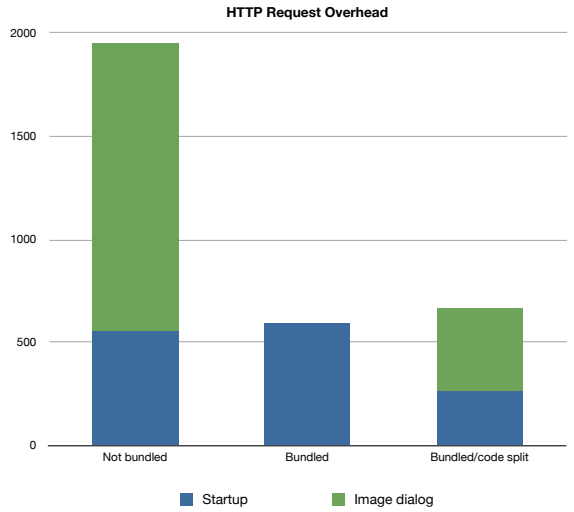
```
void onShowImagesButtonClicked() {
    GWT.runAsync(new RunAsyncCallback() {
        public void onSuccess() {
            showImagesDialog();
        }
    }
}
```

Make it Snappy

Increase responsiveness by deferring resource fetching

Total time spent waiting (ms)

- Not bundled: 1409ms
- Bundled: 594ms
- Bundled and code split: 664ms

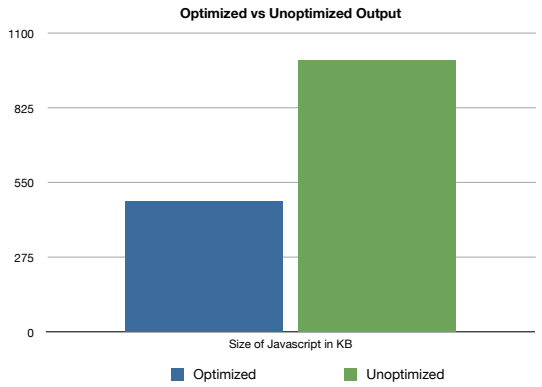


Make it Snappy

Reduce overall code size, reduce startup time

- More Javascript == more parsing == increased load time, cpu cycles
- Increased cpu cycles == battery drain == poor user experience

- GWT compiler optimizations can help
  - Dead code elimination
  - Duplicate function removal
  - Obfuscation



- Build great UIs for any form factor
- Optimize the user experience
- Make it snappy
- Work without a connection

Work Without a Connection

- Run wherever, whenever
  - Use Application Cache to load resources locally
- Access data wherever, whenever
  - Use Local Storage as a caching mechanism for RPCs

Work Without a Connection

Run wherever, whenever using Application Cache

- Part of the HTML5 feature set
- Loads resources like HTML, CSS, and JS from disk
- Works with iPhone and Android (+ Chrome, Safari, and Firefox)

Work Without a Connection

Run wherever, whenever using Application Cache

- appcache.nocache.manifest
  - Lists all of the files that you want to browser to cache
    - Include HTML, CSS, JS
      - Exclude GWT-RPC-related files (security)
      - "nocache" - always requested
- web.xml
  - mime-type for manifest files (text/plain)
- <your_app>.html
  - include <html manifest="app.nocache.manifest">

Work Without a Connection

Run wherever, whenever using Application Cache

...or check out the GWT 2.4 SDK :-)

Work Without a Connection

Run wherever, whenever using Application Cache

```
<!-- Define a custom App Cache linker -->
<define-linker name="appcachelinker"
    class="com.google.gwt.sample.mobilewebapp.linker.AppCacheLinker"/>

<!-- Use it -->
<add-linker name="appcachelinker"/>
```

Work Without a Connection

Access data wherever, whenever using Local Storage

- Part of the HTML5 feature set
- Utilizes a local database for reading/writing data
- Key/value-pair persistence
- Works with iPhone and Android (+ Chrome, Safari, and Firefox)
- Included in the GWT 2.3 SDK

Work Without a Connection

Access data wherever, whenever using Local Storage

- Grab an instance of the Local Storage interface

```
localStorage = Storage.getLocalStorageIfSupported();
```

- Read cache and then request any updates

```
public List<Tasks> getTasks() {
  String taskListString = localStorage.getItem(TASKLIST_SAVE_KEY);
  List<Tasks> tasks = deserialize(taskListString);
  requestFactory.taskRequest().queryForUpdates().fire(...) {
    ...
  }
  return tasks;
}
```

Work Without a Connection

Access data wherever, whenever using Local Storage

- Upon response, update cache and any listeners

```
public List<Tasks> getTasks() {
  ...
  requestFactory.taskRequest().queryForUpdates().fire(
    new Receiver<List<Tasks>>() {
      public onSuccess(List<Tasks> tasks) {
        String tasksListString = serialize(tasks);
        localStorage.setItem(TASKSLIST_SAVE_KEY, tasksListString);
        // update everyone listening for task list updates
      }});
  ...
}
```

Summary

- There are many challenges when building great mobile web apps
- GWT and GPE can help
- Tools for quickly building great UIs
- Frameworks for optimizing the user experience
- Compilers and code generators to improve speed
- API support for working offline

More Info...

- You can download the source and tools here:
  - http://code.google.com/webtoolkit/download.html
- Other sessions to check out
  - Use Page Speed to Optimize Your Web Site For Mobile (Check back for YouTube link)
  - Mobile Web Development: From Zero to Hero (12:30pm)
  - HTML5 versus Android: Apps or Web for Mobile Development? (3pm)

Q&A

</presentation>