Web Application Development

Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology

http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Web Applications (Apps)

Data-centric applications in
the Browser.

Agenda

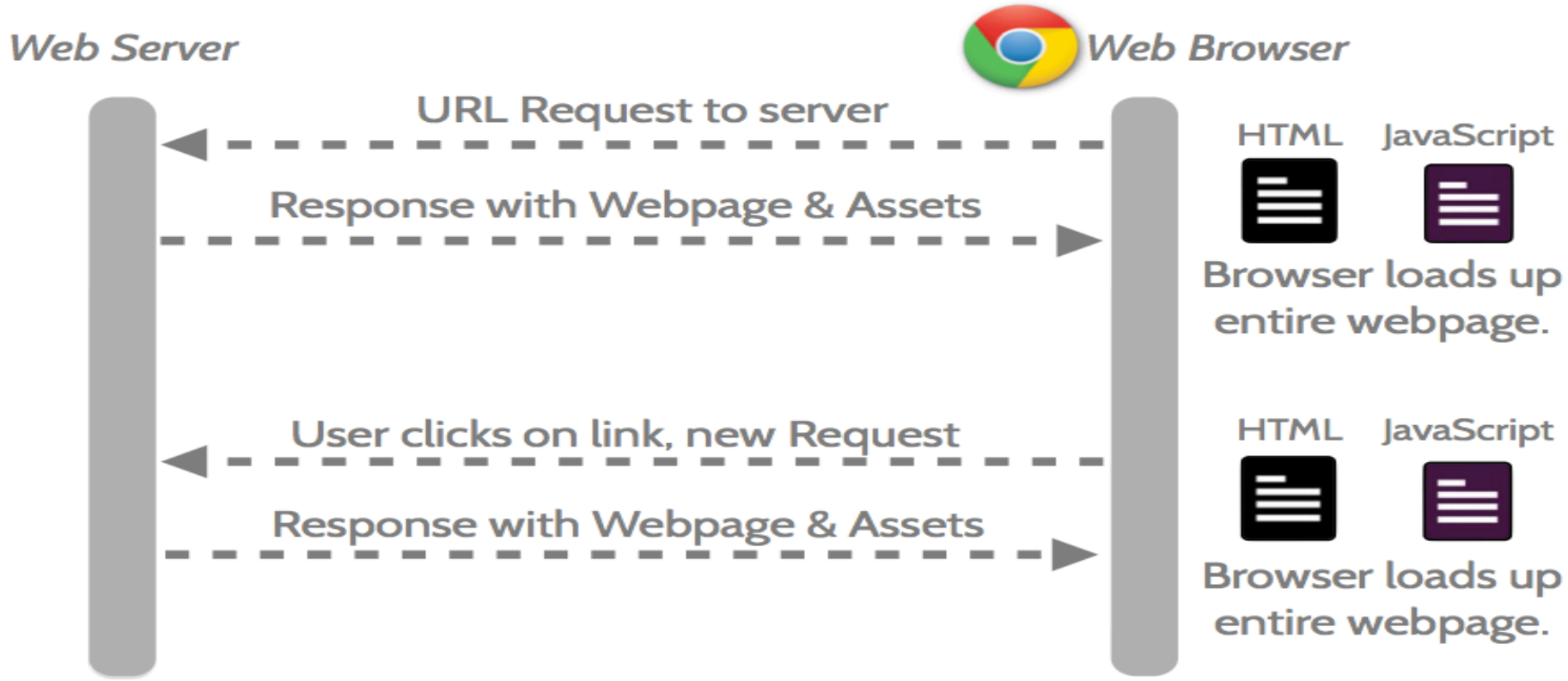
- Early Web Apps
- Web App Evolution & AJAX
- Single Page Application Frameworks (SPAs)
- Design Patterns – Module View Controller (MVC)

Early Web Apps

- Characteristics:
  - Server creates pages / browser displays.
  - Data input sent to and processed by the server
  - Updated pages created on the server and resent to browser.

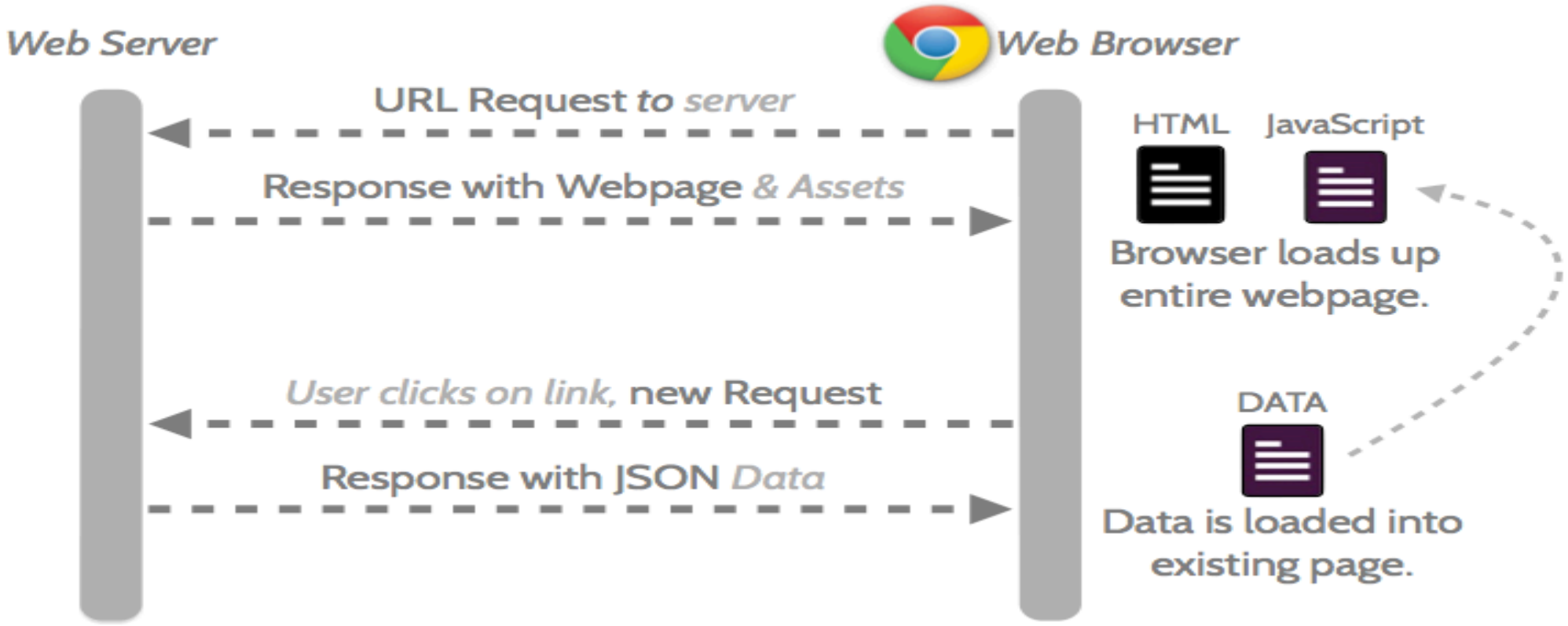
- For Example - PHP, JSP/Servlets, Struts, (and more recently) Ruby on Rails.

Early Web Apps.



- Disadvantages:
  - Bad UX (User eXperience) – reload of UI parts
  - Poor performance

Web Evolution – AJAX (Asynchronous Javascript And Xml)



What is AJAX?

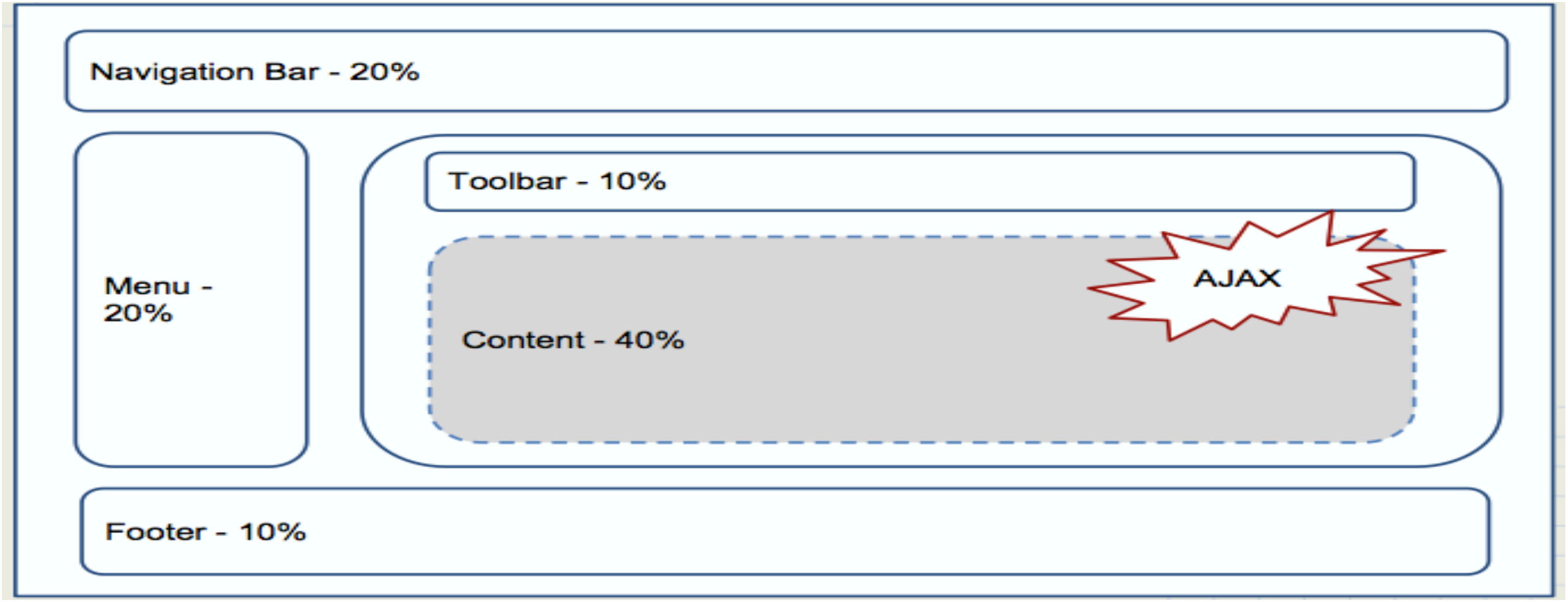
- AJAX = Asynchronous JavaScript and XML.
- Not a language but a technique.
- A technique for loading data (formatted as XML) in the background (asynchronous) and displaying it on the webpage, without reloading the whole page.
  - JSON formatting is now favoured over XML
- Examples: Gmail, Google Maps, Youtube.

Simple AJAX example (using JQuery)

- **$.get(URL,callback)** – Send HTTP request to URL; Execute callback function when response arrives.

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.
    min.js"></script>
5   <script>
6   $(document).ready(function(){
7       $("button").click(function(){
8           // AJAX request
9           $.get("http://localhost:8080/sample.txt",function(data,status){
10              $('h2').text(data)
11          })
12      })
13  })
14  </script>
15  </head>
16  <body>
17      <h2>Let jQuery AJAX Change This Text</h2></div>
18      <button>Get External Content</button>
19  </body>
20  </html>
```

Web Evolution - AJAX



Static

Dynamic

Web Apps - AJAX

- Interactive client-side web:
  - Collect input from user.
  - Update display.
  - Communicate with server (AJAX)

- Client-side processing enabled by:
  - JavaScript.
  - DOM manipulation.
  - HTTP server messaging.

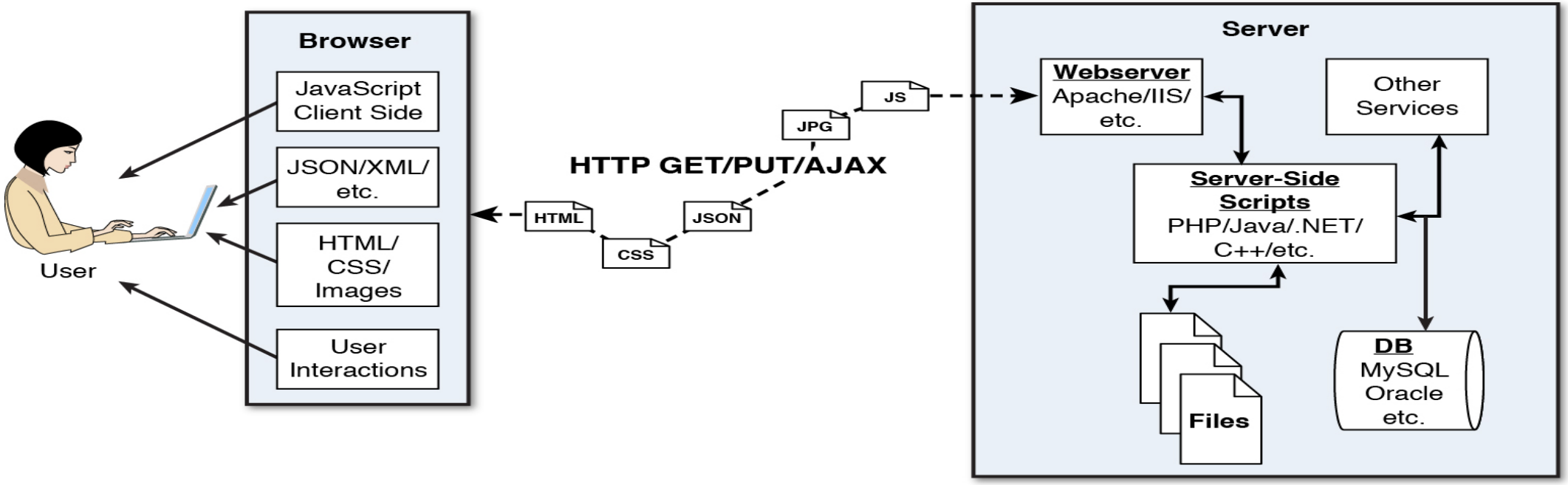
Web Evolution – SPA Frameworks

- SPA (Single Page Application) frameworks – Client-side Javascript MVC frameworks. (MVC covered below)
  - Examples:
    - AngularJS; Backbone; EmberJS; Knockout, …. many more …
- Benefits:
  - Less boilerplate code.
  - Less effort on mundane programming tasks; allowing for better focus on what is ACTUALLY VALUABLE – THE LOGIC.
  - More efficiency in development.
  - Better client-side code architecture through clearer Separation of Concerns (SoC).
- All are built on core browser functionality – event-driven, asynchronous ; DOM manipulation; HTTP server communication

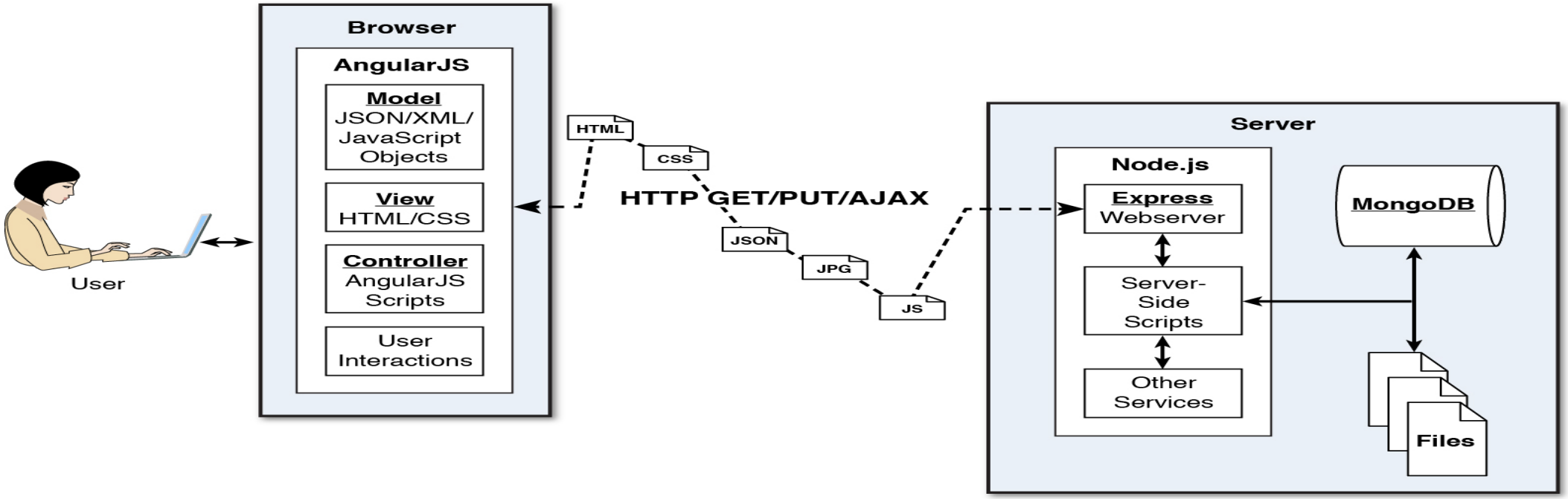
Web Evolution – SPA frameworks

- Library Vs Framework
  - Library (e.g. JQuery)
    - Passive functionality; Invoked by the application.
  - Framework (e.g. AngularJS)
    - Provides application architecture (SoC); Deals with common mundane requirements; Invokes application code.

Components of a basic Web App



Components of a MEAN Web App



Design Patterns

What is a Design Pattern? (High Level View)

A pattern describes a problem which occurs over and over again in our environment,

and then describes the core of the solution to that problem,

in such a way that you can use this solution many times over,

without ever doing it the same way twice“

(Christopher Alexander, et al : “A Pattern Language: Towns/Buildings/ Construction”, Oxford University Press, New York, 1977)

What is a Design Pattern? (S/W View)

- Description of communicating objects and classes that are customized to solve a general design problem in a particular context.

  (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns – Elements of Reusable Object-Oriented Software", Addison-Wesley, 1994 (22nd printing July 2001))

- Each pattern focuses on a particular object-oriented design problem or issue.

Elements of Design Patterns (1/2)

1. Pattern Name
   - Increases design vocabulary, higher level of abstraction.
2. Problem
   - When to apply the pattern
   - Problem and context, conditions for applicability of pattern

Elements of Design Patterns (2/2)

3. Solution
   - Relationships, responsibilities, and collaborations of design elements.
   - Not any concrete design or implementation, rather a template
4. Consequences
   - Results and trade-offs of applying the pattern
   - Space and time trade-offs, reusability, extensibility, portability

Design Pattern Space

Purpose		
Creational	Structural	Behavioral
1. Factory Method 2. Abstract Factory 3. Builder 4. Prototype 5. Singleton	1. Adapter 2. Bridge 3. Composite 4. Decorator 5. Facade 6. Flyweight ***7. Model-View-Controller*** (MVC) 8. Proxy	1. Interpreter 2. Template Method 3. Chain of Responsibility 4. Command 5. Iterator 6. Mediator 7. Memento 8. Observer 9. State 10. Strategy 11. Visitor

Architecture Pattern - MVC

- "A software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts ....."

- MVC provides clean separation of concerns (SOC) of:
  - data (Model)
  - presentation (View)
  - user input (Controller)

Architecture Pattern - MVC

Model

- **Represents knowledge or data**
- **Talks to the server**
- **Isolated from controllers and views**

View

- **Visual presentation of something related to the current state of the Model**
- **Is generated from a Template**
- **One model might have multiple views**

Controller

- Glue between Model and View
- Handles user interactions
- Might perform business logic role

The role of controller greatly varies from framework to framework

Architecture Pattern - MVC

- MVC spawned many variations:
  - Model-View-ViewModel (MVVM)
  - Model-View-Presenter (MVP)
  - http://joel.inpointform.net/software-development/mvvm-vs-mvp-vs-mvc-the-differences-explained/

- MV*/MVW (Whatever)

Questions?