ORACLE

A New Approach to the UI for Distributed Applications

Ed Burns
JavaEE UI Spec Lead
edward.burns@oracle.com
@edburns

DEVNEXUS

My Plan for Your Time Investment

- Distributed App UI History
- HTML5: What's in a name?
  - A bit of JSF Apologetics
- JavaScript/HTML5 Rich Client Landscape
- Java EE 7
- Java EE + JavaScript

Speaker Qualifications – Ed Burns

And non-qualifications

- Involved with JSF since 2002
- Spec lead since 2003
  - Most fun part of the job: cleanly integrating other people's great ideas into JSF (and hopefully improving on the in the process)
  - Not an expert in applying JSF in practice
- Author of four books for McGraw-Hill

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

HTML5

Why all the fuss?

More cute logos at http://www.w3.org/html/logo/

HTML5

Why all the fuss?

 | |

Classification

A Web App is a Distributed App

- Why?
  - Multiple Computers
  - Interconnections Between Them
  - Shared State Among Them
- Today's production Web apps are extremely complex distributed applications.

Yeah, So What?

- Why does this classification matter?
  - Because History Matters
- To understand the current state of web applications, we must go back to the history of distributed applications, and of the Internet itself.

What Makes a Distributed App

- Finding the best allocation of processing tasks to processing nodes
  - User Interface
  - Domain Logic
  - Application Logic
  - Data Persistence
  - Communication
  - Reliability, Security

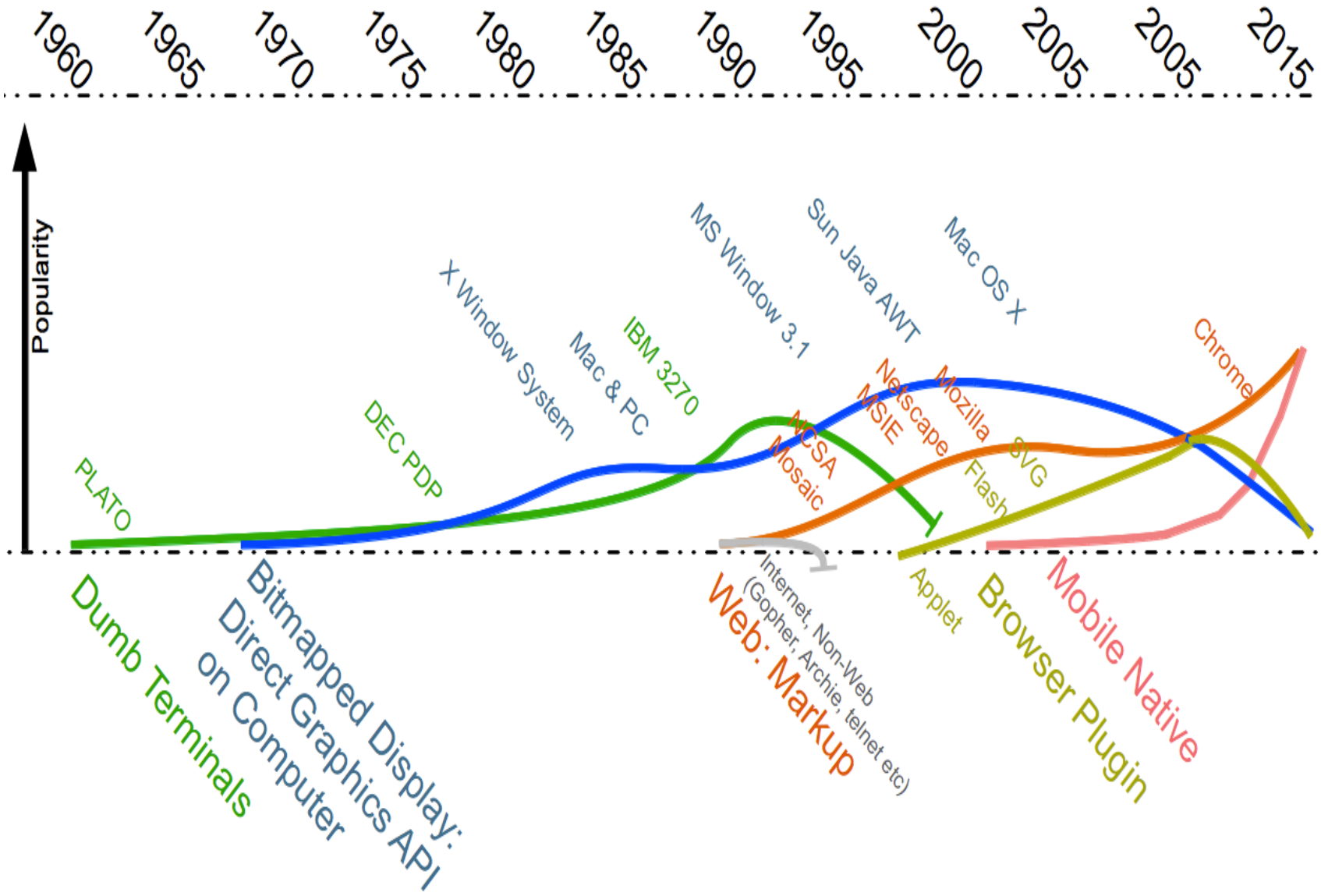
What Makes a Distributed App

- Finding the best allocation of processing tasks to processing nodes
  - **User Interface**
  - Domain Logic
  - Application Logic
  - Data Persistence
  - Communication
  - Reliability, Security

UI Considerations

- The UI is the hardest part to get right.
- The technology for building the UI is changing very rapidly, and will continue to change for the foreseeable future.
- The technology for the other aspects of application development is less volatile, more mature.
- The major software stack and device vendors are competing on the basis of their UI technology, as the gateway to the rest of their stack.

Distributed App UI History



What's in a name?

Ajax

- Remember all the fuss about Ajax in 2006?
  - Asynchronous
  - JavaScript
  - And
  - XMLHttpRequest

What's in a name?

Ajax

- Remember all the fuss about Ajax in 2006?
  - Asynchronous
  - JavaScript
  - And
  - XMLHttpRequest
- Ajax is a programming technique, not a single technology

What's in a name?

HTML4

- What do people mean when they say HTML4?
  - IE6
  - Not very high performance JavaScript
  - Lots of browser tricks
  - Use of native plugins is common
- HTML4 **is** seen as a single technology

What's in a name?

HTML5

- What do people mean when they say HTML5?
  - "Modern" browsers
  - A gigantic collection of related technologies
    - Markup
    - Offline storage
    - EventSource
    - DOM
    - JavaScript
    - CSS3
    - Canvas
    - Input controls
    - Web components
    - Application Cache
    - WebSocket
    - JSON
    - Geolocation
    - XMLHttpRequest Level 2
- HTML5 is more a marketing term than a single technology

HTML5

Is it really a big deal?

- The rise of Chrome and the end of polyfill
- Standards have finally won
  - How good is your standards body?
    - W3C, ECMA, IETF
  - HTML5: Microsoft, Google, Apple, what about Mozilla?
- Aside:
  - Is HTML5 a bloated specification?
  - Is JavaEE a bloated specification?
  - What is bloat? A indicator of how old something is.
  - http://mir.aculo.us/2010/10/19/standards-bloat-and-html5/

HTML5

Is it really a big deal?

- The death of the browser plugin: April 2010
  http://www.apple.com/hotnews/thoughts-on-flash/
- Where does the tension remain?
  - Take advantage of the power in the client
  - Minimize the complexity of distributing and maintaining the software

What's Hot Now?

- HTML5 native applications
  - JavaScript MV* frameworks
  - REST
- Dan North, thought leader
  - "The Browser is Dead…"
    - http://bit.ly/DanNorthBrowserIsDead PDF
    - http://bit.ly/DanNorthBrowserIsDeadVideo YouTube

Dan North's Assessment of What's Hot Now

- Technologies
  - Graphics: 2D and 3D + transforms
  - Client local storage
  - Sever Sent Events: Web Sockets
  - Ecmascript
- Techniques
  - Everything is asynchronous
  - Don't page template, just use the DOM (jQuery)
  - No UI state on the server

Dan North's Assessment of What's Hot Now

- Use standards…
  - W3C standards
  - JCP standards?
    - JSON JSR-353
    - WebSocket JSR-356
    - JAX-RS JSR-339

So why are these things hot now?

- Flashy results?
- Maintainability?
- Better runtime performance ➔ *potential* for better user experience?
- Wider reach?

So why are these things hot now?

JSF responses

- Flashy results?
  - This is a component library concern. Many component libraries have very flashy components.
  - Abstractions endure
- Maintainability?
- Better runtime performance ➔ *potential* for better user experience?
- Wider reach?

So why are these things hot now?

JSF responses

- ~~Flashy results?~~
- Maintainability?
  - JavaEE/JSF was designed for large teams of corporate developers producing code that needs to stick around long after said developers have moved on.
  - For example, emphasis on statically typed technologies
- Better runtime performance ➔ *potential* for better user experience?
- Wider reach?

So why are these things hot now?

JSF responses

- ~~Flashy results?~~
- ~~Maintainability?~~
- Better runtime performance ➔ *potential* for better user experience?
  - With any technology, including the HTML5 native approach, it is possible to produce a poorly performing user experience. The question is how hard is it to produce a decently performing one.
  - Stateless JSF is a step in that direction.
- Wider reach?

So why are these things hot now?

JSF responses

- ~~Flashy results?~~
- ~~Maintainability?~~
- ~~Better runtime performance ➔ *potential* for better user experience?~~
- Wider reach?
  - JSF was designed for client device independence
  - HTML5, while growing, is still not at the least common denominator level
  - Being able to support IE 6 is sometimes still important

So why are these things hot now?

JSF responses

- ~~Flashy results?~~
- ~~Maintainability?~~
- ~~Better runtime performance ➔ *potential* for better user experience?~~
- ~~Wider reach?~~

In reality, many approaches can co-exist

- JSF for one class of developers/users
- HTML5 native for another
- Re-use in the application tier

HTML5

Putting it in context

- HTML5 is a marketing term that describes a way of building the UI for a distributed system.
  - UI processing task resides entirely in the browser

Review: The Lifecycle Orchestrates MVC
The Baseball and Apple Pie of Web apps

- Data Conversion and Validation
- Page Flow
- Database integration

- I18N, L10N, A11Y
- Support CSS, Markup based layout
- User Friendliness!

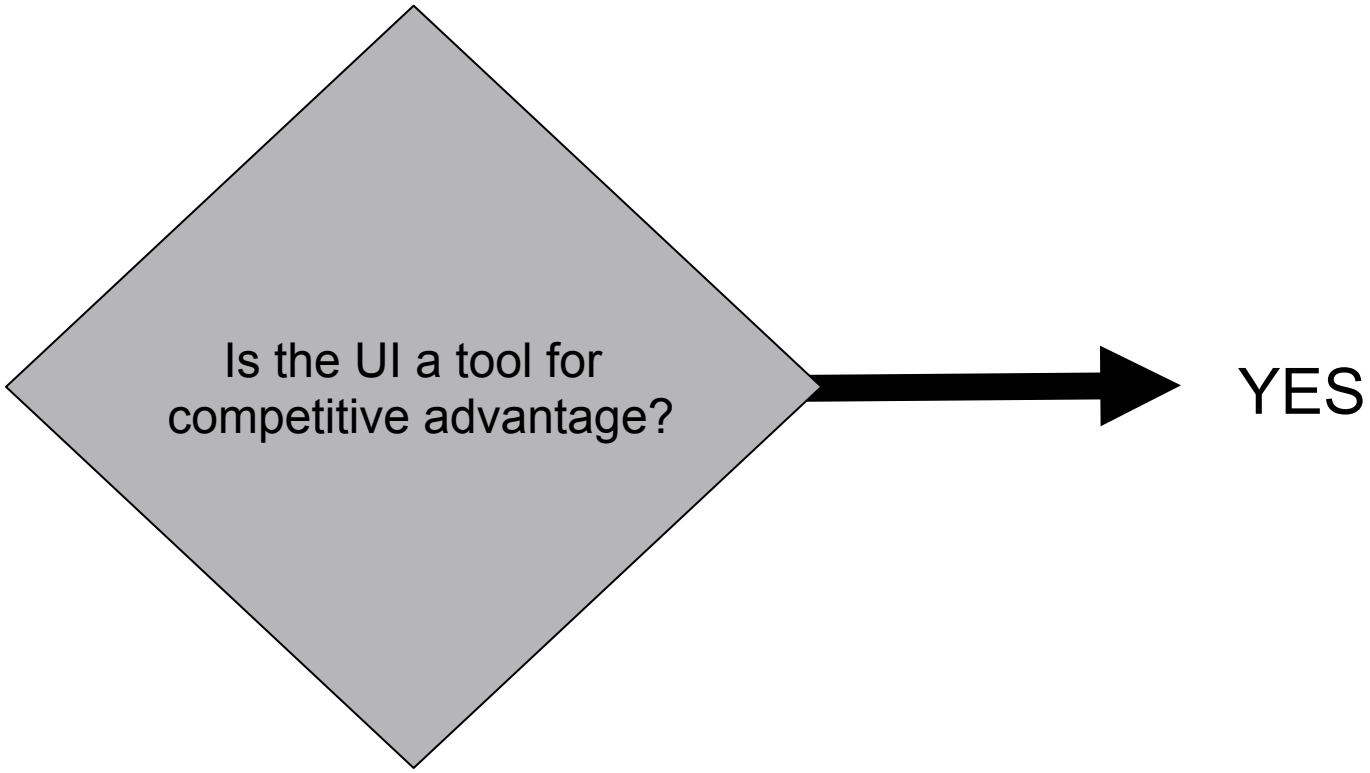
Review: The Lifecycle Orchestrates MVC
The Baseball and Apple Pie of Web apps

- Data Conversion and Validation
- Page Flow
- Database integration
- Maintainability

- I18N, L10N, A11Y
- Support CSS, Markup based layout
- User Friendliness!
- Ease of developer on-boarding
- Developer productivity
- Industry coding standards

What does an enterprise want in a UI?

What does an enterprise want in a UI?

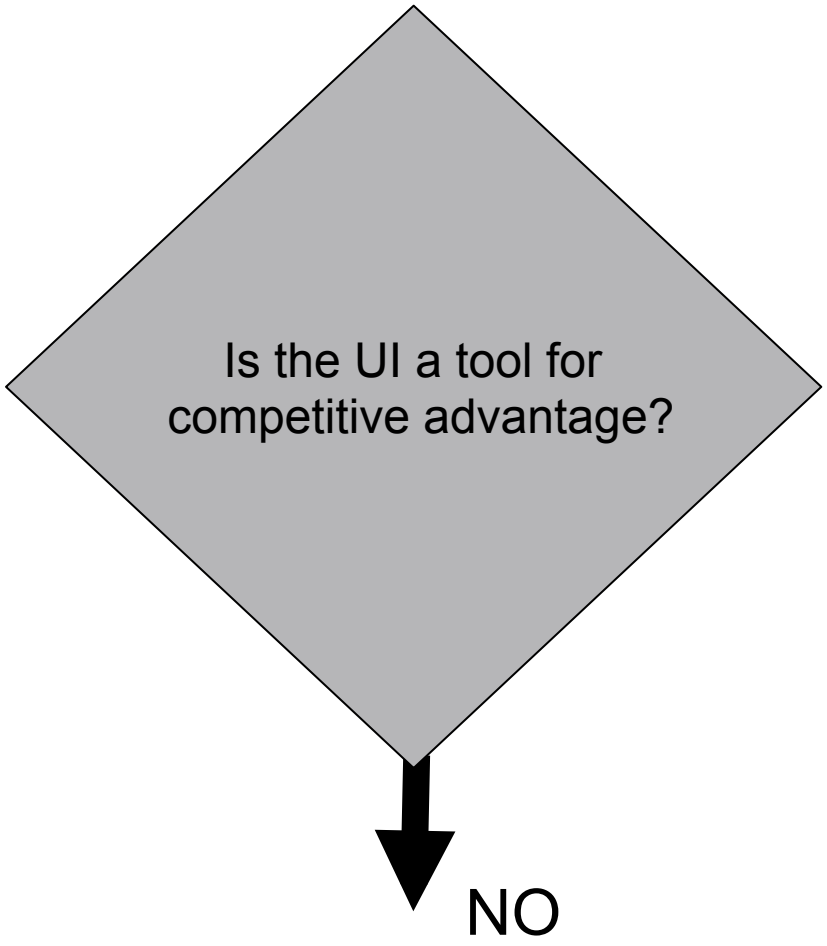


What does an enterprise want in a UI?

The UI is a tool for competitive advantage

- A beautiful UI is important when
  - B2C
  - Customer experience is important
  - The competition has a nice UI
  - The UI is external facing: Internet
  - Organizational culture places a high degree of importance on style and beauty

What does an enterprise want in a UI?



What does an enterprise want in a UI?

The UI is NOT a tool for competitive advantage

- A beautiful UI is not important when
  - (sometimes) B2B
  - Your users are all employees
  - Getting the job DONE is more important than enjoying yourself while you do it
  - The domain conforms to well understood patterns for UI
    - Data entry
    - Case management
    - Billing, Payroll, Accounts
    - Dispatching

JavaScript/HTML5 Rich Clients Rising?

- The thin client vs. rich client debate is pretty old…
- Server-side web frameworks have ruled for a while (JSF, Struts, Spring MVC)
- AJAX was a mild shift to the client (PrimeFaces, GWT, Vaadin)
- Rich clients powered by JavaScript/HTML5 appear to be making a comeback…
  - Improving JavaScript engines (V8, SpiderMonkey, Rhino, Nashorn)
  - Better tools (jQuery, MV* frameworks, Chrome, FireFox, Avatar)
  - Standards advancement (CSS3, HTML5, WebSocket, HTML Components…)

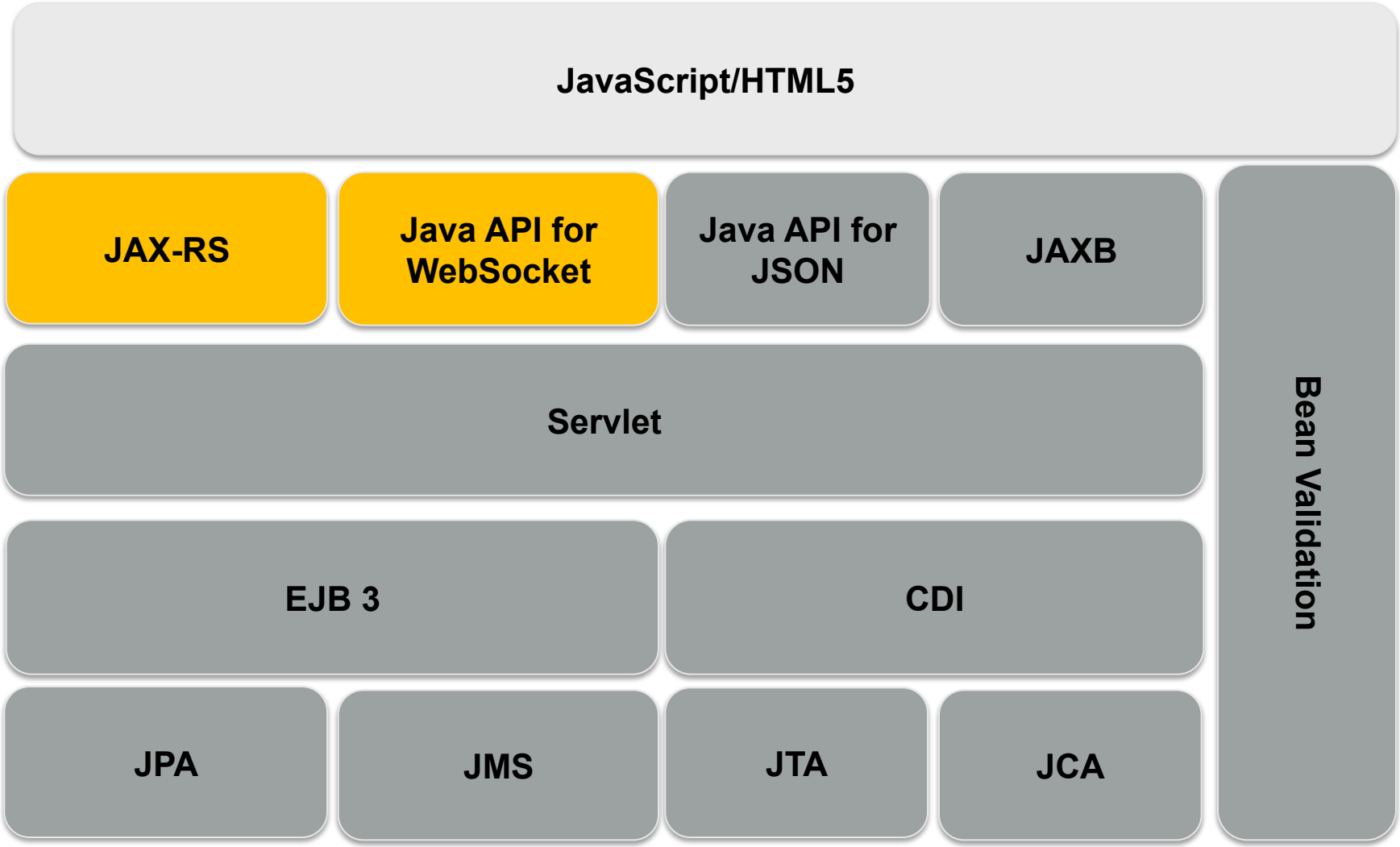
Perhaps not a Slam Dunk?

- Richer clients clearly better at some things
  - Highly dynamic, interactive interfaces
  - Complex, feature-rich UIs
  - “Single page applications” (“Applets” ☺)
- But perhaps not a panacea
  - Heavily form/workflow driven applications
  - Server-side rendering still a better bet for performance/reliability?
  - JavaScript/HTML development is not without it’s pains…
  - Server-side frameworks are a strong incumbent
- Co-existence in the short and long term?
  - Islands of rich client functionality within server-centric UIs?
  - Different strokes for different folks?

My Big Fat Rich-Client Architecture

- Very similar to client/server architecture of lore
- Client responsible for UI rendering, basic input validation, logic and state
- Server responsible for business logic, domain model, persistence
- Web/HTTP is glue that connects client and server
- Typical communication protocols
  - REST for majority of cases
  - WebSocket when full-duplex communication is needed
  - JavaScript tools support REST well, but not WebSocket (yet)
- The typical (ideal?) data interchange format is JSON
- Java EE is a great server-side platform for this architecture…

Java EE + JavaScript



JAX-RS

- REST development API for Java
- Server and client
- Annotation based, declarative
  - `@Path, @GET, @POST, @PUT, @DELETE, @PathParam, @QueryParam, @Produces, @Consumes`
- Pluggable and extensible
  - Providers, filters, interceptors

JAX-RS Example

```
@Path("/atm/{cardId}")
public class AtmService {

    @GET
    @Path("/balance")
    @Produces("text/plain")
    public String balance(
            @PathParam("cardId") String card,
            @QueryParam("pin") String pin) {
        return Double.toString(getBalance(card, pin));
    }

    ...
```

JAX-RS Example

```
    ...

    @POST
    @Path("/withdrawal")
    @Consumes("text/plain")
    @Produces("application/json")
    public Money withdraw(
            @PathParam("card") String card,
            @QueryParam("pin") String pin,
            String amount) {
        return getMoney(card, pin, amount);
    }
}
```

Java API for WebSocket

- High level declarative API for WebSocket
- Both client and server-side
- Small, powerful API
  - **`@ServerEndpoint, @OnOpen, @OnClose, @OnMessage, @OnError, Session, Remote`**
- Pluggable and extensible
  - Encoders, decoders, sub-protocols

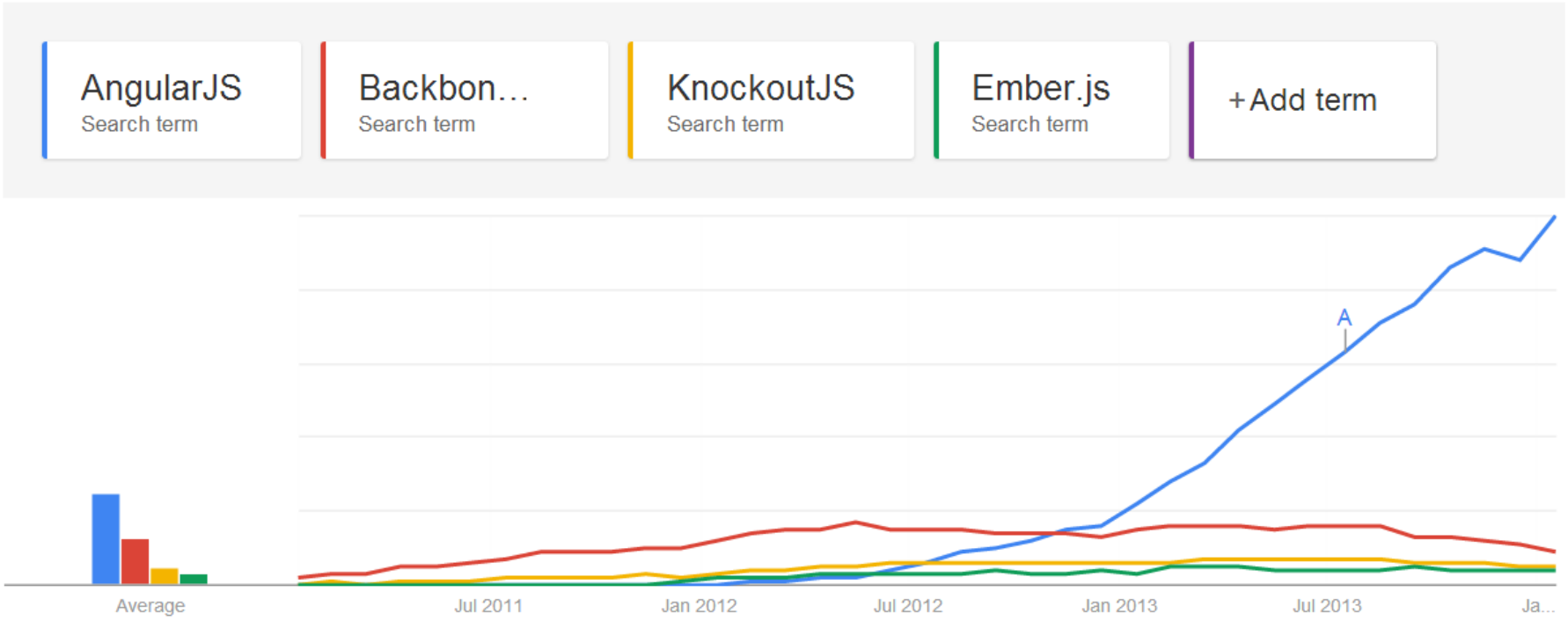
WebSocket Sample

```
@ServerEndpoint("/chat")
public class ChatBean {
    Set<Session> peers = Collections.synchronizedSet(…);

    @OnOpen
    public void onOpen(Session peer) {
        peers.add(peer);
    }

    @OnClose
    public void onClose(Session peer) {
        peers.remove(peer);
    }
    ...
```

WebSocket Sample (Continued)

```
    ...

    @OnMessage
    public void message(String message, Session client) {
        for (Session peer : peers) {
            peer.getRemote().sendObject(message);
        }
    }
}
```

JavaScript Movers and Shakers



source: Google Trends

Project Avatar

- End-to-end open source JavaScript framework from Oracle
  - JavaScript on the client and server side (can be just one or the other)
  - Uses Nashorn/JDK8
  - Utilizes some underlying Java EE capabilities
  - Runs on GlassFish, perhaps on WebLogic at some point…
- Integrated support for REST, WebSocket, Server-Sent Events (SSE)…
- PhoneGap support
- Still nascent, good time to get involved

http://avatar.java.net

Java EE + JavaScript Demo

https://github.com/m-reza-rahman/javaee-javascript

Best of Both Worlds

- AngularJS for the UI heavy lifting
- Server side ResourceBundles for Localization
- Server side templating for page modularity

Summary

- JavaScript/HTML5 clients gaining traction as opposed to server-side web frameworks
- Communication between the client and server happens via JSON over REST or WebSocket
- Java EE well positioned as a JavaScript rich client backend, especially with JAX-RS, the Java API for WebSocket and JSON-P
- JavaScript framework from Oracle - Avatar
- You can use the demo code as a starting point to exploring the emerging space
- Most importantly, have fun!

Try it Out!

http://download.java.net/glassfish/4.0/release/glassfish-4.0.zip

Resources

- Java EE Tutorials
  - http://docs.oracle.com/javaee/7/tutorial/doc/home.htm
- Digging Deeper
  - http://docs.oracle.com/javaee/7/firstcup/doc/home.htm
  - https://glassfish.java.net/hol/
  - https://java.net/projects/cargotracker/
- Java EE 7 Transparent Expert Groups
  - http://javaee-spec.java.net
- Java EE 7 Reference Implementation
  - http://glassfish.org
- The Aquarium
  - http://blogs.oracle.com/theaquarium