Google™ 09 IO

Measure in Milliseconds: Performance Tips for Google Web Toolkit

Kelly Norton
May 27, 2009

the plan

Why worry about performance?

4 things GWT does for you.

4 things you should do for yourself.

Why Worry About Performance?

Why worry about performance?

It is a usability concern

Why worry about performance?

It is a usability concern

how delay affects us? a simple model.

Why worry about performance?

It is a usability concern

how delay affects us? a simple model.

0.1 seconds - instantaneous.

Why worry about performance?

It is a usability concern

how delay affects us? a simple model.

0.1 seconds - instantaneous.

1 second - pause, but still seems like a single task.

Why worry about performance?

It is a usability concern

how delay affects us? a simple model.

0.1 seconds - instantaneous.

1 second - pause, but still seems like a single task.

10 seconds - disruption, abandonment, goes to facebook to complain about the design.

Why worry about performance?

Users notice

How many times have you encountered something that “just seems sluggish.”

We’re going to all this trouble to give user’s more control

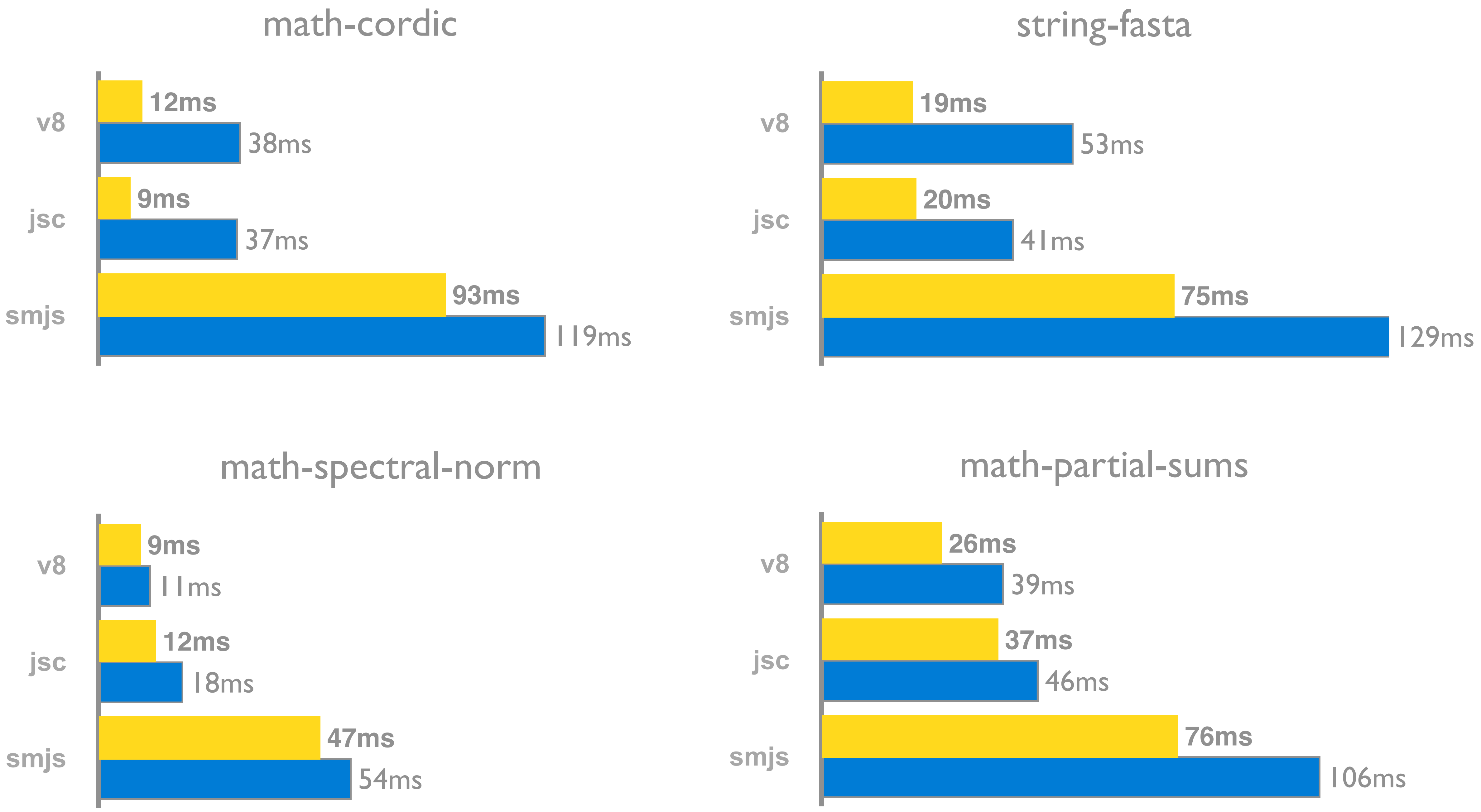
A study in 1997 even showed that users evaluate the quality of content lower when latency is increased.

Sears, A., Jacko, J.A., Borella, M.S. (1997), "Internet delay effects: how users perceive quality, organization, and ease of use of information", Late-Breaking/Short Talks, http://wwacm.org/sigchi/chi97/proceedings/short-talk/als2.htm, pp.22-7.

4 things GWT does for you.

#1. Produces fast JavaScript

Ported some SunSpider tests to Java to track compiler performance



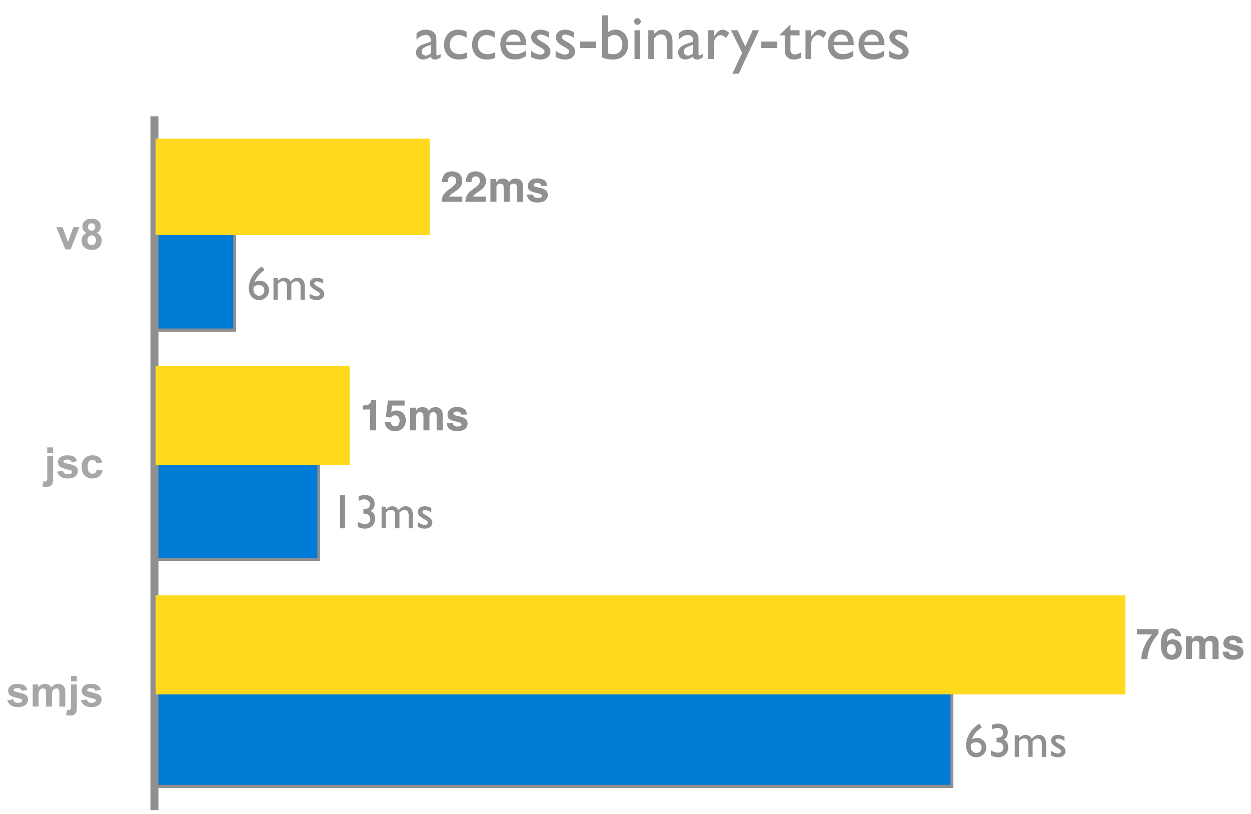
Results are from internally run tests.

TODO: Put those tests in a google code project.

Add legend

#1. Produces fast JavaScript

but lest you think we win them all



access-binary-trees revealed an opportunity to make our constructors faster.

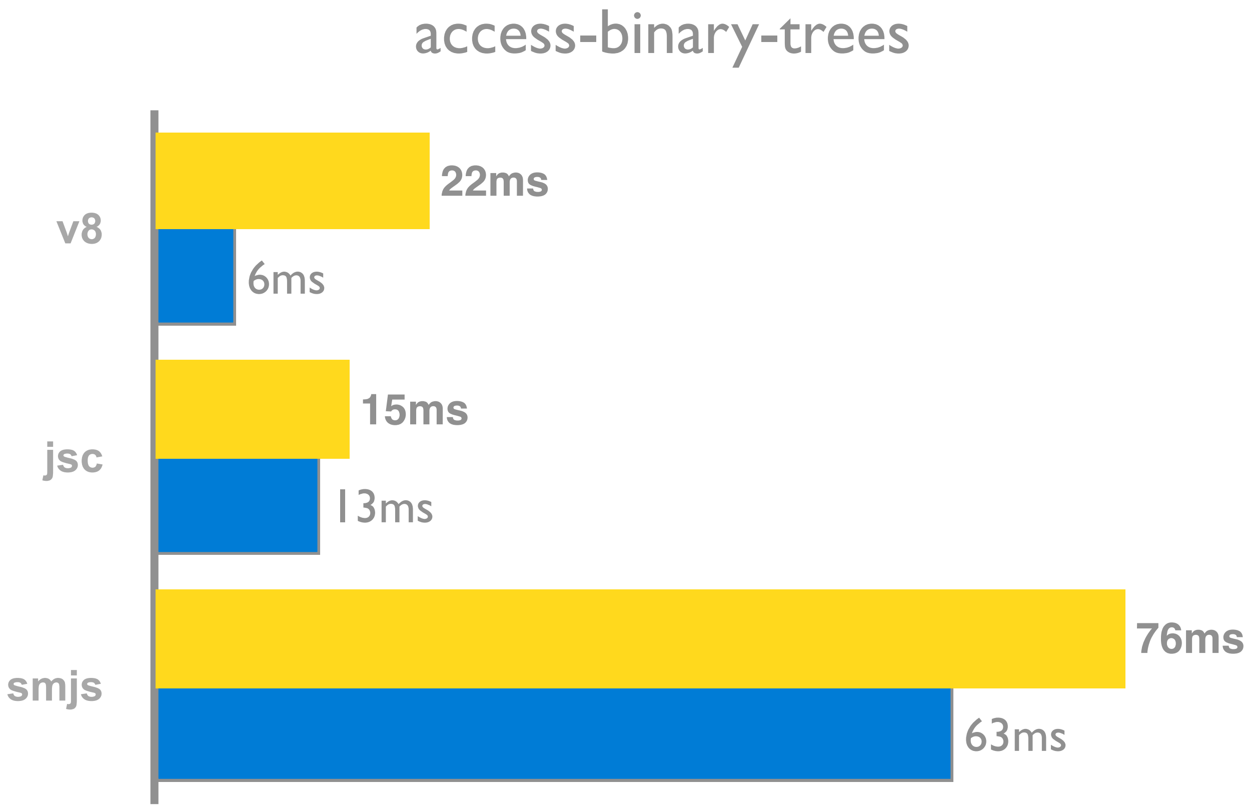
Results are from internally run tests.

TODO: Put those
a google code pr

Add legend

#1. Produces fast JavaScript

but lest you think we win them all



access-binary-trees revealed an opportunity to make our constructors faster.

that will be fixed in GWT 2.0 and you will get a speedup just by recompiling.

Results are from internally run tests.

TODO: Put those
a google code pr

Add legend

#1. Produces fast JavaScript

How is GWT faster? We cheat.

Inlining - eliminates levels of method dispatch.

```
Shape s = new Circle(aRadius);
widget.setText(“area: “ + s.getArea());

becomes:
Circle s = new Circle(aRadius);
widget.setText(“area: “ + s.getArea());

then becomes:
Circle s = new Circle(aRadius);
widget.element.textContent = “area: “ + (s.r * s.r * PI);
```

#1. Produces fast JavaScript

How is GWT faster? We cheat.

String interning - prevents unnecessary object creation and saves space.

```
Java:
void assignStyles() {
  Style style = this.getElement().getStyle();
  style.setProperty(“color”, “red”);
  style.setProperty(“border”, “1px solid black”);
}

JavaScript:
var a = ‘color’, b = ‘red’, c = ‘border’, d = ‘1px solid black’;
...
function assignStyles() {
  var style = this.element.style;
  style[a] = b;
  style[c] = d;
}
```

#1. Produces fast JavaScript

How is GWT faster? We cheat.

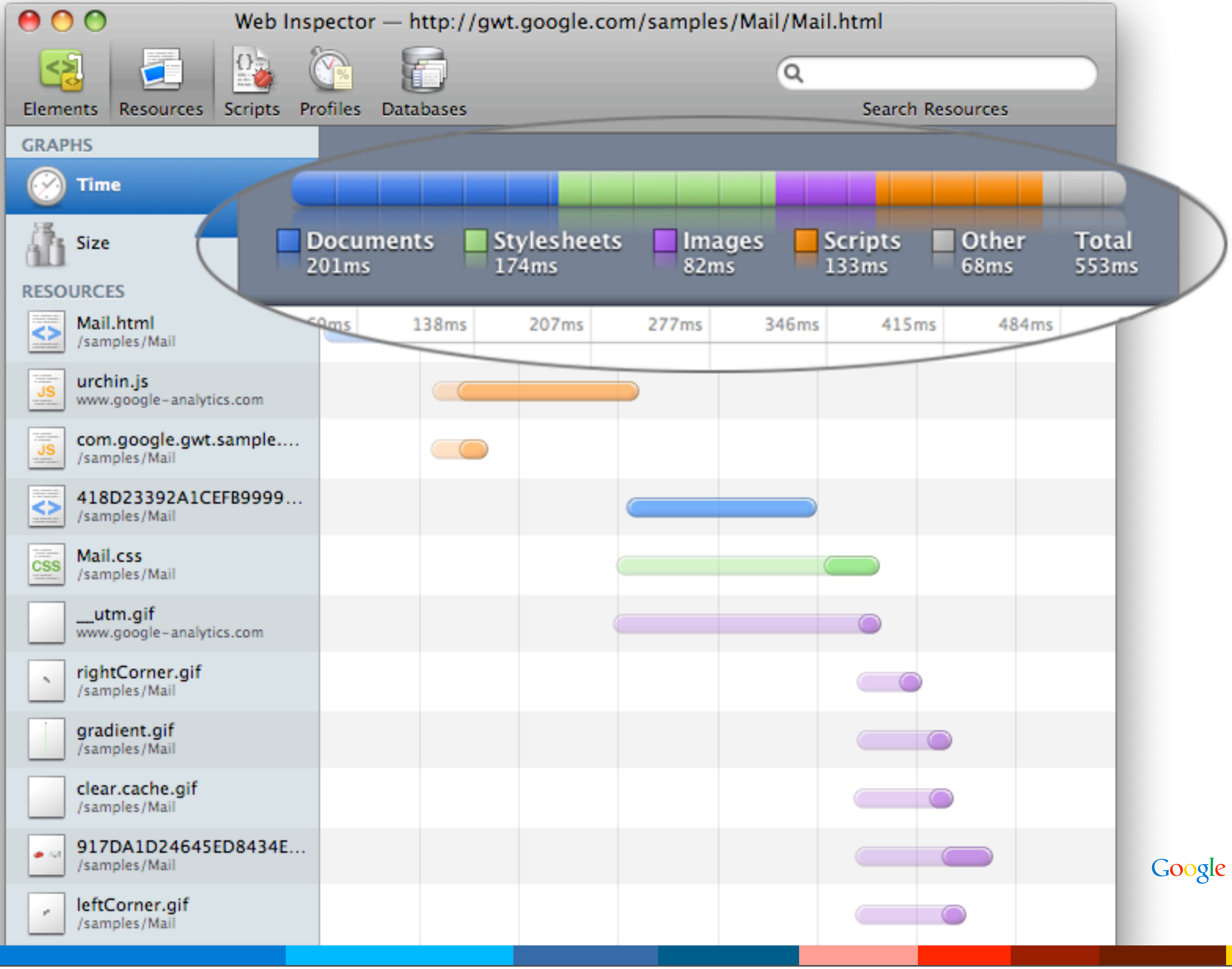
method de-virtualization - avoids prototype chain lookups.

```
class MyListener implements LibraryDelegate {
  ...
  public String getText() {
    return text;
  }
}

becomes (Java):
static String getText(MyListener self) { return self.text; }

then becomes (JavaScript)*:
function $getText(self) { return self.text; }
```

* In reality, this function would be eliminated entirely and the expression `object.text` would be inlined.

#2. Loads your non-code resources quickly

As of GWT 1.4, we bundled images but not CSS.



Web Inspector — http://gwt.google.com/samples/Mail/Mail.html

Mail.html /samples/Mail
urchin.js www.google-analytics.com
com.google.gwt.sample.... /samples/Mail
418D23392A1CEFB9999... /samples/Mail
Mail.css /samples/Mail
__utm.gif www.google-analytics.com
rightCorner.gif /samples/Mail
gradient.gif /samples/Mail
clear.cache.gif /samples/Mail
917DA1D24645ED8434E... /samples/Mail
leftCorner.gif /samples/Mail

#2. Loads your non-code resources quickly

MyBundle.java

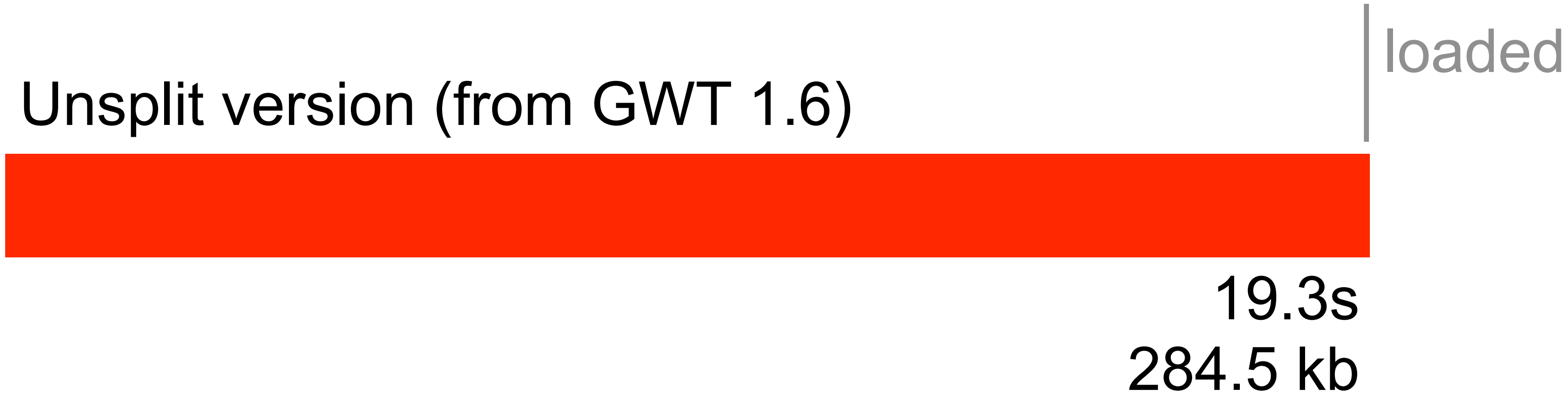
```
interface MyBundle extends ClientBundle {
  public static final MyBundle INSTANCE = GWT.create(MyBundle.class);

  @Source(“chicken.png”) ImageResource chickenImage();

  @Source("goat.png") ImageResource goatImage();

  @Source("default.css") CssResource defaultCss();
}
```

default.css

```
@if user.agent safari {
  .funky-box { -webkit-border-radius: 4px; }
} @elif user.agent gecko {
  .funky-box { -moz-border-radius: 4px; }
}

@sprite .goat-box { gwt-image: “goatImage”; color: #000; }
```

#3. Splits your code elegantly.

Loading GWT Showcase over a slow connection.



Unsplit version (from GWT 1.6)

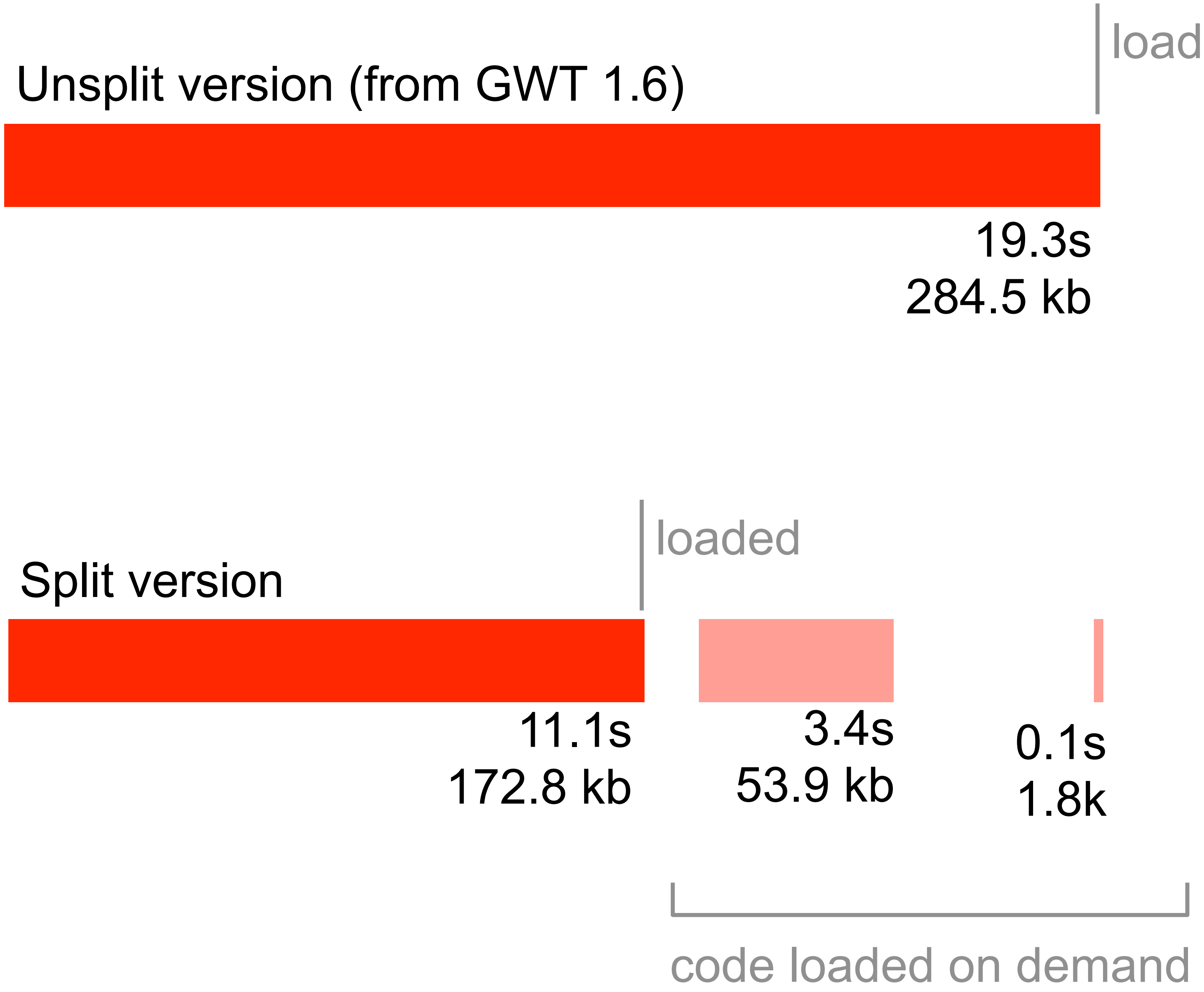
loaded

19.3s

284.5 kb

#3. Splits your code elegantly.

Loading GWT Showcase over a slow connection.



#3. Splits your code elegantly.

Just invoke GWT.runAsync and the compiler does the work.

```
GWT.runAsync(new RunAsyncCallback() {
  public void onFailure(Throwable caught) {
    openErrorNotification();
  }

  public void onSuccess() {
    openSettingsView();
  }
});
```

This will load all the code needed to run everything reachable from the callback, given all other code you've previously loaded.

#4. Interoperates with JavaScript at no cost

Consider GWT’s DOM library.

```
document.getBody().appendChild(
  document.createDivElement()).setInnerText(“Hello I/O”);

becomes (JavaScript):
$doc.body.appendChild(
  $doc.createElement(‘div’)).textContents = ‘Hello I/O’;
```

The ability to write directly to the DOM, but still get type-safety and GWT compiler optimizations.

#4. Interoperates with JavaScript at no cost

Use JavaScriptObject to work with JSON.

JSON:

```
{ ‘firstname’ : ‘Kelly’, ‘lastname’ : ‘Norton’ }
```

Java:

```
class Person extends JavaScriptObject {
  protected Person() {}
  final native String getFirstName() /*-{
    return this.firstname;
  }-*/;
}
```

At runtime, there is no overhead:

```
p.getFirstName(); => p.firstname;
```

4 things you should do yourself.

#1. Avoid unnecessary widgets.

Building up and tearing down of large Widget hierarchies is expensive.

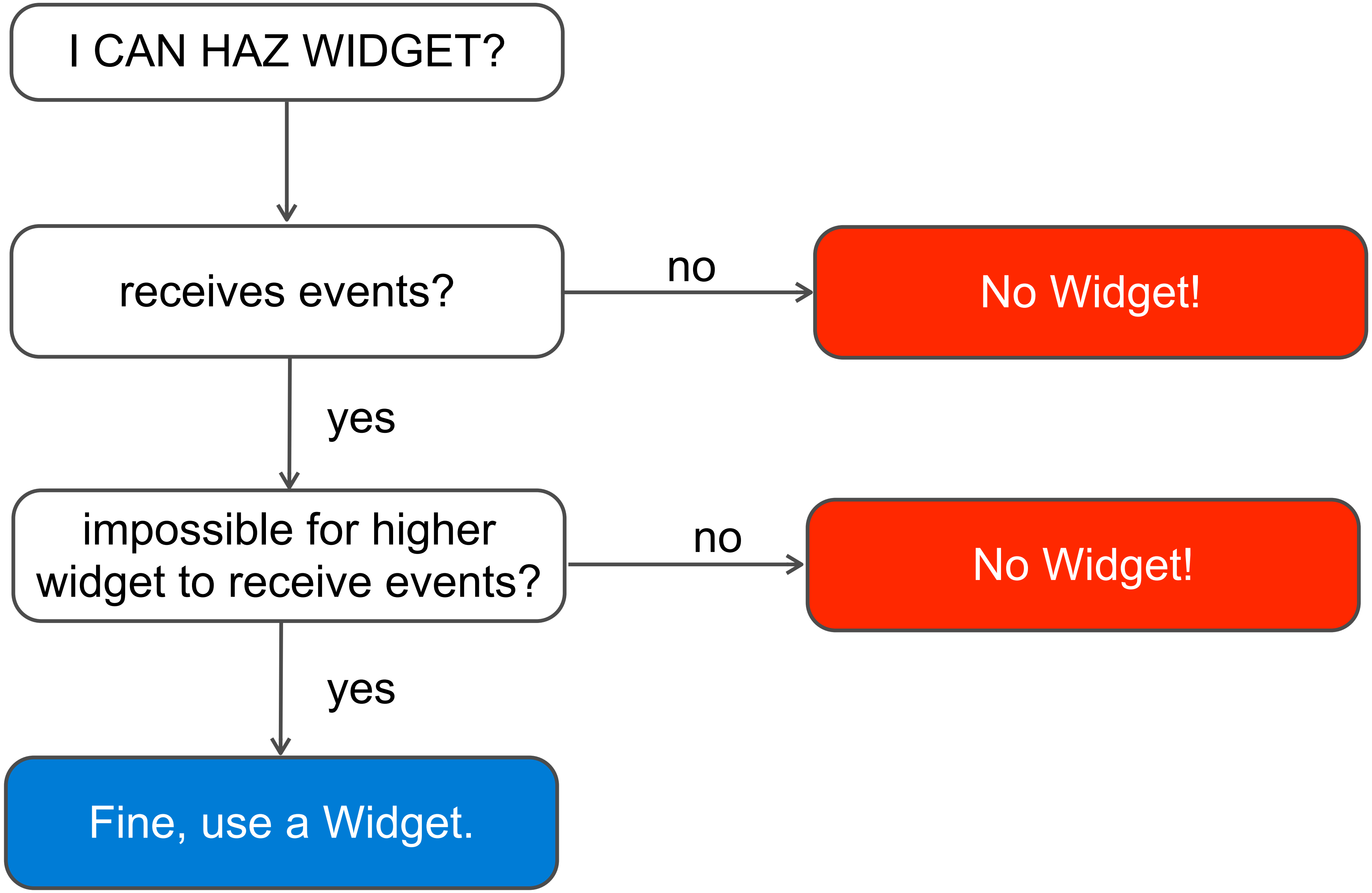
2 most common misuses:

Using nested widgets for layout.

Unnecessarily using widgets with listeners inside of panels.

#1. Avoid unnecessary widgets.

How to decide if you need a widget.



#1. Avoid unnecessary widgets.

Solution: Use HTMLPanel for layout.

```
HTMLPanel panel = new HTMLPanel("<div><div><div>” +
    “<span>Header</span><div id=\"content\"></div>” +
    “</div></div></div>");

FlowPanel content = new FlowPanel();
content.add(new TextBox());
content.add(new Button(“Save”));
panel.add(content, "content");
```

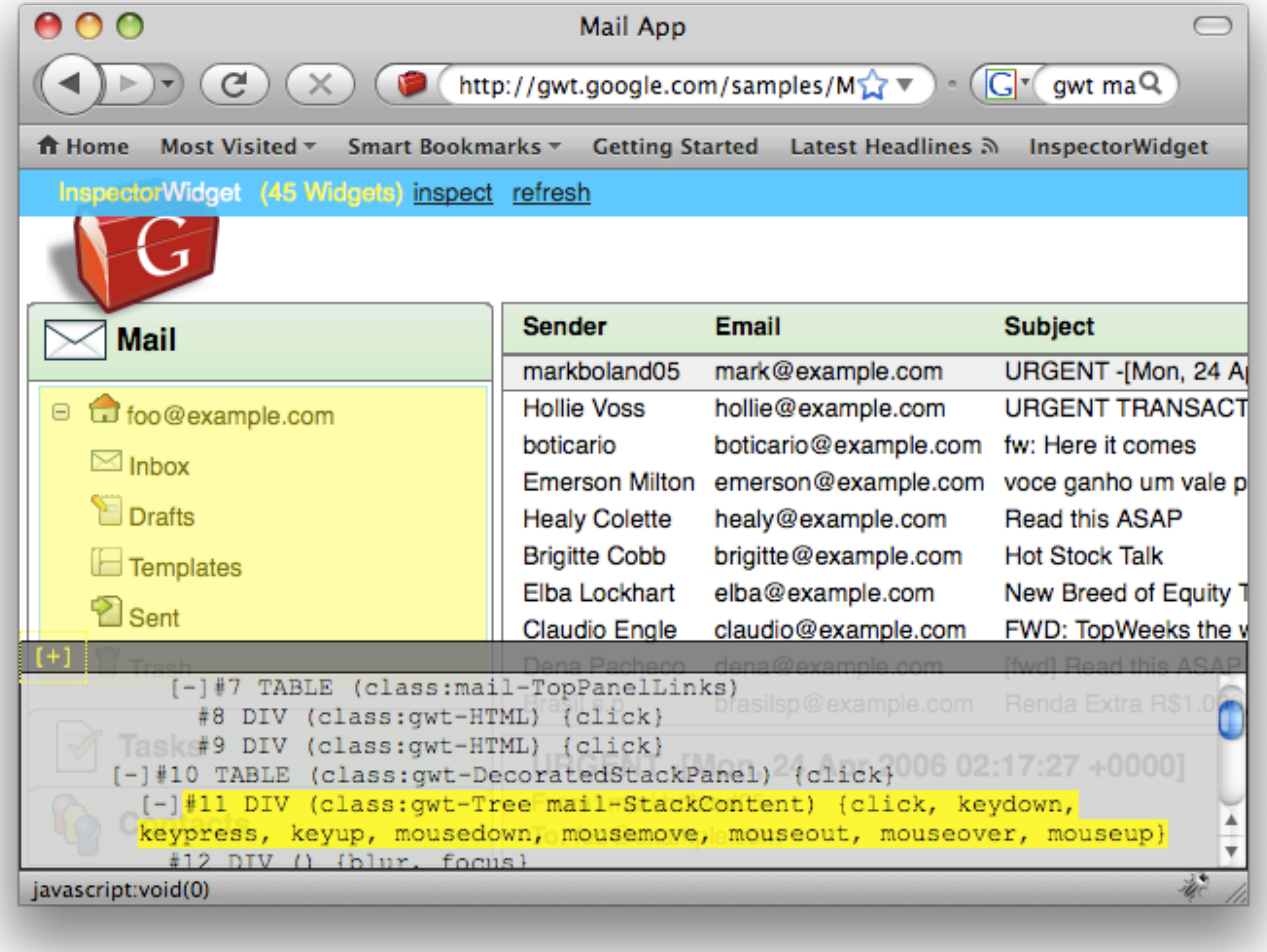
#1. Avoid unnecessary widgets.

Solution: Use event bubble to dispatch child events

```
class MyWidget extends SimplePanel {
  private DivElement header;
  private DivElement body;

  MyWidget {
    addHandler(new ClickHandler() {
      public void onClick(ClickEvent e) {
        Element el = Element.as(
           e.getNativeEvent().getEventTarget());
        if (el == header) {
          // The header was clicked.
        } else if (el == body) {
          // The body was clicked.
        }
      }
    }, ClickEvent.getType());
  }
}
```

#1. Avoid unnecessary widgets.

InspectorWidget: a bookmarklet for inspecting Widgets in a live app.



Sender	Email	Subject
markboland05	mark@example.com	URGENT -[Mon, 24 A
Hollie Voss	hollie@example.com	URGENT TRANSACT
boticario	boticario@example.com	fw: Here it comes
Emerson Milton	emerson@example.com	voce ganho um vale p
Healy Colette	healy@example.com	Read this ASAP
Brigitte Cobb	brigitte@example.com	Hot Stock Talk
Elba Lockhart	elba@example.com	New Breed of Equity T
Claudio Engle	claudio@example.com	FWD: TopWeeks the w
Dena Pacheco	dena@example.com	[fwd] Read this ASAP
Brasil s.p.	brasilsp@example.com	Renda Extra R$1.00

URGENT -[Mon, 24 Apr 2006 02:17:27 +0000]

```
      [-]#7 TABLE (class:mail-TopPanelLinks)
        #8 DIV (class:gwt-HTML) {click}
        #9 DIV (class:gwt-HTML) {click}
  [-]#10 TABLE (class:gwt-DecoratedStackPanel) {click}
    [-]#11 DIV (class:gwt-Tree mail-StackContent) {click, keydown,
    keypress, keyup, mousedown, mousemove, mouseout, mouseover, mouseup}
      #12 DIV () {blur, focus}
```

http://gwt-instrumental.googlecode.com/svn/latest/inspectorwidget/index.html

#2. Use debug builds effectively.

Remember that exceptions are for developers, not users.

```
void getThingAtIndex(int index) {
  if (index < 0 || index >= length) {
    throw new IndexOutOfBoundsException(
      “Dear User: though you can do nothing about this, ” +
      “I’m going to ensure that you download a message about” +
      “programatic indices! KTHXBAI, The programmer.”);
  }
  return things[index];
}
```

Some exceptions you will never catch.

#2. Use debug builds effectively.

Solution: Use assert instead.

```
void getThingAtIndex(int index) {
  assert index >= 0 && index < length;
  return things[index];
}
```

`assert` is turned on in hosted mode by default.

You can turn them on in web mode by passing -ea to the GWT compiler.

#2. Use debug builds effectively.

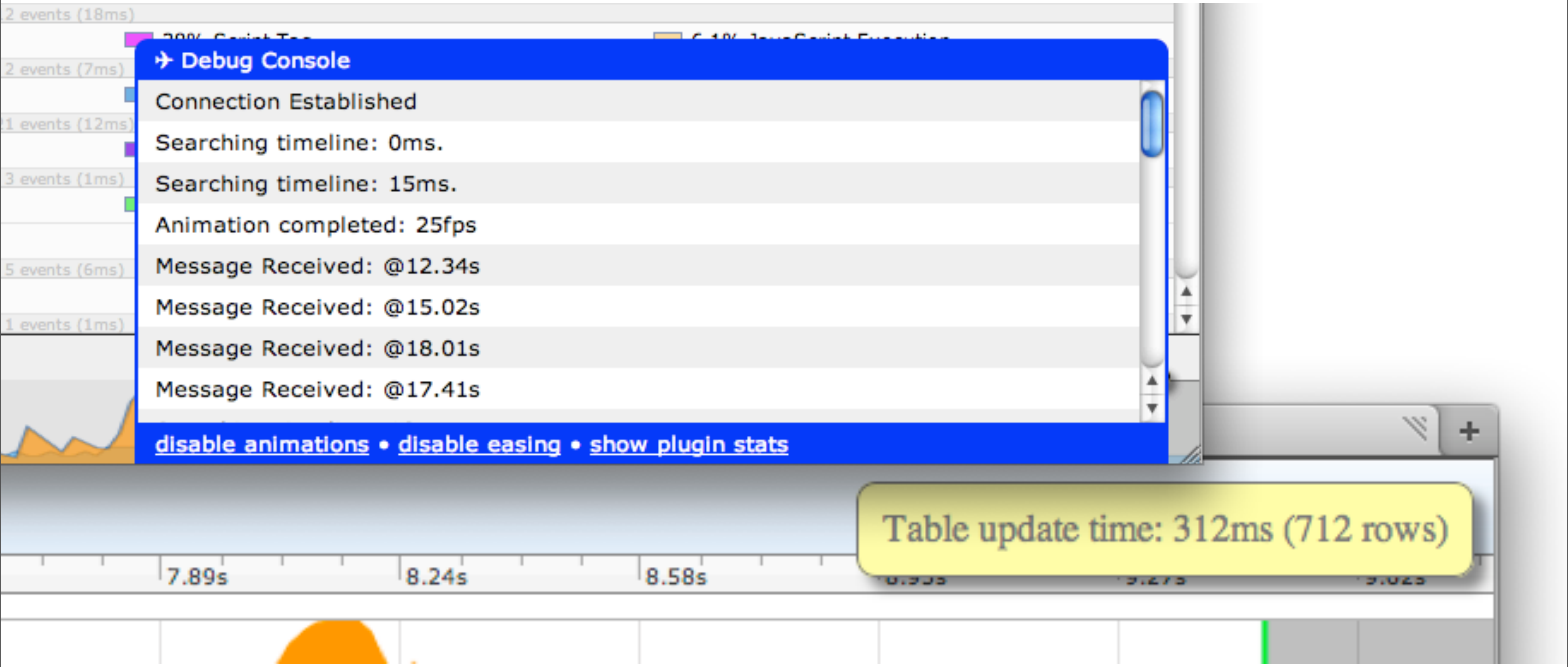
In fact, you should setup debug and release builds for your apps.

```
MyAppCommon.gwt.xml
<module>
  ...
  <define-property values="debug, release" name="app.config" />
  <replace-with class="myapp.debug.DebugConsole">
    <when-type-is class="myapp.Console" />
    <when-property-is name="app.config" value="debug" />
  </replace-with>
  ...
</module>

MyAppDebug.gwt.xml
<module>
  ...
  <set-property name="app.config" value="debug" />
</module>
```

Create this demo and
make sure it works.

#2. Use debug builds effectively.

Use debug mode to track performance constantly.

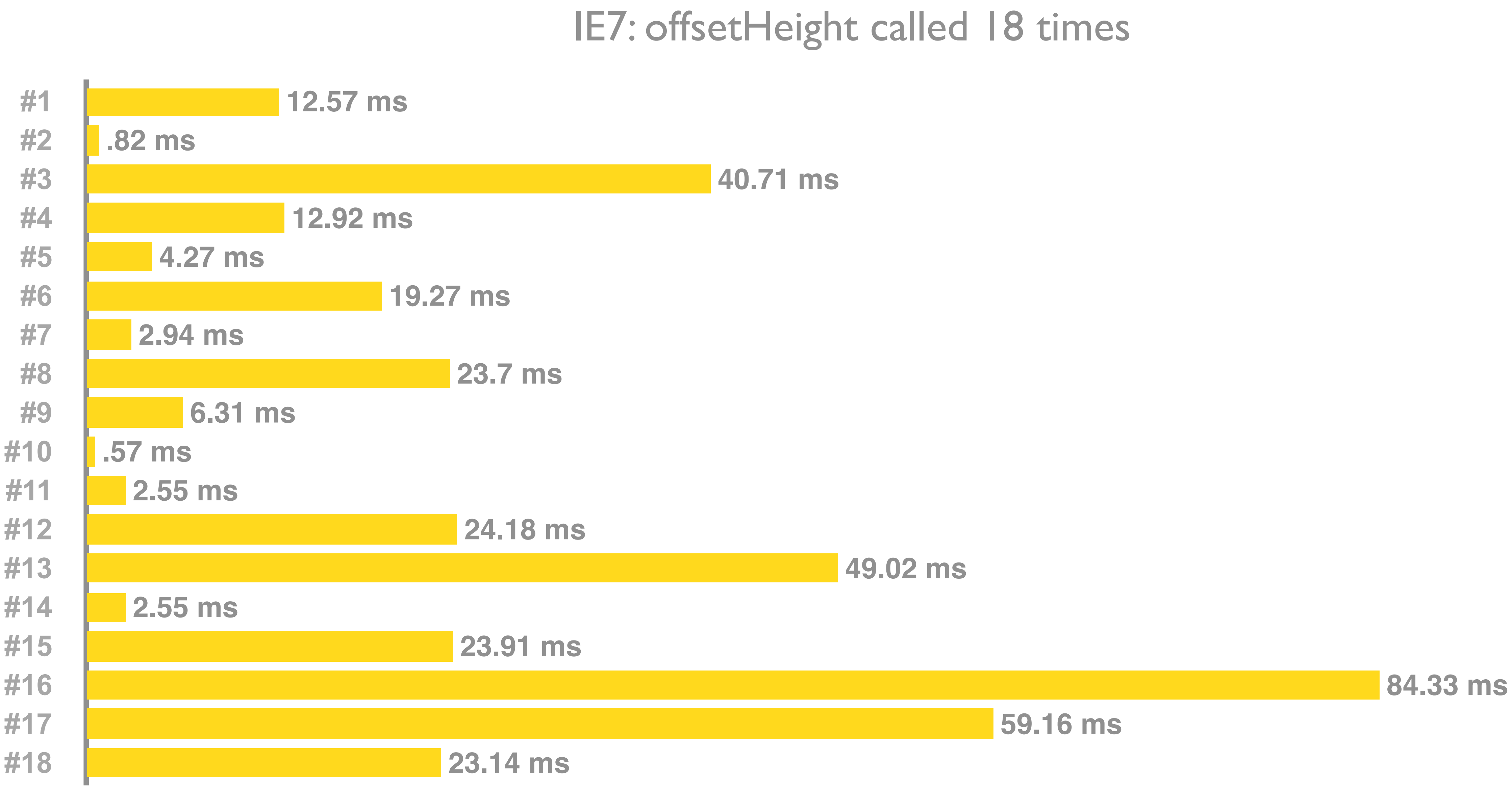


```
Connection Established
Searching timeline: 0ms.
Searching timeline: 15ms.
Animation completed: 25fps
Message Received: @12.34s
Message Received: @15.02s
Message Received: @18.01s
Message Received: @17.41s
```

Table update time: 312ms (712 rows)

#3. Avoid forcing layout.

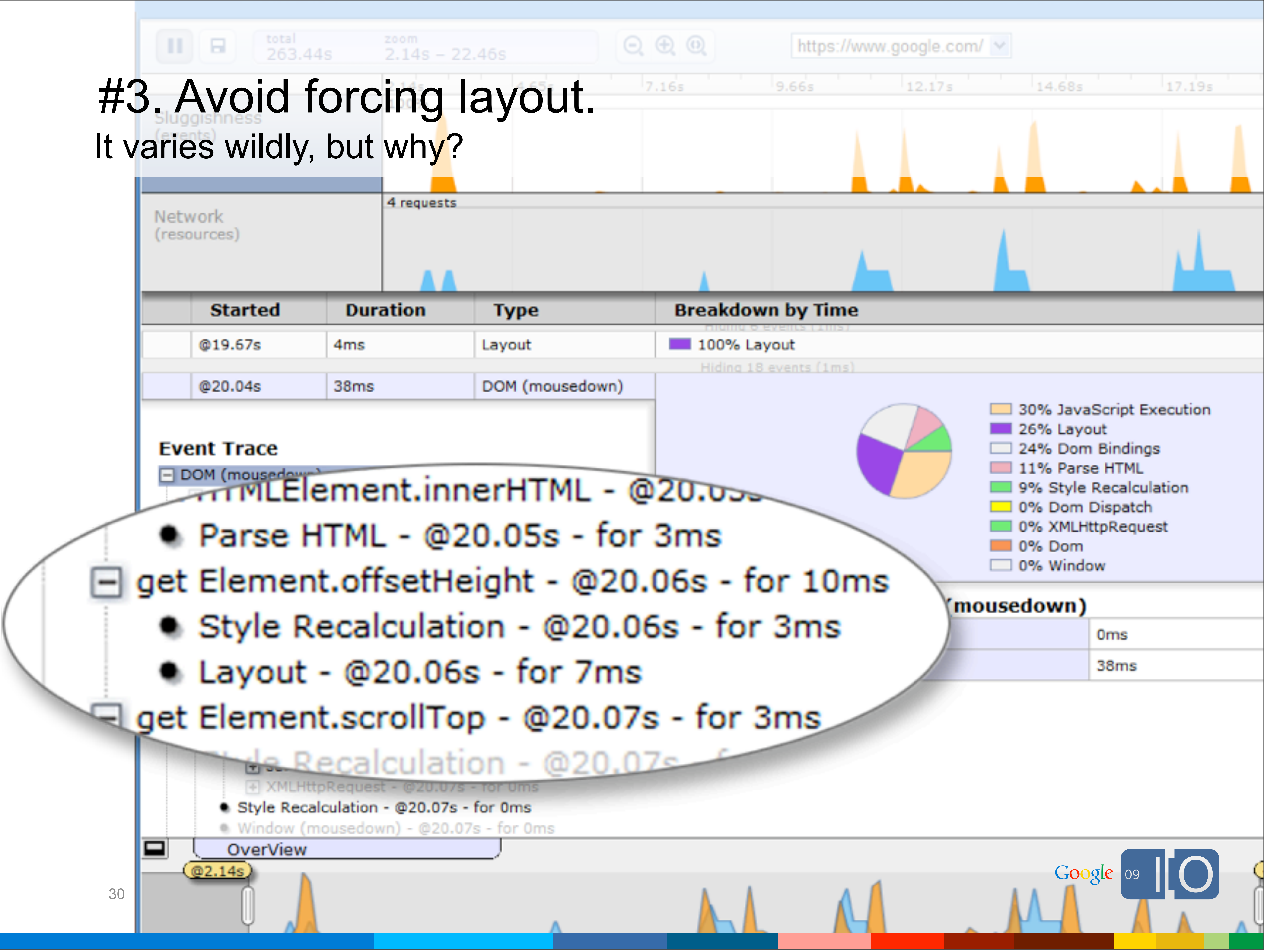
How long does it take to get an element's offsetHeight?



Data from instrumented IE7 & Google Calendar

#3. Avoid forcing layout.

It varies wildly, but why?



Started	Duration	Type	Breakdown by Time
@19.67s	4ms	Layout	100% Layout
@20.04s	38ms	DOM (mousedown)	

Hiding 18 events (1ms)

- 🟧 30% JavaScript Execution
- 🟪 26% Layout
- ⬜ 24% Dom Bindings
- 🟥 11% Parse HTML
- 🟩 9% Style Recalculation
- 🟨 0% Dom Dispatch
- 🟩 0% XMLHttpRequest
- 🟧 0% Dom
- ⬜ 0% Window

Event Trace

⊟ DOM (mousedown)

HTMLElement.innerHTML - @20.05s

- Parse HTML - @20.05s - for 3ms

⊟ get Element.offsetHeight - @20.06s - for 10ms

- Style Recalculation - @20.06s - for 3ms
- Layout - @20.06s - for 7ms

⊟ get Element.scrollTop - @20.07s - for 3ms

- Style Recalculation - @20.07s

⊞ XMLHttpRequest - @20.07s - for 0ms

- Style Recalculation - @20.07s - for 0ms
- Window (mousedown) - @20.07s - for 0ms

(mousedown)

	0ms
	38ms

#3. Avoid forcing layout.

Solution: Avoid layout from JavaScript.

Try to do what you what you need to with CSS alone.

Do style updates and measurements together:

```
Avoid:
widgetA.setStyleName(“a”);
int leftA = widgetA.getAbsoluteLeft();
widgetB.setStyleName(“b”);
int leftB = widgetB.getAbsoluteLeft();

Better:
widgetA.setStyleName(“a”);
widgetB.setStyleName(“b”);
int leftA = widgetA.getAbsoluteLeft();
int leftB = widgetB.getAbsoluteLeft();
```

#4. Fetch only what you show with RPC

GWT RPC handles arbitrary object graphs, but you rarely need that.

Often see apps where time to deserialize data structures bogs down startup. :-(

Return only what is visible to the user.

#4. Fetch only what you show with RPC

Omit large fields from your client data objects.

```
class Article implements Serializable {
  String key;
  String title;
  Date publishedOn;
  String content; // relatively large.
}

class RemoteService {
  Article[] getArticles();
}
```

When you make a list of Articles, you have to download all the content.

#4. Fetch only what you show with RPC

Omit large fields from your client data objects.

```
class Article implements Serializable {
  String key;
  String title;
  Date publishedOn;

  void getContent(AsyncCallback<String> callback);
}

class RemoteService {
  Article[] getArticles();
  String getArticleContent(String key);
}
```

that’s it. Questions?

4 things GWT does for you.
#1. Produces fast JavaScript.
#2. Loads your non-code resources quickly.
#3. Splits your code elegantly.
#4. Interoperates with JavaScript at no cost.

4 things you should do for youself.
#1. Avoid unnecessary widgets.
#2. Use debug builds effectively.
#3. Avoid forcing layout.
#4. Fetch only what you show with RPC.

InspectorWidget:
http://gwt-instrumental.googlecode.com/svn/latest/inspectorwidget/index.html

Google™ 09 I O