

Basics of Web Development

Lecture 14

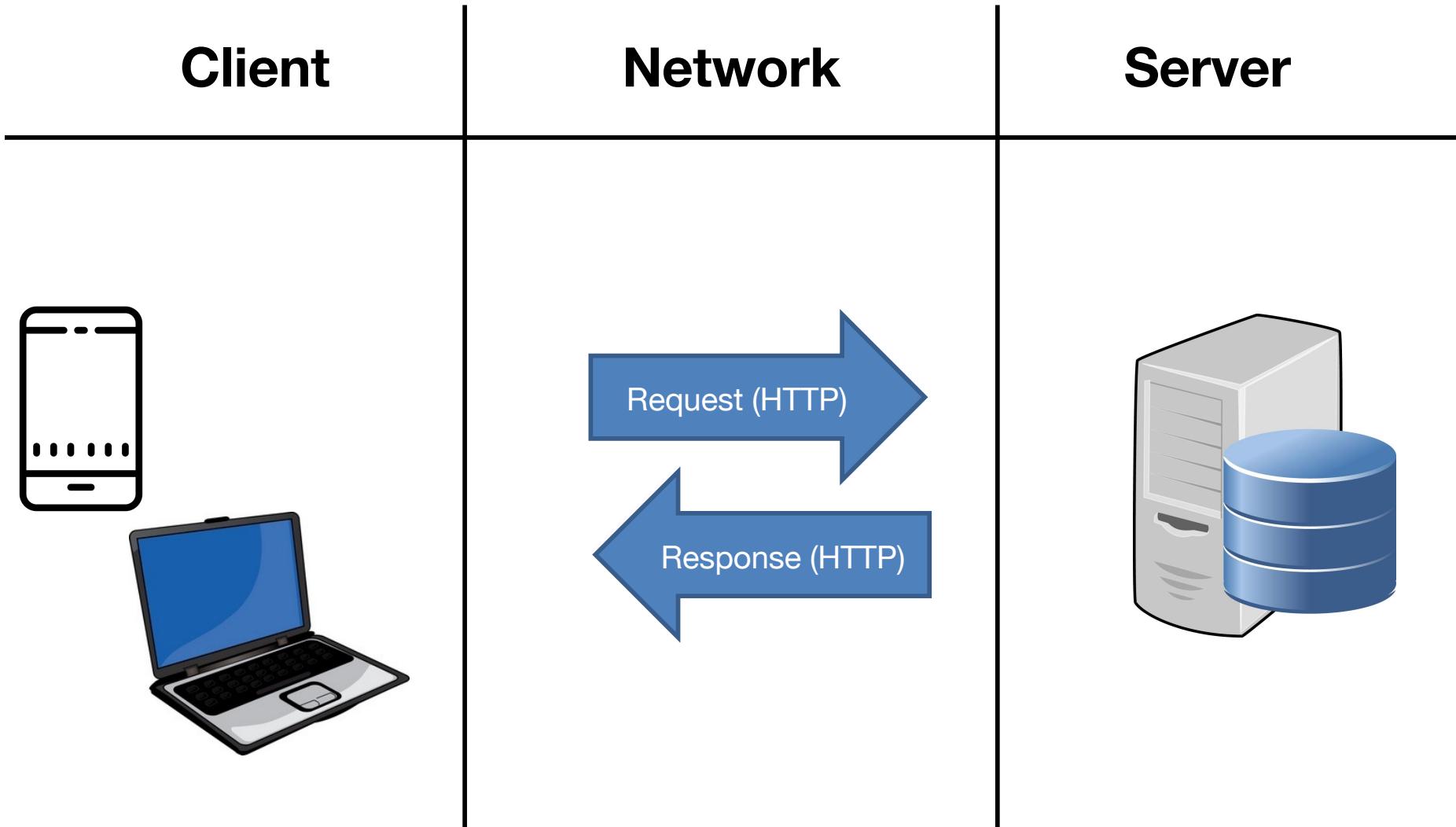


Outline

1. Big Picture
2. Client Side
3. Server Side



Big Picture



Client

- Any software capable of issuing HTTP requests (and processing responses)
 - Most common: web browser
- “Apps” commonly issue HTTP requests on your behalf as a standardized communication layer

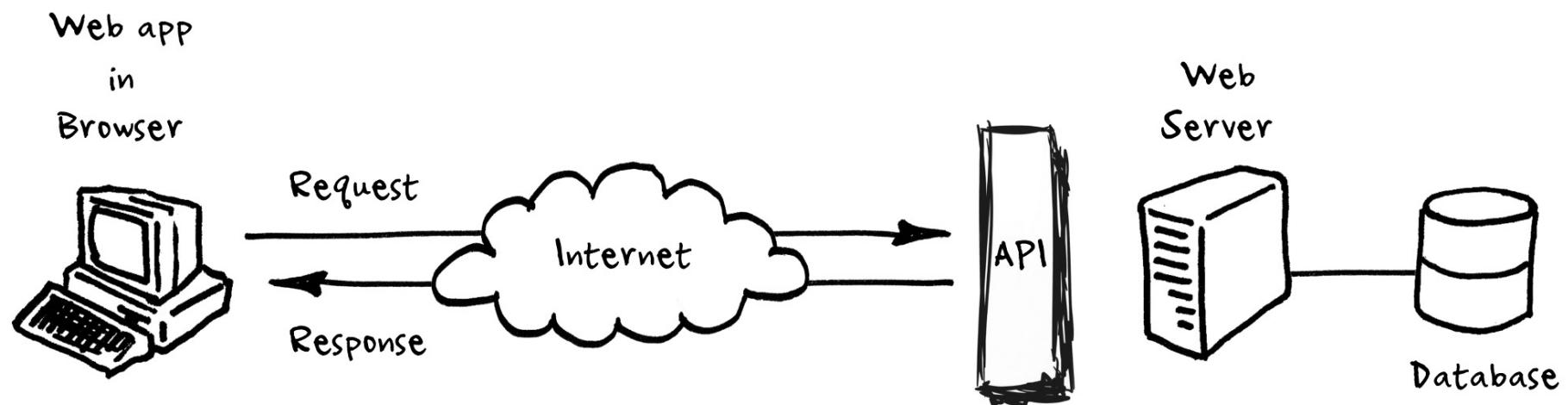


Server

- Any software listening for HTTP requests on one/more ports (and responds)
- Commonly a buffer layer in a 3 (or more) tier architecture



3-Tier Architecture



Hypertext Transfer Protocol (HTTP)

- Application protocol for distributed, client-server communication
- Session
 - Request (port, method, headers, message)
 - Response (status, headers, message)
- Stateless
 - Cookies, server sessions, hidden form data



Example

Request: www.example.com

```
GET /index.html HTTP/1.1
```

```
Host: www.example.com
```

Response

```
HTTP/1.1 200 OK
```

```
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Encoding: UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close
```

```
<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    Hello World!
  </body>
</html>
```



HTTP Request

- TCP port
 - Usually 80 (http), 443 (https)
- URL

`http(s)://user:pass@domain:port/path?query#anchor`
- Method: intended effect
 - **GET:** “safe” representation (in URL)
 - **POST:** add
 - PUT: replace/add
 - DELETE: delete
 - OPTIONS: get
- Headers: operational parameters



HTTP Response

- Status code, common...
 - 200=ok, 404=not found, 403=forbidden, 500=server error
- Headers: operational parameters
- Message body
 - Document (HTML, XML, JSON), image, ...



Maintaining State: Cookies

Client

```
GET /index.html HTTP/1.1
```

```
Host: www.example.org
```

```
...
```

```
GET /spec.html HTTP/1.1
```

```
Host: www.example.org
```

```
Cookie: theme=light;  
sessionToken=abc123
```

```
...
```

Server

```
HTTP/1.0 200 OK
```

```
Content-type: text/html
```

```
Set-Cookie: theme=light
```

```
Set-Cookie: sessionToken=abc123;  
Expires=Wed, 09 Jun 2021 10:18:14 GMT
```

```
...
```



Maintaining State: Sessions

- Basic idea: server provides client a “token” that uniquely identifies the session data
- Language support
 - e.g., PHPSESSID



Maintaining State: Form Data

- Basic idea: forms have hidden fields with any necessary information to maintain client-server synchronization



Web Development

- Creating server-side (back-end) “producers” of content and client-side (front-end) “consumers”
- Good
 - Platform independent (somewhat)
 - Lots of tools, libraries, etc.
- Bad
 - MANY languages/technologies at play
 - Baseline: HTML, CSS, JS



Client-Side Technologies

- Document structure/content
 - **HTML**, XML
- Document styling
 - **CSS**
- Dynamics
 - **JavaScript**, (Java/Flash/Silverlight/...)
- Communication
 - AJAX via XML/JSON
- Tools
 - Inspection, jQuery, Bootstrap, Angular/React, CDN



Hypertext Markup Language (HTML)

- Markup to support structured documents
- Semantics for...
 - Text (headings, paragraphs, lists, tables, etc.)
 - Multimedia (images, music, video, etc.)
 - Links to other resources
 - Forms
 - Styling
 - Scripting



HTML Hello World

```
<html>
  <head>
    <title>howdy</title>
  </head>
  <body>
    <p>hello</p>
    <p><a href="https://helloworldcollection.github.io">world</a></p>
  </body>
</html>
```



HTML Elements/Tags

Every tag has a typical syntax...

- Start/end tag

```
<element></element>
```

- Attribute/value pair(s)

```
<element key1="value1" key2="value2" ... >
```

- Content

```
<element key="value">content</element>
```

- If no content, may see

```
<element key="value" />
```



Typical HTML Document

- **head**
 - **title**: shown in browser
 - **meta**: used for automated processing
 - **style, script**, etc.
- **body**
 - **p**: paragraph
 - **img**: image
 - **ul, ol**: unordered/ordered list
 - **li**: list item
 - **a**: “anchor” (link)
 - **form, table**
 - **div**: “division” (logical grouping)



HTML Forms

- Attributes dictate where an HTTP request is intended to proceed, and how
 - Method: GET/POST
 - Action: URL
- Many elements
 - Single/multi-line input
 - Single/multi lists
 - Check/radio boxes
 - File uploads
 - Buttons

Email address

We'll never share your email with anyone else.

Password

Check me out



Extensible Markup Language (XML)

- Serves an important role for a common, computer-readable data exchange format
 - Common in software products, business exchanges
- Has fallen out of favor as a web document format
 - XHTML, XML+XSLT
 - AJAX



Document Styling

- It can be advantageous to separate document structure/content from presentation
 - Supports modularity, consistency, maintainability
- Cascading Style Sheets (CSS) is a language for describing document presentation semantics
 - Fonts, layout, colors, etc.
 - Hierarchical, object-oriented
 - Support for medium-specificity



CSS Example

```
body {  
    background-color: black;  
    color: white;  
    font-family: Georgia, "Times New Roman";  
    font-size: 16px;  
}  
  
p {  
    padding: 10px 0px;  
}  
  
.heavy {  
    font-weight: bold;  
}
```



CSS Usage

- Element

```
<p style="...">
```

- Document

```
<head>
    <style type="text/css">...</style>
</head>
```

- Linked

```
<head>
    <link rel="stylesheet"
          href="style.css"
          type="text/css"
          media="print" />
</head>
```



Typical CSS

- Good style: minimalist, descriptive HTML + site-linked CSS
 - Improves readability, consistency, accessibility
- Different browsers (e.g., mobile) have different CSS interpretations (and starting configs), hence UI libraries
 - Bootstrap! (more later)



Dynamics

- Historically webpages were static, with server interaction required
- As the web evolved, technologies emerged to make sites more interactive locally
 - Java applets, Flash, Silverlight, ...
- For reasons of security, “politics” (e.g., Apple vs Flash), etc., JavaScript is dominant



JavaScript Features

- Document Object Model (DOM)
 - Exposes HTML elements to programmatic manipulation
 - Provides event hooks (`onclick="..."`)
 - JavaScript Object Notation (JSON): human-readable text to transmit data objects
- Interpreted, C/Java-like syntax
 - Functions (higher order)
 - Classes
- Missing types
 - Hence TypeScript :)



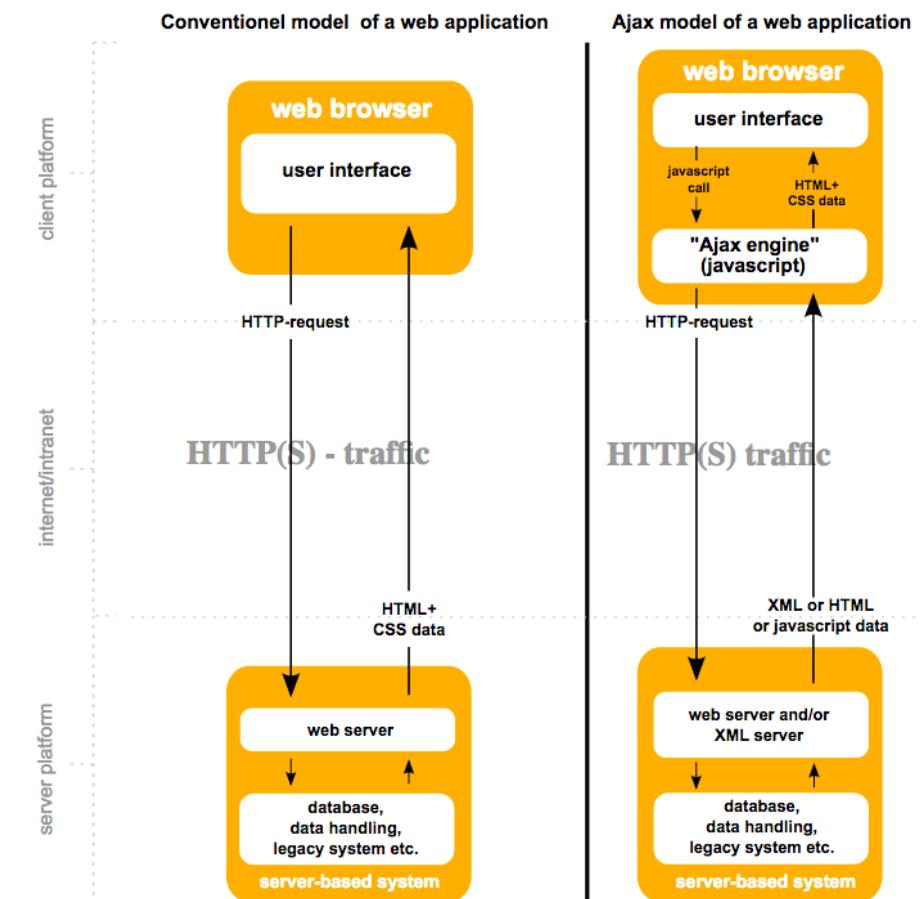
JSON Example

```
{  
    "firstName": "John",  
    "lastName": "Smith",  
    "isAlive": true,  
    "age": 25,  
    "address": {  
        "streetAddress": "21 2nd Street",  
        "city": "New York",  
        "state": "NY",  
        "postalCode": "10021-3100"  
    },  
    "phoneNumbers": [  
        {"type": "home", "number": "212 555-1234"},  
        {"type": "office", "number": "646 555-4567"},  
        {"type": "mobile", "number": "123 456-7890"}  
    ],  
    "children": [],  
    "spouse": null  
}
```



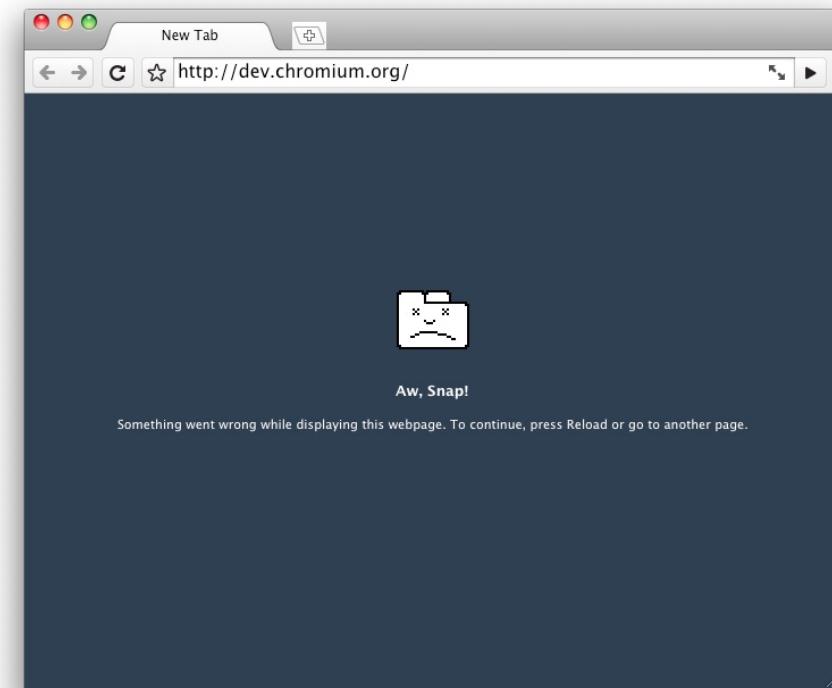
Asynchronous JavaScript (AJAX/J)

- Basic idea: don't reload the page in order to talk to the server
- Originally paired with XML, now more commonly JSON
- Makes for a consistent format of data exchange between multiple platforms (e.g., web, iOS, Android, ...)



JavaScript Pitfalls

- Differences in browser (version) support/implementation
 - jQuery + Bootstrap
 - Angular/React/Vue
- Security
 - Cross-site scripting (XSS)
- Speed
- Accessibility
 - Pages should gracefully degrade



Cross-Site Scripting (XSS)

Basic idea...

- (Using one of several methods) embed evil code into a site a user trusts
- The code acts as the user (i.e., via stored credentials/data) to steal data, perform actions, ...

Common types...

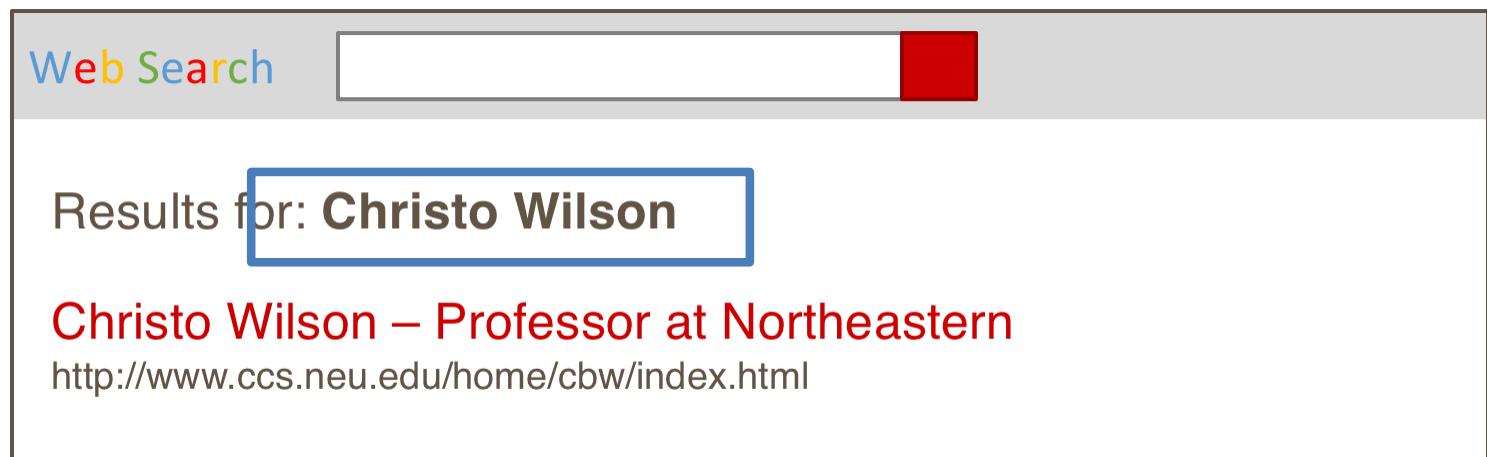
- Reflected (code comes from URL)
- Stored (code comes from backend data)



XSS Example: Reflection Opportunity

Assume a search site...

<http://www.websearch.com/search?q=Christo+Wilson>



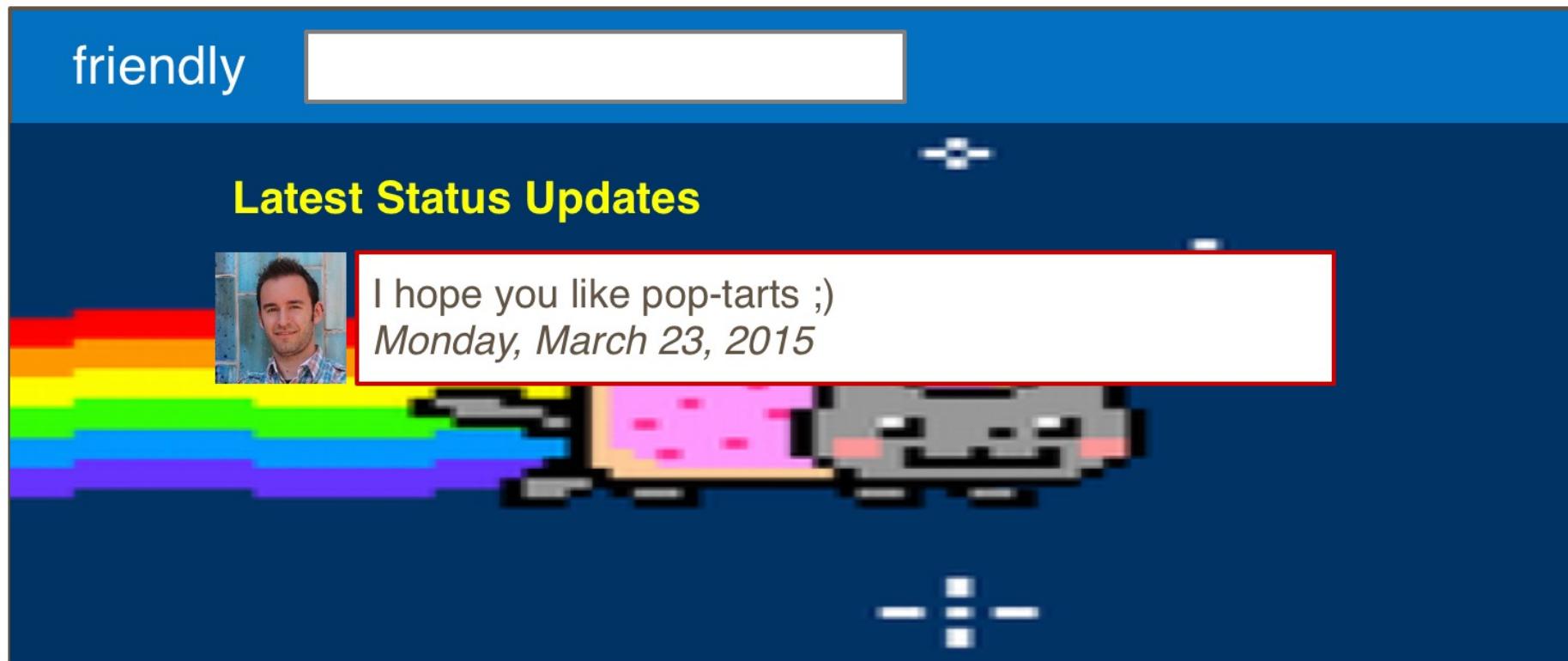
XSS Example: Reflected Attack

```
http://www.websearch.com/search?q=<script>document.write('
```

Update Status

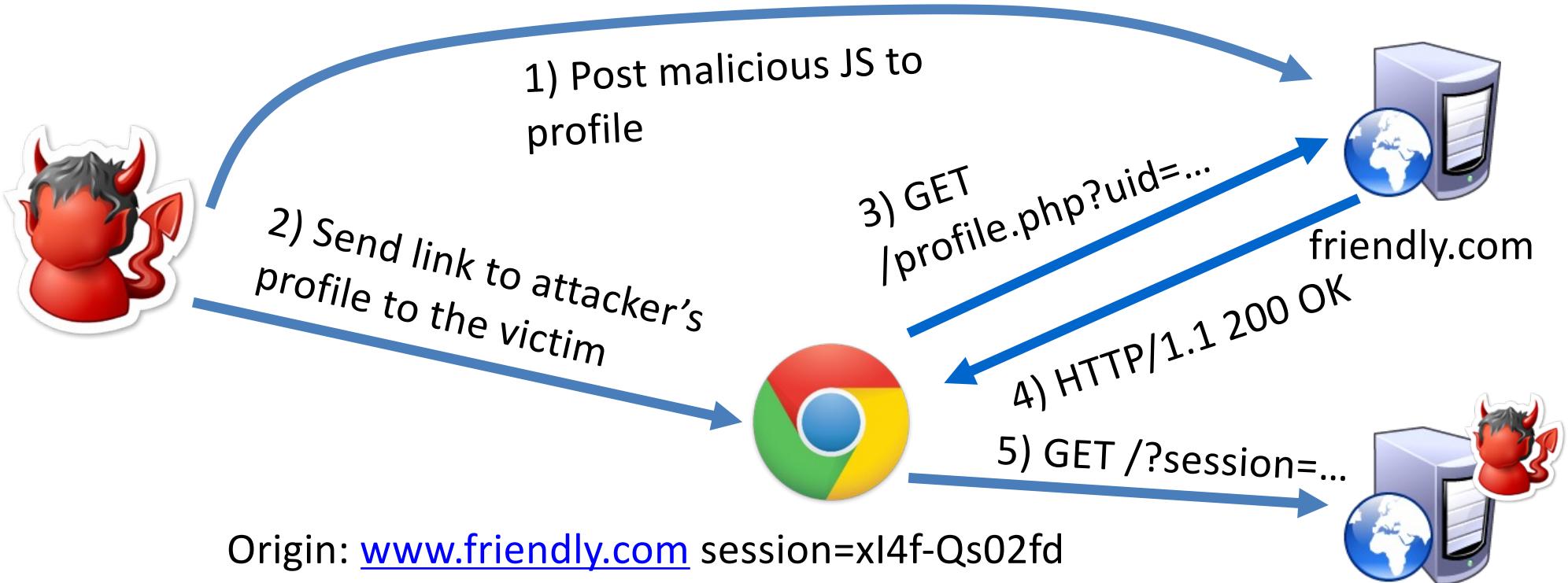


XSS Example: Stored Opportunity



XSS Example: Stored Attack

```
<script>
  document.write('');
</script>
```

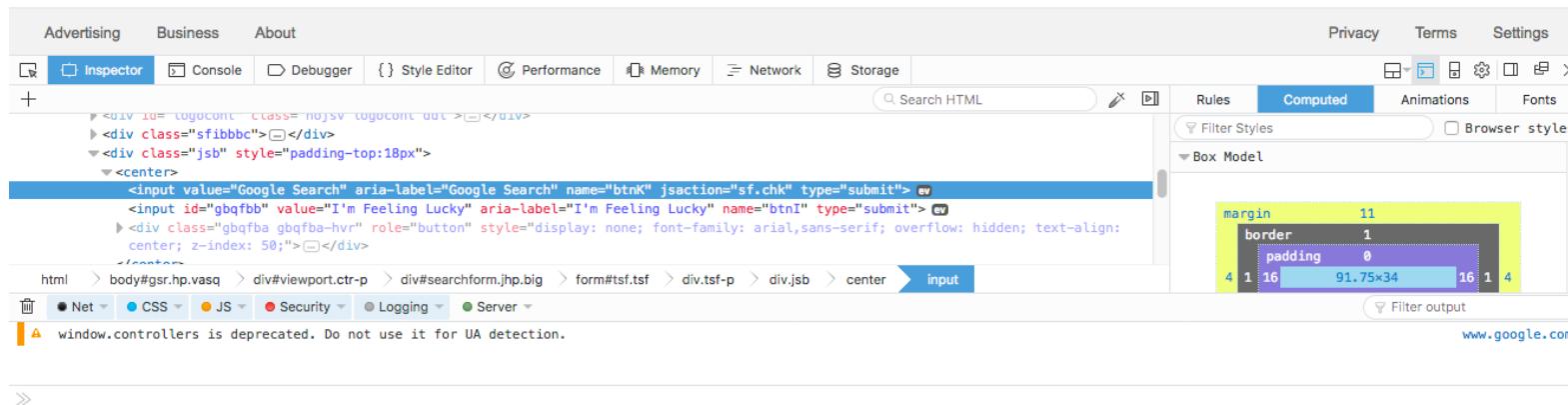
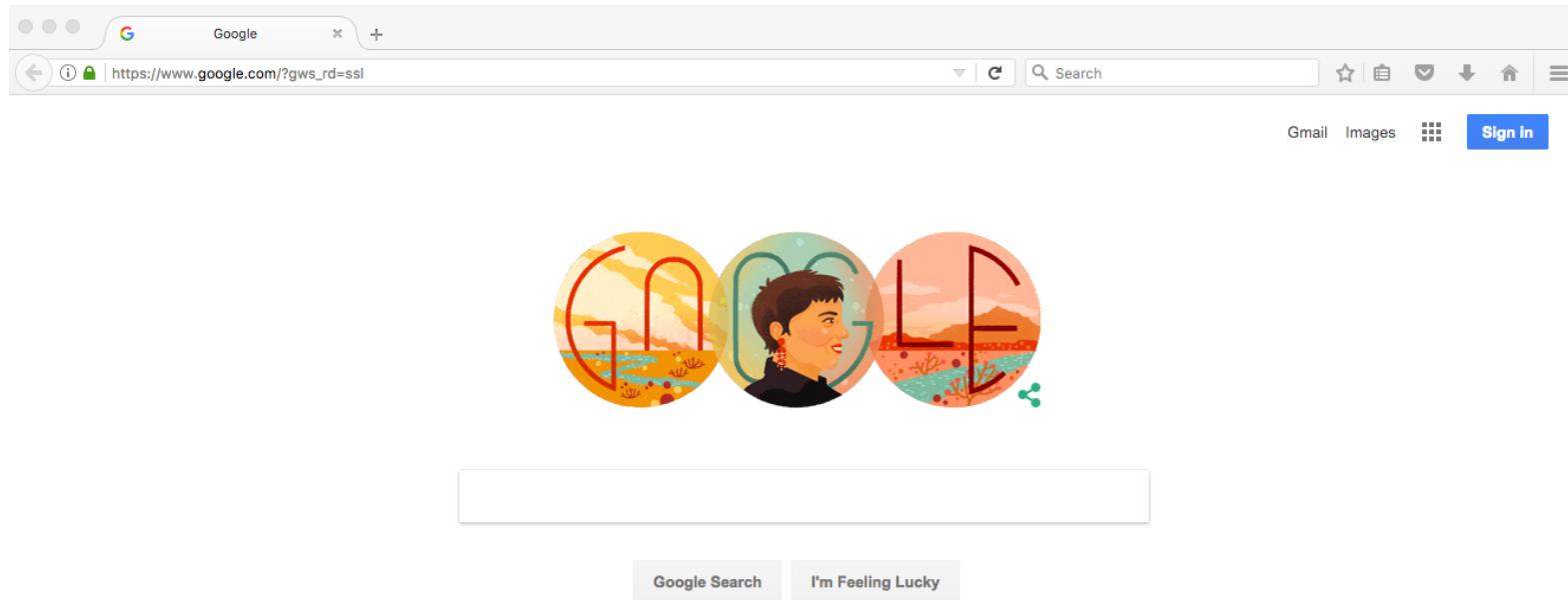


Best-Effort XSS Prevention

- **Validate all input**
- **Filter all output**



Web Inspector



Inspect!

Built into browsers, good for...

- Seeing site code
 - Understanding the DOM
 - And changing, temporarily :)
 - Seeing computed CSS
 - Debugging JS
 - Seeing different device effects
 - “Responsive” design
 - Profiling
- ...



jQuery

- Cross-platform, open-source JavaScript library designed to simplify client-side scripting
 - DOM selection/traversal/manipulation/events, plugins
- Example: when the page loads, hide any paragraph on the page that is clicked

```
$(document).ready(function(){
        $("p").click(function(){
                $(this).hide();
        });
});
```



Bootstrap

- Most commonly used front-end framework
- Standardized CSS, JavaScript, HTML
- Makes it easy to produce good-looking, responsive, cross-browser websites



Webapp Libraries/Frameworks

- Most common: Angular, React
 - Also Vue
- Try to facilitate front-end applications built on web technologies
 - Desktop & mobile

```
<div id="myReactApp"></div>

<script type="text/babel">
  function Greeter(props) {
    return <h1>{props.greeting}</h1>
  }
  var App = <Greeter greeting="Hello World!" />;
  ReactDOM.render(App,document.getElementById('myReactApp'));
</script>
```



Content Delivery/Distribution Network (CDN)

- Geographically distributed servers for quickly providing reliable access to (typically) static content
- Good for...
 - Improving user experience (e.g., Netflix)
 - Avoiding DDoS ([Distributed] Denial of Service)
 - Distributing code/fonts (e.g., Bootstrap)



Server-Side Technologies

- Web Server
 - Apache, nginx, IIS, Tomcat, Node
 - Hosting, Containers
- Languages
 - PHP, Python, Ruby, Java/JSP, ASP, CGI, JS
 - Content Management System (CMS)
- Databases
 - MySQL, SQL Server, PostgreSQL, NoSQL



Web Server

- The primary role of a web server is to satisfy client HTTP requests
 - Other: virtual hosting, throttling, scripting, security, ...
- Typical process
 1. Receive HTTP request
 2. Reference configuration
 3. Retrieve resource
 4. Send HTTP response



Hosting

- Your own machine: avoid for production
 - Security, scaling
- Remote
 - Shared, dedicated hosting
 - Virtual Private Server (VPS) = VMs
 - Control panel (e.g., CPanel)
 - SSH/SFTP
 - Containers (e.g., Heroku)
 - Cloud for storage, networking, computing, etc.
 - Amazon Web Services, Microsoft Azure, Google Cloud



Server-Side Scripting

- The web server executes a script whose result is packaged and sent to the client as an HTTP response
 - HTML/JSON document
 - Image
 - Download
- ...



Example PHP Script

```
<?php
    echo '<html>';
        echo '<head>';
            echo '<title>howdy</title>';
        echo '</head>';
        echo '<body>';
            echo '<p>hello world</p>';
        echo '</body>';
    echo '</html>';
?>
```



Model-View-Controller (MVC)

- A design pattern to separate code for...
 - Business logic (model)
 - Output representation (view)
 - Routing requests (controller)
- Common, embedded within many frameworks



Content Management Systems (CMS)

- Web application that provides infrastructure for managing data
 - Data management, editing, workflows, syndication, collaboration/delegation, etc.
 - Standardized client- and server-side components
- Examples
 - Wikis (MediaWiki), Blogs (WordPress)
 - Drupal, Joomla
 - SharePoint



Favorite Resource: W3Schools

<https://www.w3schools.com>

- Quick Start + Reference
 - HTML
 - CSS
 - JavaScript
 - Bootstrap, JQuery, React, Angular
 - PHP, Python, Node.js
 - SQL



Summary

- Web development usually involves developing client- & server-side components (talk via HTTP)
- Client-side (front-end) is usually HTML, JS, CSS
 - With great frameworks
 - Mostly stateless
- Server-side (back-end) is usually some language (e.g., PHP, Python, Ruby, JS) with a data layer
 - Commonly talking to a relational database directly or possibly via cloud services

