SPA, Web API, Serverless
the change of web app architectures

IT 4403 Advanced Web and Mobile Applications

Jack G. Zheng
Fall 2019

KENNESAW STATE UNIVERSITY

https://www.edocr.com/v/j23zyenr/jgzheng/Single-Page-Applications-Serverless

Overview

- This lecture notes provides a high level overview of single page applications in the context of web application architecture changes
  - The context: web application architectures
  - What is SPA
  - Key principles and concepts
  - Enabling technologies

The Context

- Rise of JavaScript/HTML5
  - JavaScript/HTML5 handle the user interface and interaction functionalities
- Changing of web application architectures
  - Backend and frontend is even more loosely coupled.

Rise of JavaScript

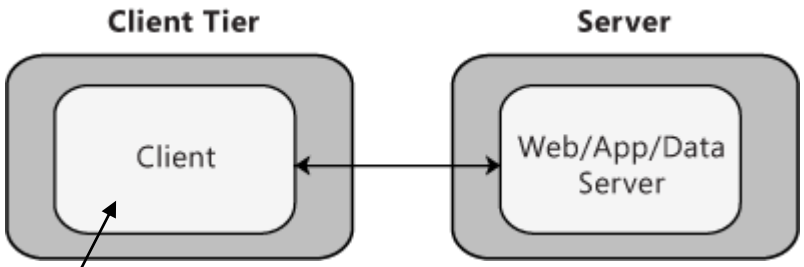
- #1 in GitHub since 2014
  - https://octoverse.github.com/projects#languages
- #1 in StackOverflow since 2013
  - https://insights.stackoverflow.com/survey/2018
- Most in demand and salaries, 2015
  - https://gooroo.io/GoorooTHINK/Article/16300/Programming-languages--salaries-and-demand-May-2015
  - http://www.techrepublic.com/article/here-are-the-3-most-in-demand-coding-languages-and-where-you-can-find-a-developer-job/
- Other rankings
  - #1 The RedMonk Programming Language Rankings since 2015 https://redmonk.com/sogrady/2018/08/10/language-rankings-6-18/
  - #2 in student programmers in hackathon (#1 is HTML/CSS) http://studenthackers.devpost.com/2015.html
  - #2 in Developer Economics survey https://dashboard.developereconomics.com/?survey=de15#mobile
- SharePoint/WordPress is moving to JavaScript
  - https://arc.applause.com/2015/11/24/wordpress-javascript-calypso/
  - https://developer.wordpress.com/calypso/
  - https://developer.wordpress.com/2015/11/23/the-story-behind-the-new-wordpress-com/

Change of Web App Architecture

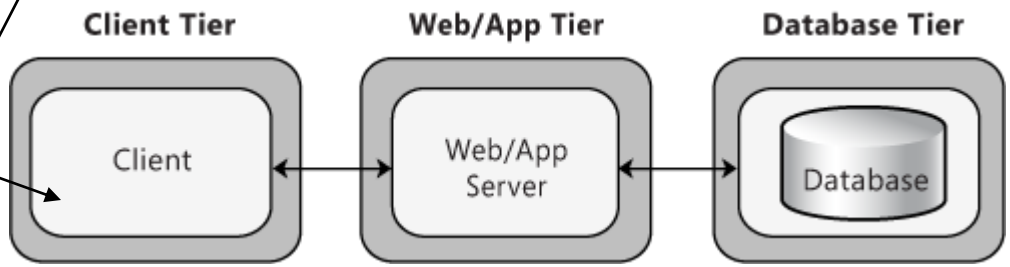
- Wave 1: multi-layer/tier systems
  - Rich (multiple) servers
- Wave 2: service oriented
  - Further separation of server side and client side
  - Services are more among servers
- Wave 3: growing focus on the client side and widely use of AJAX
  - Rich client development, SPA
  - More open service model; clients consume services
- Wave 4: consuming services from the cloud (third parties)
  - BaaS and serverless; the client is not tied to its own initiating server

Multi-Tier Web Architectures

- Two tier



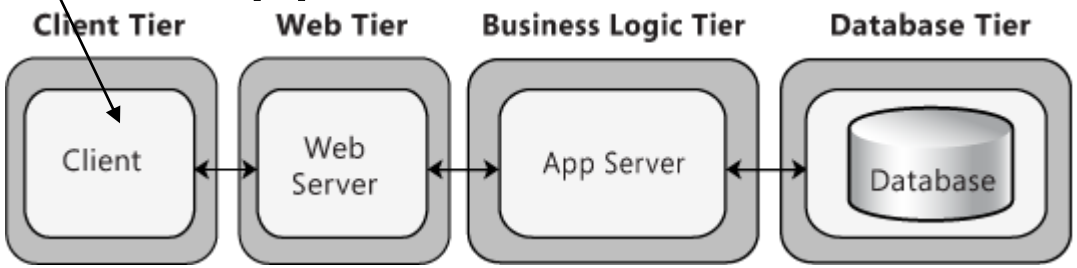
- Three tier (web app)



In the first wave of architecture change, there was no significant change on the client side

Changes were more focused on the server side

- Four tier web app

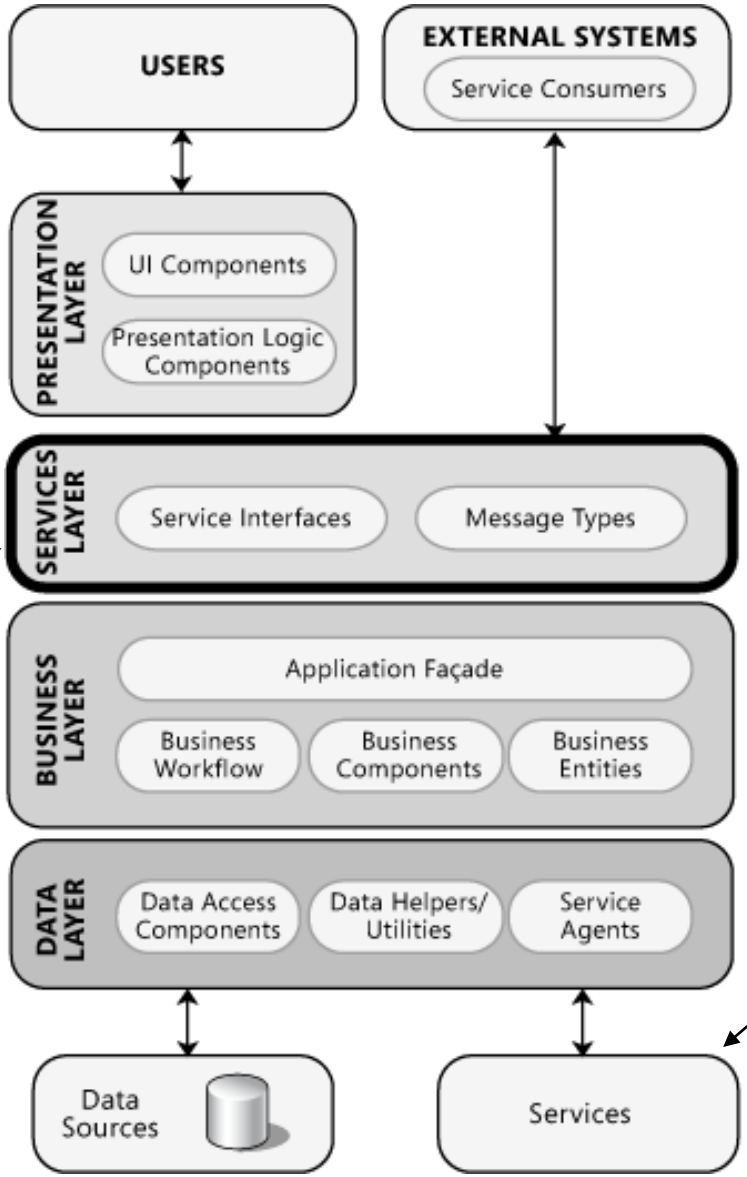


http://msdn.microsoft.com/en-us/library/ee658109.aspx

Service Oriented Architecture

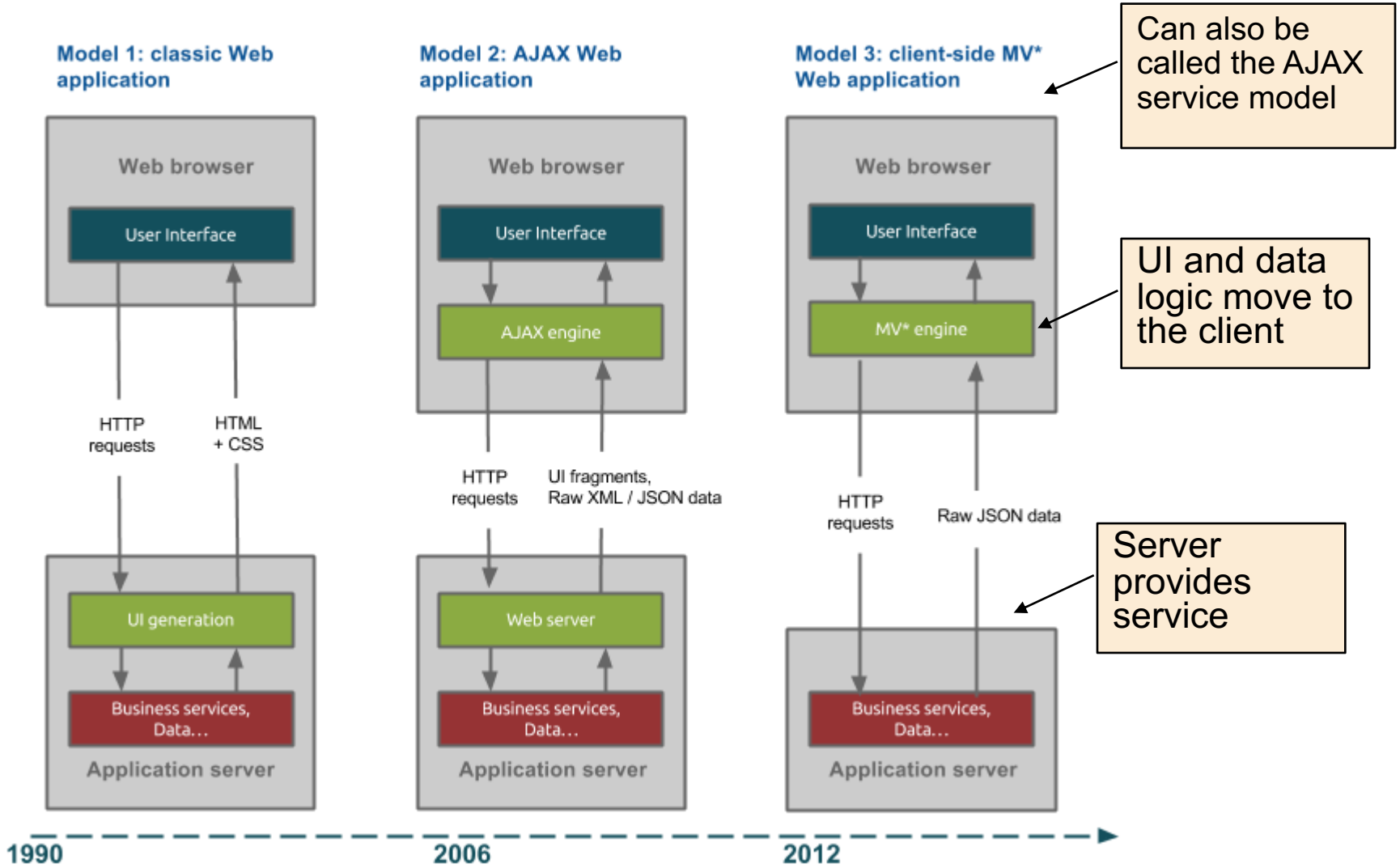
Presentation layer can flexibly moved to the client side.

The service layer effectively makes clients more like an independent and autonomous service consumer, and less owned by its server part – making things “loosely coupled”.



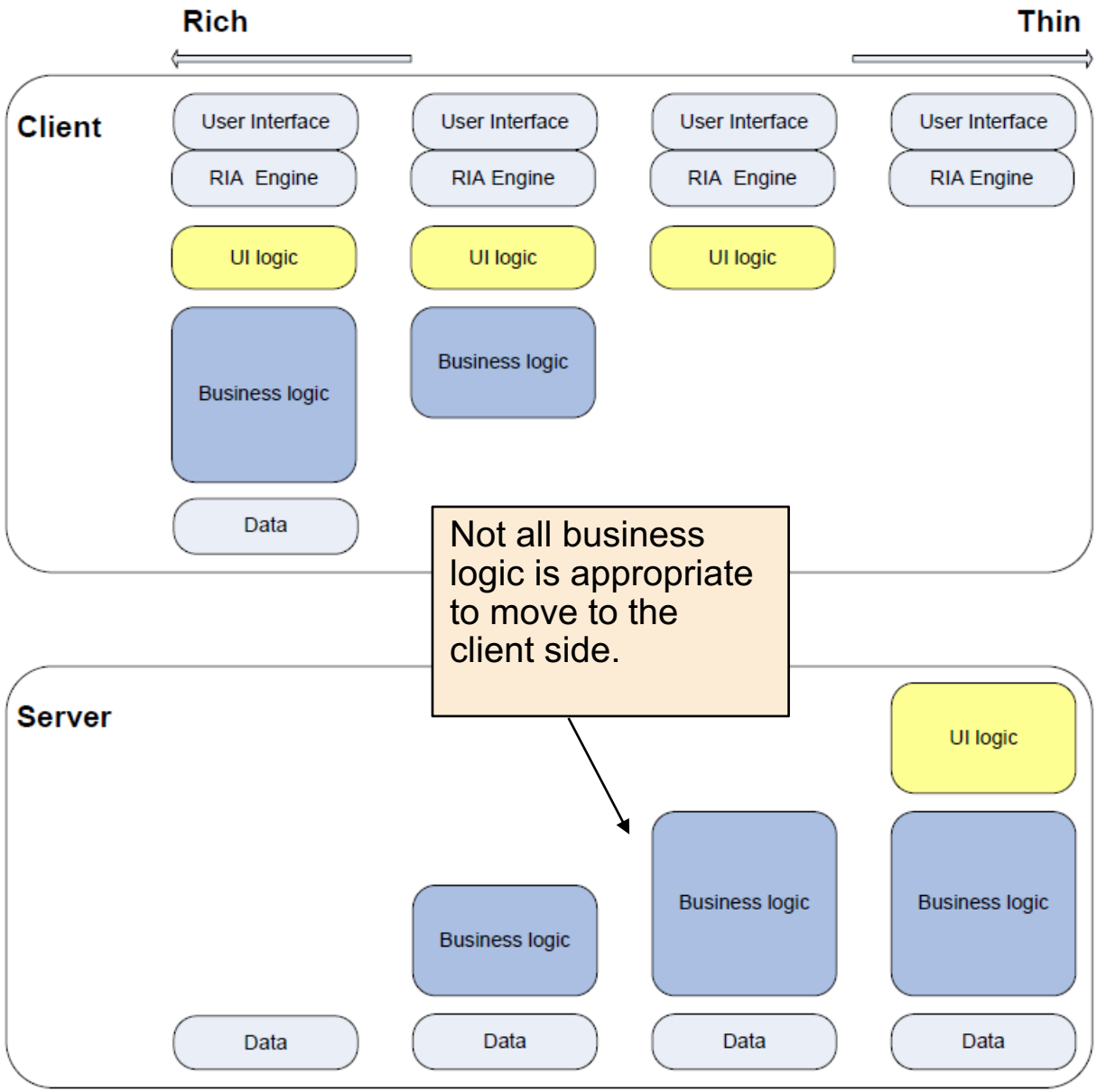
Data can also be services

Changing of C/S Interaction



http://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-1/

Placement of Logic

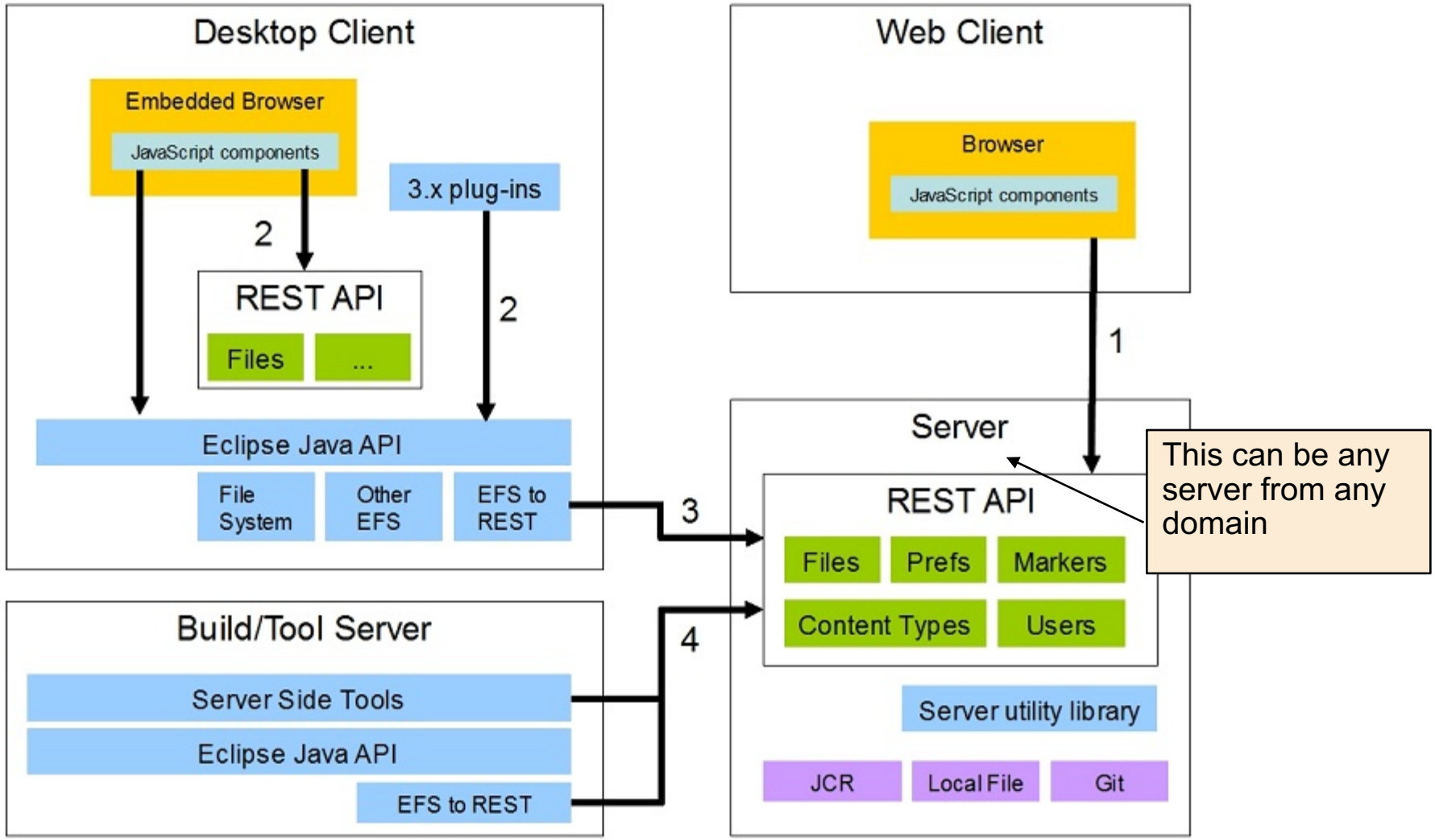


http://www.openajax.org/member/wiki/images/8/89/NexawebAjaxCharacteristics.pdf

Transfer of UI Logic

- UI logic moves to the client side, which can include
  - Layout
  - User interaction
  - Dynamic effects
  - Animation
  - Content presentation update/refresh
  - These functionalities are handled by HTML5/JavaScript https://stackoverflow.com/questions/1059832/is-client-side-ui-rendering-via-javascript-a-good-idea
- The server's diminished role
  - The server maintains no UI state.
  - The server, however, does provide resources to the client. Those resources start out with the initial HTML for the web page (the shell) and may in response to client HTTP requests for data in the form of HTML fragments, JSON data, or on-demand JavaScript/CSS.

AJAX + (RESTful) Services



http://stage.vambenepe.com/archives/1801

Serverless

- The term serverless application has two distinct but related meanings (https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/serverless/web-app):
  - **Backend as a service** (BaaS). Backend cloud services, such as databases and storage, provide APIs that enable client applications to connect directly to these services.
  - **Functions as a service** (FaaS). In this model, a "function" is a piece of code that is deployed to the cloud and runs inside a hosting environment that completely abstracts the servers that run the code.

- Both definitions have in common the idea that developers and DevOps personnel don't need to deploy, configure, or manage servers.

- Serverless applications are event-driven cloud-based systems where application development rely solely on a combination of **third-party services, client-side logic** and cloud-hosted remote procedure calls **(Functions as a Service)**.

- *Serverless architectures are application designs that incorporate third-party "Backend as a Service" (BaaS) services, and/or that include custom code run in managed, ephemeral containers on a "Functions as a Service" (FaaS) platform. By using these ideas, and related ones like single-page applications, such architectures remove much of the need for a traditional always-on server component. Serverless architectures may benefit from significantly reduced operational cost, complexity, and engineering lead time, at a cost of increased reliance on vendor dependencies and comparatively immature supporting services.* https://martinfowler.com/articles/serverless.html

- Microservices: http://www.forbes.com/sites/jasonbloomberg/2015/02/09/service-oriented-architecture-enabler-of-the-digital-world/

Is it good?

- See some discussion here
  - https://softwareengineering.stackexchange.com/questions/140036/is-it-a-good-idea-to-do-ui-100-in-javascript-and-provide-data-through-an-api

What is SPA

- A single-page application (SPA) is a web application or web site (the whole application or site) that fits on a single web page with the goal of providing a more fluid user experience similar to a desktop application.
- The page does not reload at any point in the process, nor does control transfer to another page.
- The application or the page involves dynamic communications with the server behind the scenes.
- It’s an extreme version of rich client focused web application architecture

SPA Key Principles and Concepts

- Page/shell
- View
- Data/resource

https://johnpapa.net/pageinspa/

Page/Shell

- The “page” in SPA is the single web page that the server sends to the browser when the application starts. It’s the server rendered HTML that gets everything started. No more, no less.
- Initial page acted as a holder for content, UI, style, and other logic.
- The shell is the layout and structure of your app. The visual parts that rarely change.
- The shell is generally loaded on the initial page load from the server and serves as the placeholder for all of your views. It can also be dynamically loaded progressively.
- The page/shell does not reload at any point in the process, nor does control transfer to another page.

https://johnpapa.net/pageinspa/

https://developers.google.com/web/fundamentals/architecture/app-shell

View

- The Views are HTML (UI) fragments that make up what the users commonly call screens or pages.
- A particular “sub-page” “partial page” within the shell.
  - Could be the complete page, a section (div) of the page, or just certain areas all round the page.
  - Different views can be presented at the same time and do not interfere each other (multi-view, multi-doc interface)
- Transition (navigation) between views
  - URL routing by simulating HTTP request

https://johnpapa.net/pageinspa/

User Interactions

- All user interactions and actions happen within the page.
- All results (view changes) due to these actions are presented directly in the page.
- Multiple actions/interactions are possible and should not interfere one other (multi-tasking).

Data/Resources

- SPA makes requests for resources from the server on-demand
- Those resources may be JSON data, HTML fragments, JavaScript or CSS
- The server provides resources/data, not pages (except for the initial page)

https://johnpapa.net/pageinspa/

Applications

- SPA is good for many applications that try to replicate desktop app experience where user interactions are more intensive
- Examples
  - Many email apps
    - Google Gmail, Outlook Web
  - Office/editing apps
    - Office 365
    - Google Docs
  - http://www.zillow.com
  - http://cc-ng-z.azurewebsites.net

SPA is not (Exactly) Single Page Website

- Single or one page web site is a website designed in one (physical) web page
- Rooted from the web design community as a UI design style.
- Focus on content and the flow of content; does not focus on interactions between server and client
- Definitions did have changed to accommodate trends; and more recently came across SPA
  - https://onepagelove.com/what-exactly-is-a-one-page-website
- Typical examples
  - http://desrist2017.kit.edu
  - http://www.100yearsofnps.com
  - More examples: https://onepagelove.com, http://www.awwwards.com/websites/single-page/

Technical Approaches

- AJAX – the focus of this course
- WebSocket
- Browser Plugin

Driving Technologies

- Key elements (this course will focus on bare-bone SPA without frameworks)
  - JavaScript
  - AJAX
  - Standard data transfer format especially JSON
  - RESTful web services (APIs)
- More (a research project?)
  - HTML 5 (web storage, web sockets, history)
  - Data model/storage
  - Frameworks
  - Browser plugin

Three Key Parts

- Server side: services, APIs
  - A server provides data or functionalities through standard service APIs (RESTful service APIs)
- Communication: AJAX
  - Client communicates with service providers via AJAX
  - AJAX provides dynamic and continuous user experience
- Client side: JavaScript (jQuery)
  - handles communication and consumes services
  - controls logic (UI, data, business) and events

Web API Overview

- Web APIs are interfaces exposed based on web technologies
  - http://code.tutsplus.com/articles/the-increasing-importance-of-apis-in-web-development--net-22368
  - Server side: similar as web services, interface exposed through HTTP methods
  - https://en.wikipedia.org/wiki/Web_API
- Major API techniques
  - SOAP web services
  - RESTful web services
  - JavaScript
- Major data/message format
  - XML and XML-based: RSS/Atom (feeds)
  - JSON
- Directory of web APIs
  - http://www.programmableweb.com/apis/directory
  - http://www.programmableweb.com/api-research

RESTful Web Services

- The RESTful Web Service is an approach for using REST as a communication technology to build SOA
  - In other words, services are exposed using REST style interfaces
  - RESTful web services = REST + Web Services
- Key principles and practices
  - Expose directory structure-like URIs.
  - Use HTTP methods explicitly.
  - Use multiple representations (XML, JSON, or both)
  - Be stateless.
- More
  - http://www.ibm.com/developerworks/webservices/library/ws-restful/
  - http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069

RESTful web services URI

- General format

```
Protocol://ServiceName/ResourceType/ResourceID
```

- Examples
  - http://MyService/Persons/1
  - http://it-ebooks-api.info/v1/book/2279690981
  - http://it-ebooks-api.info/v1/search/php/page/3
- Parameters
  - Parameters are used for options to serve the representation, like
    - http://MyService/Persons/1?format=xml&encoding=UTF8
  - NOT for part of the resource
    - http://MyService/Persons?id=1

JSON

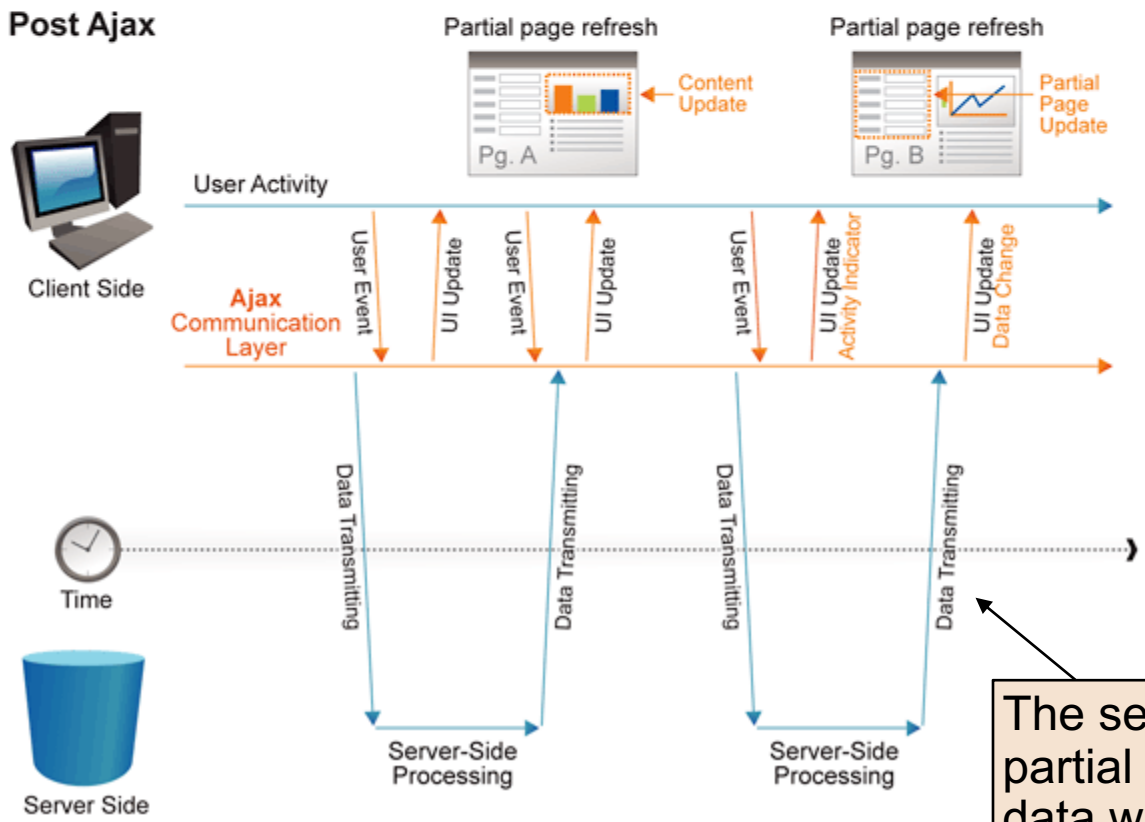
- JSON (JavaScript Object Notation) is a lightweight data-interchange format.
  - It is easy for humans to read and write.
  - It is easy for machines to parse and generate.
  - It is based on a subset of the JavaScript Programming Language.
- JSON is built on two structures:
  - A collection of name/value pairs.
  - An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.
- Detailed definition at http://www.json.org

AJAX

- AJAX (Asynchronous JavaScript and XML) is a group of interrelated web development techniques used on the client-side to create interactive web applications.
  - Despite the name, the use of XML is not actually required, nor do the requests need to be asynchronous.
- AJAX incorporates:
  - standards-based presentation using HTML and CSS;
  - dynamic display and interaction using the Document Object Model;
  - data interchange and manipulation using XML, JSON, or preformatted HTML;
  - asynchronous operations and communications using XMLHttpRequest;
  - JavaScript binding everything together.

Ajax Model

- With Ajax, web applications can communicate with servers in the background without a complete page loading after every request/response cycle.



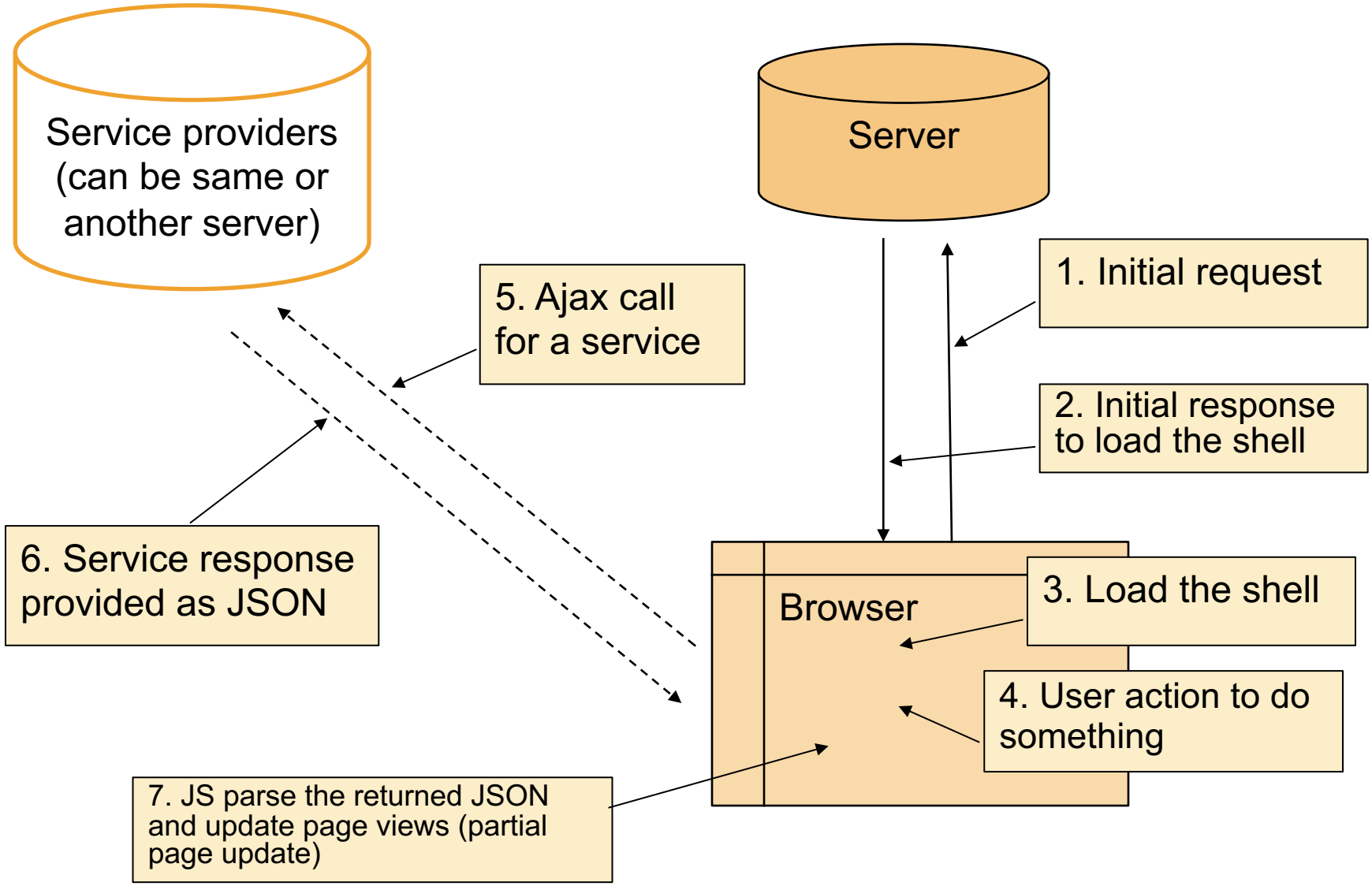
The server prepares partial pages or just data without markups.

http://www.websiteoptimization.com/secrets/ajax/8-1-ajax-pattern.html

JavaScript/jQuery

- JavaScript unified everything together
- AJAX is handled by JavaScript and the browser
- JSON is a native JavaScript object and can be directly parsed and read by JavaScript once it is returned from the server
- UI (views) is managed by JavaScript through DOM manipulation (dynamic HTML)
- Actions/events are handled by JavaScript
- JavaScript consumes services
- https://stateofjs.com

A Simple Complete Cycle



Benefits

- Dynamic and interactive
- Loads quickly, smooth transition between views
- Client based processing, highly responsive
- See some discussion here
  - https://stackoverflow.com/questions/21862054/single-page-application-advantages-and-disadvantages

Challenges and More Issues

- Navigation, browser history
- Client data model/storage
- Session/state control - on the server or client?
- Two way communication (duplex)? (Web socket, push)
- Data binding methods
- Frameworks
- Web analytics of SPA
- SEO, not search friendly

Impact on Job/Career

- Frontend developer
  - https://css-tricks.com/the-great-divide/

Good Resources

- Core readings
  - JavaScript's role: https://www.applause.com/blog/javascript-is-eating-the-world
  - The new Web application architectures and their impacts for enterprises
    - http://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-1/
    - https://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-2/
  - Client centered web app architecture: http://www.slideshare.net/bastila/sencha-web-applications-come-of-age
  - An introduction to single page applications: http://www.johnpapa.net/pageinspa/ - SPA will be our focus of the course (project) throughout the semester. Get some initial conceptual understanding first. Read the comments as well.
  - What is Serverless Architecture? https://hackernoon.com/what-is-serverless-architecture-what-are-its-pros-and-cons-cc4b804022e9

- JavaScript's role:
  - https://dannorth.net/2011/12/19/the-rise-and-rise-of-javascript/comment-page-1/
  - https://www.itprotoday.com/web-development/explore-new-world-javascript-focused-client-side-web-development

- The new Web application architectures and their impacts for enterprises
  - https://mobidev.biz/blog/3_types_of_web_application_architecture
  - http://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-1/
  - https://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-2/
  - Client centered web app architecture: http://www.slideshare.net/bastila/sencha-web-applications-come-of-age

- Single page app
  - https://johnpapa.net/spa/
  - http://singlepageappbook.com
  - http://tutorialzine.com/2015/02/single-page-app-without-a-framework/
  - https://en.wikipedia.org/wiki/Single-page_application

- Serverless
  - https://www.fullstackfirebase.com/introduction/what-is-serverless
  - https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/serverless/web-app