Introduction to Web Architectures

Andrea Nuzzolese

Cretids: Chhorn Chamnap

Outline

- Basic Web Architecture
  - HTML
  - URI
  - HTTP
- Web Architecture Extension
  - Cookie
- Database-driven Website Architecture
  - AJAX
- Web Services
  - XML
  - JSON
- RESTful Web Service
  - The REST Architectural Style
  - Resources and Resource Oriented Services

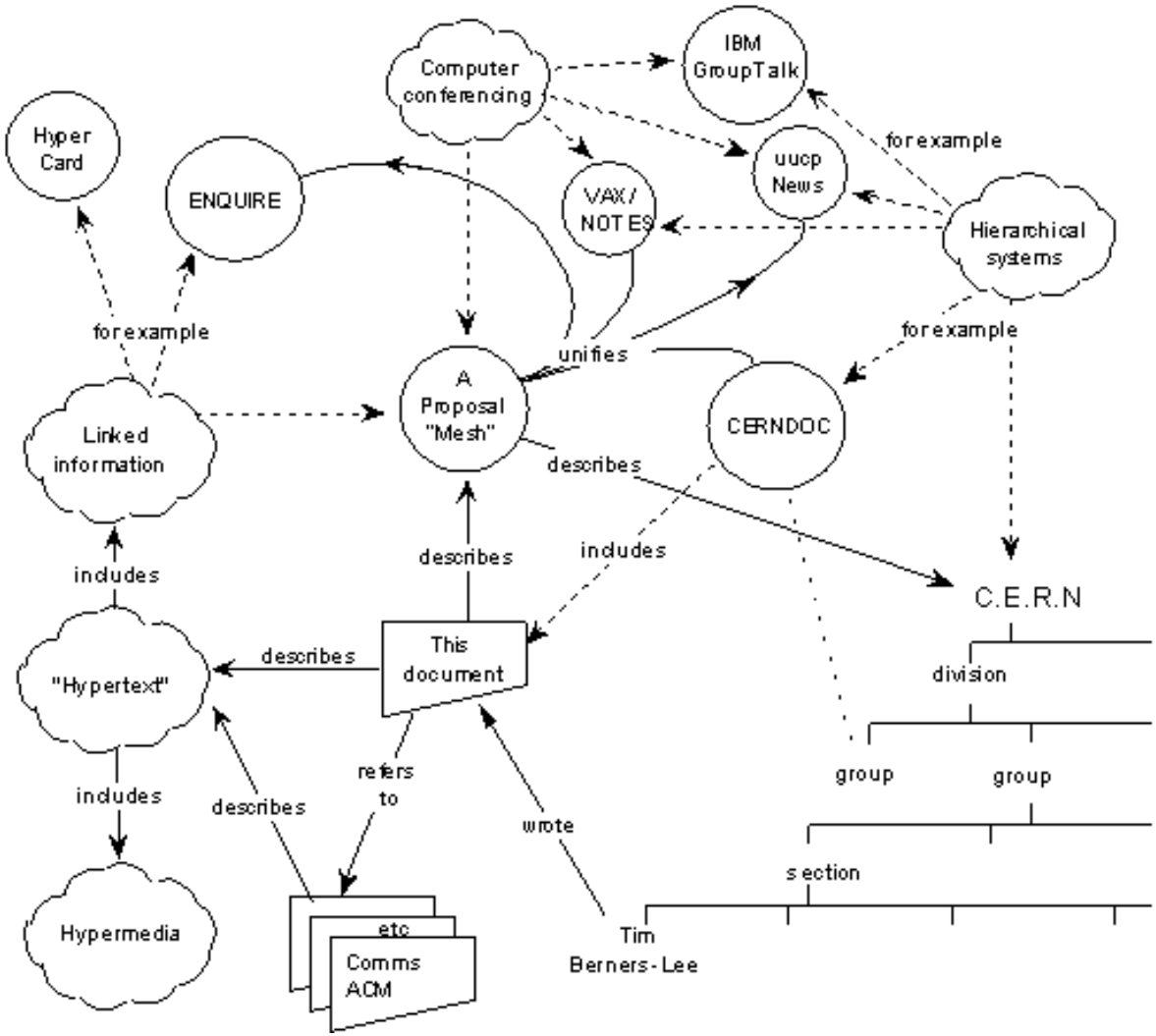
The World Wide Web

The World Wide Web

- In 1989, Tim Berners-Lee had suggested a way to let all users, but particularly scientists, browse each others' papers on the Internet.
- He developed HTML, URLs, and HTTP.

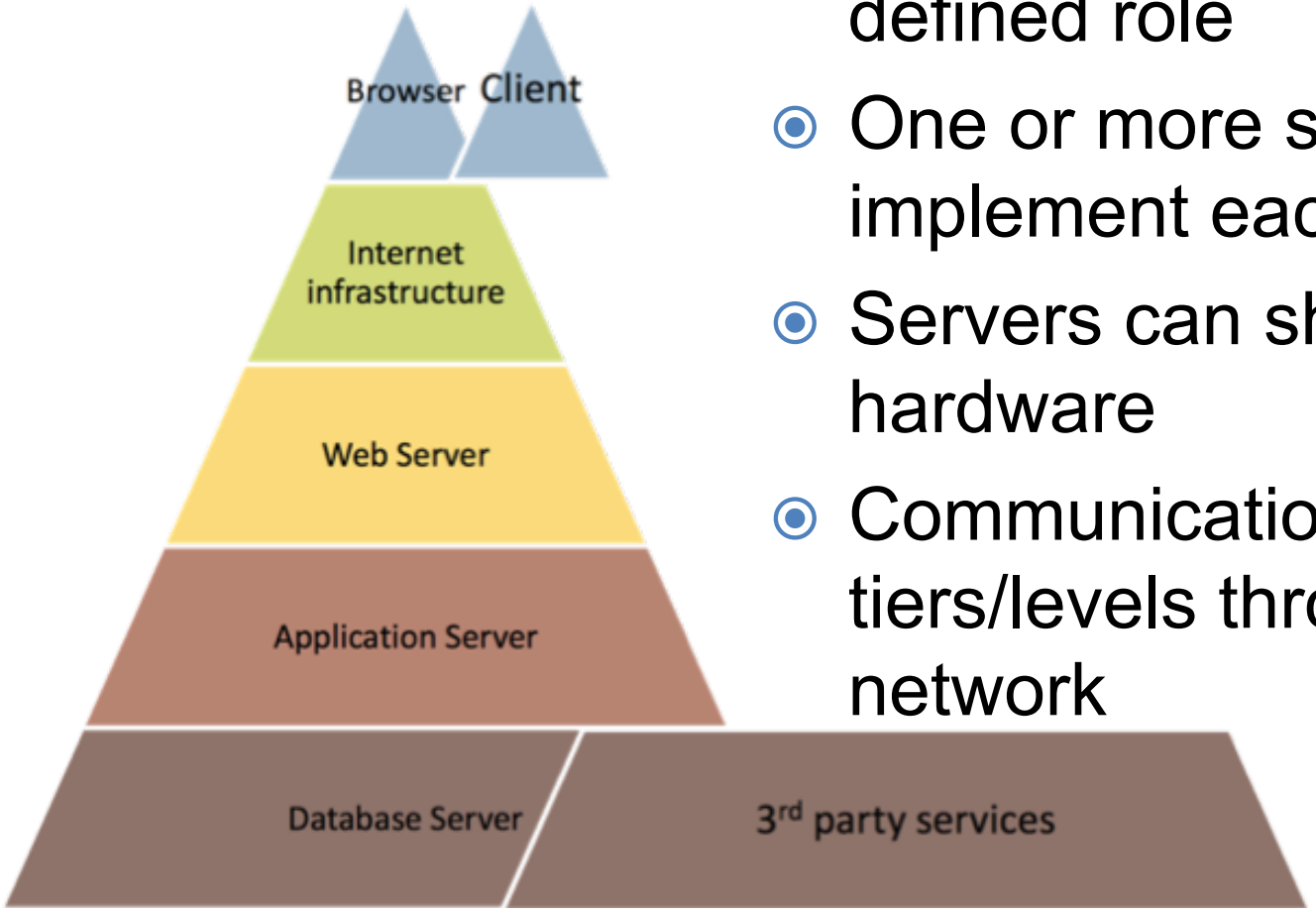
[107] **Tim Berners-Lee,** the father of the World Wide Web.

TBL's Web architecture



General Web Architecture

N-tier Architecture

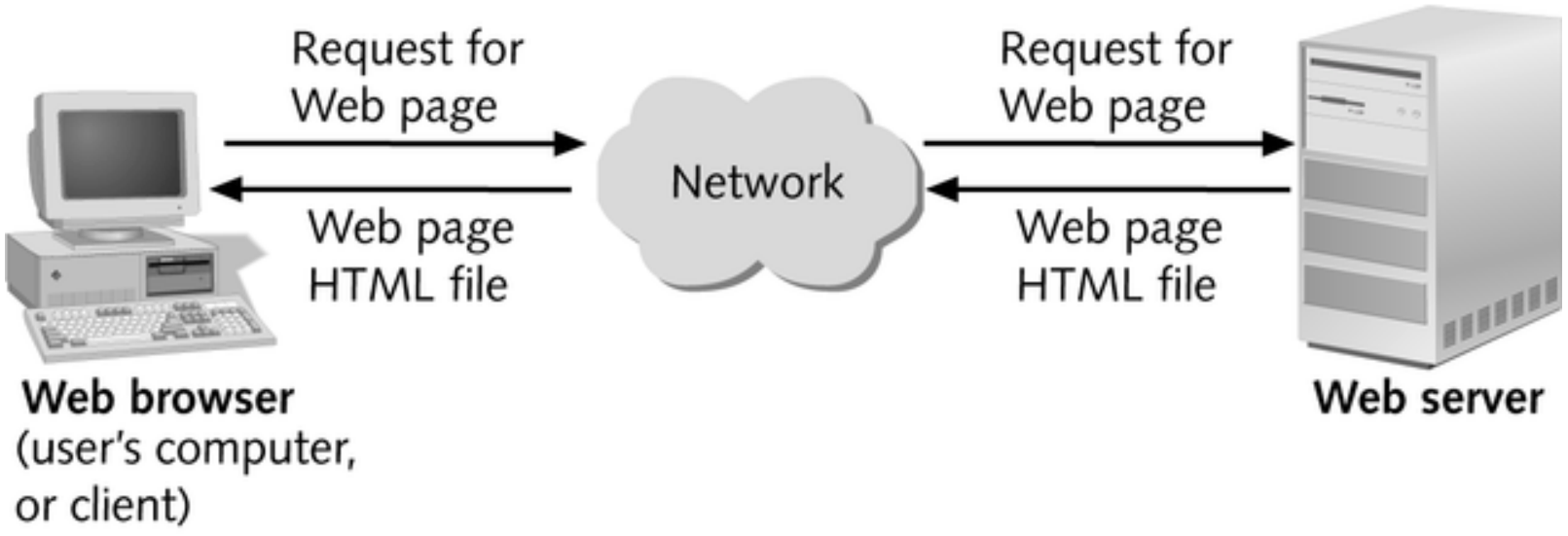


- Each level/tier has a well defined role
- One or more servers implement each tier/layer
- Servers can share hardware
- Communication between tiers/levels through the network

Basic Web Architecture

Basic Web Architecture

- The web is a 2-tiered architecture.
  - A web browser displays information content,
  - and a web server that transfers information to the client.



Web Browser

- The primary purpose is to bring information resources to the user.
- An application for retrieving, presenting, and traversing information resources.

Web Server

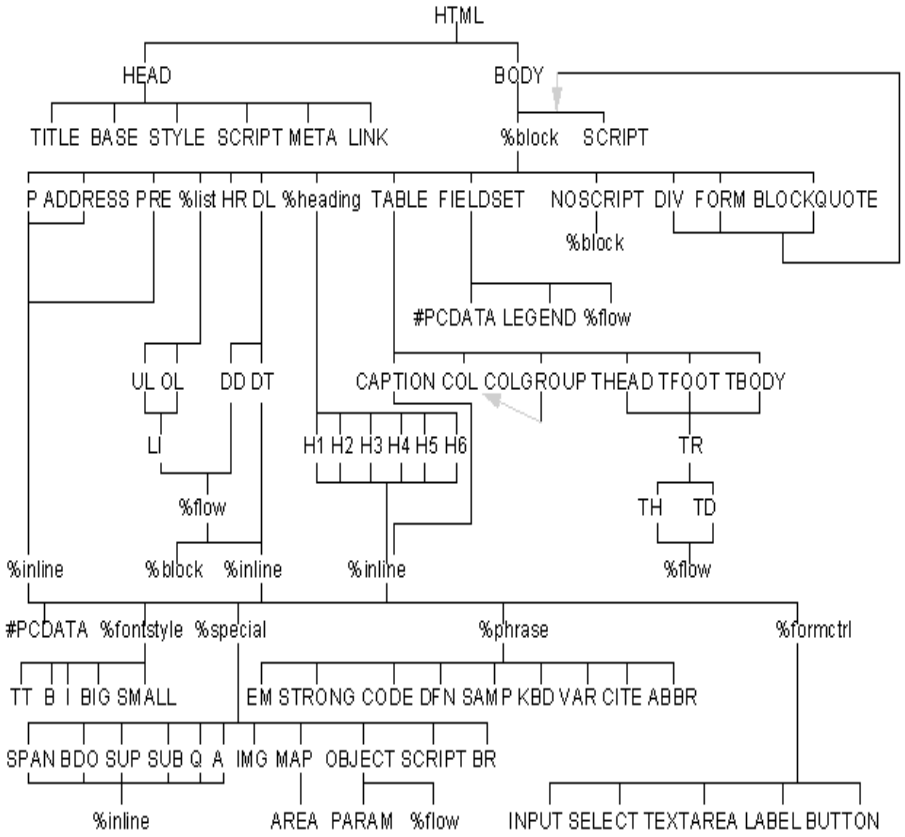
- The term **web server** or **webserver** can mean one of two things:
  - A computer program that accepts HTTP requests and return HTTP responses with optional data content.
  - A computer that runs a computer program as described above.

HTML

HyperText Markup Language

HTML

- Document layout language (not a programming language)
- Defines structure and appearance of Web pages

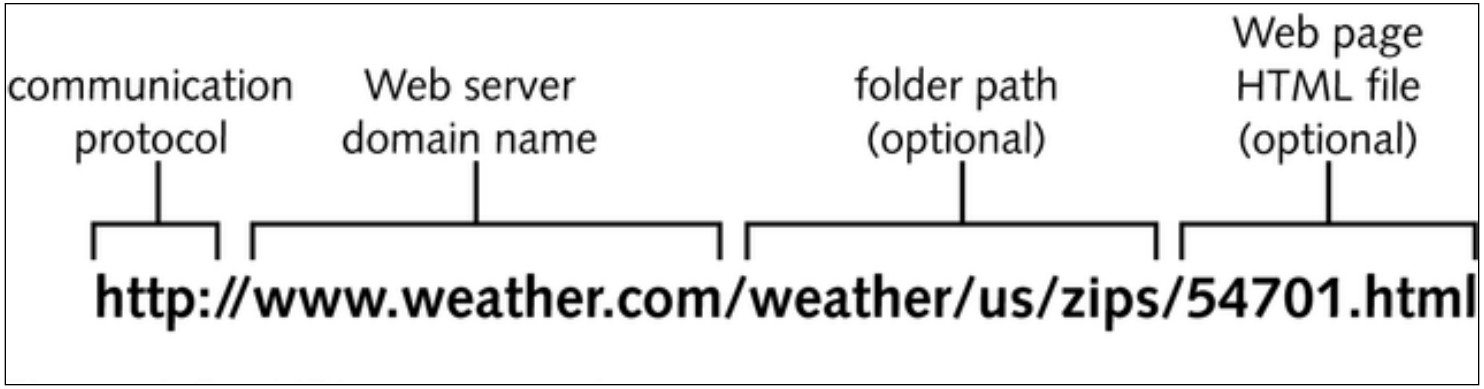


URI

Universal Resource Identifier

URI

- URLs are location dependent
- It contains four distinct parts: the protocol type, the machine name, the directory path and the file name.
- There are several kinds of URLs: file URLs, FTP URLs, and HTTP URLs.



HTTP

HyperText Transfer Protocol

HTTP

- HTTP is a request/response standard of a client and a server.
- Typically, an HTTP client initiates a request.
- Resources to be accessed by HTTP are identified using Uniform Resource Identifiers (URIs).

Request message

- The request message consists of the following:
  - Request line
  - Headers (Accept-Language, Accept, ….)
  - An empty line
  - An optional message body

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Request methods

- HTTP defines eight methods (sometimes referred to as "verbs") indicating the desired action to be performed on the identified **resource**.
  - HEAD
  - **GET**
  - **POST**
  - PUT
  - DELETE
  - TRACE
  - OPTIONS
  - CONNECT

Safe methods

- **HEAD**, **GET**, **OPTIONS** and **TRACE** are defined as *safe* (no side effects).
- **POST**, **PUT** and **DELETE** are intended for actions which may cause side effects either on the server.

Status Codes

- The first line of the HTTP response is called the *status line*.
- The way the user agent handles the response primarily depends on the code and secondarily on the response headers.
- Success: 2xx
- Redirection: 3xx
- Client-Side Error: 4xx
- Server-Side Error: 5xx

HTTP session state

- HTTP is a stateless protocol.
- Hosts do not need to retain information about users between requests.
- Statelessness is a scalability property.
- For example, when a host needs to customize the content of a website for a user. Solution:
  - Cookies
  - Sessions
  - Hidden variables (when the current page is a form)
  - URL encoded parameters (such as `/index.php?session_id=some_unique_session_code`)

Sample HTTP Request and Response

- Client request

```
GET /index.html HTTP/1.1
Host: www.example.com
```

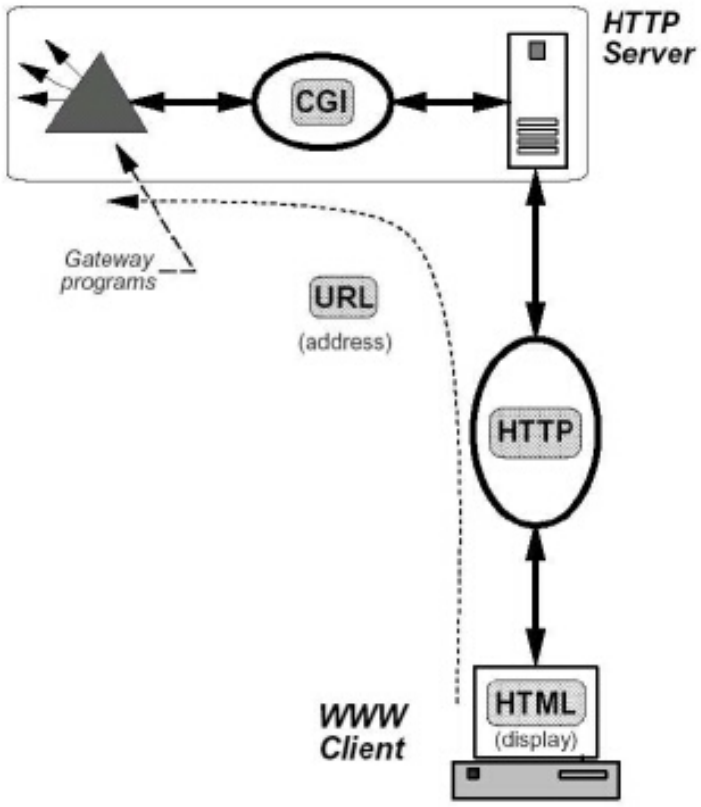
- Server response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix)  (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

Web Architecture Extension

Web Architecture Extension

- CGI extends the architecture to 3-tiers by adding a back-end server that provides services to the Web server.



Traditional uses of JavaScript

- JavaScript is a scripting language designed for creating dynamic, interactive Web applications that link together objects and resources on both clients and servers.
  - Getting your Web page to respond or react directly to user interaction with form elements and hypertext links
  - Preprocessing data on the client before submission to a server
  - Changing content and styles

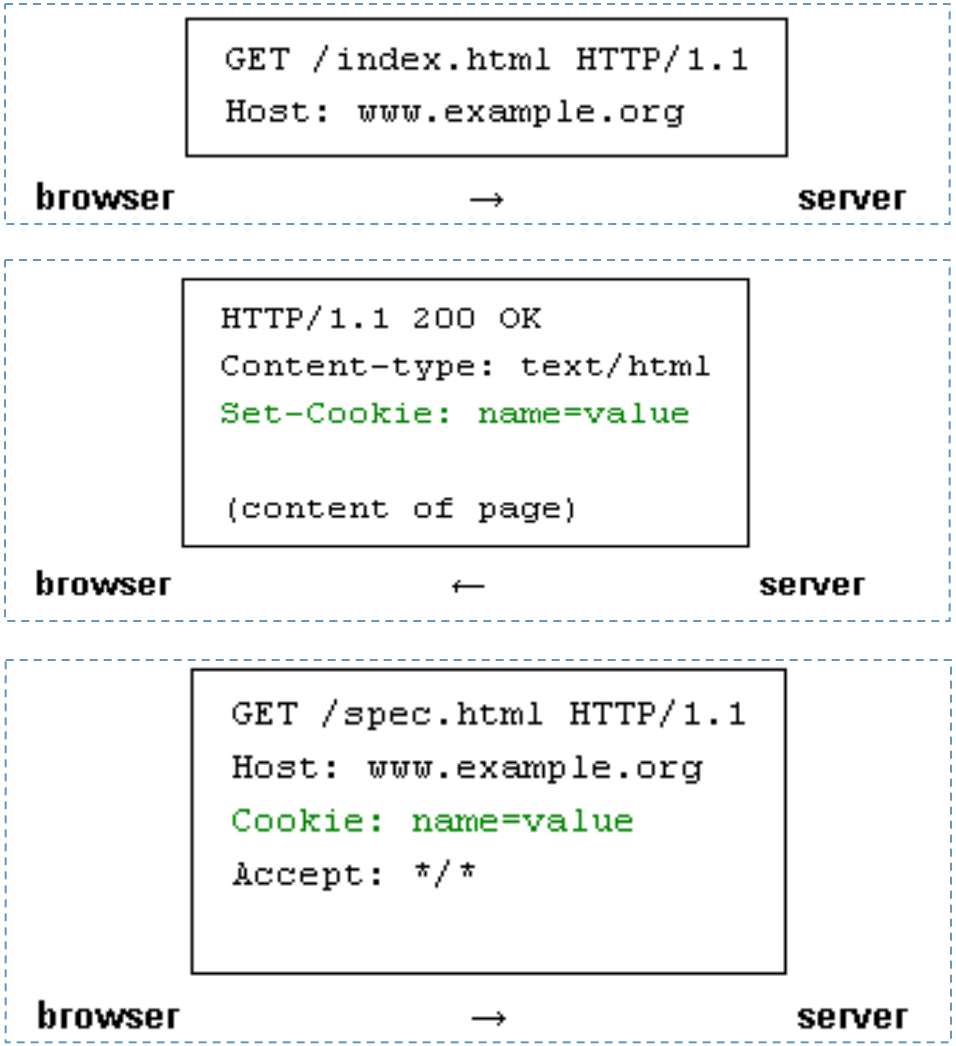
Cookie

tracking cookie, browser cookie,
or HTTP cookie

Cookie

- Cookie is a small piece of text stored on a user's computer by a web browser.
- A cookie consists of one or more name-value pairs containing bits of information such as user preferences.
- A cookie can be used for:
  - authenticating,
  - session tracking, and
  - remembering specific information about users.

Setting A Cookie

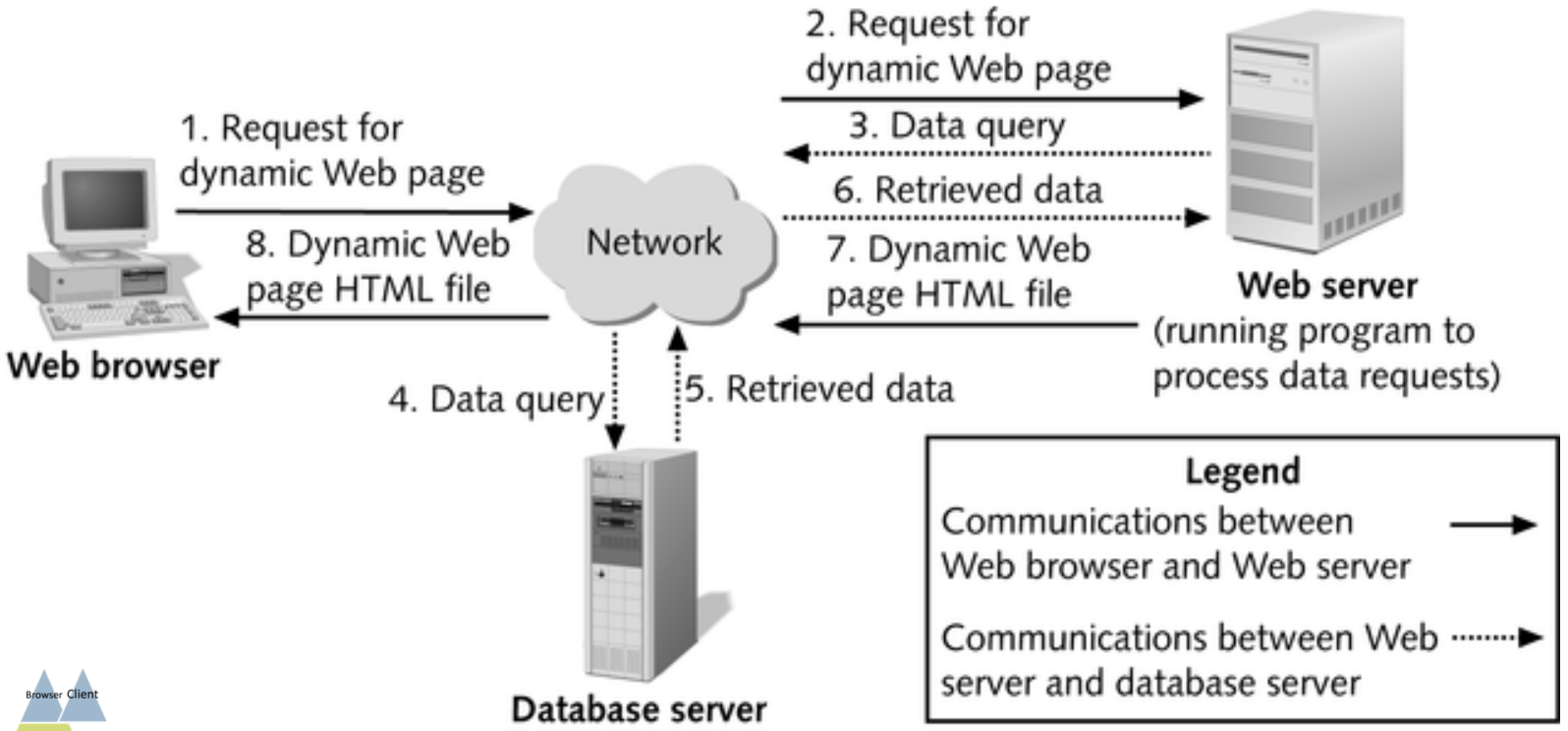


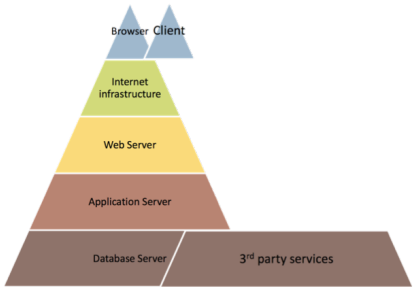
Cookie Expiration

- Cookies expire, and are therefore not sent by the browser to the server, under any of these conditions:
  1. At the end of the user session if the cookie is not persistent
  2. An expiration date has been specified, and has passed
  3. The expiration date of the cookie is changed to a date in the past
  4. The browser deletes the cookie by user request

Database-driven Website Architecture

Database-driven Website Architecture





Server-side processing

- In server-side processing, the Web server:
  - Receives the dynamic Web page request
  - Performs all of the processing necessary to create the dynamic Web page
  - Sends the finished Web page to the client for display in the client's browser

Client-side processing

- Client-side processing
  - Some processing needs to be “executed” by the browser, either to form the request for the dynamic Web page or to create or display the dynamic Web page.

  Eg. Javascript code to validate user input

Server and Client side processing

- Server-side processing
  - PHP
  - ASP
  - ASP.NET
  - Perl
  - J2EE
  - Python, e.g. Django
  - Ruby, e.g. Ruby on Rails
  - ColdFusion
- Client-side processing
  - CSS
  - HTML
  - JavaScript
  - Adobe Flex
  - Microsoft Silverlight

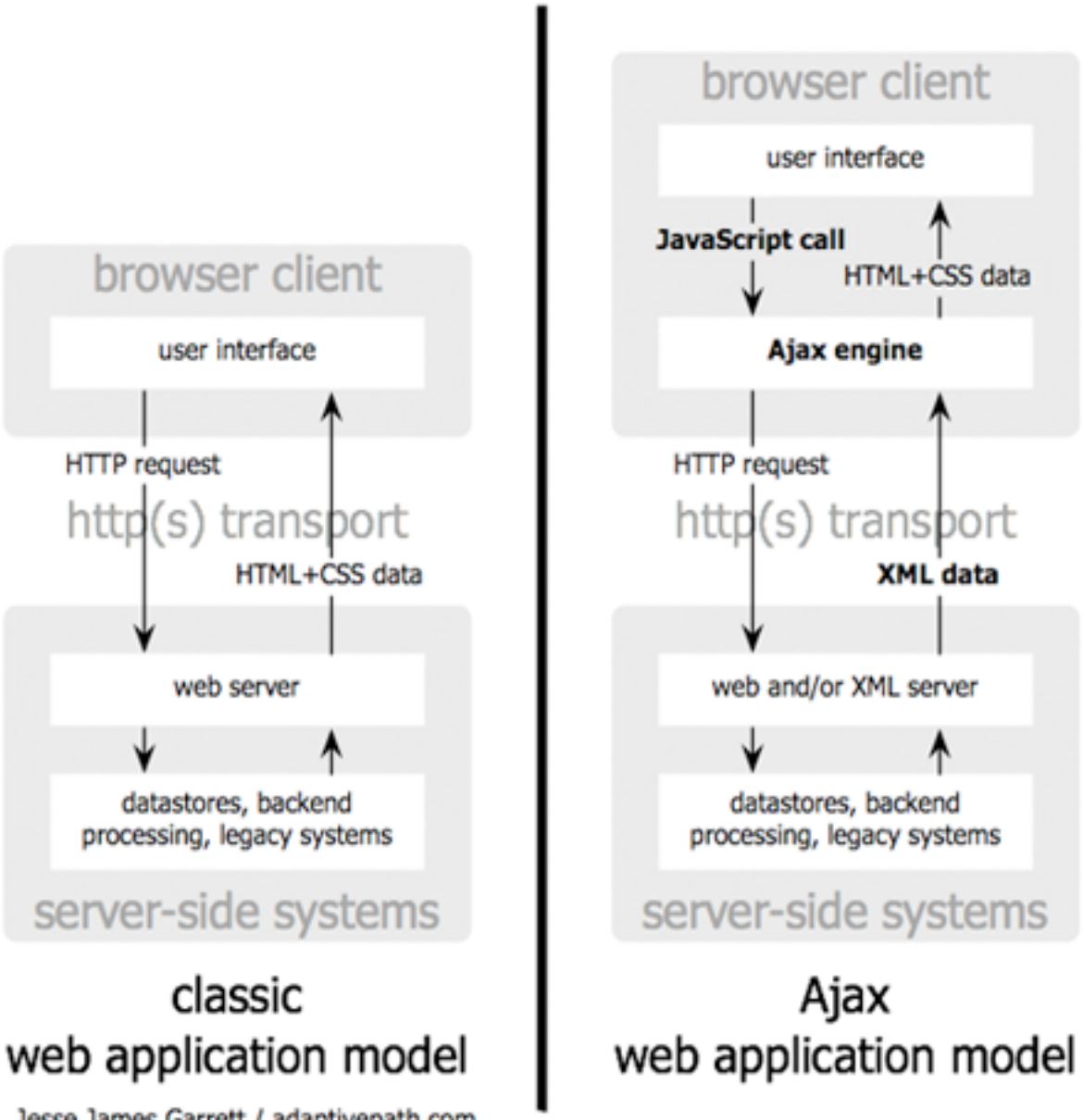
AJAX

Asynchronous JavaScript and XML

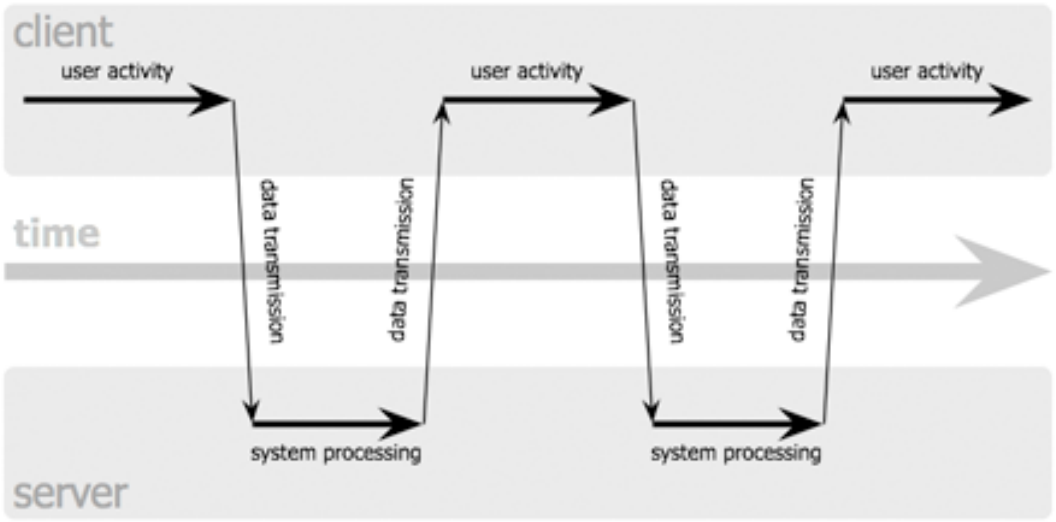
Defining Ajax

- Ajax isn't a technology. It's really several technologies, each flourishing in its own right, coming together in powerful new ways. Ajax incorporates:
  - XHTML and CSS;
  - Document Object Model;
  - XML and XSLT;
  - XMLHttpRequest;
  - JavaScript

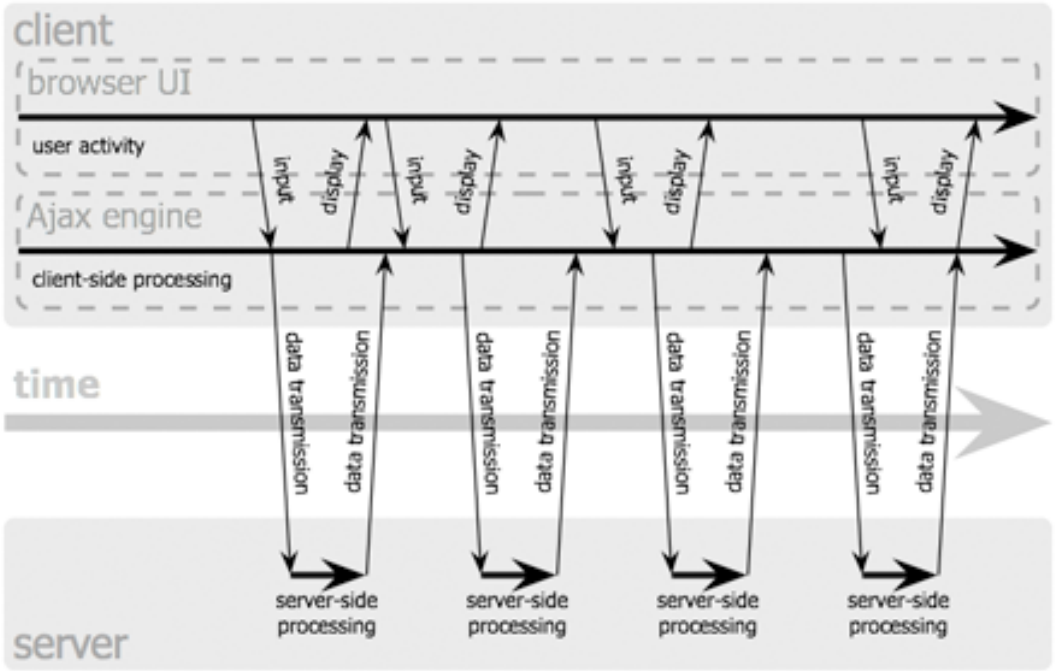
Jesse James Garrett, essay in february 18, 2005
Ajax: A New Approach to Web Applications


browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend
processing, legacy systems
server-side systems
classic
web application model
Jesse James Garrett / adaptivepath.com
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
HTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend
processing, legacy systems
server-side systems
Ajax
web application model

classic web application model (synchronous)



Ajax web application model (asynchronous)



Jesse James Garrett / adaptivepath.com

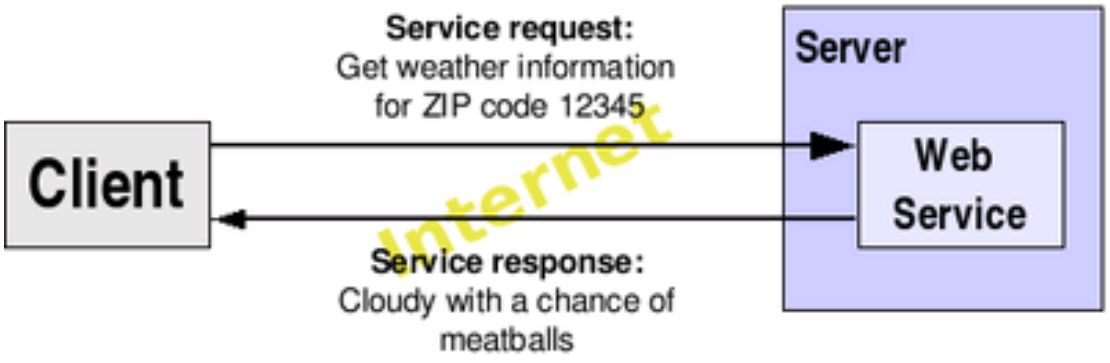
Drawbacks of AJAX

- It breaks browser history engine (Back button).
- No bookmark.
- The same origin policy.
- Ajax opens up another attack vector for malicious code that web developers might not fully test for.

Web Services

Web Services

- Web Service is a software system designed to support machine-to-machine interaction over a network.
- Web services are frequently just Internet Application Programming Interfaces (API) that can be accessed over a network.



Web Services (cont.)

- Web Services are platform-independent and language-independent, since they use standard XML languages.
- Most Web Services use HTTP for transmitting messages (such as the service request and response).
- Style of Use
  - RPC
  - SOAP
  - REST

XML

eXtensible Markup Language

XML

- XML is a universally agreed markup meta-language primarily used for information exchange.
- The two primary building blocks of XML are elements and attributes.
  - Elements are tags and have values.
  - Elements are structured as a tree.
  - Alternatively, elements may have both attributes as well as data
  - Attributes help you to give more meaning and describe your element more efficiently and clearly.

XML (cont.)

```
<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<person>
   <id type="integer">1111</id>
   <last_name>Smith</last_name>
   <first_name>John</first_name>
   <address>
    <city>New York</city>
    <street>21 2nd Street</street>
    <postal_code type="integer">10021</postal_code>
    <state>NY</state>
   </address>
</person>
```

JSON

JavaScript Object Notation

JSON

- JSON is a lightweight computer data interchange format.
- JSON is based on a subset of the JavaScript programming language.
- It is considered to be a language-independent data format.
- It serves as an alternative to the use of the XML format.

Douglas Crockford is a senior JavaScript Architect at Yahoo! He is well known for his work in introducing JavaScript Object Notation (JSON).

JSON (cont.)

```
{
        "firstName": "John",
        "lastName": "Smith",
        "address": {
            "street": "21 2nd Street",
            "city": "New York",
            "state": "NY",
            "postalCode": 10021
        },
        "phoneNumbers": [
            "212 555-1234",
            "646 555-4567"
        ]
 }
```

RESTful Web Services

What is REST

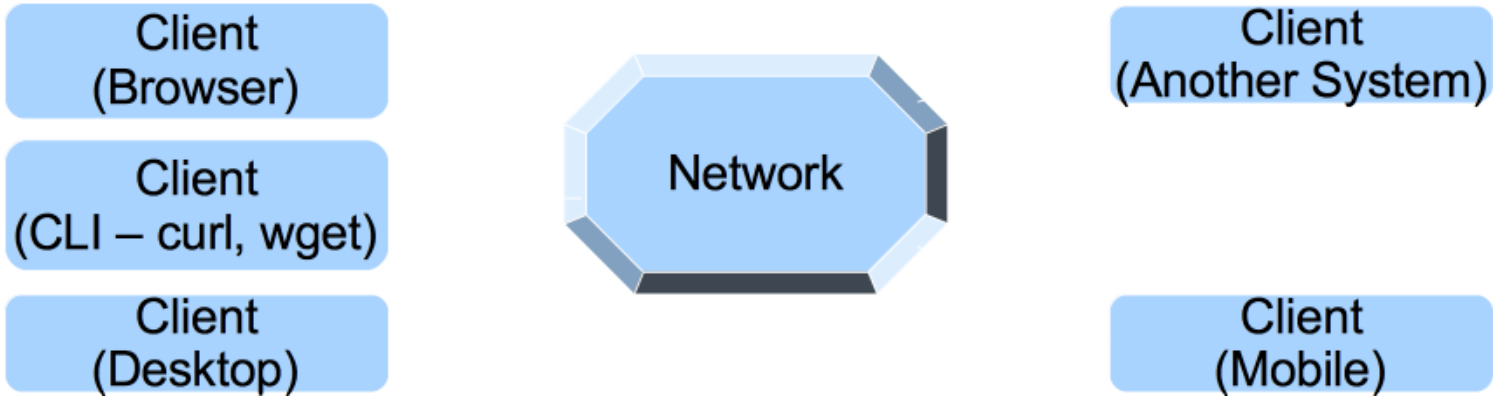
- REpresentational State Transfer
- Proposed by **Dr. Roy Thomas Fielding** in his PhD dissertation titled - **“Architectural Styles and the Design of Network-based Software Architectures”**

What is REST

- REpresentational State Transfer
- REST is an architectural style composed of specific constraints
  - Client-Server
  - Cache
  - Tiered System
  - Sateless
  - Uniform Interface
  - Code-on-Demand

REST Constraints

- **Client-Server**
  - No restrictions on the **nature of the client**
  - No restrictions on the **number of the clients**
  - No restriction on **communication medium** / **protocol**



REST Constraints

- **Cache**
  - *"... the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests."*
- **Advantages**
  - **Efficient**
  - **Scalability**
  - **Performance**

REST Constraints

- **Stateless**
  - *"... each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server."*
- **Advantages**
  - **Visibility**
  - **Scalability**
  - **Reliability**

REST Constraints

- **Layered System**
  - The n-tier architectures constrain component behavior such that each component cannot "see" beyond the immediate layer with which they are interacting.
- **Uniform Interface**
  - Identification of ***resources***
  - Manipulation of ***resources*** through representations
  - Self-descriptive messages

RESTful Web Service

- What is RESTful Web Service or API?
  - **Any system following fulfilling the constraints, thus definition, of REST is a RESTful Web Service.**
- RESTful Web Service system communicates over HTTP protocol
- RESTful Web Service design and architecture grows around **resources**.

Resource Oriented Architecture

- Introduced in the book “RESTful Web Services”
- Resource-Oriented Architecture is about a RESTful system based on the identification of any resource (data, services, etc.) provided by a Web Service by means of Unique Identifiers, i.e., URI

Resources

- A Resource is anything, a concept, that is worth having a URI to linked to.
  - E.g. http://basis.com.bd/softexpo/2011/
- A URI is a name and address of a resource.
- A Resource may have many URIs but needs to have **at least one**.
- A Resource may have one or more representations; i.e. it may not have any representations at all.

Resources + HTTP requests