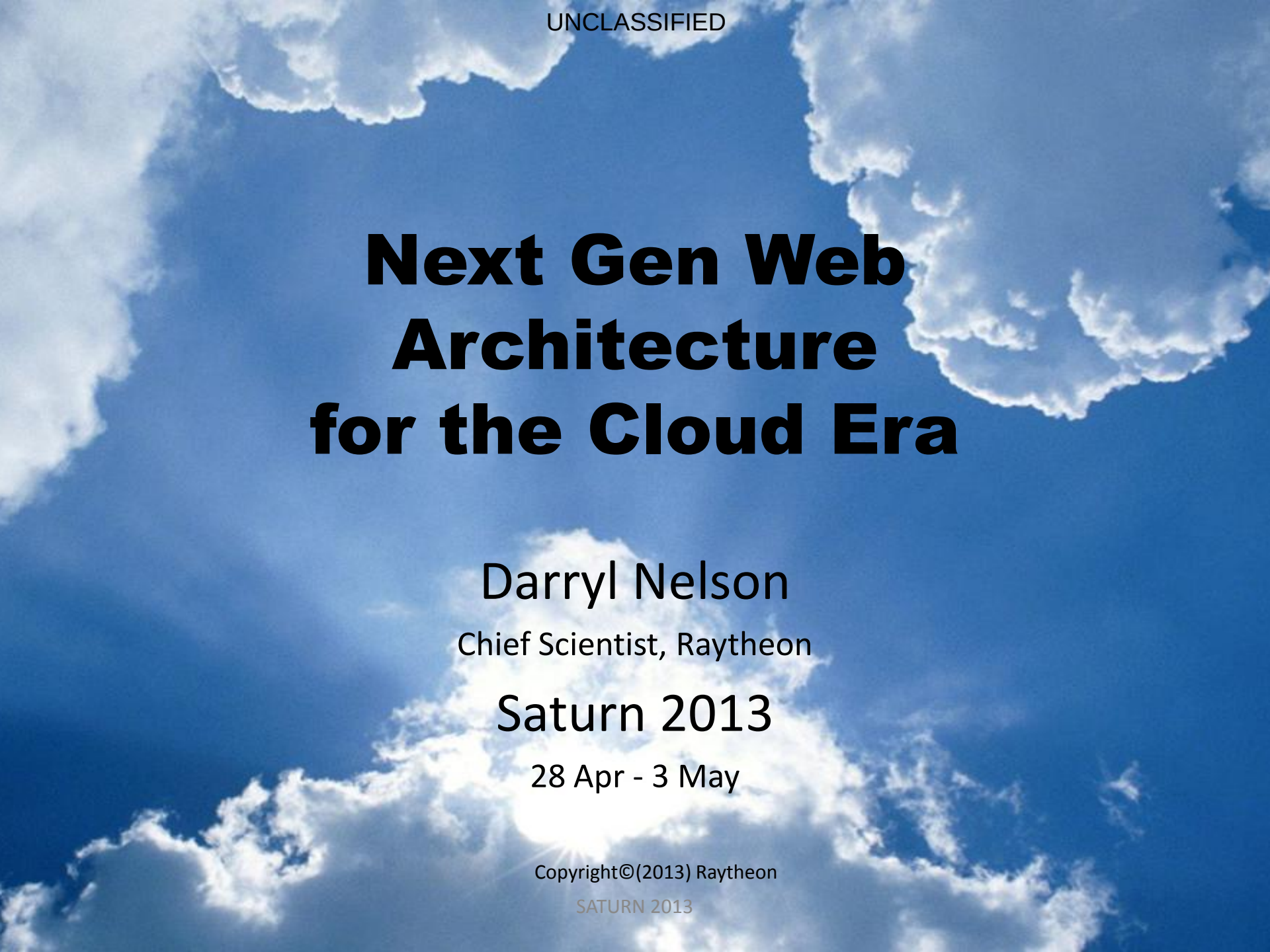


Next Gen Web Architecture for the Cloud Era

Darryl Nelson

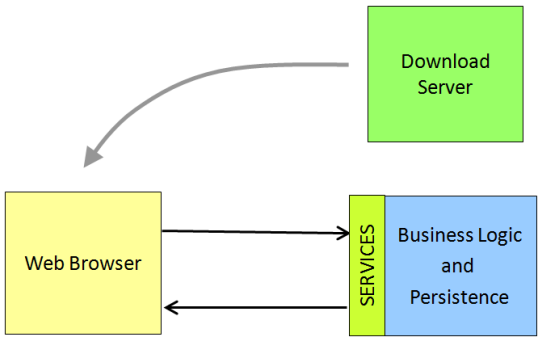
Chief Scientist, Raytheon

Saturn 2013

28 Apr - 3 May

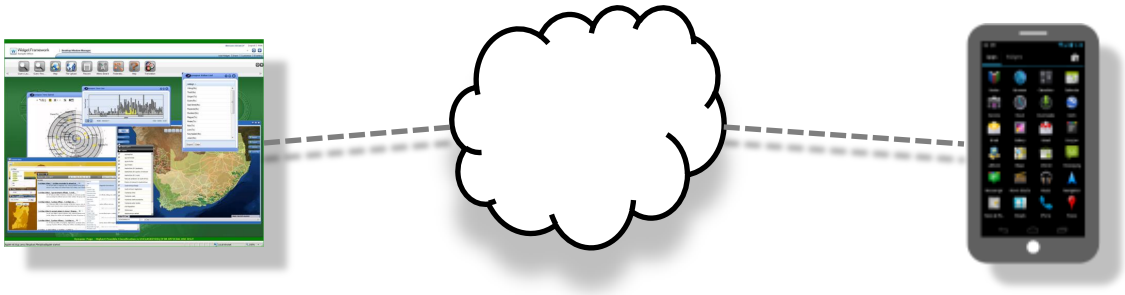
Agenda

- Existing Web Application Architecture
- SOFEA
- Lessons learned

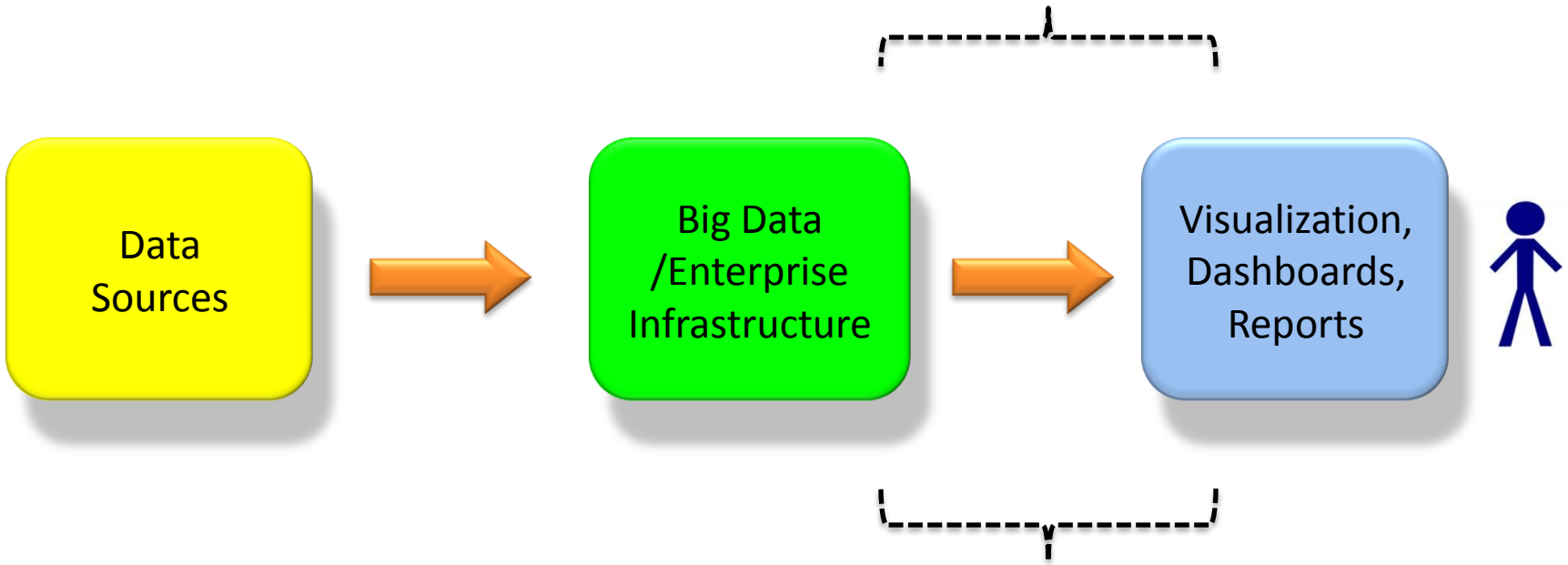


Audience

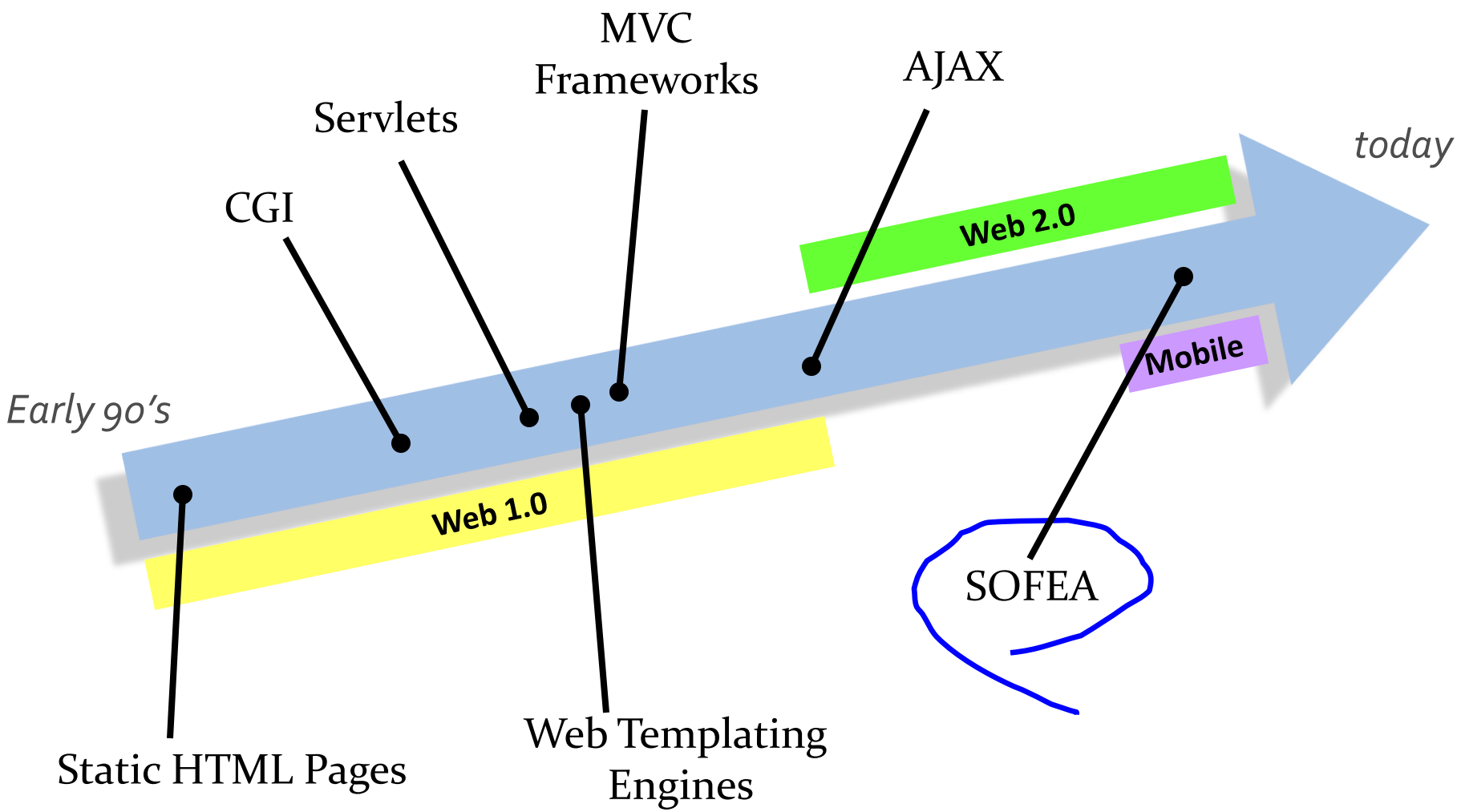
Anyone interested in web technology
who has a *basic* understanding of
web applications and
Service Oriented Architectures (SOA)



Focus



Arc of Web App Architecture History *

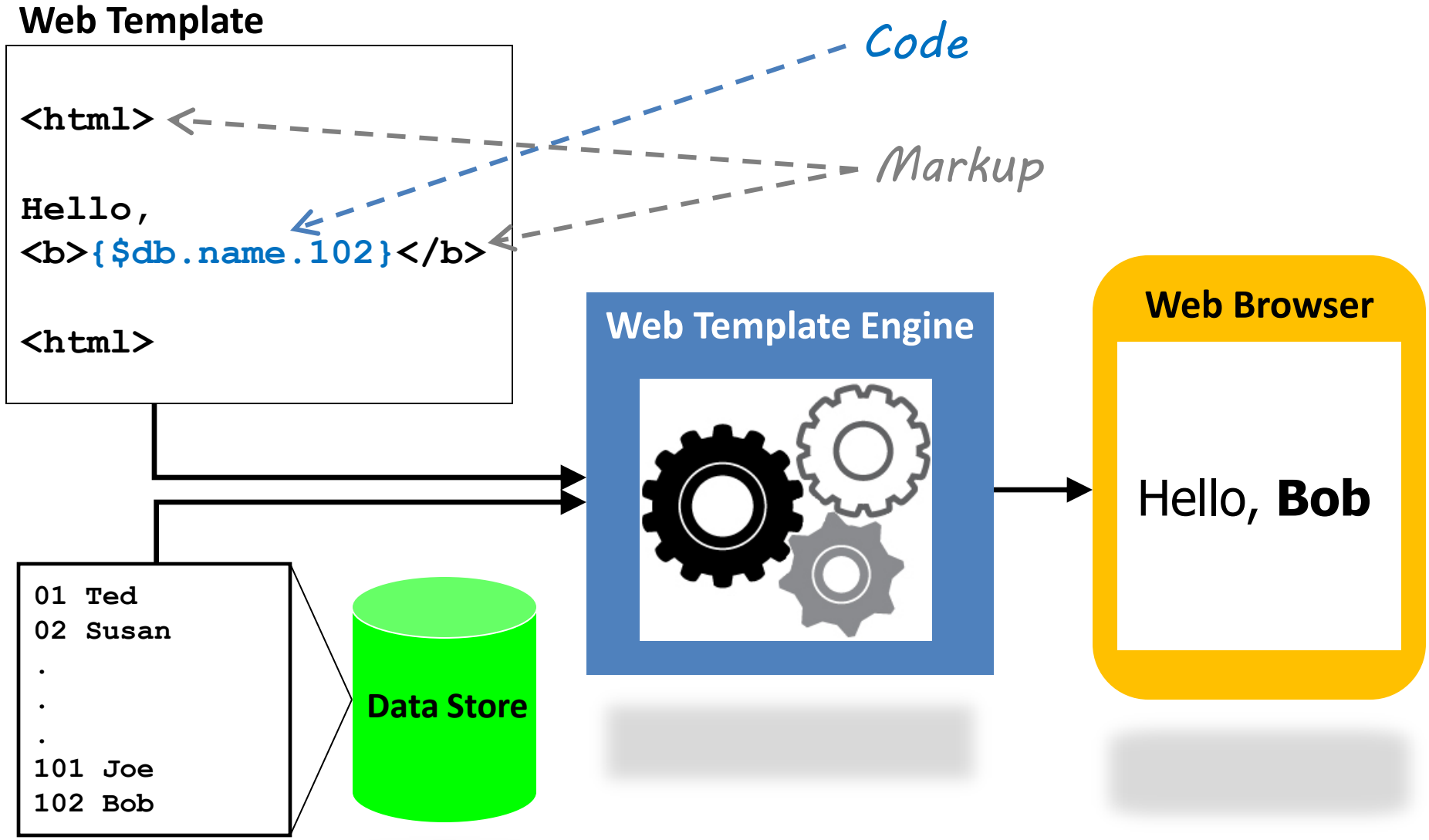


* NOT to scale!

Web Templating Engines

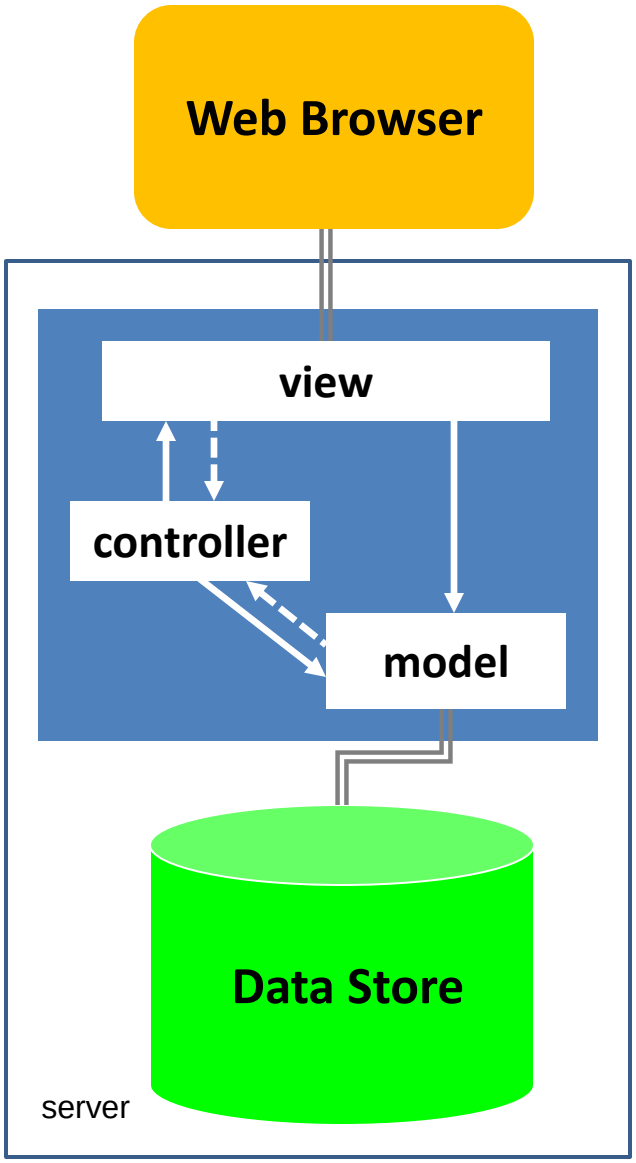
- Embedded code within static HTML elements
- Mix of static and dynamic HTML
- "Model 1" Architecture
- Examples
  - Java Server Pages (JSP)
  - PHP
  - Active Server Pages (ASP)

Web Templating Engines cont.



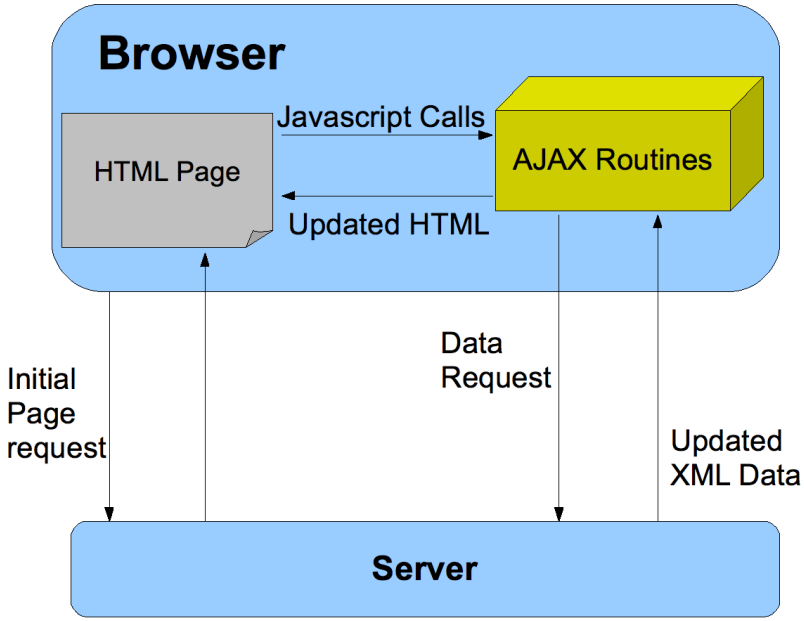
MVC Frameworks

- Model View Controller pattern
- Server side framework
- "Model 2" Architecture
- Examples
  - ASP.NET MVC Framework (.Net)
  - Struts, Spring MVC (Java)
  - Ruby on Rails (Ruby)
  - Django (Python)
  - Grails (Groovy)



AJAX

- **A**synchronous **J**avaScript **A**nd **X**ML
- Dynamic content changes without reloading the entire page
  - interactive and dynamic web apps approaching rich client capability
- HTML/CSS + DOM + *XmlHttpRequest* Object + JavaScript + JSON/XML

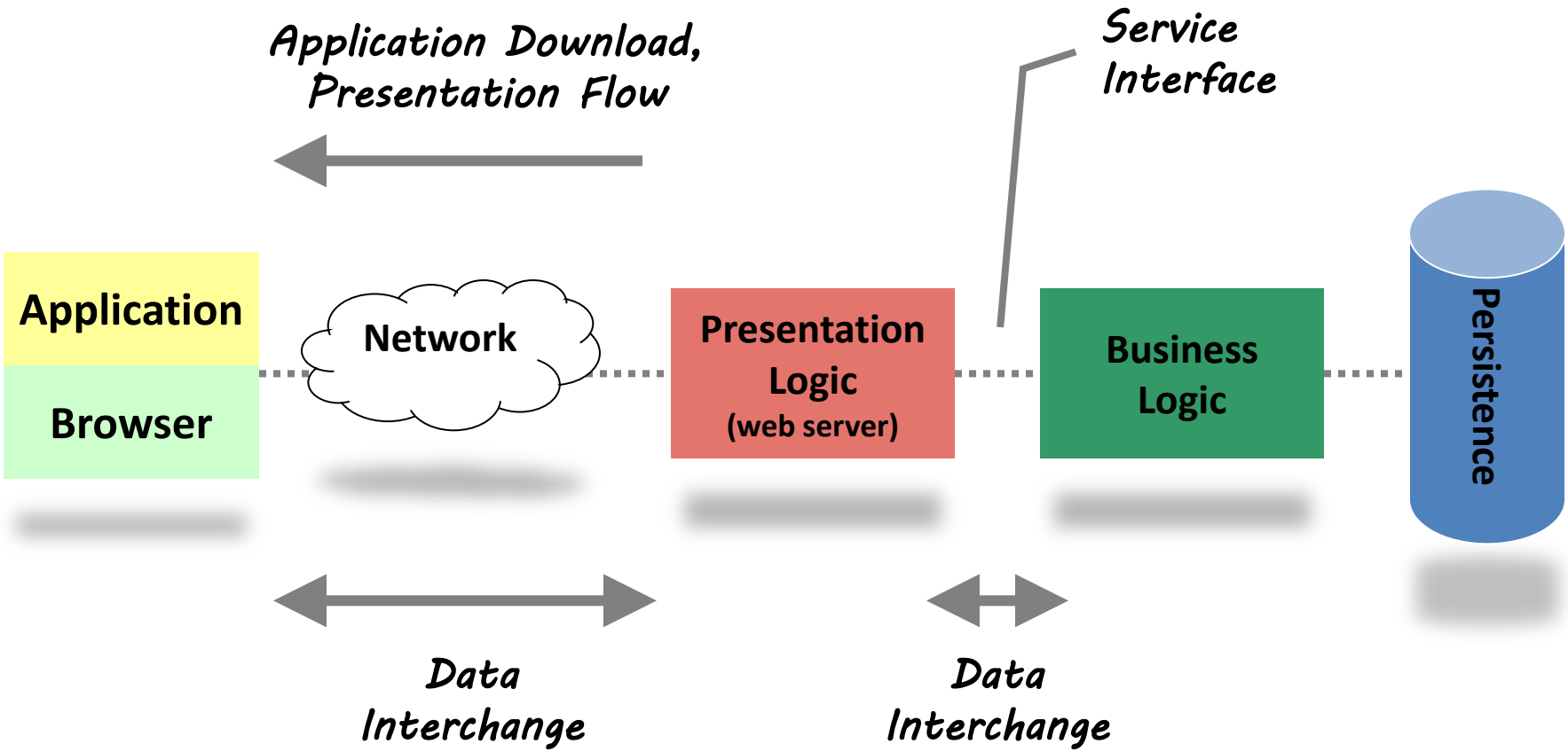


Img: http://courses.ischool.berkeley.edu/i153/s11/webapps-advanced

3 Processes of Web Applications

1. **Application Download**
   Mobile code (JavaScript, HTML, Applets, Flash) download to the client (web browser)
2. **Presentation Flow**
   Dynamic visual rendering of the UI (screen changes, new screens, etc) in response to user input and data state changes
3. **Data Interchange**
   The exchange of data between two software components or tiers (search, updates, retrieval, etc)

Process Allocation for Web Templating Engines Frameworks

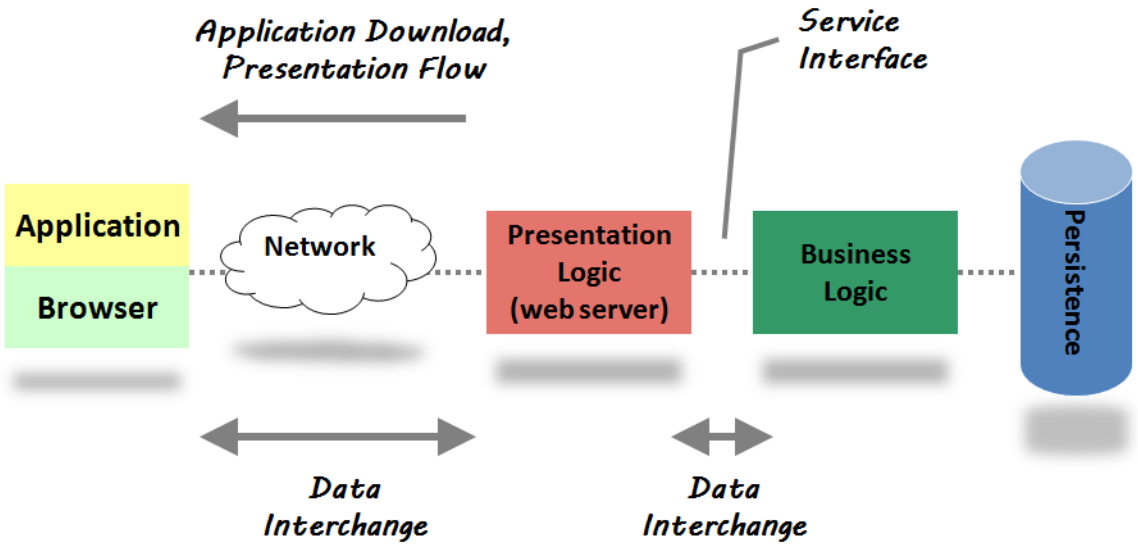


Characteristics of Web Templating Engines and MVC Frameworks

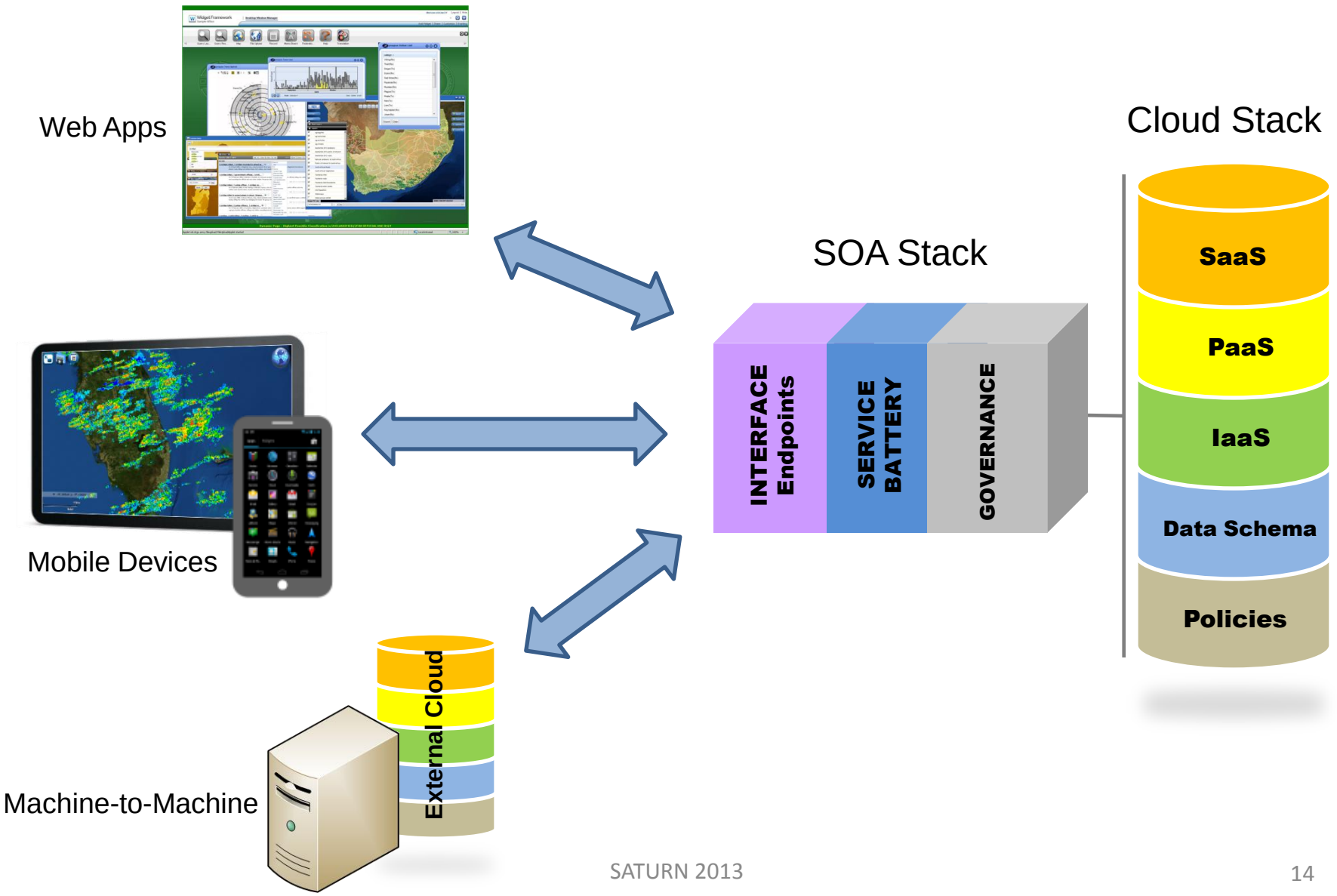
- **Tight coupling between presentation flow and data interchange (both in the web server)**
  - Triggering a Presentation Flow in a web application always initiates a Data Interchange operation
  - Every Data Interchange operation results in a Presentation Flow operation
- **Presentation flow and data interchange are orthogonal concerns that should be <u>decoupled</u>**
  - Separate concerns

Today

web templating engines +
MVC frameworks +
a *sprinkling* of Ajax



SOA & Cloud


Web Apps
Mobile Devices
Machine-to-Machine
External Cloud
SOA Stack
INTERFACE Endpoints
SERVICE BATTERY
GOVERNANCE
Cloud Stack
SaaS
PaaS
IaaS
Data Schema
Policies

SOFEA

An architectural style
for web applications in
SOA (& Cloud) environments

SOFEA

- **Service Oriented Front End Architecture**
  - *Synonymous with "Single Page"* Web Applications
- ***Life above the Service Tier***
  How to Build Application Front-ends in a Service-Oriented World
  - Ganesh Prasad, Rajat Taneja, Vikrant Todankar
- ***Architectural Style***
  - Not an implementation
- Prasad, et al propose that **the SOA revolution has left behind application front ends/UI's**

SOFEA is now…

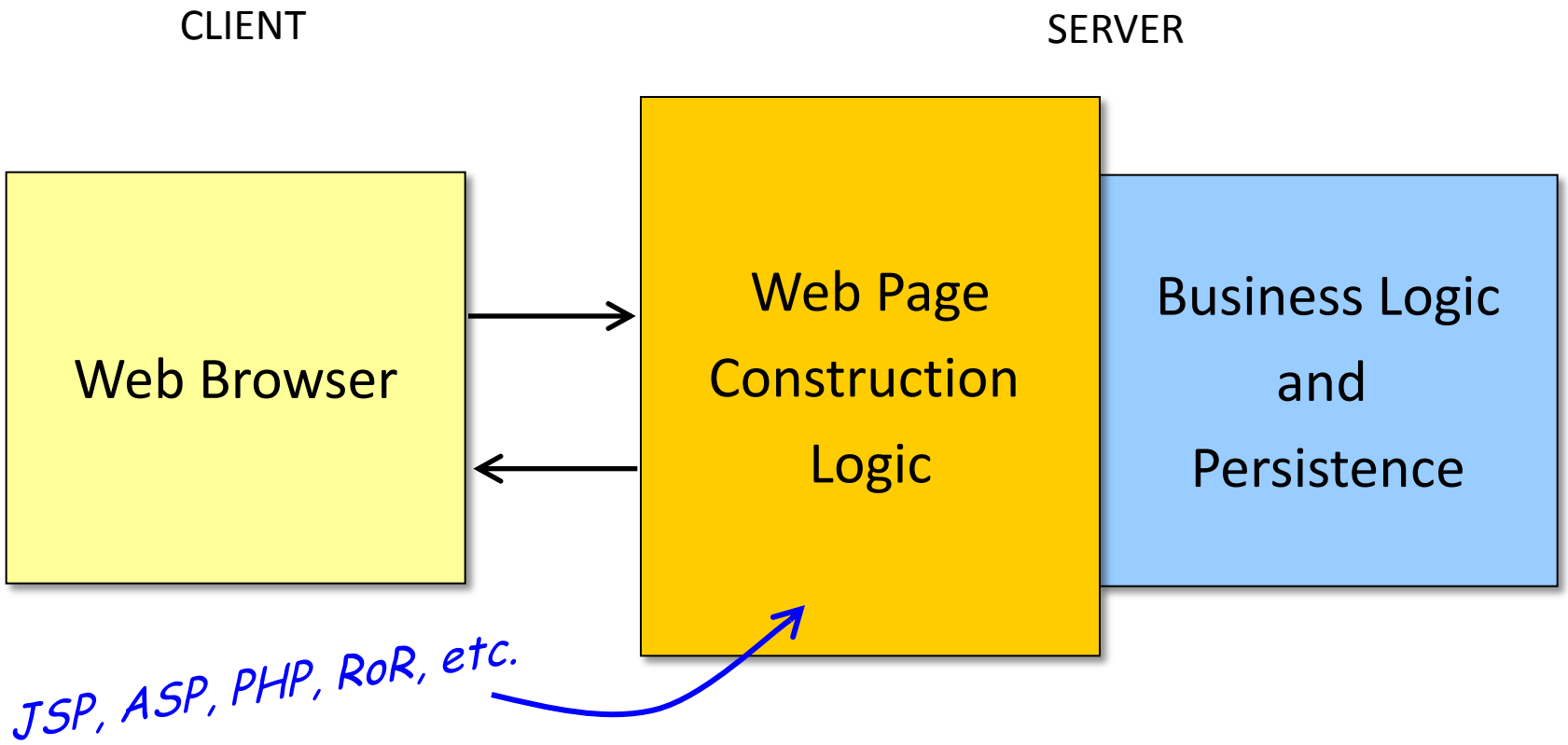
Feasible because

1) Maturity of the SOA paradigm in theory and practice
2) Advancements in browser-based client technologies, especially JavaScript browser engines and AJAX toolkits

Necessary because

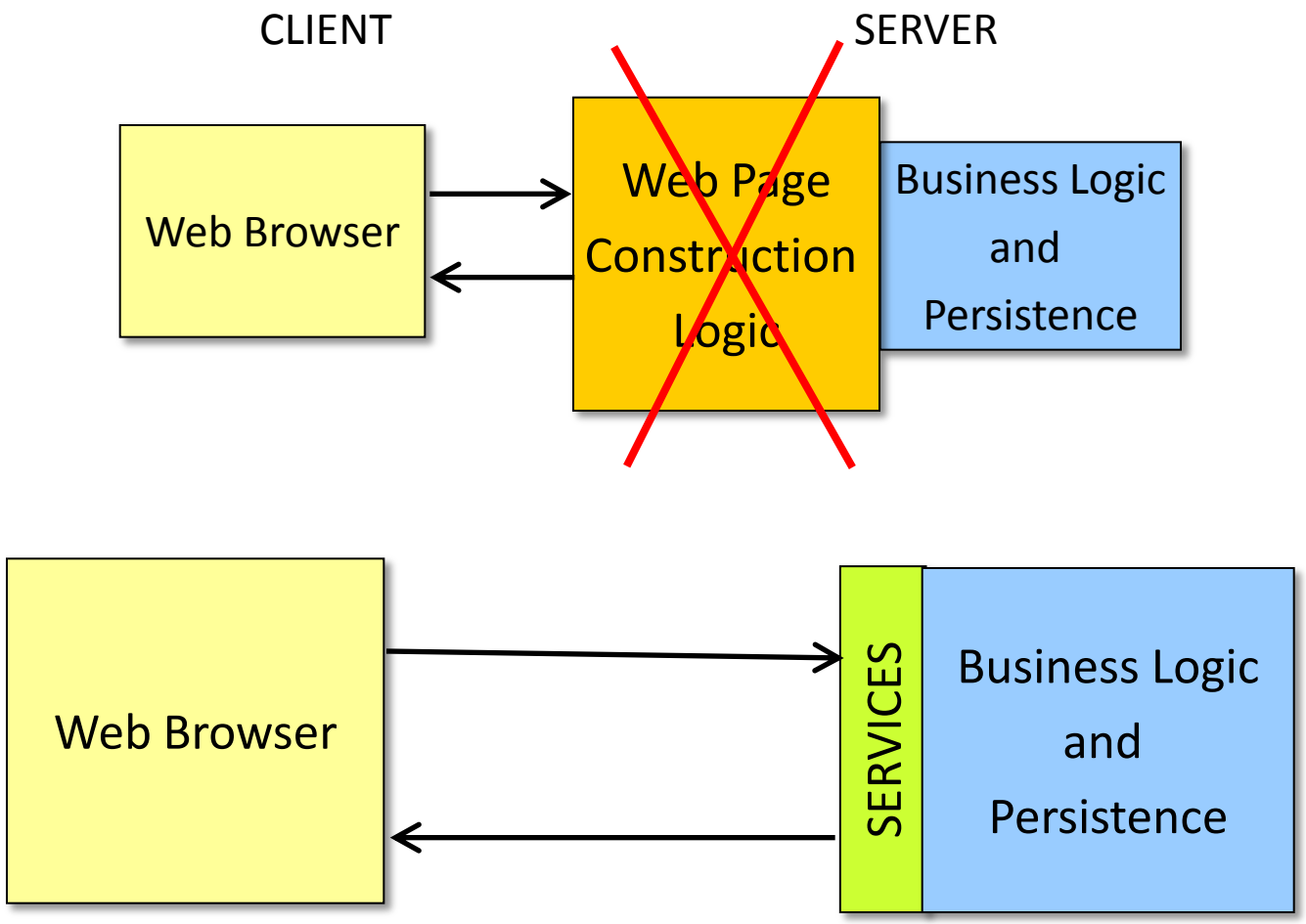
1) SOA is the defacto delivery mechanism for cloud-based services (Cloud and SOA are complementary technologies)
2) Diversity of client platforms
   - Growing dominance of Mobile clients

Legacy Enterprise Web Architecture

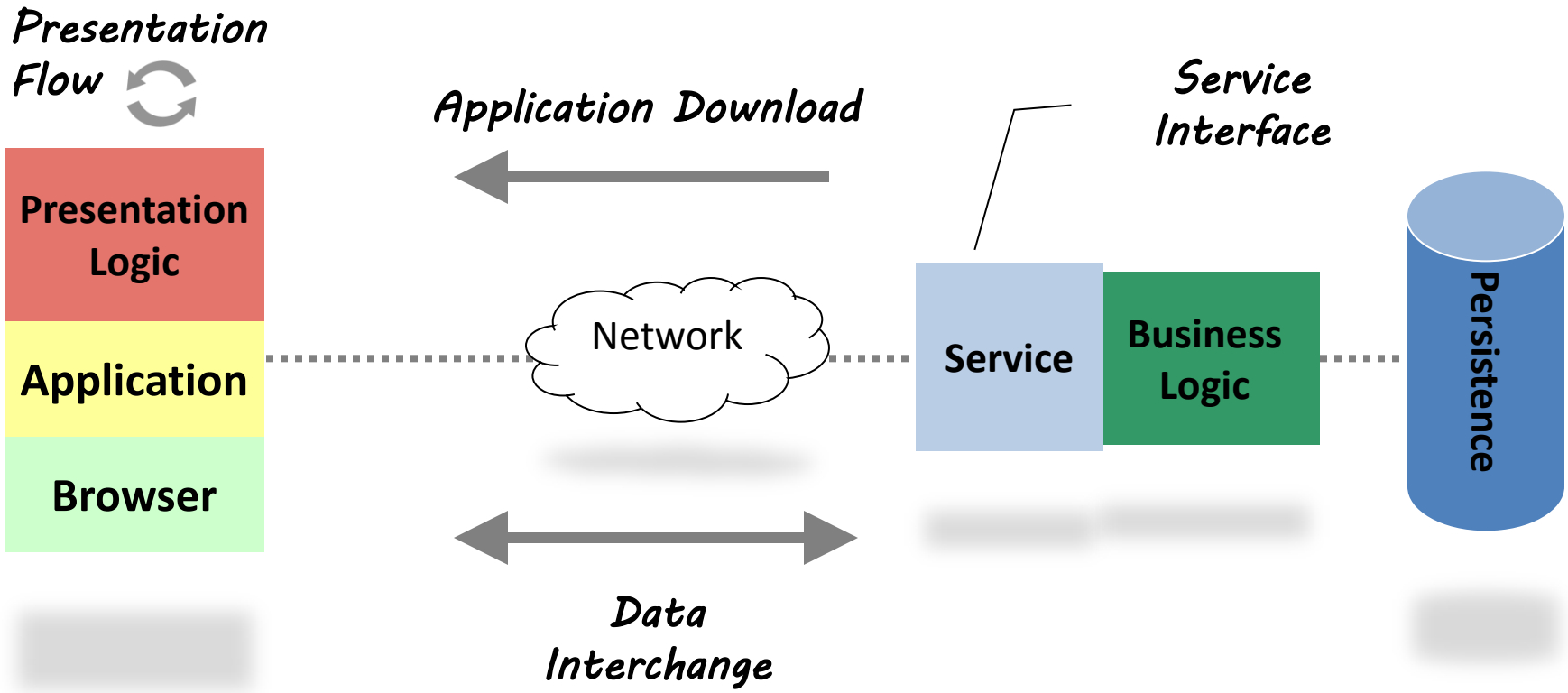


Typical Enterprise Web Application Architecture

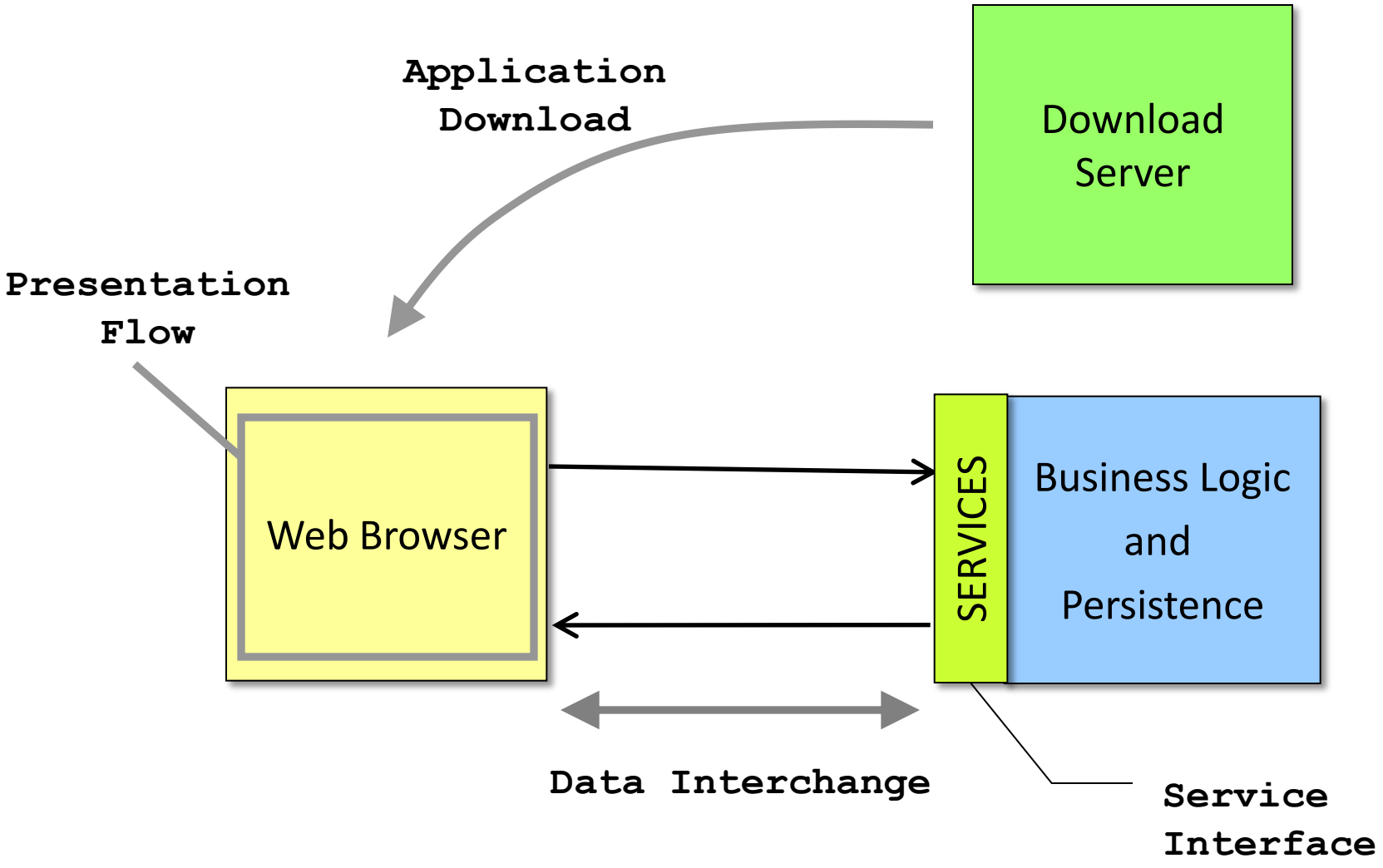
SOFEA



Process Allocation for SOFEA



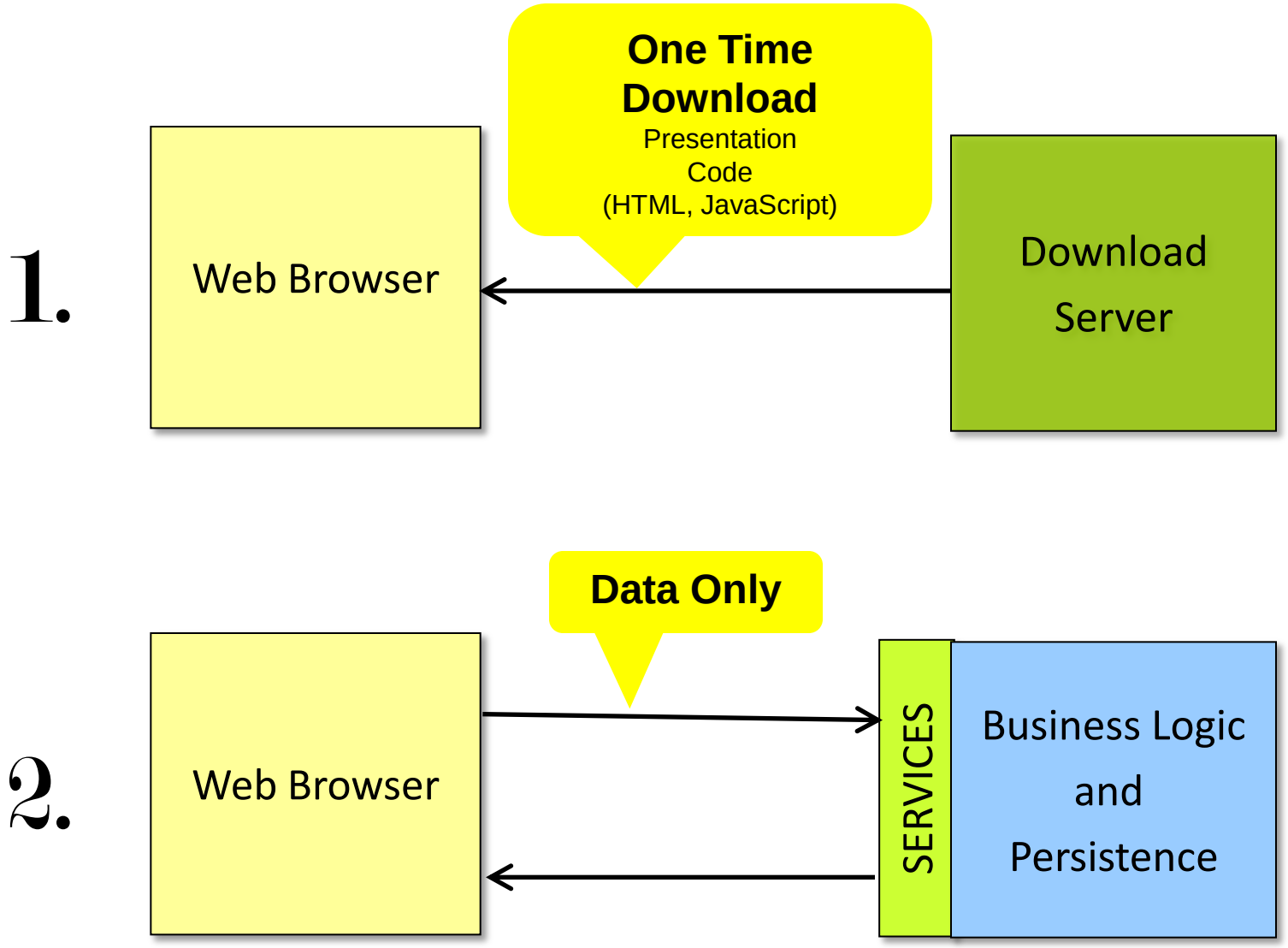
3 Web Processes and SOFEA



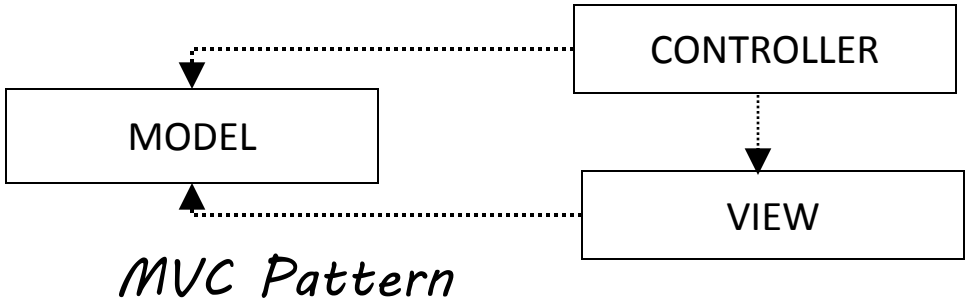
SOFEA Principles

1. Application Download, Data Interchange, and Presentation Flow must be decoupled
   - No part of the client should be evoked, generated or templated from the server-side.
2. Presentation Flow is a client-side concern only
3. All communication with the application server should be using services (REST, SOAP, etc)
4. The MVC design pattern belongs in the client, not the server

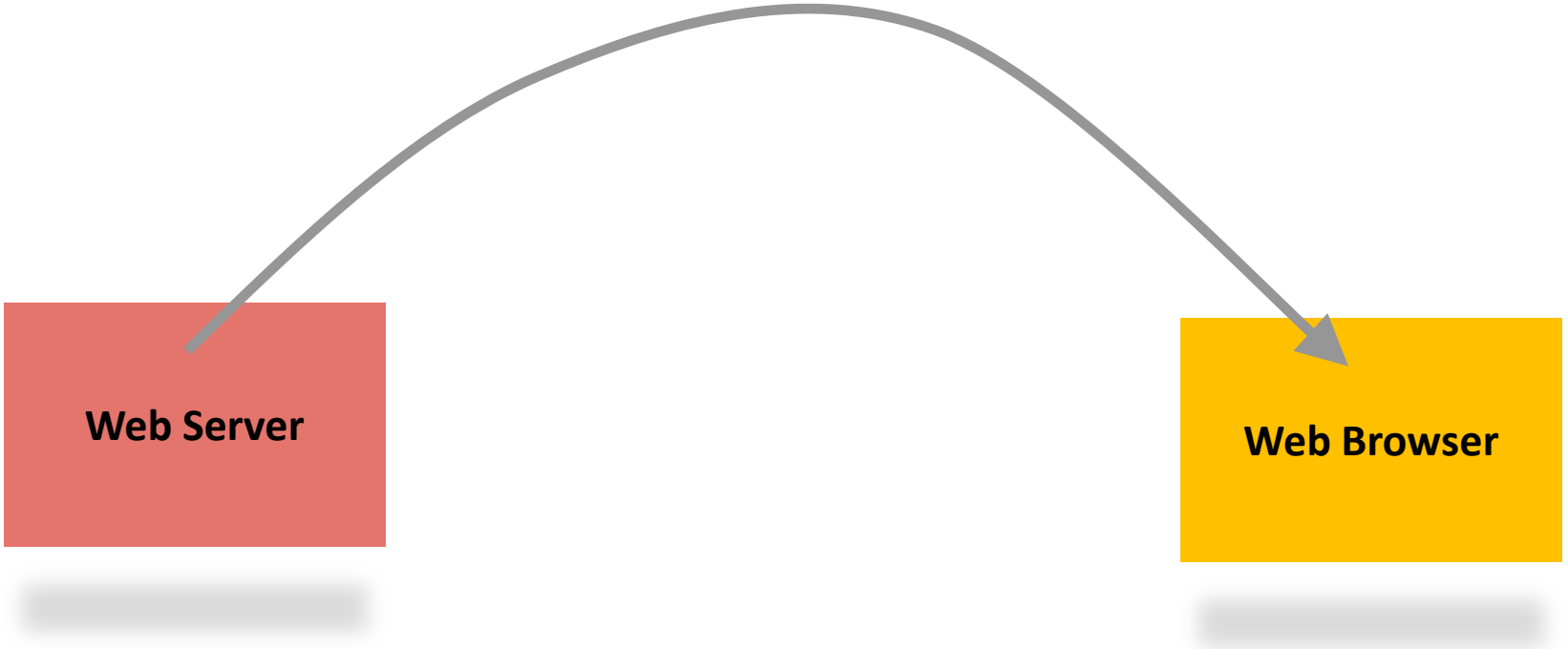
SOFEA Lifecycle



MVC in the Web Browser



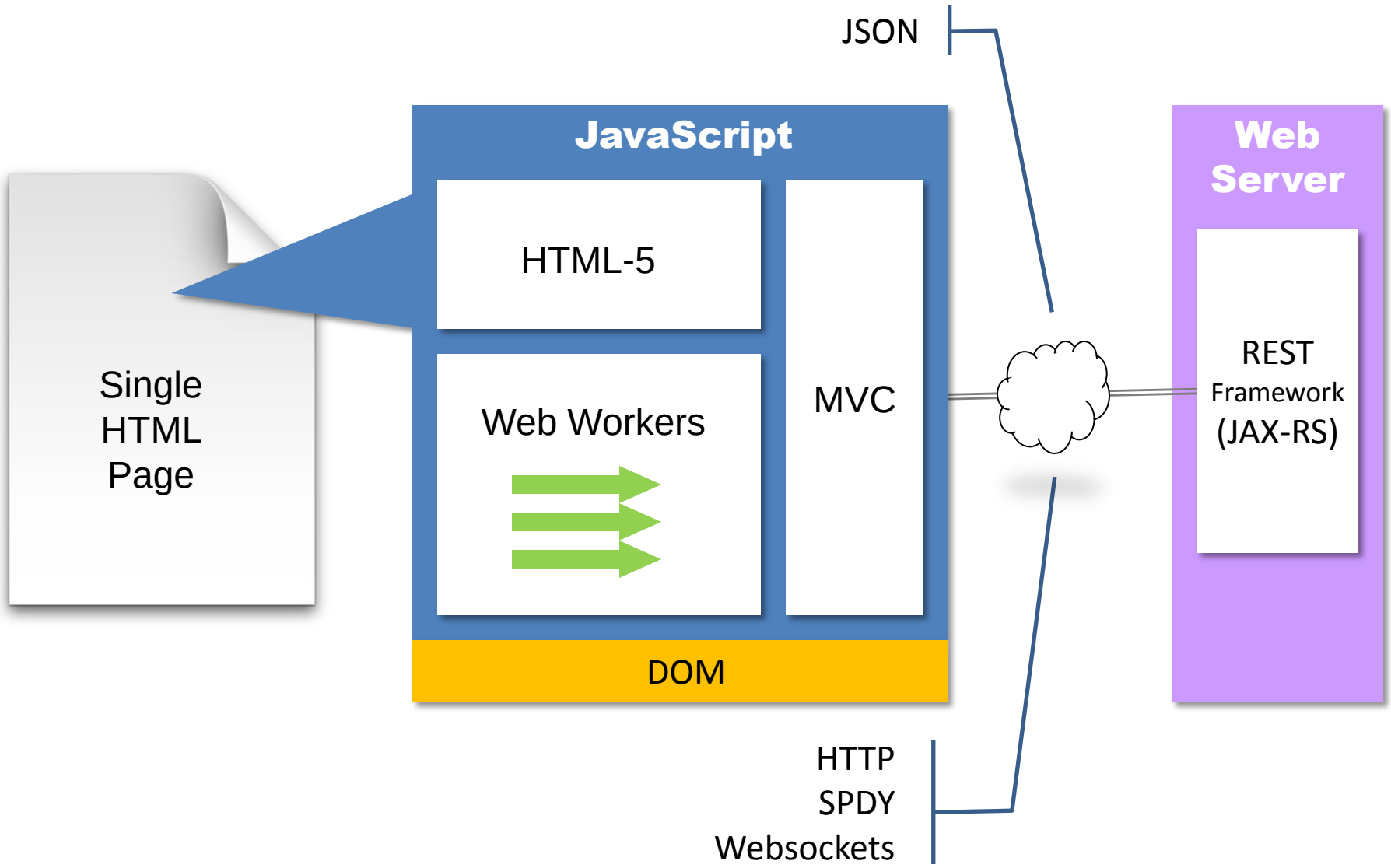
MVC Pattern



Benefits of SOFEA

- **Scalability**
  - Server has less work to do; no more presentation generation, just provide a services
- **Higher ROI for each LOC**
  - Expanded opportunity space due to the inherent reusable nature of SOA
- **Better user response**
  - Low latency == happy end users
  - After the app download, no presentation is transported over the wire, only business data
- **Natural fit into SOA and Cloud environments**
- **Organized programming model**
  - Client developers concentrate on the UI
  - Server-side developers concentrate on Services
- **Offline applications**
  - When the network crashes, decoupled client can dynamically switch out their model objects
- **Interoperability**
  - Easier integration with lower overhead from multiple platforms
  - Clients don't care if services are Java, C#, Python, Cobol or a heterogeneous mix

SOFEA Client Implementation Archetype



Lessons Learned

- **The web client is a "Priority 1" architecture tier, not an after thought**
  - Object-Oriented Analysis and Design principles
  - Design Patterns
  - Continuous integration, performance testing, etc
  - Critical to expend significant engineering time and energy on the client architecture
- **Use a mature client-side frameworks**
  - Dojo, JQuery, AngularJS, etc
- **The RESTful model is natural fit for SOFEA systems**
- **Architects & developers should "bake-in" *asynchronicity* between the server and client layers**
- **Leverage newer technologies where appropriate**
  - HTML-5 Web Workers & Websockets
  - Google's SPDY
- **Start cross-browser compatibility testing early in the development cycle**
  - Fight the "add IE support later" temptation
- **SOFEA excellent choice for our customer's bandwidth starved environments**
  - Very low latency for those customer's with average-good network pipes

Resources

- ***Life Above the Service Tier***
  by Ganesh Prasad, Rajat Taneja and Vikrant Todankar
  - http://wisdomofganesh.blogspot.com/2011/10/life-above-service-tier-change-of-links.html
- **JavaScript Frameworks**
  - Dojo: http://dojotoolkit.org/
  - JQuery: http://jquery.com/
  - AngularJS: http://angularjs.org/
  - KnockoutJS: http://knockoutjs.com/
- **JavaScript Design Patterns Book**
  - http://addyosmani.com/resources/essentialjsdesignpatterns/book/
- **SOA & Cloud**
  - http://www.infoq.com/articles/ieee-software-engineeering-services-cloud-computing
- **Web Sockets**
  - http://www.websocket.org/
- **Google SPDY**
  - http://www.chromium.org/spdy

Darryl Nelson

Chief Scientist

Raytheon Intelligence and Information Services

Darryl.Nelson@Raytheon.com

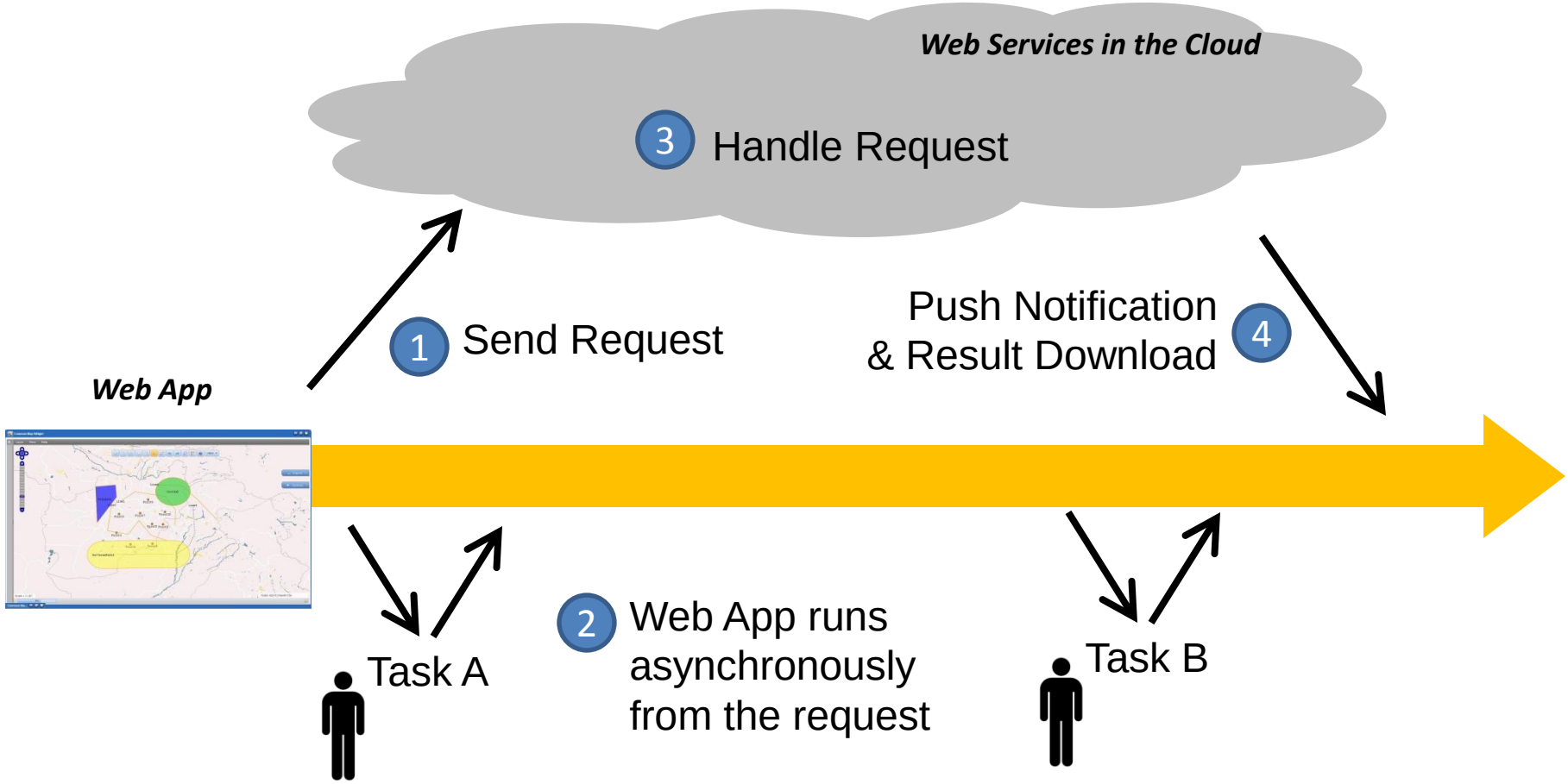
Darryl.Nelson.Tech@Gmail.com

backup

SOFEA Implementation Examples

- Client
  - JavaScript: Dojo, JQuery, ExtJS, angularjs.org, knockoutjs.com, Twitter Bootstrap
  - Flex*
  - Silverlight*
  - Java Applets*
- Services
  - WS-* (SOAP/WSDL)
    - Axis, Weblogic, Websphere
  - REST ☺
    - Jersey, RESTEasy, RESTlets, Drop Wizard

Processing Request with Push Response Design Pattern



Adapted from J. Christensen, *Using REST Web-Services and Cloud Computing to Create Next Generation Mobile Applications*, ACM 978-60558-768-4/09/10