



Effective GWT: Developing a complex, high-performance app with Google Web Toolkit

Examples

Code for the examples is available from
development.lombardi.com

GWT – What and Why?

- ▶ GWT is a toolkit that allows you to write Java code that will be compiled into JavaScript and executed client side in a user's browser.
- ▶ You can use your favorite Java based tools to develop, test and debug your web application.
- ▶ It handles some of the differences between how different browsers handle JavaScript and the HTML DOM so you don't have to.
- ▶ The compiler optimizes JavaScript allowing you to write faster than possible JavaScript.
- ▶ Can integrate with existing JavaScript code and libraries.

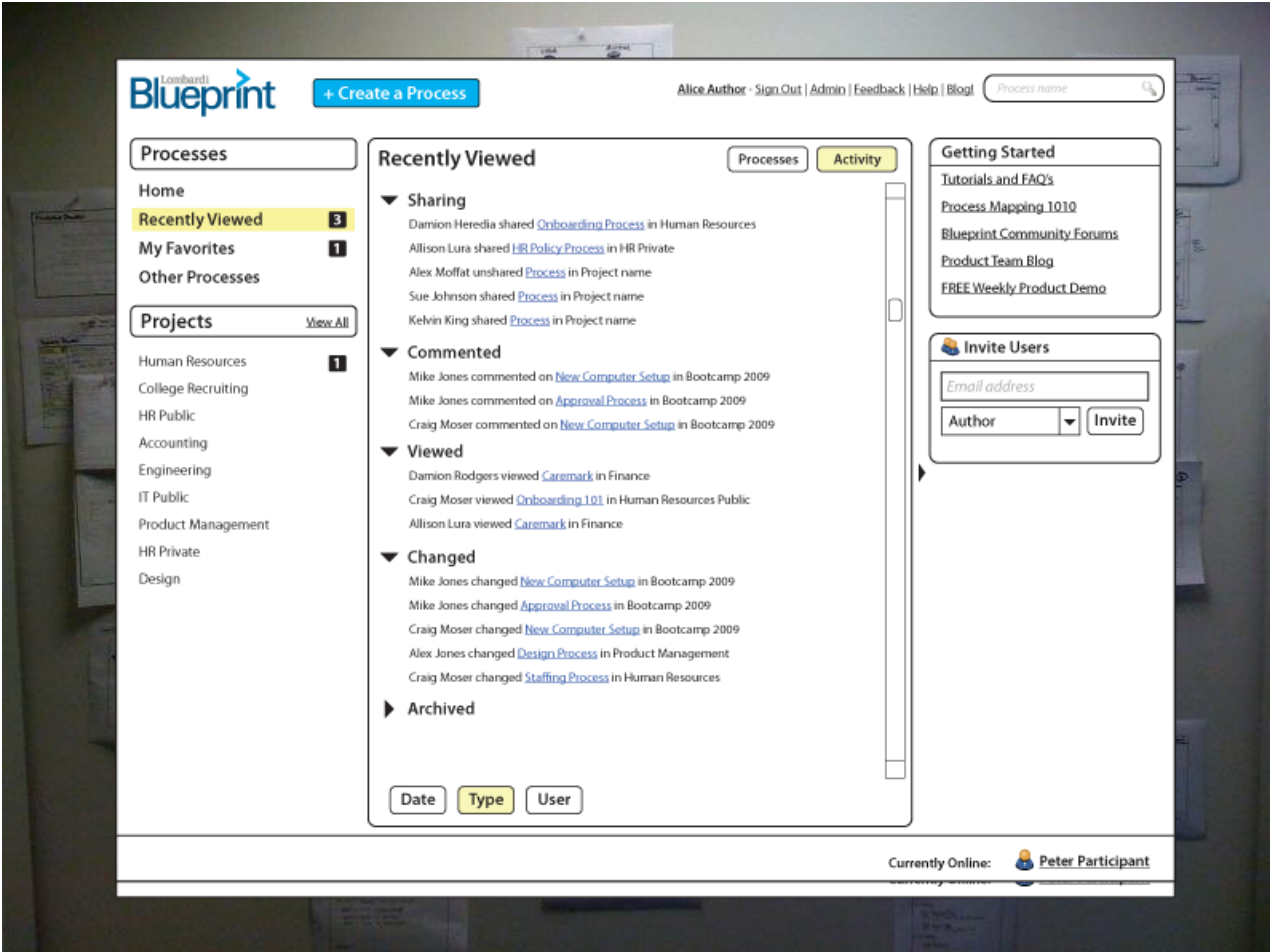
To provide some context if you've not seen Blueprint before

QUICK DEMO

UI Development is an infinite capacity for taking pains

UI DEVELOPMENT

Wireframe



Interaction Mockup

- ▶ Still an open question for us.
- ▶ We use PowerPoint or Keynote at the moment.
- ▶ Investigating iPlotz

High Fidelity Mockup

The mockup shows a web application with a dark blue header and a light blue sidebar. The main content area is white with a blue border. The sidebar contains navigation menus for 'Processes' and 'Projects'. The 'Processes' menu has 'Home', 'Recently Viewed' (3), 'My Favorites' (2), and 'Other Processes'. The 'Projects' menu has 'View All' and a list of project categories with counts: Human Resources (10), College Recruiting, Accounting, HR Public, Product Management (3), Design, Pre Sales, Onboarding (8), IT Public, Training, Outsourcing, Engineering, Human Resources, College Recruiting, Accounting, HR Public, and Product Management. The main content area features a 'What's New' section with a star icon, categorized by date: Today, Yesterday, and Feb 1, 2009. Each category lists recent activity with user names and process titles. At the bottom of the 'What's New' section are filter buttons for 'Date', 'Type', and 'User'. The right sidebar contains a 'Time is ticking!' warning about a 15-day free trial, a 'Getting Started' section with links to tutorials, process mapping, forums, and a product demo, and a 'Currently Online' status bar at the bottom showing three active users: Allison Lura, Stu Rodgers, and Craig Moser, with a '3 More' dropdown.

Lombardi Blueprint | Mindi Acosta | Sign Out | Feedback | Help | Blog! | process name

Processes

- Home
- Recently Viewed (3)
- My Favorites (2)
- Other Processes

Projects [View All](#)

- Human Resources (10)
- College Recruiting
- Accounting
- HR Public
- Product Management (3)
- Design
- Pre Sales
- Onboarding (8)
- IT Public
- Training
- Outsourcing
- Engineering
- Human Resources
- College Recruiting
- Accounting
- HR Public
- Product Management

What's New

Today

- Damion Kelley shared [Onboarding Process](#) in Human Resources
- Damion Kelley changed [Onboarding Process](#) in Human Resources
- Allison Lura shared [HR Policy Process](#) in HR Private
- Alex Moffat joined your account as a designer by invitation from Chris Miller
- Sue Johnson created Accounting project
- Kelvin King joined your account as a participant by invitation from Mike Thompson
- Damion Kelley archived [Hiring Process - OLD](#) in Human Resources

Yesterday

- Mike Jones shared [New Computer Setup](#) in Bootcamp 2009
- Beth Rodgers joined your account as a participant by invitation from Chris Miller
- Cliff Vars created [New Hire Orientation](#) in College Recruiting

Feb 1, 2009

- Stu Rodgers shared [Caremark](#) in Finance
- Beth Rodgers joined your account as a designer by invitation from Allison Howell
- Cliff Vars shared [New Hire Orientation](#) in College Recruiting

Date Type User

Time is ticking!

You only have **15 days** left on your free trial account

Click below to purchase your account now and keep full use of Blueprint past the end of your trial.

[Learn More](#)

Getting Started

- [Tutorials and FAQ's](#)
- [Process Mapping 101](#)
- [Blueprint Community Forums](#)
- [Product Team Blog](#)
- [FREE Weekly Product Demo](#)

Currently Online: Allison Lura | Stu Rodgers | Craig Moser | 3 More

Going to code

- ▶ Be involved in the design.
- ▶ You need to know CSS and the HTML DOM.
- ▶ Fundamentally.
 - What's the appropriate DOM structure?
 - How create and manipulate with GWT?
- ▶ GWT removes browser specific JS and DOM issues
 - But you still have to deal with cross browser CSS
 - And the browser quirks


Example UI


▼ **Sharing**

Everyone (All account authors can edit all processes.)

Authors (3) Add

Can create and edit processes in this project.

-  Alex Moffat ×
-  JC Denton ×
-  Alice Author ×

 Admin  Author  Participant

Example Four



Back



Forward



Refresh



Stop



Compile/Browse

http://localhost:8888/four.html



Users (1)

Have access to this project.

Add



Alice



http://localhost:8888/four.html

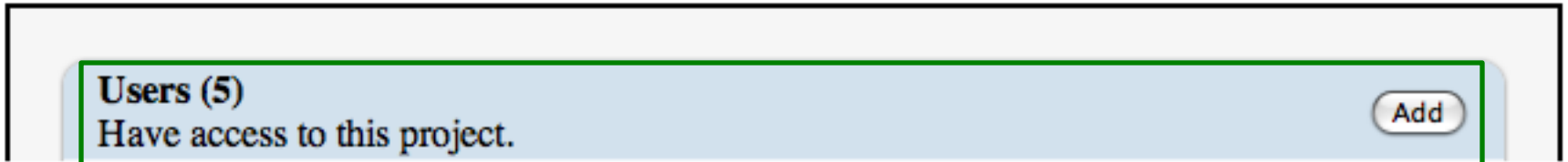
Header

Users (5)

Have access to this project.

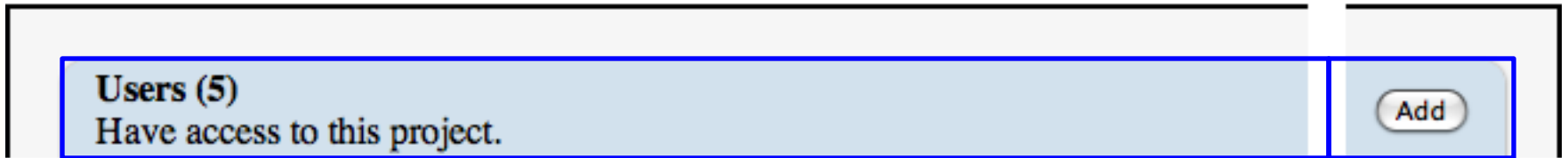
Add

Header



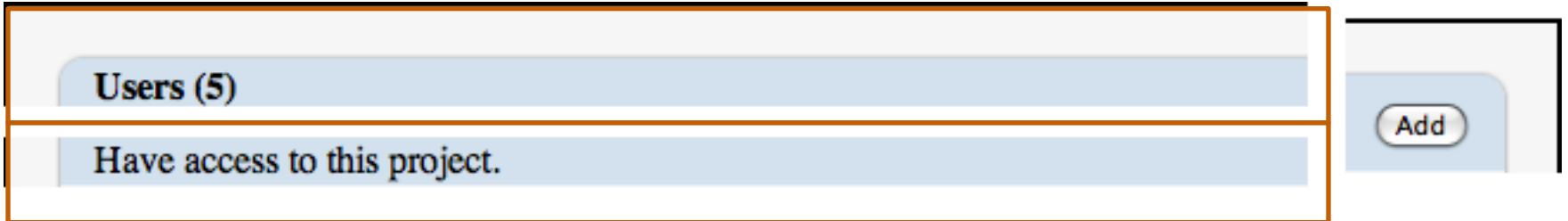
```
<div> <!-- FlowPanel -->
</div>
```

Header



```
<div>  
  <div> <!-- FlowPanel -->  
  </div>  
  <div> <!-- FlowPanel -->  
  </div>  
</div>
```

Header



```
<div>
  <div>
    <div> <!-- Label -->
    </div>
    <div> <!-- Label -->
    </div>
  </div>
  <div>
  </div>
</div>
```

Building the header (1)

```
private Widget buildHeader(ClickHandler buttonHandler) {  
  
    Label authorCount =  
        new Label(messages.numberOfAuthors(0));  
    Label description =  
        new Label(messages.authorsDescription());  
  
    FlowPanel left = new FlowPanel();  
    left.setStyleName("left");  
    left.add(authorCount);  
    left.add(description);  
  
    addButton = new Button(messages.add());  
    addButton.addClickHandler(buttonHandler);  
}
```


Building the header (2)

```
FlowPanel right = new FlowPanel();  
right.setStyleName("right");  
right.add(addButton);
```

```
FlowPanel header = new FlowPanel();  
header.setStyleName("header");  
header.add(left);  
header.add(right);
```

```
return header;
```

```
}
```

Outer Panel

Users (5) Add

Have access to this project.
Have access to this project.

- Alice ✕
- Bob ✕
- Carol ✕
- Dave ✕
- Emily ✕

Outer Panel

```
public class RoundedContainerWithHeader
    extends Composite {
...
    public RoundedContainerWithHeader(Widget header,
        Widget content) {

        this.header = header;
        this.content = content;

        DecoratorPanelWithHeader panel =
            new DecoratorPanelWithHeader();
        initWidget(panel);

        setStyleName("roundedContainerWithHeader");
    }
}
```

Rounded Corners CSS

```
.roundedContainerWithHeader .topLeft {  
    background-image: url("images/top_left.gif");  
    background-repeat: no-repeat;  
    width: 12px;  
    height: 43px;  
}  
  
.roundedContainerWithHeader .topCenter {  
    background-image: url("images/top_repeat.gif");  
    background-repeat: repeat-x;  
    height: 43px;  
}
```

Handling Window Resizing

- ▶ The content in the center cell reveals a resizing issue.
- ▶ With static HTML you're limited to what you can achieve with CSS.
- ▶ With GWT you can do programmatic layout.
- ▶ Listen for the `ResizeEvent` from the window and propagate sizes down to children.

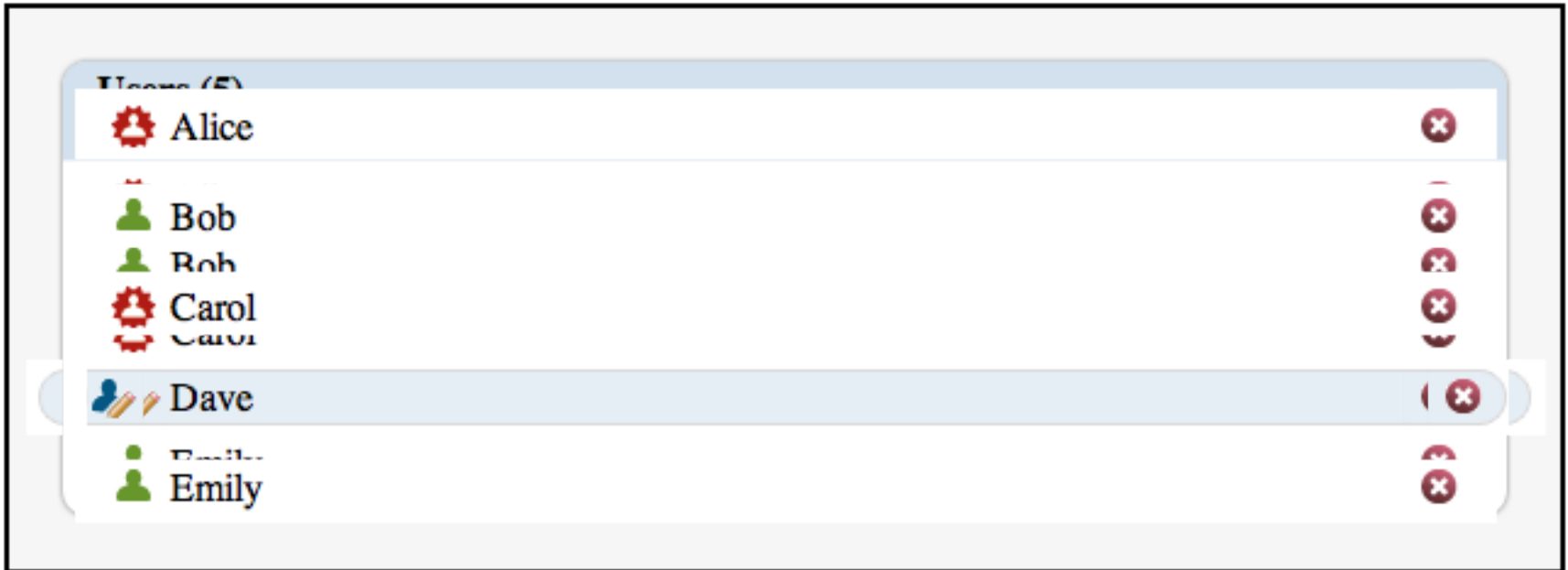
Extending DecoratorPanel

```
private void setHeader(Widget header) {
    DOM.appendChild(getCellElement(0, 1),
        header.getElement());
    adopt(header);
}

public Iterator<Widget> iterator() {
    // Need to return header here.
}
```

<http://development.lombardi.com/?p=644>

List and Row



Highlighting Rows (1)

- ▶ Done by adding and removing a CSS class
- ▶ Use `mouseover` and `mouseout`

```
private static MouseOverHandler mouseOverHandler =
    new MouseOverHandler() {
        public void onMouseOver(MouseOverEvent
            mouseOverEvent) {
            Row row = (Row) ((FocusPanel)
                mouseOverEvent.getSource()).getParent();
            row.mouseOver();
        }
    };
...
private void mouseOver() {
    getWidget().addStyleName("focussed");
}
```


Highlighting Rows (2)

- ▶ Additional class added to row
- ▶ Used to set background images

```
.row .rightEndCap {  
    float: right;  
    width: 10px;  
    height: 23px;  
    background-color: white;  
}  
  
.row.focussed .rightEndCap {  
    background-image: url("focus_right_end_cap.gif");  
}
```

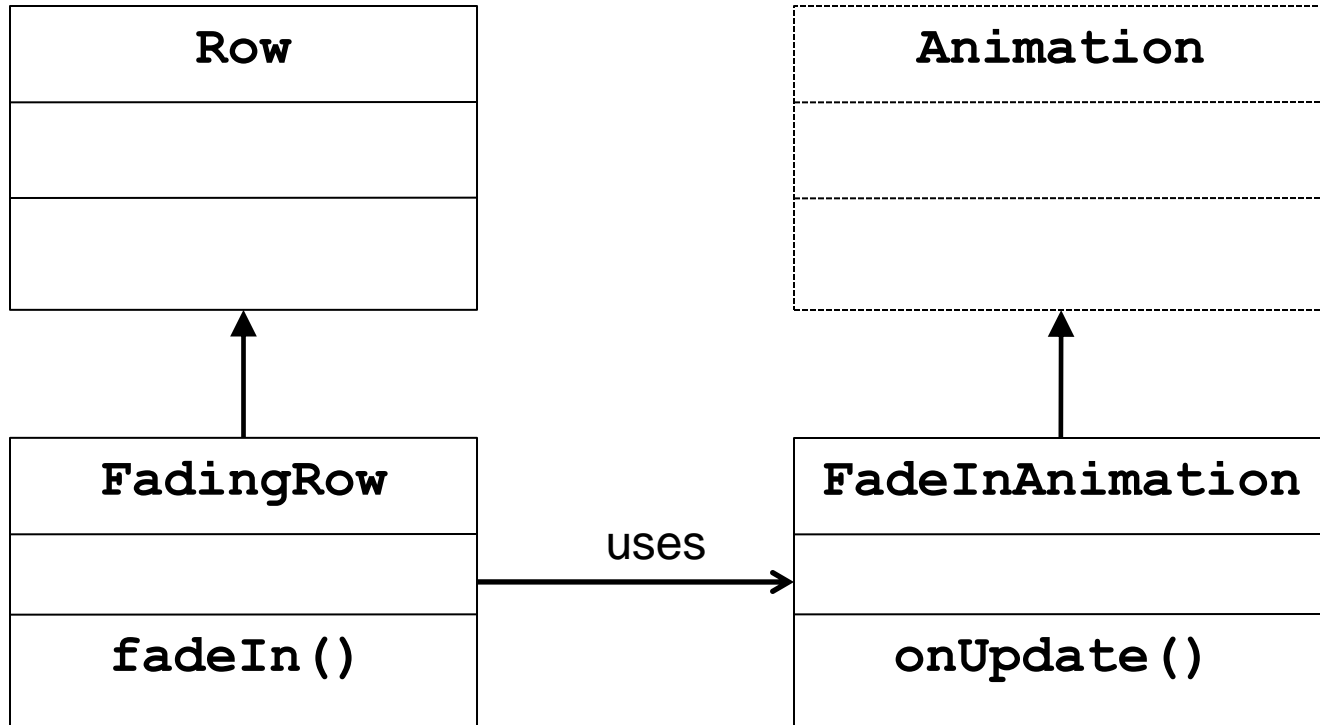
Animation

- ▶ You must use JavaScript, and therefore GWT (no CSS3 yet)
- ▶ A little animation goes a long way.
- ▶ Helps users understand the behavior of the application.
- ▶ Generally you change
 - Size
 - Position
 - Opacity

Fading rows

- ▶ Rows fade in when added and out when removed.
- ▶ Subclass GWT Animation class
- ▶ Each animation has
 - onStart- Some initial processing
 - onUpdate- Called from timer on regular basis
 - onComplete- After animation is finished
 - onCancel – If animation is cancelled

Fading Rows



Adding the row

```
private FlowPanel listPanel;  
...  
private void addUser() {  
    FadingRow row = new FadingRow(...);  
    listPanel.add(row);  
    row.fadeIn();  
}
```

Fade In

```
public class FadingRow extends Row {  
    ...  
    public void fadeIn() {  
        runningAnimation = new FadeInAnimation(this);  
        runningAnimation.run(750);  
    }  
    ...  
  
    public class FadeInAnimation extends Animation {  
        ...  
        private Element e;  
        ...  
        protected void onUpdate(double progress) {  
            DOMHelper.setOpacity(e, progress);  
        }  
    }  
}
```

Demo ...

PERFORMANCE

Original Implementation

- ▶ Typical MVC Design
- ▶ Created GWT widgets for each item on the diagram and attached handlers to each widget.

for each item (complete object containing all our data properties)

```
ActivityWidget widget = new ActivityWidget()  
widget.addKeyPressHandler(...)  
widget.addMouseDownHandler(...)  
root.add(widget)
```

```
ActivityWidget()  
    FlowPanel panel = new FlowPanel()  
    TextBox textBox = new TextBox()  
    Image gotoLinkedImage = new Image()  
    panel.add(textBox)  
    panel.add(gotoLinkedImage)  
    ...
```

Create a complex widget

Add the widget to the root

This Has Some Problems

- ▶ This design is very heavy. It creates lot of Javascript objects including multiple UI widget objects for each item and multiple handler objects.
- ▶ Handler objects could have been reused since they provide the appropriate Widget when called.
- ▶ But requires attaching handlers to each widget.
- ▶ This standard design is used for most of our application, but the diagram was too complicated for it.

New Implementation

- ▶ Goal #1: render as quickly as possible.
- ▶ Generate raw HTML in Javascript.
- ▶ Use a fly-weight pattern for event handling.
- ▶ Two classes and instances for each type of object (Activity, Decision, Line, Swimlane, etc.). One for rendering HTML and one for event handling.
- ▶ One handler for the entire diagram.

Rendering

```
StringBuilder buffer = new StringBuilder()  
for each item  
    switch (item.type)  
        case Activity: ActivityRenderer.render(buffer, item)  
    ...  
DOM.setInnerHTML(rootElement, buffer.toString())
```

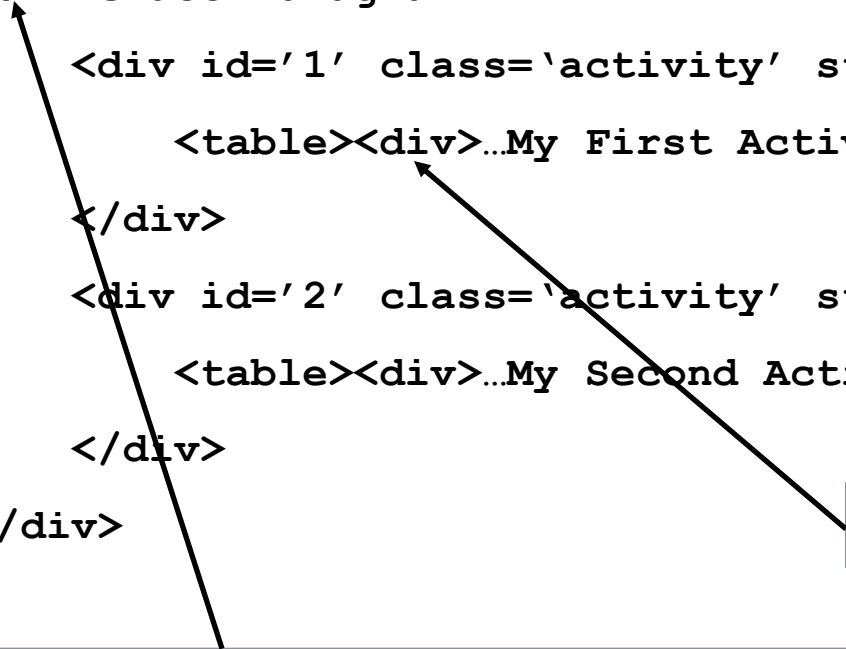
Create a buffer for all the HTML

Stuff all of it into the DOM in one go

```
ActivityRenderer.render(StringBuilder buffer, Activity item)  
    buffer.append("<div id='")  
    buffer.append(item.getId())  
    buffer.append("' class='activity' style='left:")  
    buffer.append(String.valueOf(item.getX())  
    buffer.append("px' >")  
    buffer.append(item.getName())  
    buffer.append("</div>")
```

Event Handling

```
<div class='diagram'>  
  <div id='1' class='activity' style='left:10px;top:10px'>  
    <table><div>...My First Activity Name...</div></table>  
  </div>  
  <div id='2' class='activity' style='left:50px;top:10px'>  
    <table><div>...My Second Activity Name...</div></table>  
  </div>  
</div>
```



The diagram illustrates event bubbling in a nested DOM structure. It shows a root `<div class='diagram'>` containing two `<div id='1' class='activity' style='left:10px;top:10px'>` and `<div id='2' class='activity' style='left:50px;top:10px'>` elements. Each of these `<div>` elements contains a `<table>` element, which in turn contains a `<div>` element. The text inside these innermost `<div>` elements is "...My First Activity Name..." and "...My Second Activity Name...". A blue box on the right contains the text "User clicks on innermost DIV". Two arrows originate from this box: one points to the innermost `<div>` inside the first `<table>`, and the other points to the root `<div class='diagram'>` element, demonstrating how the event bubbles up the DOM tree.

User clicks on innermost DIV

Event bubbles up to the top element and includes the deepest element that was clicked on.

Event Handling

Diagram()

```
    sinkEvents(Event.ONMOUSEDOWN | Event.ONDBLCLICK |...)
```

```
public void onBrowserEvent(Event event)
    Element target = event.getEventTarget();
    String itemId;
    do {
        itemId = target.getAttribute("id")
        if (itemId == null) {
            target = target.getParentElement();
        }
    } while (itemId == null);
```

```
int type = getType(itemId)
EventHandler handler = getHandler(type)
switch (event.getTypeInt()) {
    case Event.ONMOUSEOVER: handler.onMouseOver(event, itemId)
    ...
}
```

Enable a single handler for the entire diagram. on.

Walk up the tree until we find the root element for the item.

Let the specific handler handle the event.

Event Handling...Need Some Help

- ▶ All we have to work with after rendering is HTML (the DOM).
- ▶ Need useful data structures to handle events.
- ▶ Construct those data structures later after rendering by using the information in the DOM.
- ▶ Data structures can be simpler than a complete UI Widget.

Event Handling...Need Some Help

```
public void setRenderedHTML(String html)
    DOM.setInnerHTML(root, html)
    DeferredCommand.addCommand(new Command() {
        public void execute() {
            createCache()
        }
    })
}

public void createCache() {
    for (int index; index < root.getChildNodes().getLength(); index++) {
        Element item = root.getChildNodes().getItem(index);
        String id = item.getAttribute("id");
        int x = item.getStyle().getProperty("left");
        ...
        new DiagramObject(id, x, y, ...)
    }
}
```

All the HTML for the diagram

Execute deferred to allow the browser to display the HTML

The DOM has all the data we need

When All Else Fails

- ▶ Dual compile your code to Javascript and Java bytecode.
- ▶ If code is too slow to run on the client, run it on the server.
- ▶ The Java VM is *much* faster than the Javascript engine in IE6 (it's faster in IE7 and FF).

- ▶ A simple algorithm:
 - Run the command on the client the first time.
 - If at any point, the average client time is slower than some threshold, run the next command on the server.
 - From then on, run the command on whichever has the best average time.

- ▶ For us the RPC interface is almost entirely HTML. This works well since we already have the ability to generate the data structures we need from HTML.

Server Side In Action

- ▶ Demo ...

Miscellaneous Tips

- ▶ To save compile time during development, compile only for the browser you're actively working on and only compile the modules you've changed.

```
<module>
  <inherits name='com.google.gwt.user.User' />
  <source path="client" />

  <!-- ie6, gecko, gecko1_8, safari or opera -->
  <!--
  <set-property name="user.agent" value="ie6" />
  -->
</module>
```

And Finally...

- ▶ Put as many style constants as possible in CSS and avoid calling `Element.getStyle().setProperty()`.
- ▶ Avoid using iterators over lists (unnecessary object creation).
- ▶ Use `IncrementalCommand` to handle large lists.
- ▶ For performance or non-hosted mode testing create your own “console” using an HTML frame and `document.write`. You can also use the `gwt-log` library or the Logging library from the gwt incubator.



Lombardi