

Real World GWT

Why Lombardi built Blueprint with GWT and why you should use GWT.

What is Blueprint?

- Show some of the things we've done with GWT.
- Provide some context for later parts of the presentation.

Blueprint History

- Dojo and Flash
- Dojo and HTML
- GWT
- 3-4 developers

**GWT is the best way
to build browser
applications.**

GWT

- You write Java
- You debug Java
- GWT compiles to standard JavaScript
- You deploy JavaScript

But

- GWT doesn't restrict what you can do
- Same access as if you wrote JavaScript
- Use as much or as little as you want

GWT Fundamentals

- Compiler - Faster JavaScript than you would write by hand.
- Hosted Mode - Rapid development without compilation to JavaScript. Unit testing.

User Interface

- Static HTML
- Widgets
- DHTML
- A combination of all three



[Process Mapping 101](#) is the easiest way to get started with Blueprint! The self paced training course teaches simulated sessions and hands on exercises. [Check it out today!](#)

Welcome to Blueprint... the fast and easy way to map, model
mapping your processes, or view our [tutorials and FAQ](#)



- Map and mod
- Collaborate an
- Identify proble
- Prioritize obje

Get Starte

Recently Viewed Processes

Other Recent Process

Home Page


Static HTML

```
addListener("accountHomeCreateProcessButton2",
    createProcessListener, Event.ONCLICK);

private void addListener(String id,
    EventListener listener, int events) {
    Element element = DOM.getElementById(id);
    DOM.sinkEvents(element, events);
    DOM.setEventListener(element, listener);
}
```


► **Secure Billing Information:** for purchase orders, contact [sales](#)


Credit Card Number


CSC # 



Credit Card Expiration

1 - January  2008 

Billing Cycle 

3 months 

Billing Address (must match the address on your credit card statement)

City

State / Province

Zip / Post

► **Terms and Conditions**

PLEASE READ THIS SUBSCRIPTION AGREEMENT ("AGREEMENT") CAREFULLY BEFORE PRESSING THE "ACCEPT" BUTTON. BY PRESSING "ACCEPT," YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, PLEASE CLOSE YOUR WEB BROWSER AND YOU WILL BE UNABLE TO USE THE LOMBARDI SOFTWARE.

Billing Information

Widgets

```
additionalUsers.addFocusListener(new  
FocusListenerAdapter() {  
    @Override  
    public void onLostFocus(Widget sender) {  
        estimateProvider.requestEstimate();  
    }  
});
```


User Management


File Management

Account Information

Billing Details

Security

User Accounts

 [Invite New User](#) (2 additional user account licenses available)

Admin

Full Name

Email Address



Invited User

alex.moffat@lombardi.com



[Alex Moffat](#)

amoffat@lombardi.com

You are using **2 of the 4** available user licenses.

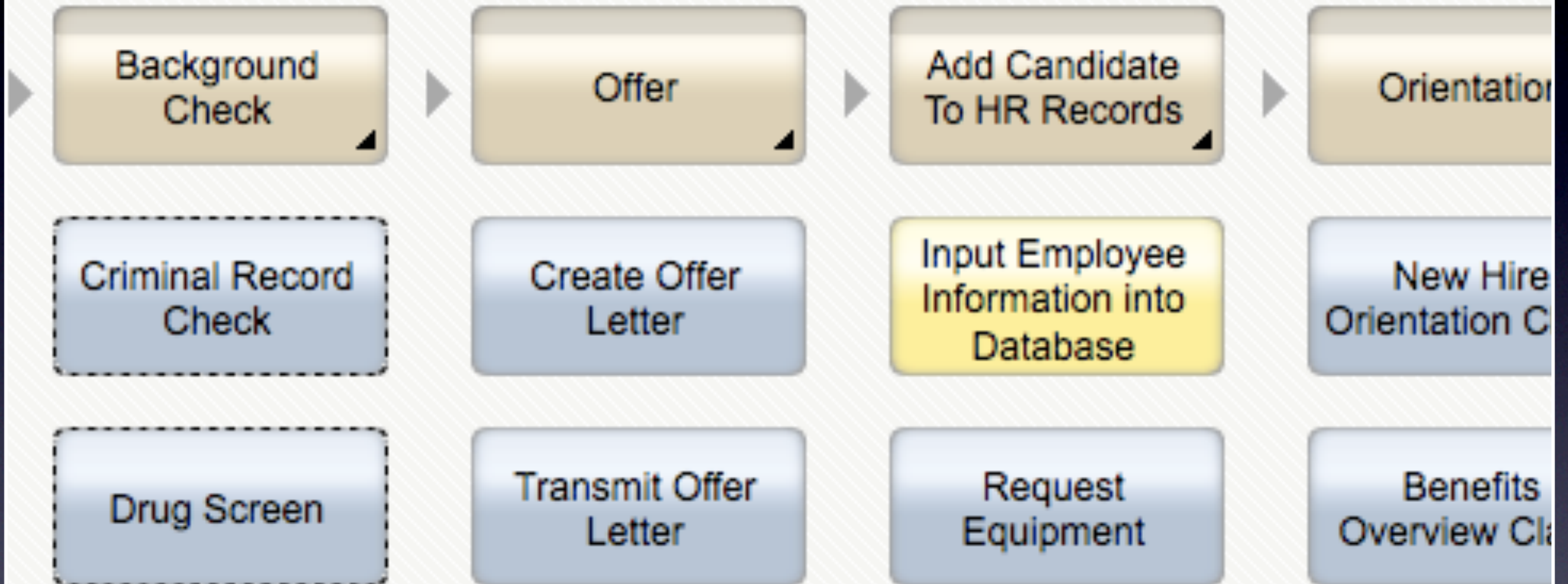
Account: acme

User admin

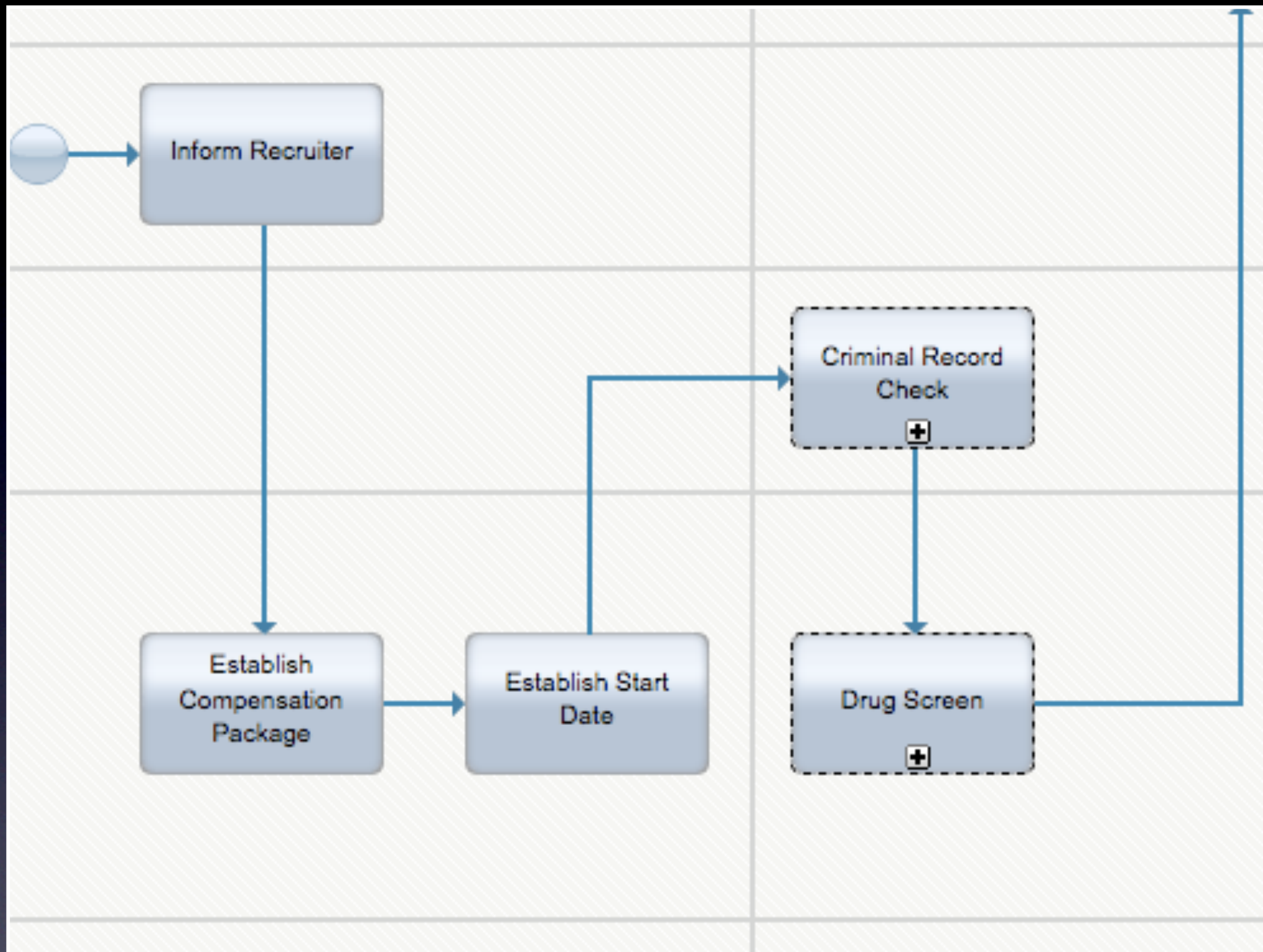
HR Specialist

Facilities T...

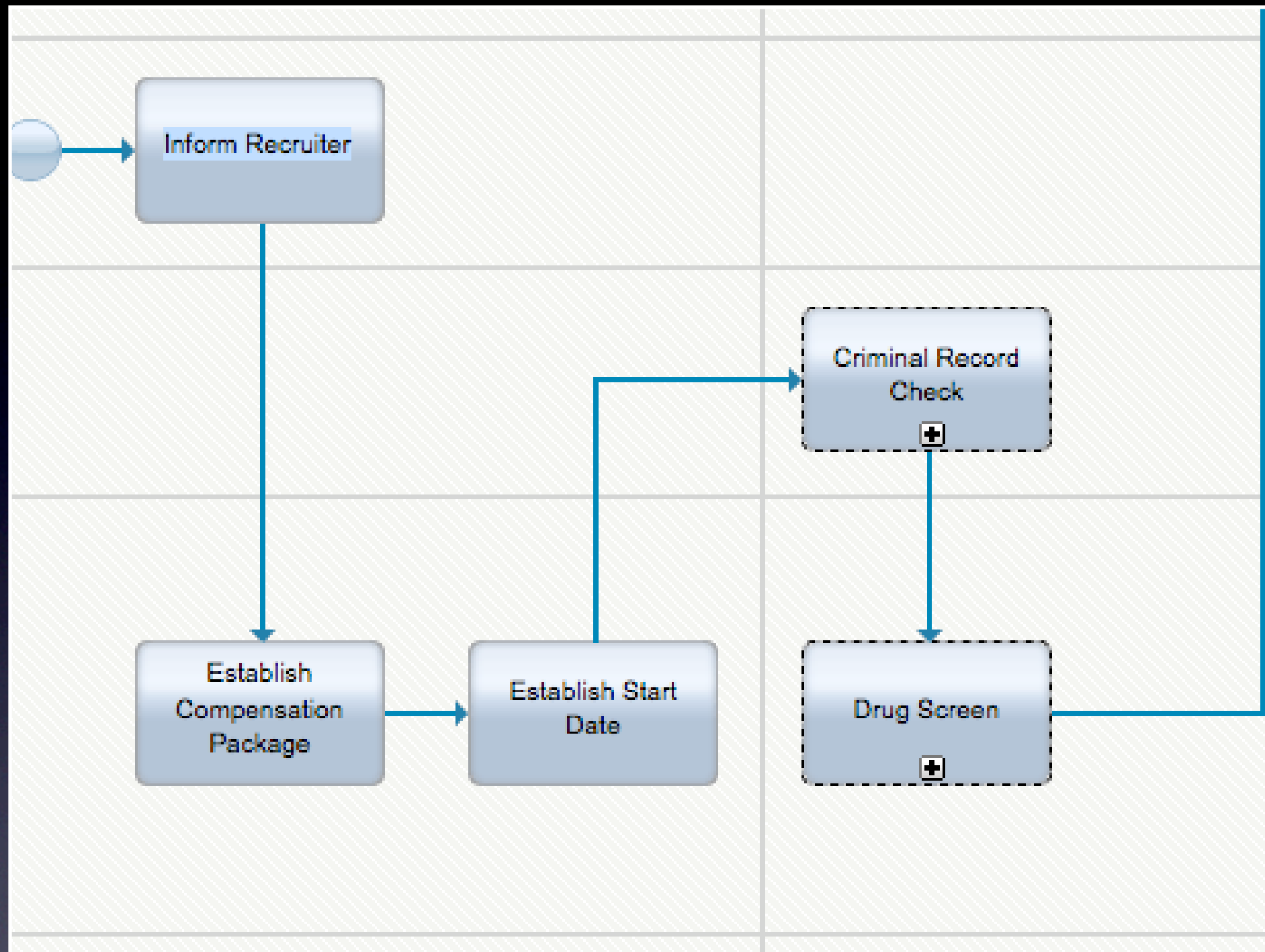
- Onboarding



Map



Diagram

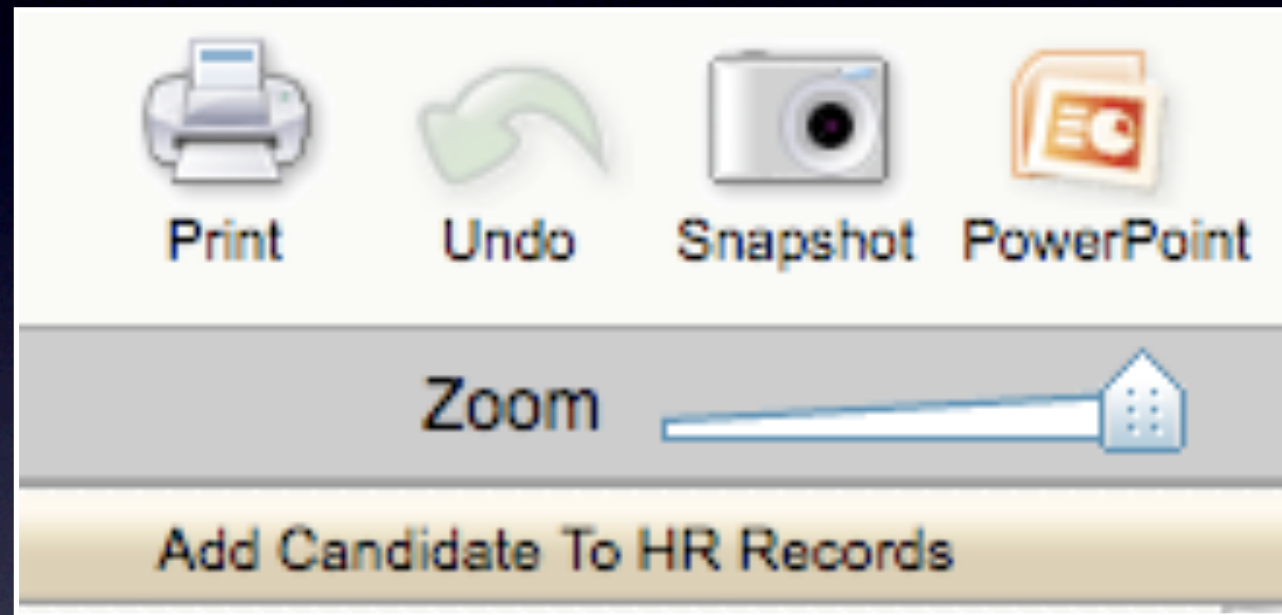


Inline editing

JavaScript Integration

- JSNI

```
native static String  
    getValue (JavaScriptObject object) /*- {  
        return (object.value == undefined) ?  
            null : object.value;  
    }-*/;
```

Zoom Control

Zoom Control

```
private String name;  
...  
public native void  
  setPosition(int p) /*-{  
    var name =  
    this.@com...JQuerySlider::name;  
    $wnd.$("#"+name).sliderMoveTo(p);  
  }-*/;
```

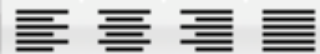

Inform Recruiter

Details

Problems

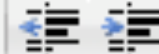
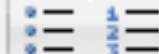
Documentation

B *I* U



Font family

Font size



A



Once we have settled on the right candidate, we need to update our recruiters so that they know the position is filled. This is also the trigger for them generating an invoice to us for their placement services. For this reason, it is essential that all communication with Recruiters is managed by the Human Resources team. We have agreed with all of our Recruiters that we will notify them at the end of each week.

NOTE: Recruiters may not contact Hiring Managers directly! It is essential they work with their Human Resources contact.

Tiny MCE

Calling the server

- Plain old HTTP calls
- RPC

HTTP

```
try {
    RequestBuilder builder = new RequestBuilder(RequestBuilder.GET, url);
    Request req = builder.sendRequest(null,
        new RequestCallback() {
            public void onError(Request request, Throwable t) {
                // Handle the error
            }
            public void onResponseReceived(Request request, Response response) {
                if (200 == response.getStatusCode()) {
                    // Process the response in response.getText()
                } else {
                    // Handle the error
                }
            }
        });
} catch (RequestException e) {
    // Couldn't connect to the server.
}
```


RPC

```
try {
    RepositoryServiceAsync service = RepositoryService.App.getInstance();
    RequestBuilder builder = service.getNotifications(currentVersionId,
        new AsyncCallback<ClientNotification>() {
            public void onFailure(Throwable t) {
                ClientRepositoryManager.logError("Error doNotification[onFailure]", t);
            }
            public void onSuccess(ClientNotification notification) {
                handleNotificationResults(notification.changeNotification);
            }
        });
    builder.setTimeoutMillis(REQUEST_TIMEOUT);
    builder.send();
} catch (Exception e) {
    ClientRepositoryManager.logError("Error doNotification", e);
}
```


Why Java?

- Tooling
- Static typing

Modern Editing

- You can use the full power of your IDE.
- Refactoring
- Where used
- etc.

Debugging

- Hosted mode enables rapid development.

Build and Deploy

- We use standard tools
- IDEA and Eclipse
- Maven and Ant
- Perforce
- TeamCity and Jira
- Apache and Jetty

How fast is this?

Damn Fast!

- Dead code elimination.
- Method inlining.
- String interning.
- Reduced JavaScript size.
- And the compiler's getting faster.

Faster Than Possible

```
<div id="text">  
  <div>Hello World</div>  
</div>
```


Faster Than Possible

```
public void onModuleLoad() {  
    Element e1 =  
        DOM.getElementById("text");  
    Element.as(e1.getChildNodes()  
        .getItem(1))  
        .setInnerText("Goodbye World");  
}
```


Overlay Types

- Treat native JavaScript objects as Java objects.
- No modification to the underlying JavaScript object.

GWT Generated

```
var $intern_5 = 'Goodbye World', ...
$intern_4 = 'text';
function init() {
    var el_0;
    el_0 =
        $doc.getElementById($intern_4);
    DOMImpl_$setInnerText(
        el_0.childNodes[1], $intern_5);
}
```


Browser specific code

```
if (browser == IE) {  
    el.innerText = text;  
} else if (browser == FF) {  
    el.textContent = text;  
} else {  
    el.removeAllChildren();  
    el.insertTextNode(text);  
}
```


Deferred Binding

- Browser specific code at compile time
- Locale specific as well

IE6 Permutation

```
var $intern_5 = 'Goodbye World', ...
$intern_4 = 'text';
function init() {
    var e1_0;
    e1_0 =
        $doc.getElementById($intern_4);
    e1_0.childNodes[1].innerText =
        $intern_5;
}
```


FF Permutation

```
var $intern_5 = 'Goodbye World', ...
$intern_4 = 'text';
function init() {
  var e1_0;
  e1_0 =
    $doc.getElementById($intern_4);
  e1_0.childNodes[1].textContent =
    $intern_5;
}
```


OBF

```
var q = 'Goodbye World', ... p =  
'text';  
function rb(){var a;a=  
$doc.getElementById(p);a.childNodes[1]  
.innerText=q}
```


Blueprint Example

- Diagram rendering

Original Implementation

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

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

```
ActivityWidget widget = new ActivityWidget()
widget.addKeyboardListener(...)
widget.addMouseListener(...)
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 with
Add listeners to the 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 listener objects.
- Listener objects could have been reused since they are passed the target Widget when called.
- But requires attaching listeners 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 and rendering.
- Two classes and instances for each type of object (Activity, Decision, Line, Swimlane, etc.). One for rendering HTML and one for event handling.
- One listener for the entire diagram.

Rendering

```
StringBuffer buffer = new StringBuffer()
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(StringBuffer 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>
```

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 |...)
```

Enable a single listener for the entire diagram.

```
public void onBrowserEvent(Event event) {
```

```
    Element target = DOM.eventGetTarget(event)
```

```
    String itemId;
```

```
    do {
```

```
        itemId = DOM.getElementAttribute(target, "id")
```

```
        if (itemId == null) {
```

```
            target = DOM.getParent(target);
```

```
        }
```

```
    } while (itemId == null);
```

```
    int type = getType(itemId)
```

```
    EventHandler handler = getHandler(type)
```

```
    switch (DOM.eventGetType(event)) {
```

```
        case Event.ONMOUSEOVER: handler.onMouseOver(event, itemId)
```

```
        ...
```

```
    }
```

```
}
```

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

Let the specific handler handle the event.
this item type.

Event Handling

- All we have to work with after rendering is HTML (the DOM).
- Need additional 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

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

public void createCache() {
    for (int index; index <DOM.getChildCount(root); index++) {
        Element item = DOM.getChild(root, index);
        String id = DOM.getElementAttribute(item, "id")
        int x = DOM.getStyleAttribute(item, "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 (JavaScript engines are getting faster though).
- 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.

In Conclusion

- We love GWT
- You should use GWT
- Give Blueprint a try
blueprint.lombardi.com
- If you're interested, read our blog
development.lombardi.com