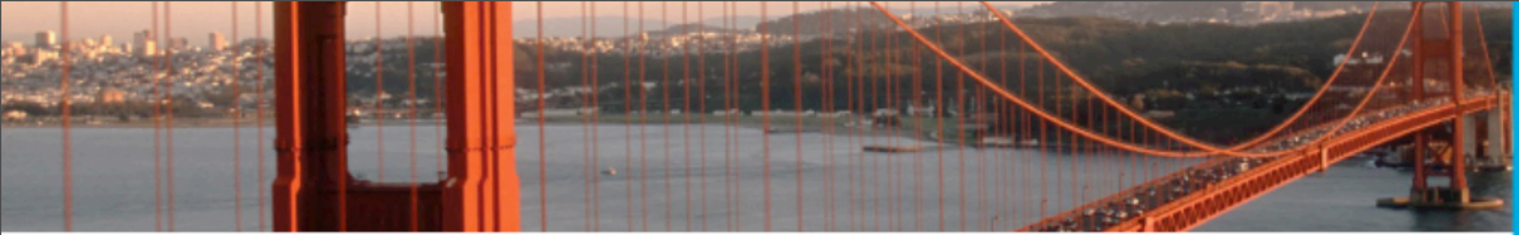


VOICES THAT MATTER CONFERENCE

GOOGLE WEB TOOLKIT

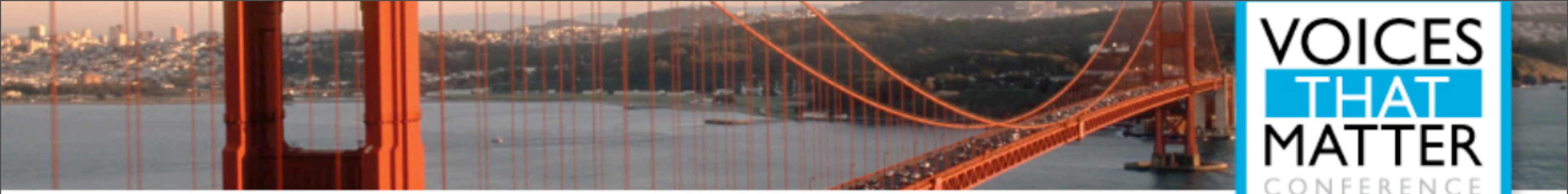
fast is better than slow

Google mantra. We will always prefer fast over "convenient for developers".

VOICES THAT MATTER CONFERENCE

GOOGLE WEB TOOLKIT

don’t forget this is a web browser



VOICES THAT MATTER CONFERENCE

GOOGLE WEB TOOLKIT

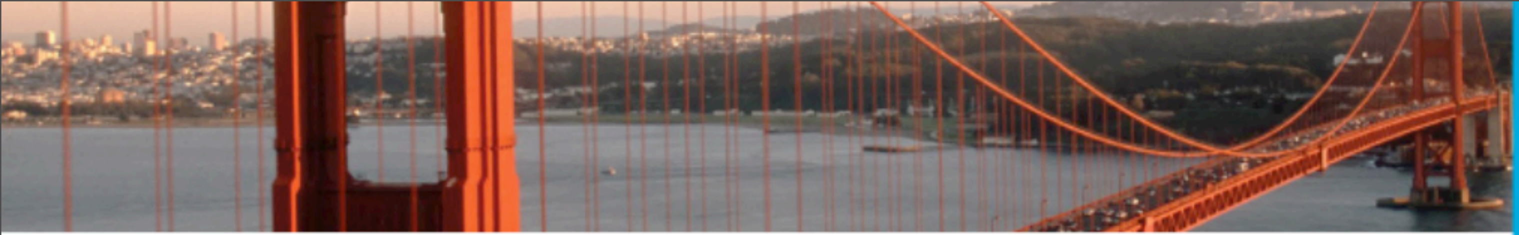
javascript is **not** fast!

The interpreter alone is on the order of **2 orders of magnitude slower** than native code

$\cong$

?

That's just the reality of the situation, and it's not changing any time soon.
On my 2.3 GHz MacBook Pro, I can run an empty 10,000,000 iteration for loop in ~2 s.
That's over 100 cycles to do essentially nothing.



dom manipulation is even slower

how long to add an item to a list box?

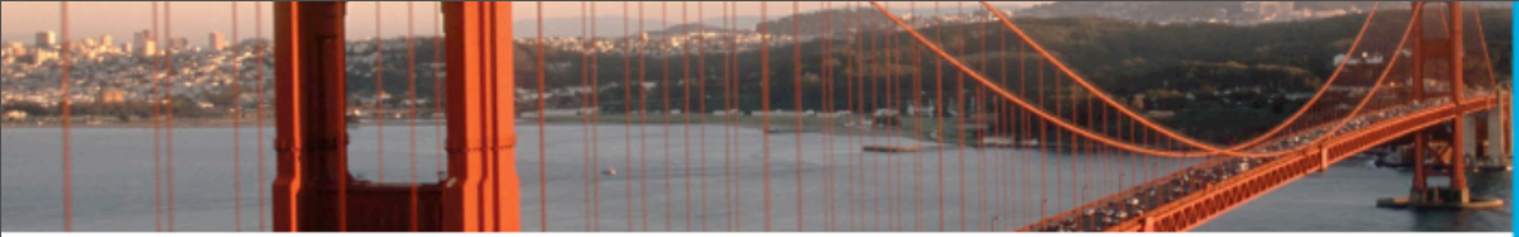
200 µs

(that’s 460,000 cycles on my laptop)

perception is all that matters

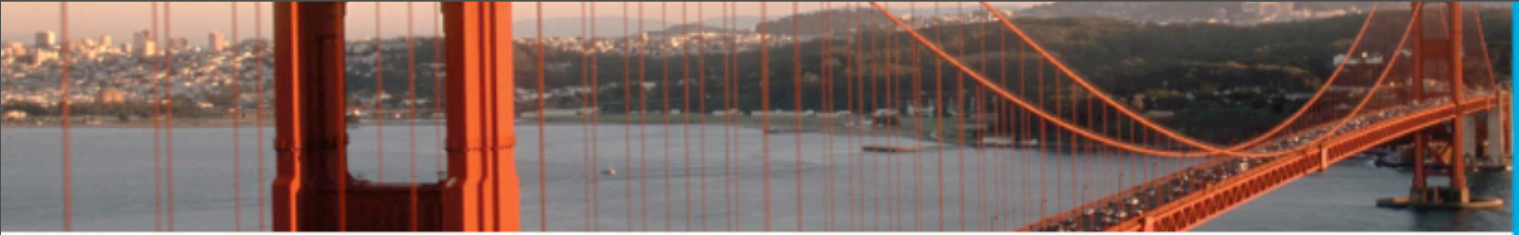
*All that matters is what the user *feels*. This gives us lots of wiggle room!*

startup time



"Every web usability study I have conducted since 1994 has shown the same thing: Users beg us to sped up page downloads."

- Jakob Nielsen



code size

cached? ⟶ download ⟶ parse ⟶ execute

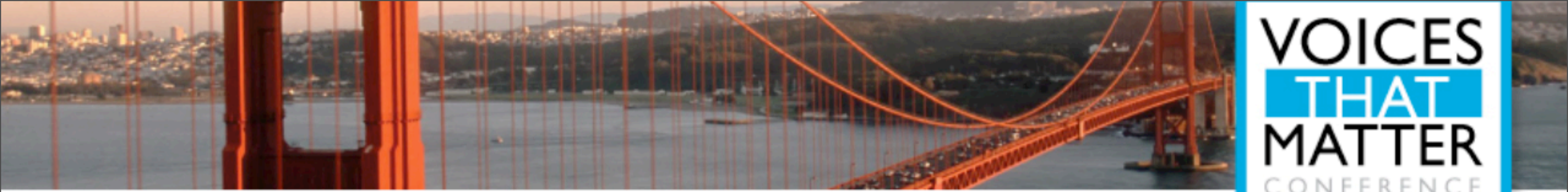
cache your scripts

gzip your scripts

Code size matters, but less than you may think.
*Download time is important, but make sure you cache large scripts *forever* (GWT does this automatically).*
*Parse time is damned near irrelevant (hard to even *measure* parse time, it's so fast).*
The GWT compiler truly and safely obfuscates, which is only safe in a statically-typed language (note all the warnings about js obfuscation errors).

GWT allows you to cache large scripts forever.
*Yahoo study: ~50% of *users* with empty cache; ~20% of *page views* with empty cache. Know your use patterns.*

*You can *always* gzip standard GWT script output.*
One reason we throw the code into HTML in an iframe (to work around IE gzip bug).



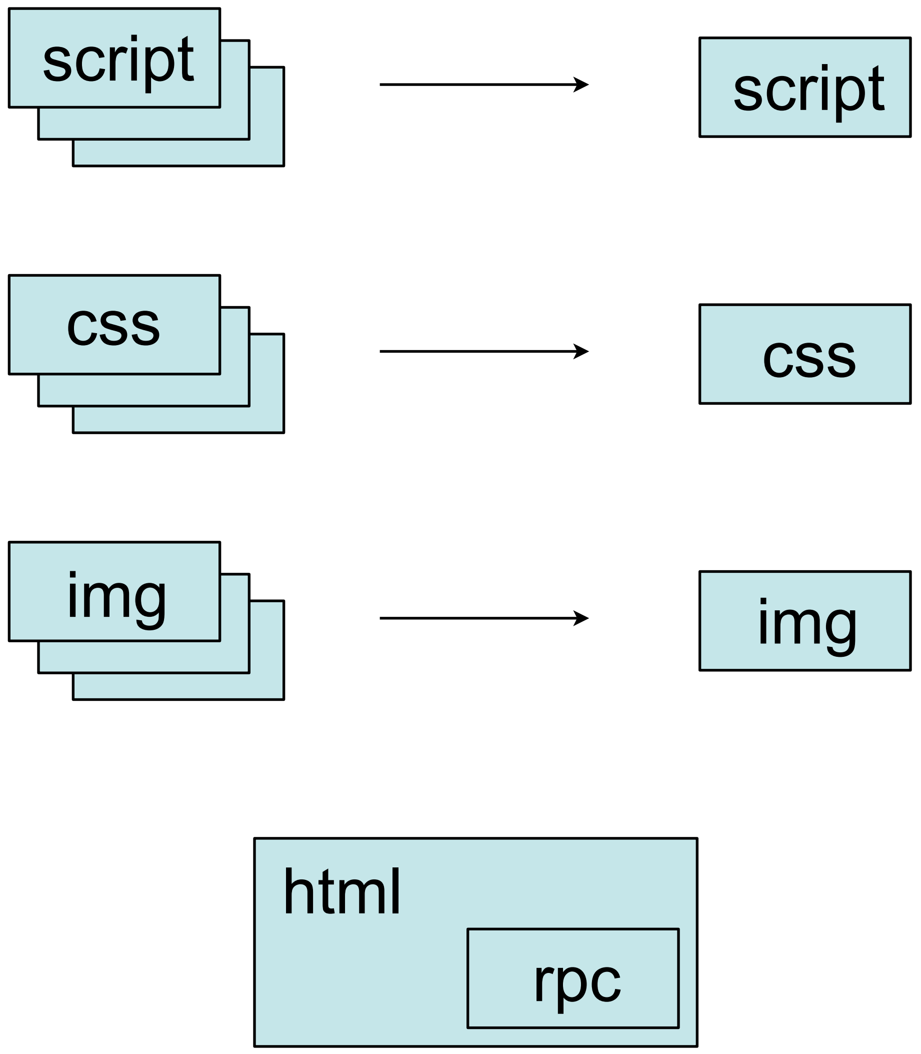
VOICES THAT MATTER CONFERENCE GOOGLE WEB TOOLKIT

http requests

the slowest thing you will ever do in the browser

html styles images scripts rpc

*HTTP requests *will eat your startup time for breakfast*.*
Minimize them at all costs; this includes external js files, css files, and rpc requests.

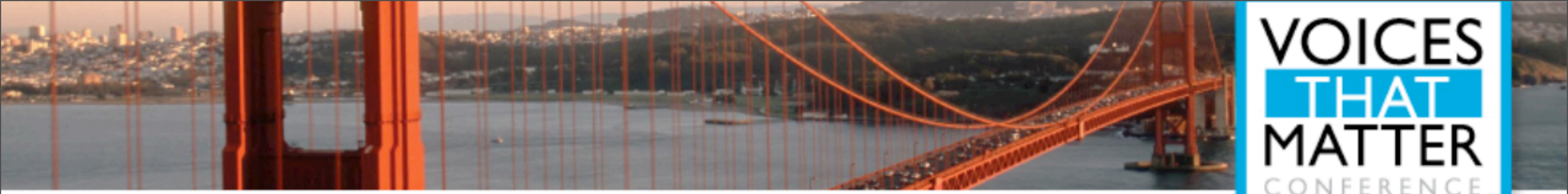


Don't load multiple stylesheets; aggregate them into one request.

If your app needs data to get started, consider pre-loading it into the outer HTML.

Images tend to generate lots of small requests (especially toolbars and such). Use ImageBundle to automatically 'sprite-ify' your images.

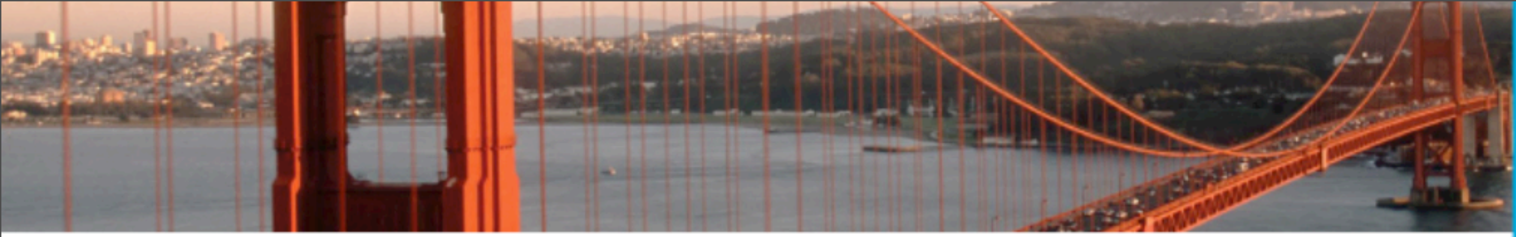
*HTML evaluation model *requires* scripts to be executed serially (defer attribute only exists on IE).*
GWT 1.5 (CHECK THIS) will allow external js to be slurped into the compiled script.



get started early

When does `EntryPoint::onModuleLoad()` run?

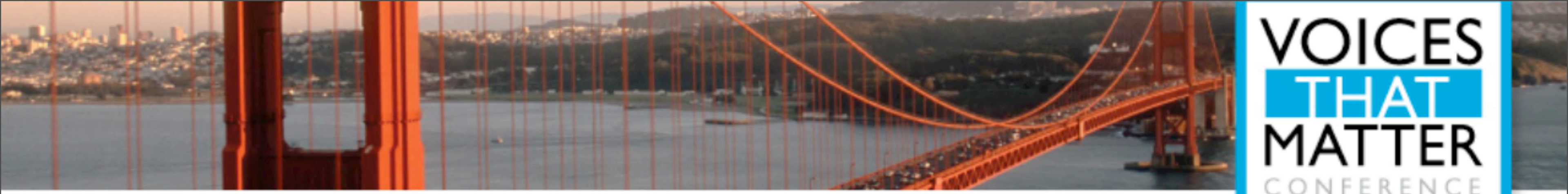
body.onload() only fires once all of your resources are loaded. This can make a huge difference if you wait until then to run your script.



profile the network

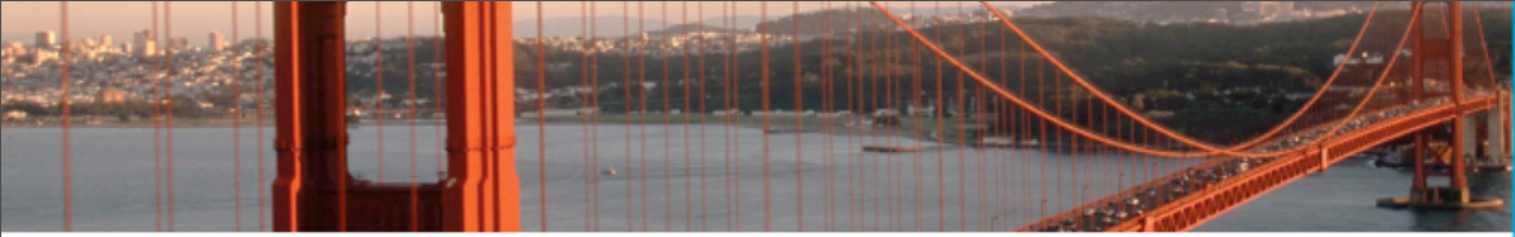
http://xk72.com/charles/

Charles (or any other logging HTTP proxy) is your friend.
It’s really hard to have a sense for what’s happening on the network unless you actually look at it.



runtime

Once your app gets started, it needs to feel snappy. If not, then you might as well go back to using static HTML.



lazy ui creation

The fastest code is that which never executes

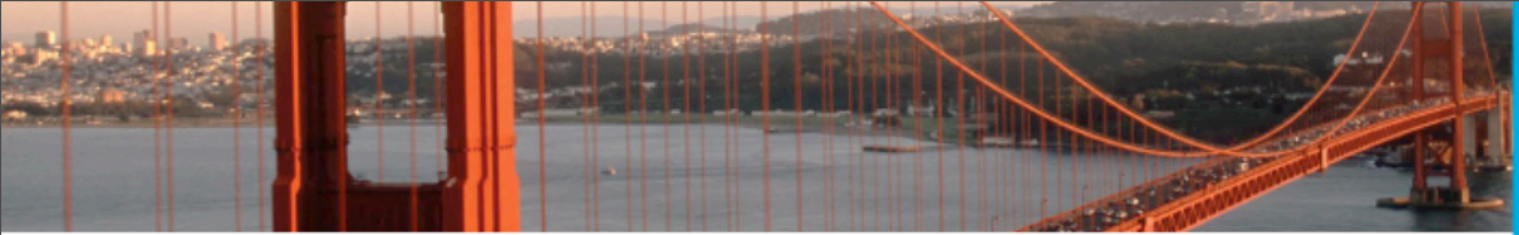
```
public class LazyDataView extends Composite {

  private Panel p = new VerticalPanel();
  private Label header = new Label("My Stuff:");
  private HeavyFreakingView dataView;

  public MyComposite() {
    initWidget(p);
  }

  public void showData(DataObject dto) {
    if (dataView == null) {
      p.add(dataView = new HeavyFreakingView());
    }
    dataView.showStuff(dto);
  }
}
```

Defer creation of Widgets and Elements until they are actually needed.

use innerHTML

```
String id = HTMLPanel.createUniqueId();
String html = "<table>" +
  "  <tr><td>Header</td></tr>" +
  "  <tr><td id='" + id + "'/></tr>" +
  "</table>";
HTMLPanel p = new HTMLPanel(html);
p.add(new MyComposite(), id);
```

It’s generally faster to use innerHTML (over direct DOM manipulation) when possible.
A number of our widgets are going to be optimized to take advantage of this.



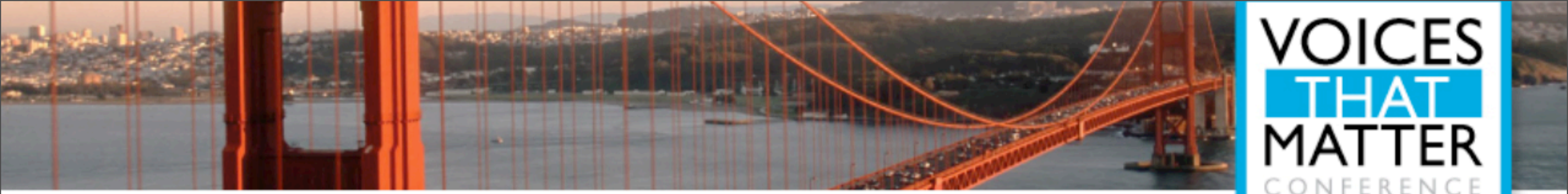
use stylesheets

```
setStyleName("foo");
```

-vs-

```
DOM.setStyleAttribute("color", "red");
DOM.setStyleAttribute("border", "1px");
...
```

Using stylesheets (as opposed to direct property sets) gives you:
- less code
- faster code
- style-ability



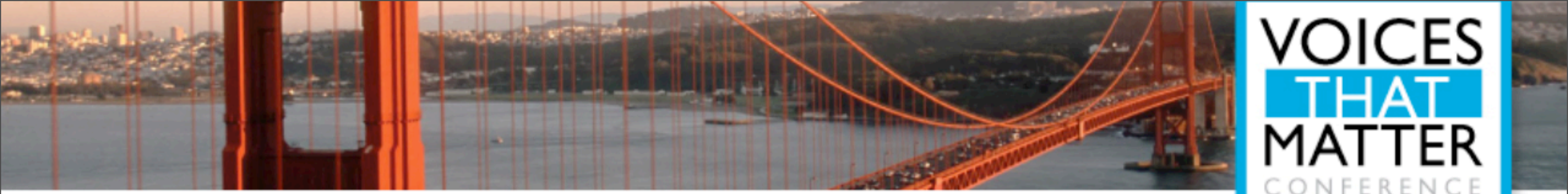
don’t block the ui thread

```
DeferredCommand.addCommand(new Command() {
  public void execute() {
    // Do heavy stuff later.
  }
});
```

*Remember that there's only one thread for the *entire* UI.*

Don't do too much in an event handler

(Hint: All code runs in event handlers).

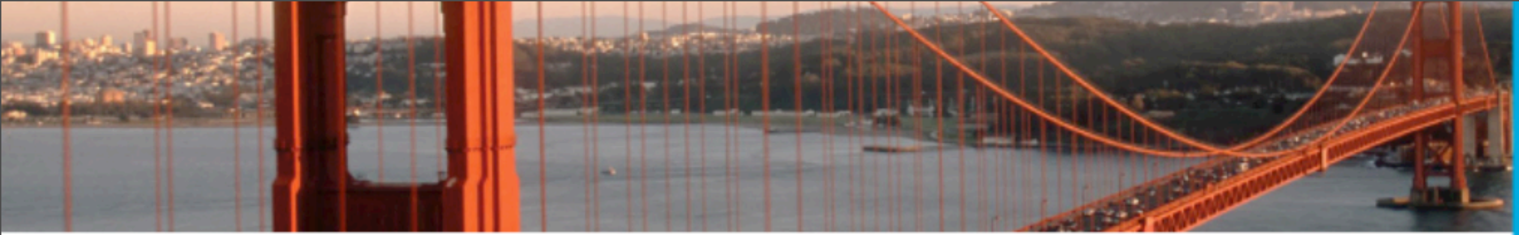


fetching data

minimize http requests

cache data

Too many requests can kill your runtime performance just as easily as it can your startup time.
Using Java objects for your data model makes it much easier to cache data.
Need to be careful about how you handle invalidation.



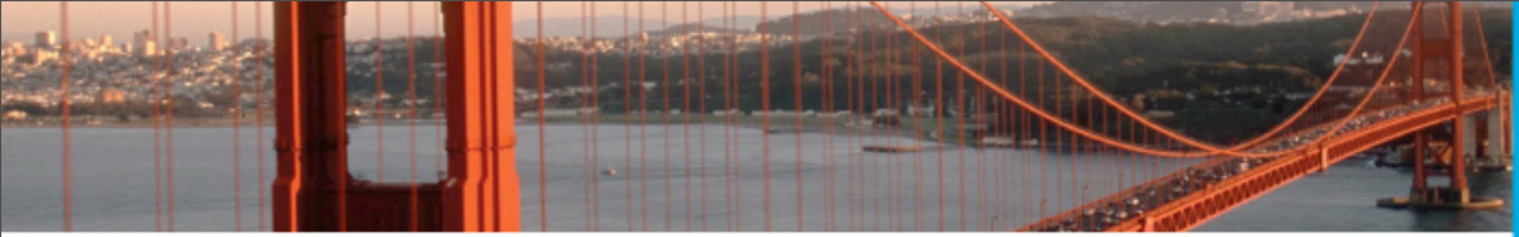
aggregate rpc calls

```
public interface MyService {
  String login(String user, String pass);
  Settings getSettings(String authToken);
  Contact[] getContacts(String authToken);
}
```

You could login, then ask for the initial settings and contacts separately, but you **know** you're going to do all three together.

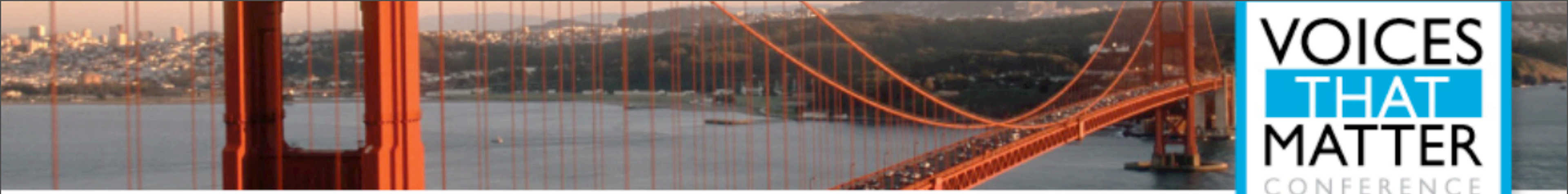
```
public class LoginResult {
  public String authToken;
  public Settings settings;
  public Contacts[] contacts;
}

public interface MyService {
  LoginResult login(String user,
                    String pass);
}
```

So just aggregate them into a single call. GWT RPC makes this easy.



compiler optimizations

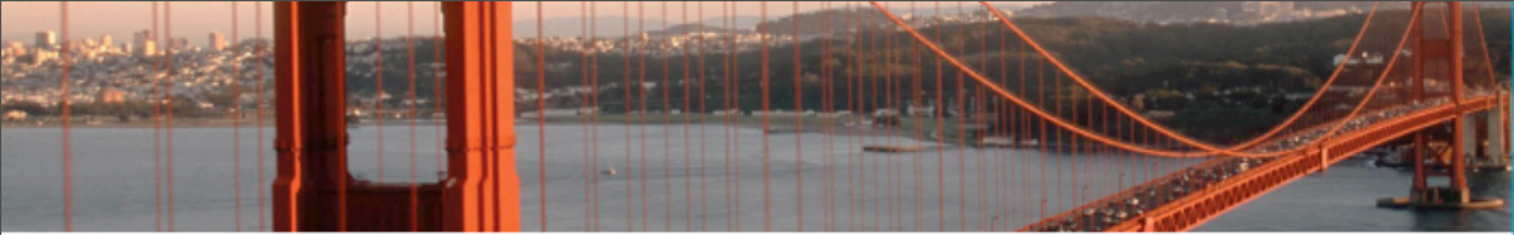
“making abstraction affordable since 1949”



java source

```
class Button {
  public void setText(String text) {
    DOM.setInnerText(getElement(), text);
  }
}
class Hello {
  public void onModuleLoad() {
    Button b = new Button();
    b.setText("w00t!");
  }
}
```

Using abstractions like Widgets, interfaces, and such is really important.
Necessary for engineering leverage over a large code base.



generated code: webkit

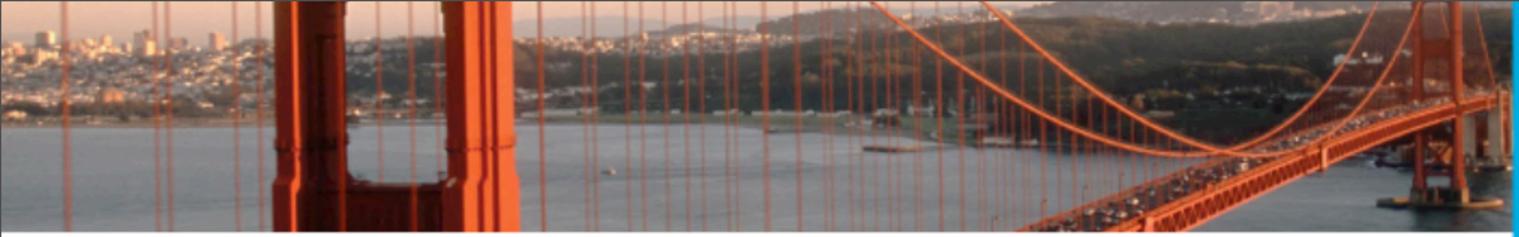
```
function $onModuleLoad(){
  var b;
  b = $Button(new Button());
  $setInnerText(b.element, 'w00t!');
  $add(($clinit_28() , get(null)), b);
}
```

But there’s no reason for your abstractions to live on in the generated code.
Generate the best possible code for each use case (in this case, WebKit).

generated code: ie

```
function $onModuleLoad(){
  var b;
  b = $Button(new Button());
  b.element.innerText = 'w00t!' || '';
  $add(($clinit_28() , get(null)), b);
}
```

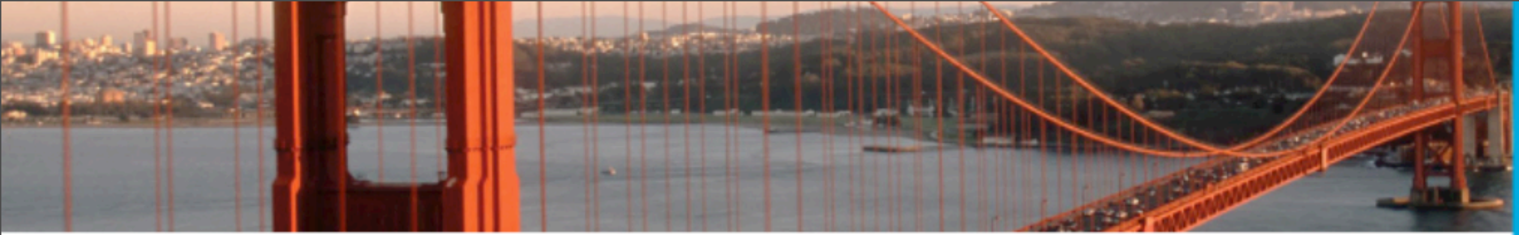
In this case, IE.



pay as you go

there is no ‘gwt library’

Aggressively deadstrip code!
Large libraries are only affordable if you can do this.



VOICES THAT MATTER CONFERENCE
GOOGLE WEB TOOLKIT

questions?