Web Services

Today's Topics

- Web Services
  - Services as RPC
    - Sending objects between clients and servers over HTTP (SOAP)
    - Describing what services are available (WSDL)
  - Services with REST
    - Exposing information as URLS
    - Supporting REST actions

Web Services

- Web Applications: Client is a web browser
  - Requests are made, along with query parameters
  - HTTP Requests are made to server, HTTP Response is generally a new HTML page

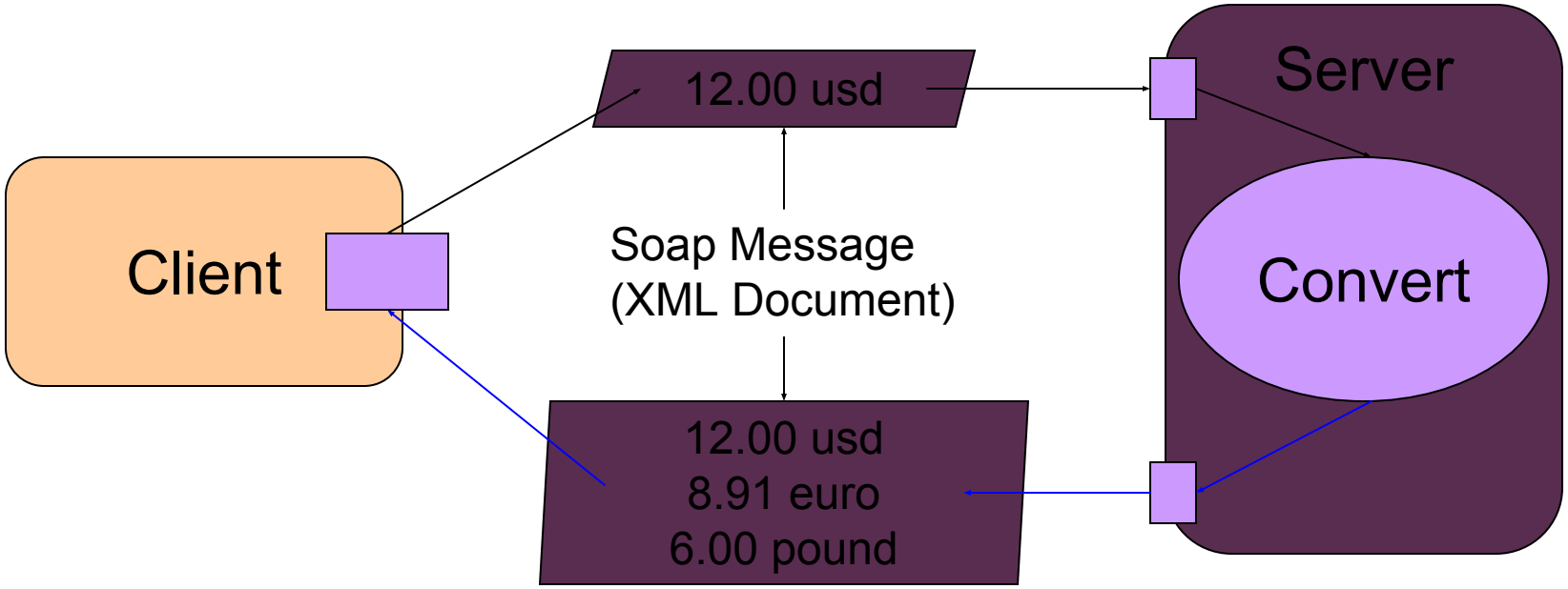
- Web Service:Client is ***any*** program
  - Messages are still passed using HTTP
  - Requests have parameters, but not usually part of query string
  - Response is data - not presentation!

Web Services as RPC

- RPC: Remote Procedure Call
  - Web Services are really just functions that are called by the client, executed by the server, which then returns the result.
  - RPC has been around forever - but the communication between client/server was not “standardized”
  - Web Services: Communication via XML over HTTP

SOAP

SOAP is a way to pass parameters and return values between client and server.



Note, a single server may have multiple services

SOAP Details

- SOAP: **S**imple **O**bject **A**ccess **P**rotocol
- The reason we need SOAP is that often parameters and return values are not simply “primitives” - they can also be complex objects
- SOAP is an XML Specification for encoding arbitrary, complex data types
- It is useful to understand the protocol - however we will generally automate the encoding/decoding process

WSDL

- SOAP is a protocol for *sending* messages
- WSDL: Web Service Description Language
  - Describes what services (functions) are available
  - Describes **what** messages will be sent back and forth, and their types
- WSDL is XML
- A server posts a WSDL file in an accessible location
  - Client software (or programmers) can access it to determine how to use the services being provided

REST

- RPC-Style services with SOAP tend to be a bit complex (at least to start with)
- REST offers a more basic approach:
  - **Re**presentable **S**tate **T**ransfer
  - Developed by one of the authors of HTTP
  - Think of services are resources, and a URL as a mechanism to browse through

REST

- REST is more than web services - it's the idea that all information (whether dynamic or static) should be accessible as a URL
  - The WWW **is** RESTful
- Web Services can fit within this model too - which is why we hear about "RESTful" web services

REST Design

- Each “thing” in your application should have an ID.
  - Map URL to **list** things, which contains URLS to get details of things
  - Use HTTP Verbs to indicate to the server what your want to do.
    - GET: Retrieve information (read only)
    - POST: Add new information
    - PUT: Update existing information
    - DELETE: Remove information

Example

- Server has a list of customers.
  - You want a web service that allows a client to **list** customers
  - View customer details
  - Edit customers
  - Create a new customer
  - Delete a customer.

Example

Expose a simple URL structure:

URL: http://www.example.com/customers

HTTP GET: Returns a list of customers (JSON, usually)

HTTP POST: Add new customer (likely JSON data)

Supply links to all customers in the GET response	http://example.com/customers/10101 http://example.com/customers/10102 http://example.com/customers/10103

Example

Individual resources identified by ID (perhaps DB ID)

URL:	http://www.example.com/customers/10102
HTTP GET:	Returns details of customer #10102
HTTP POST:	Updates customer #10102
HTTP DELETE:	Delete customer #10102

Example

- Often we associate lists of things with some things - like a customer who has many orders
  - We would have a URL scheme for orders
    - http://example.com/orders
    - http://example.com/orders/14
  - But we could get orders by a particular customer too
    - http://example.com/customer/10102/orders
    - http://example.com/customer/10102/orders/14

Server-Side - Routes

Notice the URL's have ID's in them - this can easily be accomplished with express

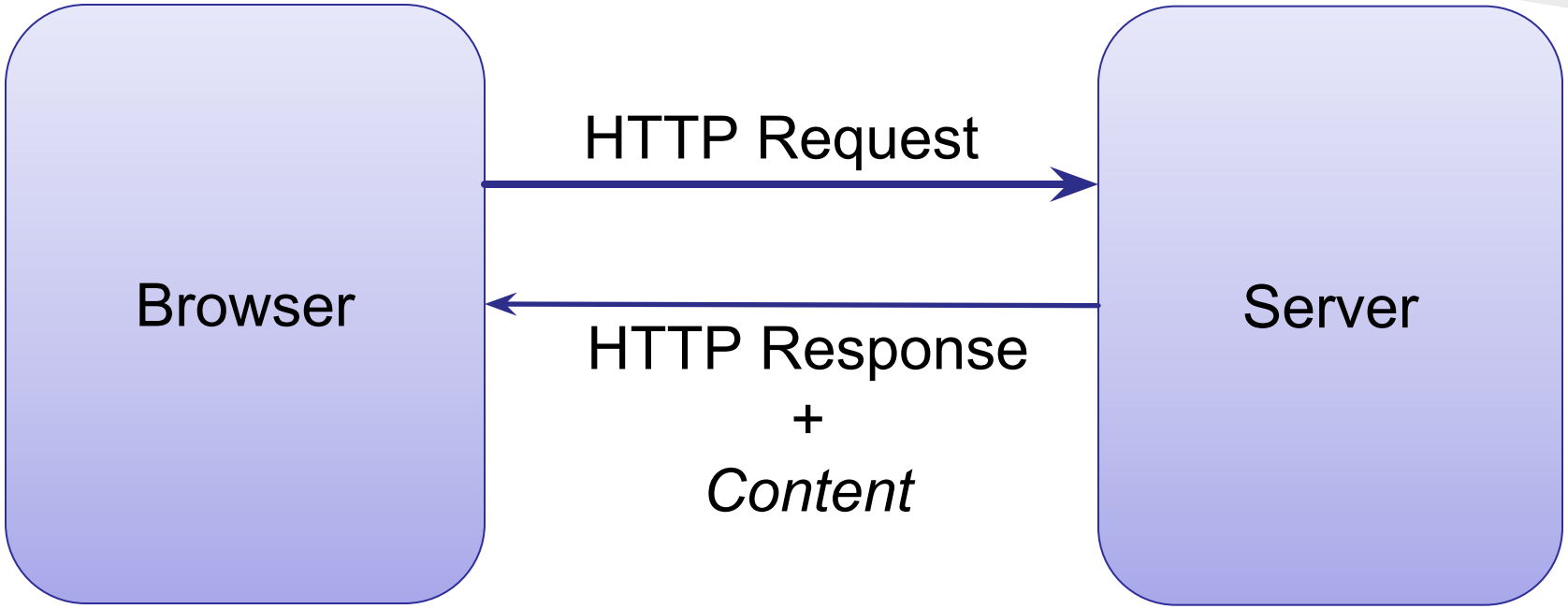
```
router.get('/users/:user', function(req, res, next) {

    You can use req.user to access whatever string was provided in the URL,

    for example:

    if /users/13 was the request, req.user would be '13'

}
```

Important Point

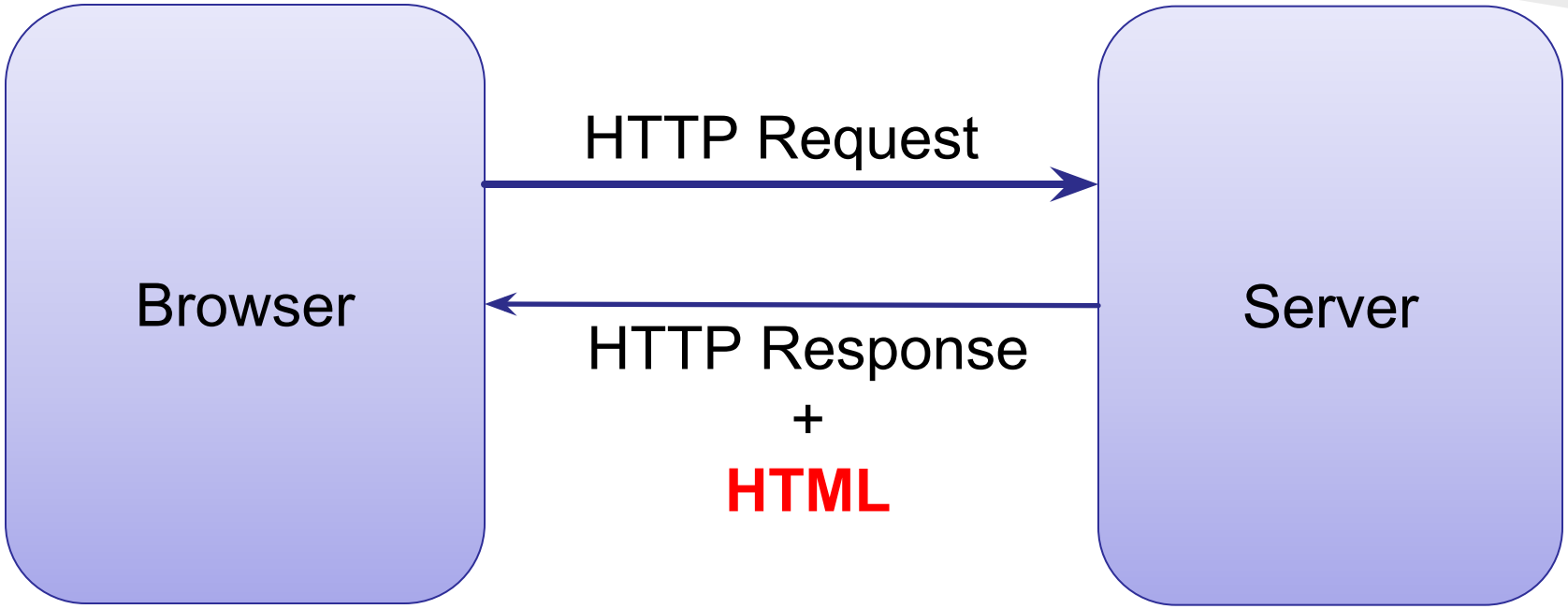
- REST requires that all operations are **stateless**.
  - This means that the server treats each HTTP request as independent - as if its never received another request from that client before.
  - This is sometimes harder than it sounds - but the limitation allows implementation to be more scaleable and maintainable

Web Technologies Overview

Big Picture



Big Picture



What's in the Response

- We've looked at "pure" HTML
  - XHTML, HTML 4.01, HTML 5
  - CSS (CSS 2, CSS 3)
  - JavaScript

CSS and JavaScript

- We've learned that CSS is *declarative*
  - There is no such thing as variables, functions, etc. in CSS
  - It would be nice though…

```
@mythemecolor:  blue;
p {
    background-color:  @mythemecolor;
}
table {
    background-color:  @mythemecolor;
}
```

LESS

- LESS is a *JavaScript* library that parses .less files, which look a lot like CSS, with variables functions,etc.
- The JavaScript executes, transforming (compiling) the LESS file into CSS for the browser

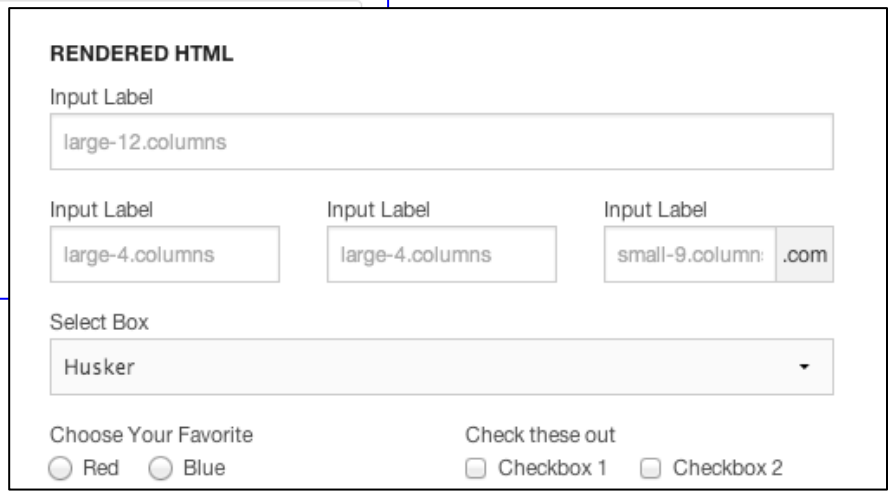
http://lesscss.org/

```
var lessMiddleware = require('less-middleware');

var app = express();
app.use(lessMiddleware(__dirname + '/public'));
app.use(express.static(__dirname + '/public'));
```

CSS and JavaScript Frameworks

Not everyone is a designer… for those of us artistically challenged… we can still have nice things.

Popular Frameworks: Bootstrap and Foundation



JavaScript Applications

- There are many frameworks designed to make life easier on the client.
- They are advanced - but once you get the hang of them, your productivity soars

What else is Client Side?

- The major alternative to HTML/CSS/JS is Adobe Flash
  - Provides complete app development framework
  - Flash scripts are embedded in the HTML (still transmitted from the server)
  - Scripts execute on the Flash Player (plugin)
  -

What else is Client Side?

- Microsoft pushed XAML / Silverlight as a Flash competitor
  - Officially Killed by Microsoft in late 2013

New Clients

Web Development and Mobile Development are highly related

Native clients:

- Android (Java)
- IOS (Objective C)

Data on server - accessible through REST

Creating the Content

- We used Node.js and EJS to generate content
- There are MANY ways to do this though
  - Main differentiation is how they separate markup from processing ocde
- Either way, browser only sees markup

Google Web Toolkit

- If you want to create complex UI's in HTML/CSS/JavaScript, but really like Java…
  - Write Java UI code (like Swing)
  - GWT *compiles* the code into JavaScript for you

May be replaced by Dart in the future...

Less Separation

- There is a whole set of technologies that started as “embed some code” within the markup
  - ASP
  - PHP
  - JSP

Ancient ASP

- Active Server Pages
  - Microsoft only (Server side)
  - Server *executes* asp page – sends result

```
<html>
<body>
<%
response.write("My first ASP script!")
%>
</body>
</html>
```

PHP

- PHP is very similar in design to ASP
  - A PHP page has PHP code within it.
  - The PHP page is executed, output sent to client


```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
   <?php echo '<p>Hello World</p>'; ?>
 </body>
</html>
```

JSP

- Java Server Pages pages are actually compiled into servlets at runtime – so you need a servlet container

```
<HTML>
 <HEAD>
  <TITLE>Hello World</TITLE>
 </HEAD>
 <BODY>
  <H1>Hello World</H1>
  Today is: <%= new java.util.Date().toString() %>
 </BODY>
</HTML>
```

Problems…

- Embedding small snippets of code within markup is fine if most of your app is markup
  - Over time – this has become increasingly rare
  - We need markup separated from code
    - Markup editors
    - Code editors

ASP.NET

- Microsoft completely switched gears when the .NET framework was released
  - C++, C#, VB.NET code running in a virtual machine (CLR)

  - ASP.NET allows the author to split a “page” into an ASP markup file and a C# (or other language) ***code behind*** file

ASP.NET MVC

While a step in the right direction, the initial "WebForms" implementation of ASP.NET wasn't great.

Recently, ASP.NET MVC has risen to dominance

- Fully supports MVC architecture
- Very similar to Node's style - including package management, Razor templates, etc...

Towards Separation in PHP

- A PHP file need not have any markup in it – it can be all procedural code
- Many frameworks and template tools in PHP that allow you to keep markup in separate file

Laravel is very much similar to the Node.js environment (aside from the asynchronous stuff)...

Ruby

- Ruby is more or less a general purpose language
- Its quite different than C-style languages

- Optional Parentheses
  - `foo bar => foo(bar)`
- Additional Control flow
  - `if, unless`
- Additional Characters
  - Methods ending in ! and ?
  - instead `isDefined(v), defined? v`

Ruby on Rails

- Just as C needs CGI and Java needs servlets – Ruby needs help..
- Ruby on Rails is a development framework and platform for writing server code in Ruby
- Huge following, large community

The reason ASP.NET MVC “feels” similar to Node and Laravel is because they ALL ripped Rails off!

Spring Framework

- The Spring framework is the most dominant Java framework around
  - Robust MVC Model
  - Enhanced DB access
  - Easy configuration
  - Security / Web Services / etc.

Future of Java

- The JVM is beginning to become more important than Java itself
  - Many developers prefer scripting languages like Ruby over the “heavy-ness” of Java
  - JVM now supports execution of many other languages
    - Ruby, Python, Scheme, JavaScript, even C!

Future of Java

- Java is still fairly dominant, huge set of libraries and existing code.
- Groovy
  - Dynamically typed
  - Very terse, very powerful
  - Compiles to .class files for JVM
  - Completely compatible with Java code
    - You can rename your .java to .groovy and the groovy compiler will compile it

Groovy + Web: GRAILS

- Groovy has most of the benefits of Ruby, plus can natively use all your existing Java code!

- Grails is a complete application framework for using Groovy for server-side development
  - "Spring on steroids"

And the copying continues…
Ruby => Groovy
Rails => Grails

Deployment

- We never talked about deployment much…
  - ASP in IIS
  - PHP on Apache
- Source Code Control
- Continuous Integration

Trend towards Cloud

- It’s a maintenance headache to host your own server
- The trend is to host your application “in the cloud”
  - This doesn’t mean tradition hosting plans..
  - This means “Platform as a Service” (PaaS)
    - Deployment / Administration through API (web services)
    - Dynamically grows to meet capacity

Platform as a Service

Amazon Elastic
Cloud Compute (EC2)

Google App Engine
(Java / Python)

.NET

Heruku (Salesforce)
Ruby, Node.js,
Java, Others

Cloud Foundry (vmware)
Java, Grails, Other