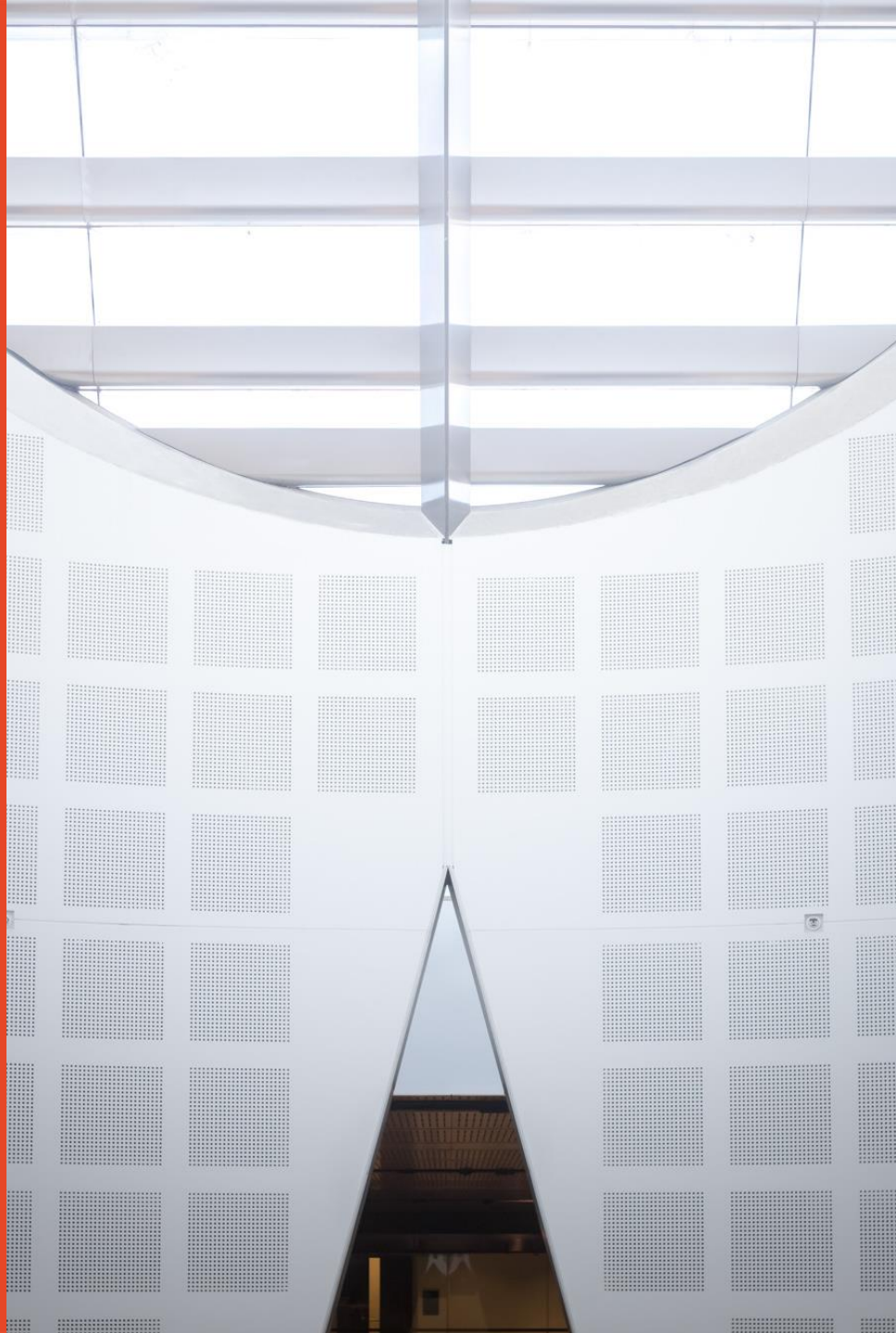
COMP5347/COMP4347 Web Application Development

Introduction – How The Web Works JavaScript (Intro.)

Dr. Basem Suleiman

School of Computer Science

THE UNIVERSITY OF SYDNEY



Agenda

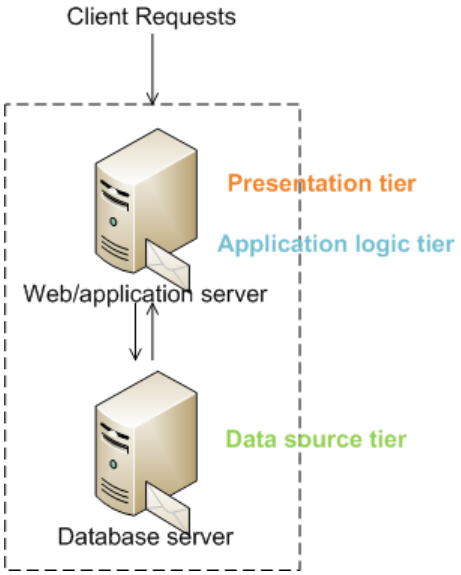
- Web Application architecture in general
  - Web application architecture
  - What is covered in this course
  - Course resources
- Basic concepts behind web browsers
  - Web page
  - HTTP protocol
- **JavaScript (Intro.)**
  - Location and Basic Syntax
    - Variables, Control Structure, Function, Object, Array, Function

World Wide Web

- The *hypertext system* proposal in 1990 led to WWW
  - Jointly published by Tim Berners-Lee and Robert Cailliau at CERN in Switzerland
- Core Features of the Web:
  - A URL to uniquely identify a resource on the WWW
  - The HTTP protocol to describe how requests and responses operate
  - A software program (later called web server software) that can respond to HTTP requests
  - HTML to publish documents
  - A program (later called a browser) to make HTTP requests from URLs and that can display the HTML it receives.
- Berners-Lee founded the World Wide Web Consortium (W3C)

https://www.w3.org/Proposal.html

Web Application Architecture



Simple web app with three principle tier/layers

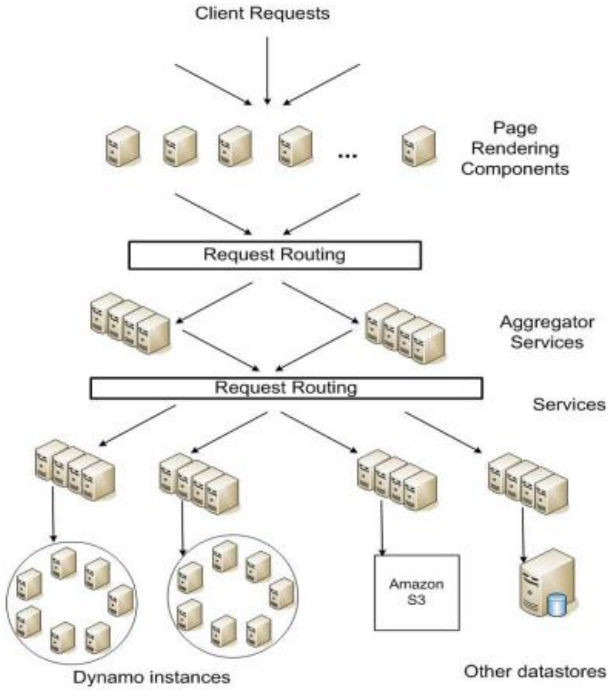
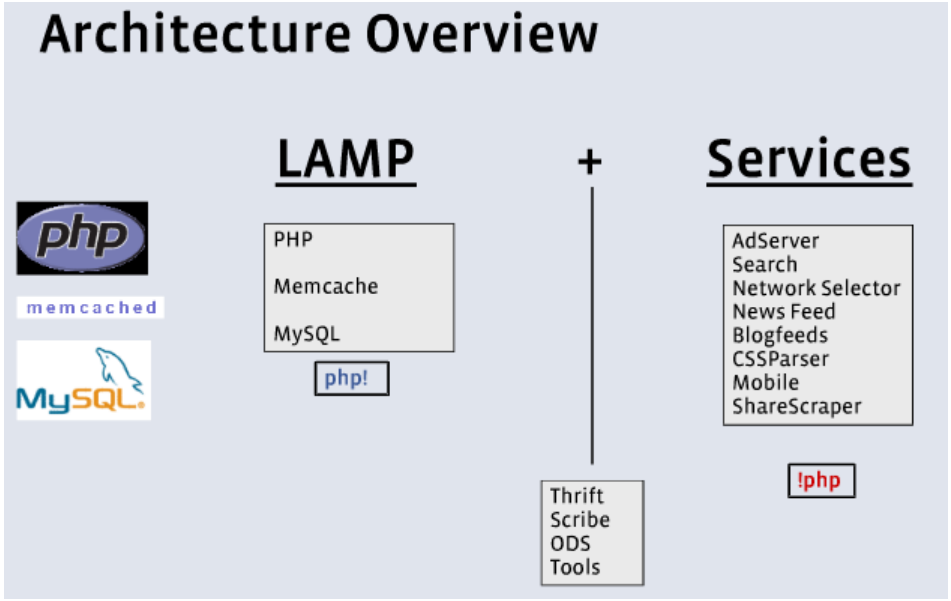


Figure 1: Service-oriented architecture of Amazon's platform [2]

Amazon.com started **10 years** ago as a **monolithic application**, running on a Web server, talking to a database on the back end.… Over time, this grew into **hundreds** of services and a number of application servers that aggregate the information from the services.… If you hit the Amazon.com gateway page, the application calls more than 100 services to collect data and construct the page for you. **A Conversation with Werner Vogels**, ACM Queue, May, **2006** [3]

Web Application Technology Stack (Facebook)

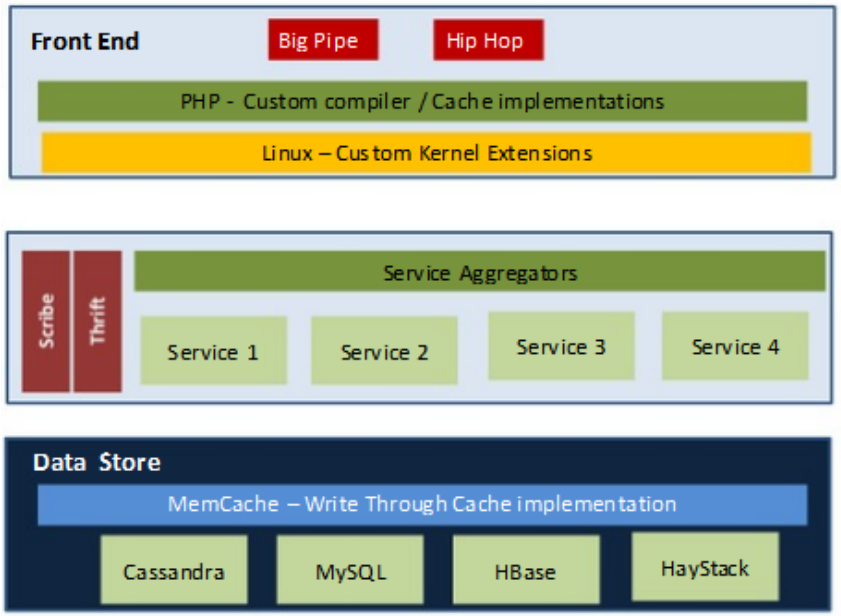


Other technology stacks:

- **WISA**: Windows, IIS Web Server, SQL server and ASP.NET
- **MEAN**: MongoDB, Express.js, Angular and Node.js

Aditya Agarwal, Facebook: Science and the Social Graph, Qcon San Francisco 2008 [http://www.infoq.com/presentations/Facebook-Software-Stack]

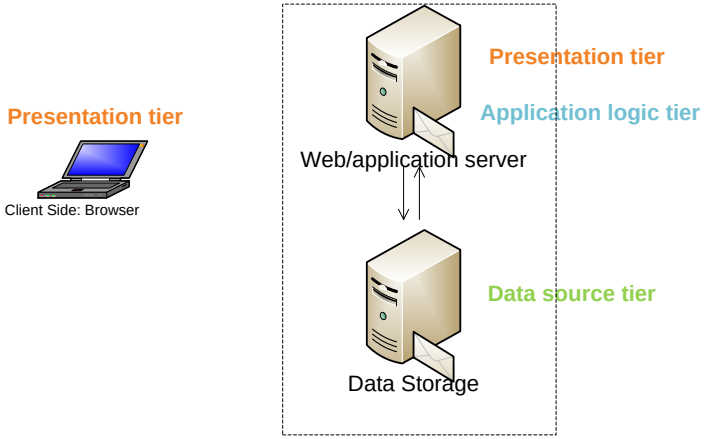
Web Application Technology Stack (Facebook)



Facebook Architecture: Breaking it Open, Slide 21 [https://www.slideshare.net/AditiTechnologies/facebook-architecture-breaking-it-open]

What is covered in this course

- Three-tiered web application
  - Client-Side Technology
  - Server-Side Technology
  - Communication between the Client and the Server
  - Performance and security
- Integrating services from various components
  - web services
- Assumed Knowledge
  - Competency in a programming
  - Basic understanding of data model and storage systems



Topics Covered – Course Schedule

Week	Topic
Week 01	Unit Introduction, How the web works, JavaScript (introduction)
Week 02	Introduction to HTML and CSS
Week 03	Advanced HTML, CSS and JavaScript: client side scripting
Week 04	Browser and rendering process
Week 05	Server-side development with node.js and express.js
Week 06	Sessions and routes - MVC implementation
Week 07	Connecting to database
Week 08	Client-side frameworks
Week 09	Javascript (cont.), Client-side frameworks (cont.)
Week 10	Web services
Week 11	Web application security
Week 12	Industry speakers
Week 13	Review, exam structure

COMP5347 Web Application Development

Fundamentals of Web Application

- It is of Client/Server architecture
- Client-Side Basics
  - Displaying content: **HTML**
  - Displaying content with desirable styles: **CSS**
  - Interacting with content: **JavaScript**
  - A central point of contact: **the browser**
- Server-Side Basics
  - A web application written in
    - **Python, PHP, Java, C#, Ruby, JavaScript,…**
  - Some system that stores the content/data: **file system, database system, …**
  - Some system that listens/handles common network activities: **web/application server**
- In between
  - A way to locate the server given a name: **DNS services**
  - A way to send Instruction/Content between client and server: **protocols**
  - A way to secure the content: **security mechanism**

Ever Changing Fields

- Both HTML and CSS have evolved rapidly
- JavaScript is getting more powerful
  - Lots of libraries
  - It can be used on server-side as well
- Browsers become more complicated
- New programming languages/frameworks for writing server applications
- New architecture/programming style to cater for scalability and cloud hosting
  - Eg. Flux vs. MVC as proposed by Facebook
  - When Netflix migrated to AWS in 2010, they have to re-implement many services
- The communication core, e.g. protocols are relatively stable

Agenda

- Course Information
  - Web application architecture in general
  - What are covered in this course
  - Course resources
- Basic concepts of web (from end user perspective)
  - Web page
  - HTTP protocol
- **JavaScript (Intro.)**
  - Location and Basic Syntax
    - Variables, Control Structure, Function, Object, Array, Function

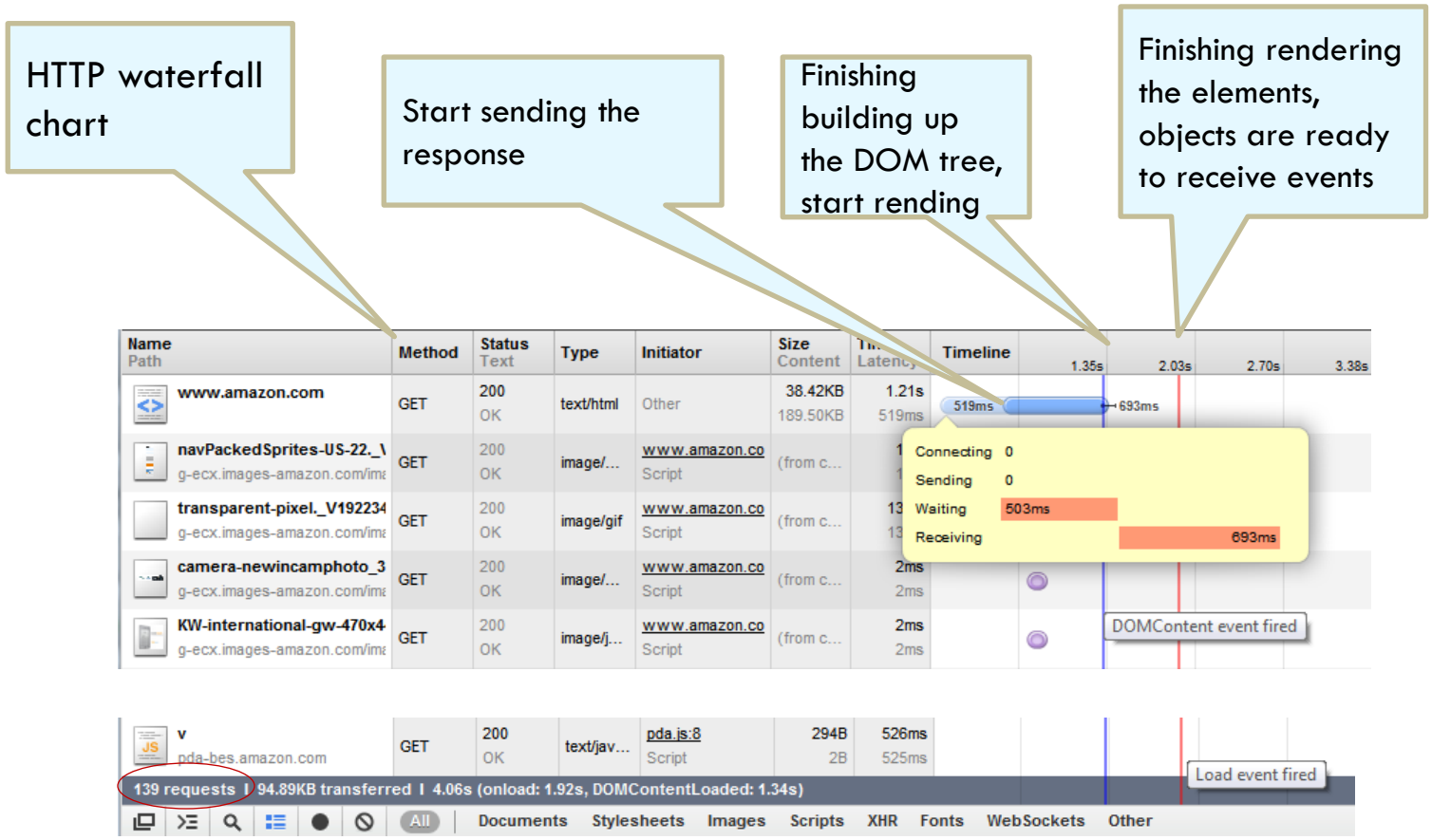
Based on Chapter 1 of Fundamentals of Web Development

Web Browser

- Main responsibility of browser
  - Generate and submit requests to web servers
  - **Accept response and render results**

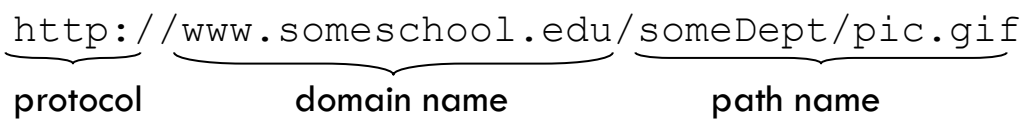
- More specifically
  - **Caching**
  - Authentication and authorization
  - State maintenance
  - **Requesting support data items**
  - **Taking actions in response to other headers and status codes**
  - Rendering complex objects
  - **Dealing with error conditions**

When you send a request from a browser



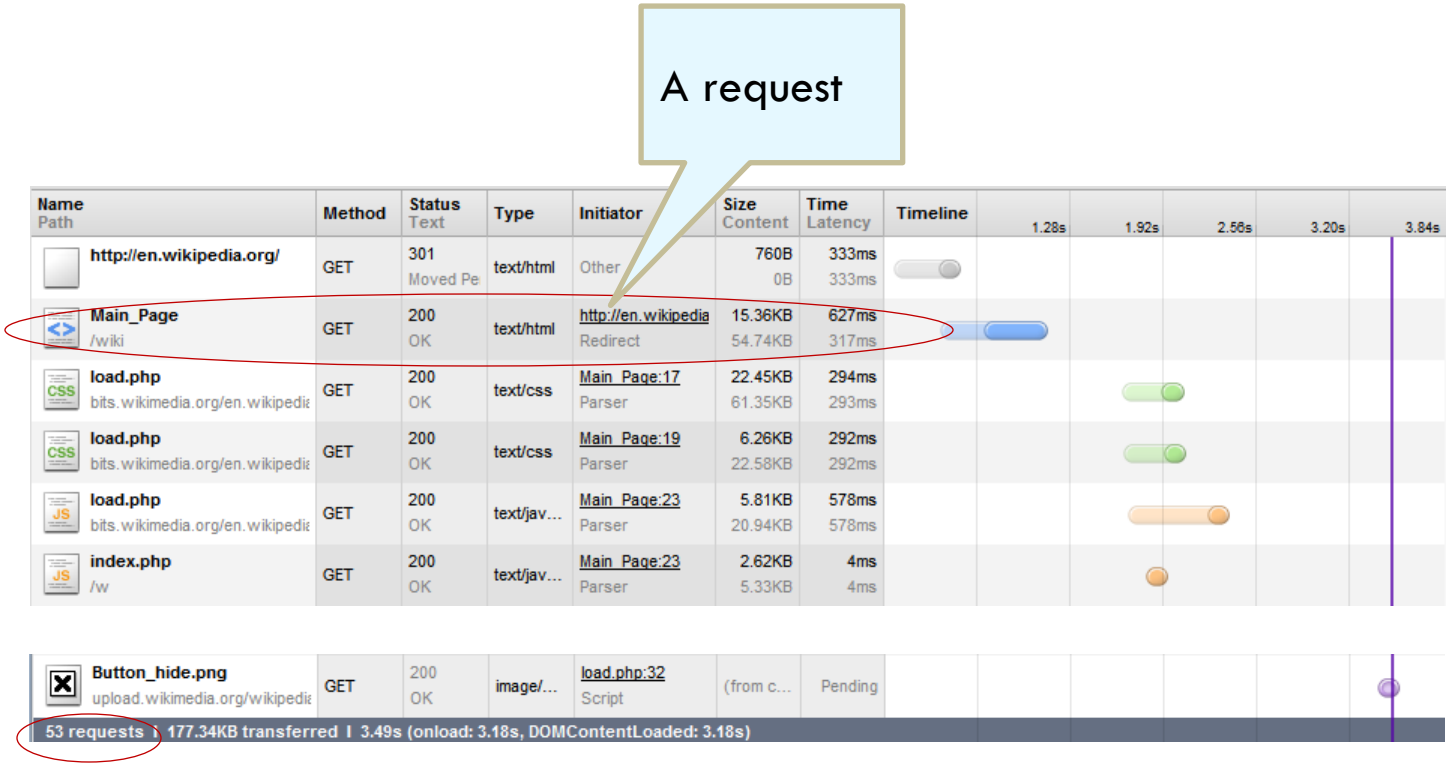
A Web Page

- Jargons everyone is familiar with
  - Web page consists of objects
  - Object can be HTML file, scripts, JPEG image, video/audio file,…
  - Web page consists of a base HTML-file which includes several referenced objects
  - Each object is addressable by a URL (Uniform Resource Locator)
  - Example URL:



A web page consists of many objects

53 requests to show the English wikipedia page on your browser



Support for content types

- To view content on the web, your browser might do
  - Render the content as a text page or an HTML page
  - Present content inline
  - Run scripts
  - Launch a helper application capable of presenting non-HTML content
  - Get confused into showing the content of an HTML file (or a server-side script) as plain text without attempting to render it or execute it

- Browser determines the content type and performs actions appropriate for that type
  - HTTP borrows its content typing system from Multipurpose Internet Mail Extension(MIME)
    - Content-encoding
    - Content-type

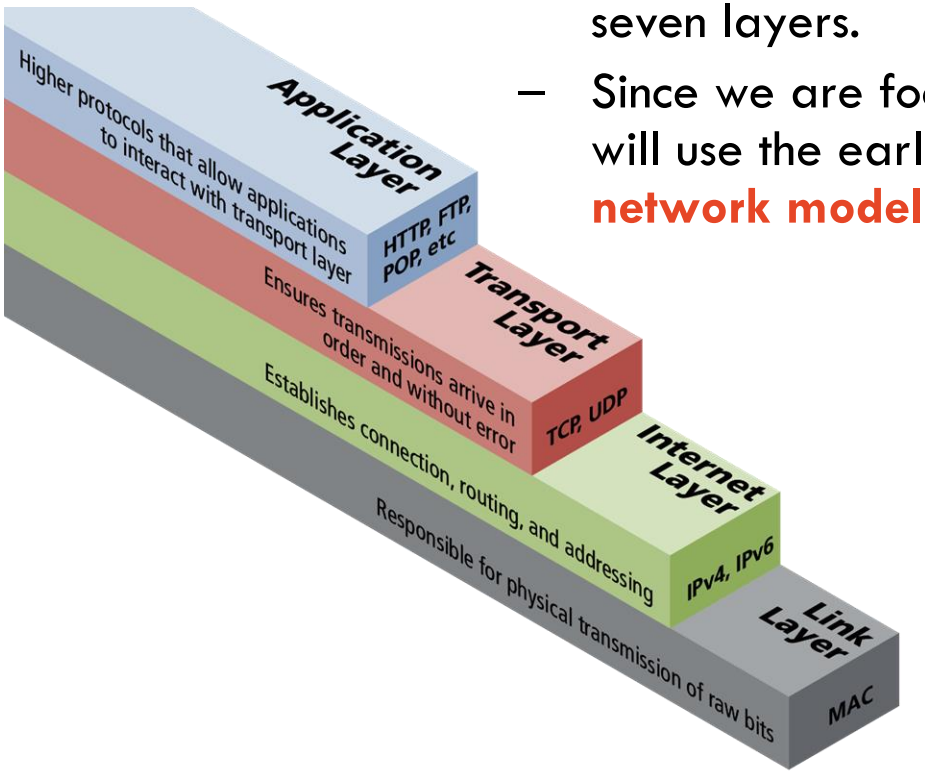
What is a Protocol?

- A **protocol** is a set of rules that partners in communication use when they communicate.

- The internet exists today because of a **suite of interrelated communications protocols**
  - The research network ARPANET created in the 1960s and was used exclusively for academic and scientific purposes
  - Grew from a handful of connected campuses (1969) to a few hundred (early 1980s)
  - To promote the growth and unification of the disparate networks a suite of protocols was invented to unify the networks together
  - By 1981, new networks built in the US began to adopt the **TCP/IP** communication model, while older networks were transitioned over to it.

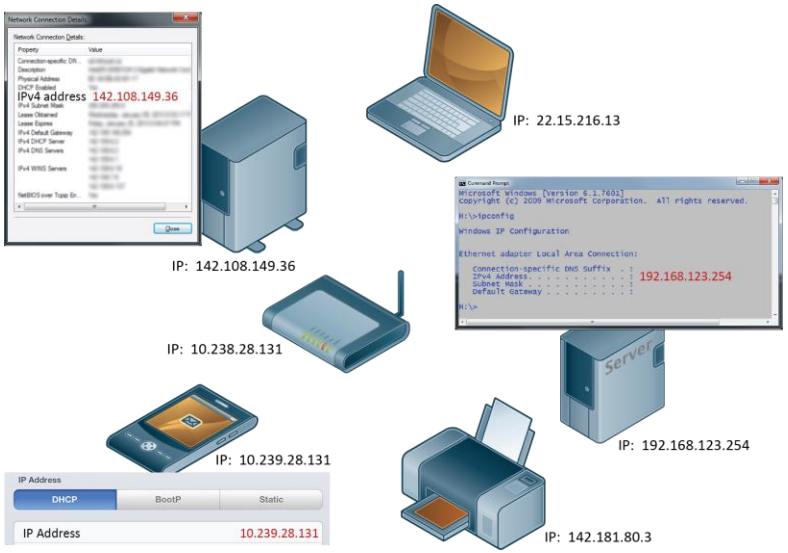
A Layered Architecture

- The TCP/IP Internet protocols were originally abstracted as a four-layer stack.
- Later abstractions subdivide it further into five or seven layers.
- Since we are focused on the top layer anyhow, we will use the earliest and simplest **four-layer network model**.



Internet Protocol

- The Internet uses the **Internet Protocol** (**IP**) addresses to identify destinations on the Internet.
- Every device connected to the Internet has an **IP address**, which is a numeric code that is meant to uniquely identify it.



Transport Layer and Application Layer

- **Internet layer protocol** (e.g. IP) provides logical communication between hosts identified by `IP` address

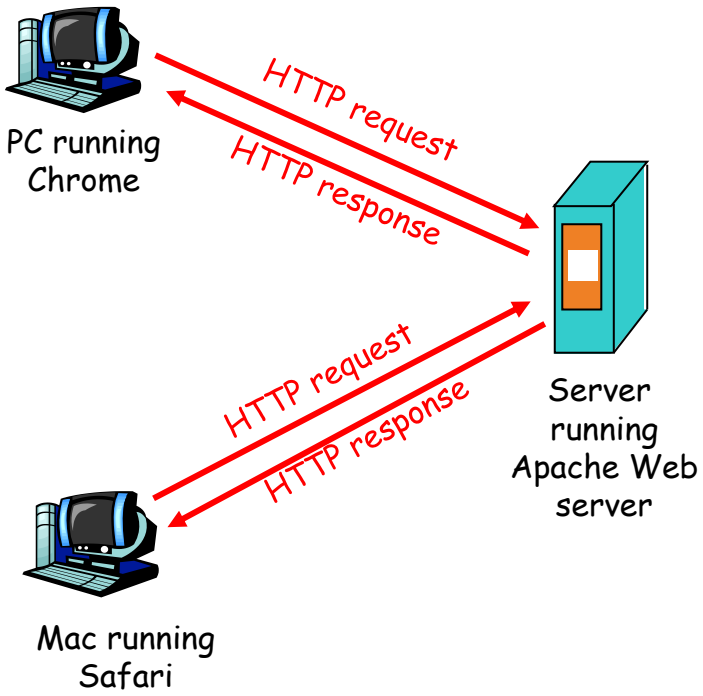
- **Transport layer protocol** (e.g. TCP) provides logical communication between processes running on the hosts identified by `ip:port`
  - TCP ensures that transmissions arrive, in order, and without error

- **Application layer protocol** (e.g. HTTP) defines the syntax and semantics of the message send between processes

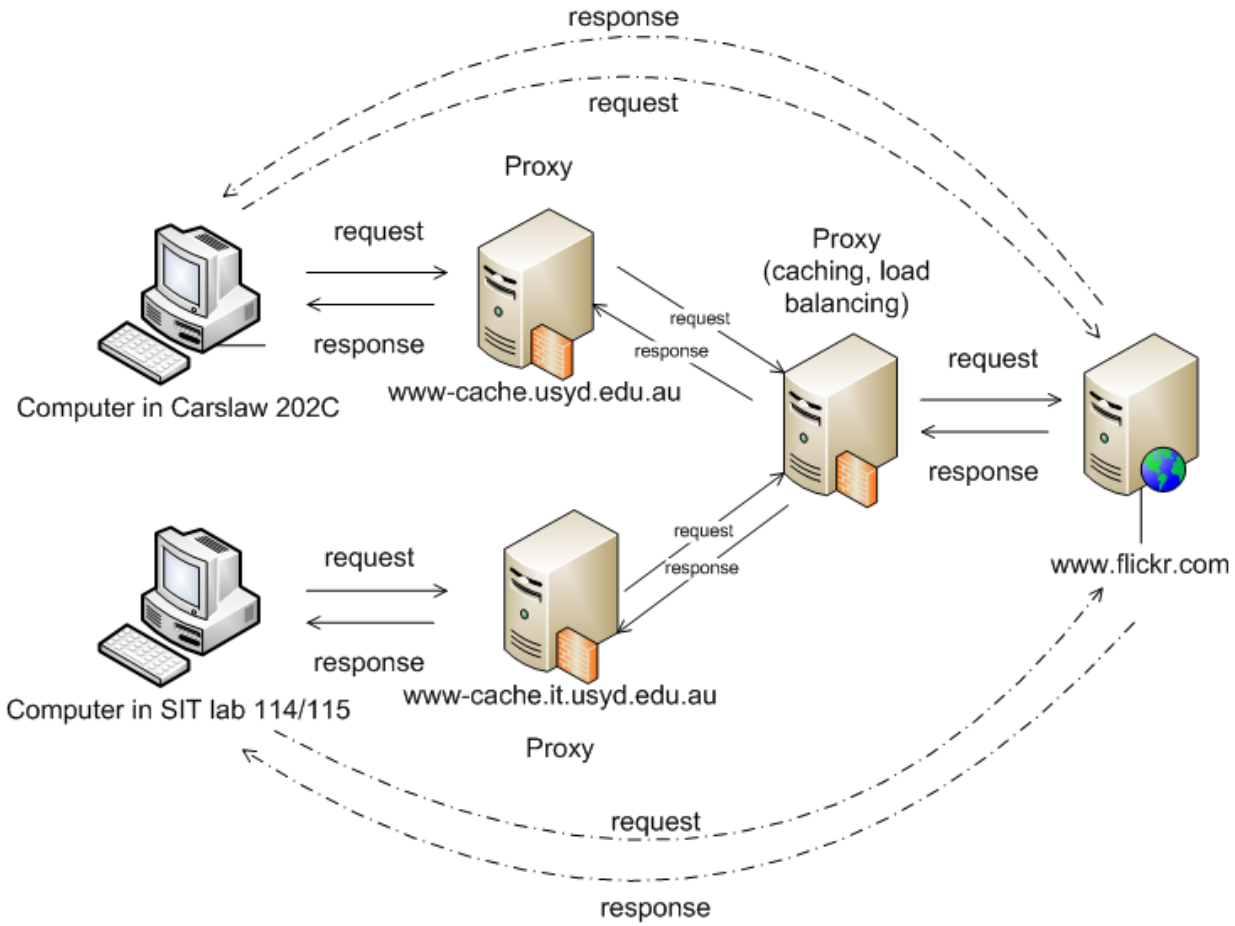
HTTP

HTTP: hypertext transfer protocol

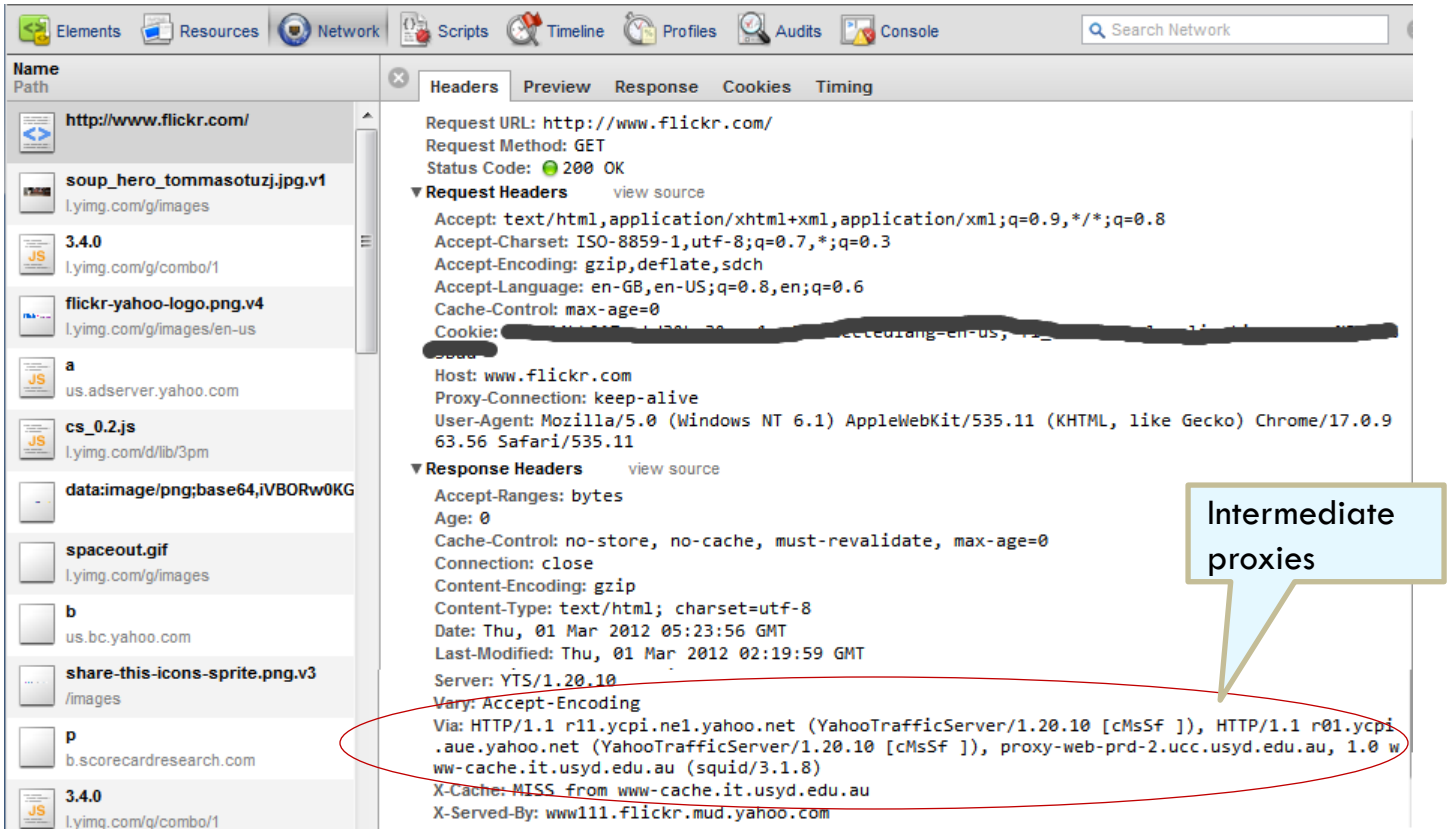
- Web's application layer protocol
  - RFC 2616 (specifications)
- Request-Response paradigm
  - *client:* browser that requests, receives, "displays" Web objects
  - *server:* Web server sends objects in response to requests



Request-response paradigm

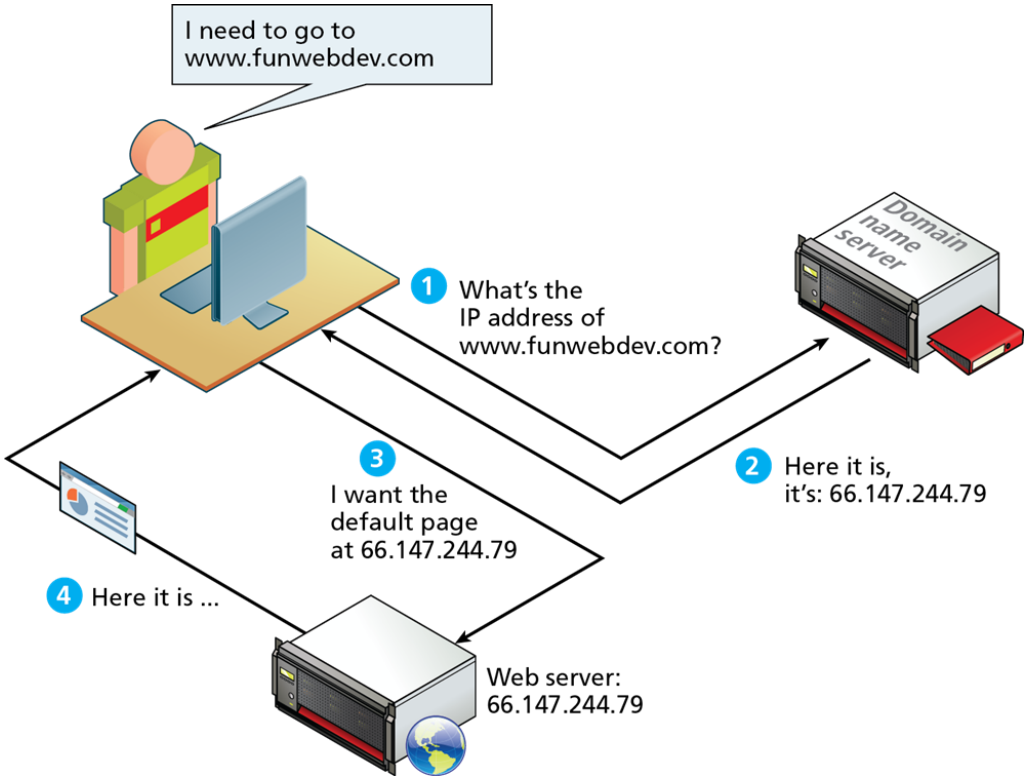


Intermediate proxies

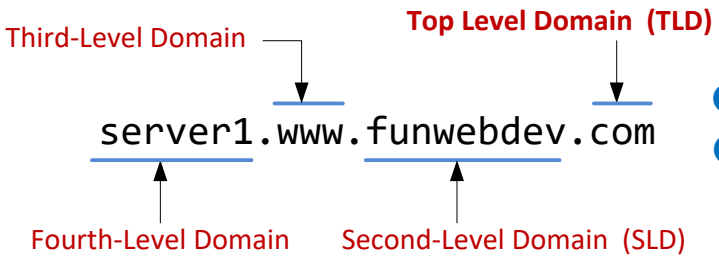


Domain Name System

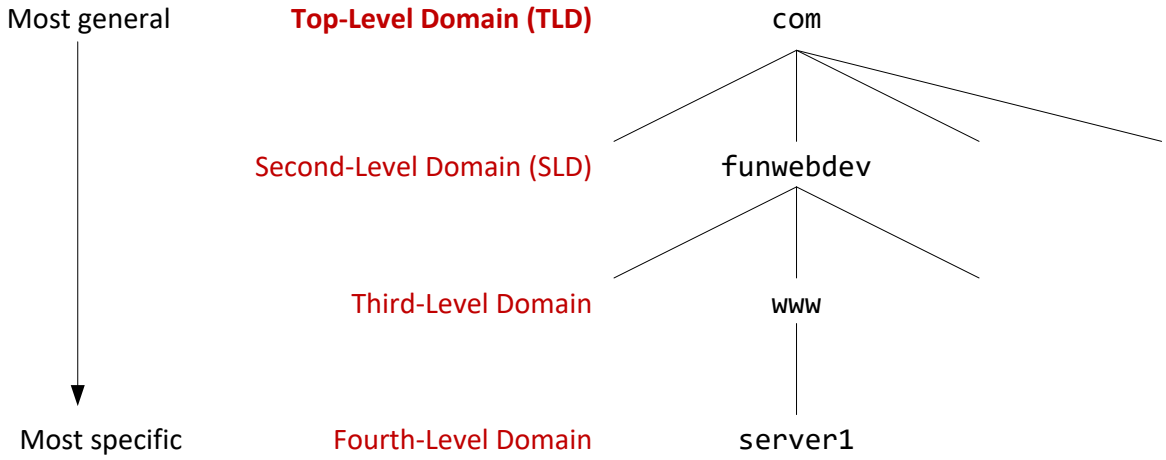
- Instead of IP addresses, we use the **Domain Name System** (**DNS**) to identify hosts on Internet



Domain Levels



Generic top-level domains (gTLD)
Country code top-level domain (ccTLD)



HTTP (cont'd)

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

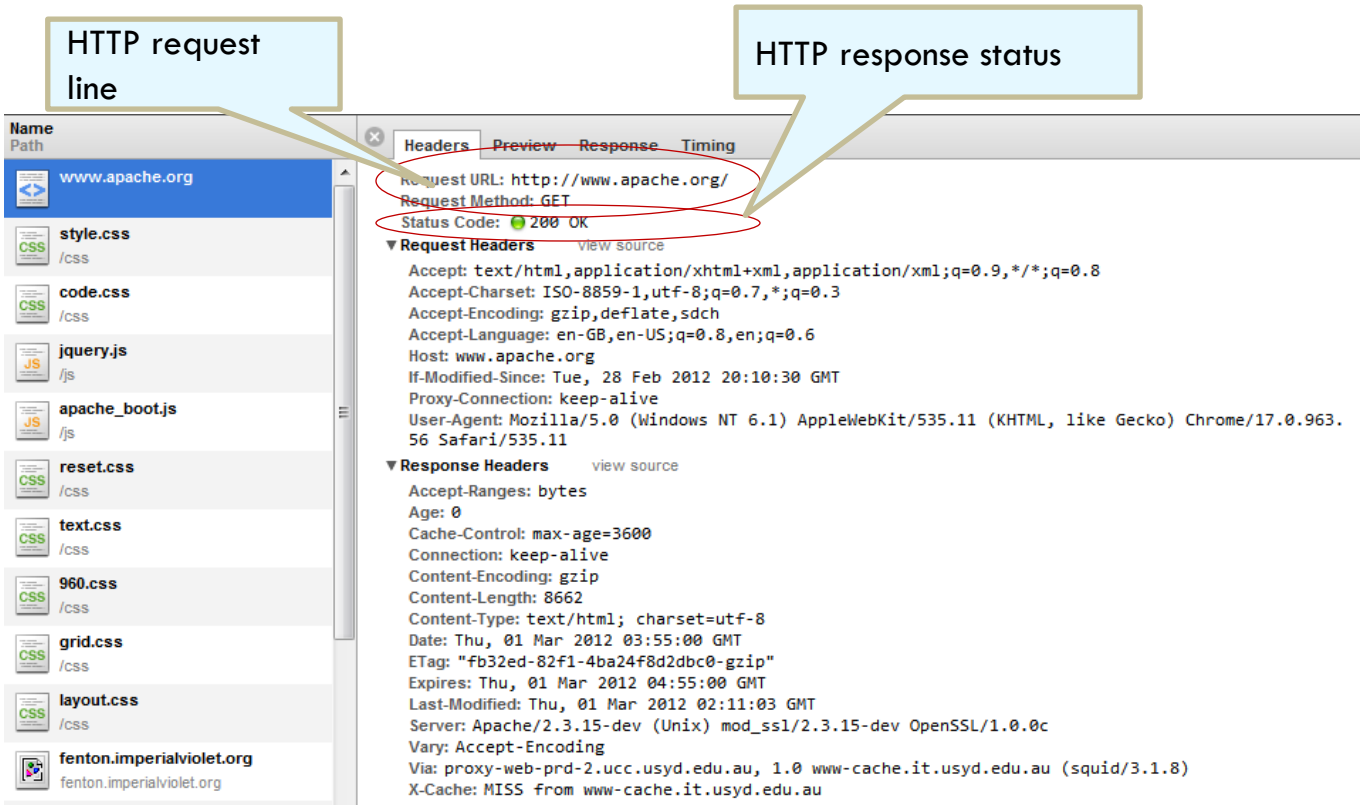
- server maintains no information about past client requests
- In contrast to protocols like FTP, SMTP

aside

Protocols that maintain "state" are complex!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

Request/Response message



Request Methods

HTTP/1.0

- GET
- POST
- HEAD
  - asks server to leave requested object out of response

HTTP/1.1

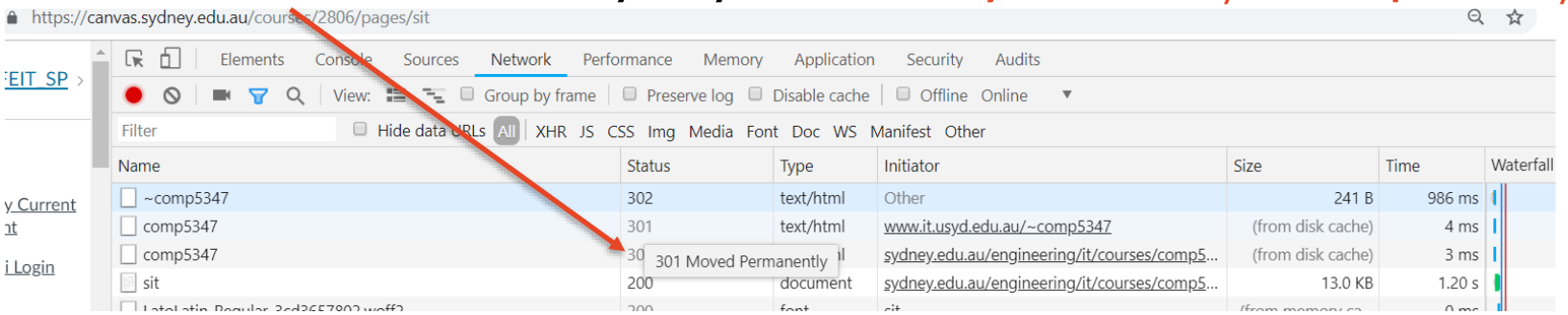
- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field
- …

HTTP Response Status Code

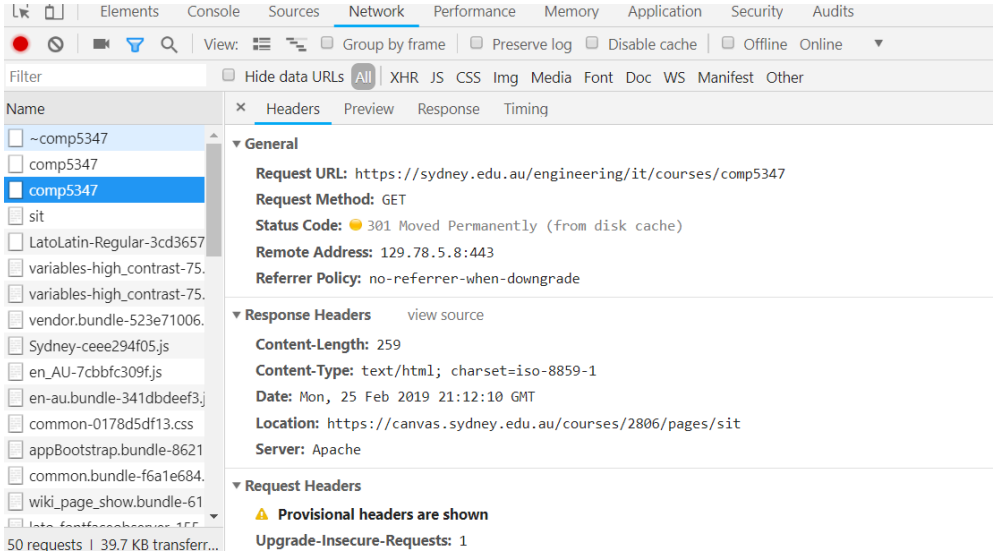
- 3-digit response code
  - 1XX – informational
  - 2XX – success
    - 200 OK
  - 3XX – redirection
    - 301 Moved Permanently
    - 302 Found/Moved temporary
    - 303 Moved Temporarily
    - 304 Not Modified
  - 4XX – client error
    - 404 Not Found
  - 5XX – server error
    - 505 HTTP Version Not Supported

Redirection status code

- 301 Moved Permanently: try www.it.usyd.edu.au/~comp5347/

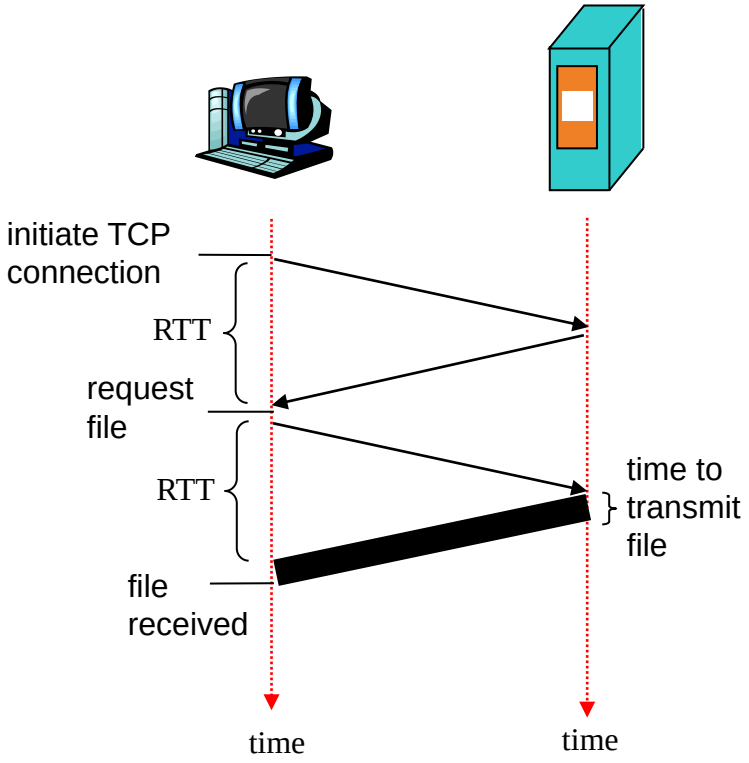


- Details of the request response can also be investigated

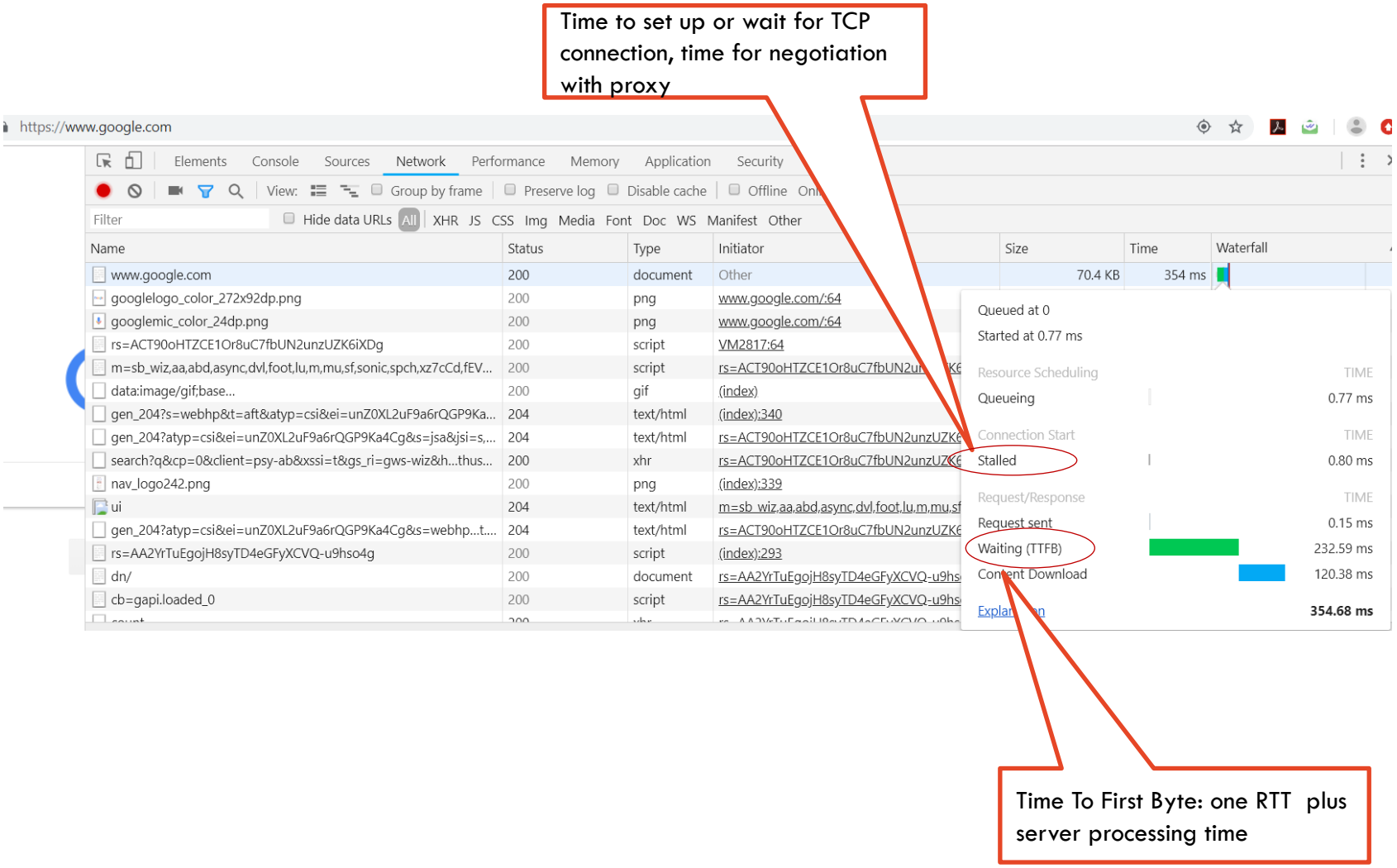


HTTP Performance

- Response Time Modeling
  - Round Trip Time (RTT): time to send a small packet to travel from client to server and back.
  - Response time:
  - one RTT to initiate TCP connection
  - one RTT for HTTP request and first few bytes of HTTP response to return
  - file transmission time
  - total = 2RTT+transmit time



HTTP performance



Nonpersistent HTTP v. Persistent HTTP

- Nonpersistent HTTP
  - At most one object is sent over a TCP connection.
  - HTTP/1.0 uses nonpersistent HTTP
- Persistent HTTP
  - Multiple objects can be sent over single TCP connection between client and server.
  - HTTP/1.1 uses persistent connections in default mode

Individual Exercise/Question

- Suppose user enters URL www.someSchool.edu/someDepartment/home.index
- Assume home.index contains text, and references to 10 jpeg images
- Ignoring server processing time, what is the approximate response time (in RTT) for HTTP 1.0 and HTTP1.1 respectively ?

Caching in HTTP

- Goal of caching in HTTP
  - Eliminate the need to send requests in many cases
    - reduce the number of network round-trips required for many operations
    - an "expiration" mechanism
  - Eliminate the need to send full responses in many other cases.
    - reduce network bandwidth requirements
    - a "validation" mechanism

- Level of caches: server side, client side (proxy and **browser**)

- Cache Correctness
  - It has been checked for equivalence with what the origin server would have returned by revalidating the response with the origin server
  - It is "fresh enough". In the default case, this means it meets the least restrictive freshness requirement of the client, origin server, and cache
  - It is an appropriate 304 (Not Modified), 305 (Proxy Redirect), or error (4xx or 5xx) response message.

Outline

- Course Information
  - Web application architecture in general
  - What are covered in this course
  - Course resources

- Basic concepts of web (from end user perspective)
  - Web page
  - HTTP protocol

- JavaScript (introduction)
  - Location and Basic Syntax
    - Variables, Control Structure, Function, Object, Array, Function

JavaScript

- Object-based, dynamically typed scripting language
  - Client-side scripting language for HTML and CSS
  - Has server-side implementation
  - "*the most popular programming language in the world*" - W3C school

- As a client-side scripting language
  - It runs inside the browser
  - Able to interact with many browser managed resources: DOM, Browser's object (BOM) such as windows, screen, history, cookies and more
  - It can be written as inline (discouraged!), embedded or as external file

Brief History

- Created in 10 days in May 1995 by Brendan Erich, then working at Netscape and now of Mozilla
  - Considered as not properly defined and targeting amateurs
- Integral part of web development in the mid 2000s with **AJAX**
  - Initial effort from Microsoft late 1999, get adopted by other browsers
  - Made very popular by Gmail and Google maps
  - Receives a lot more professional programming attention
- Many JavaScript frameworks have since created: jQuery, Prototype, AngularJS, etc.
- Server-side JavaScript also gaining popularity

JavaScript Code – Location

Inline

```
<a href="JavaScript:OpenWindow();">more info</a>
<input type="button" onClick="alert('Are you sure?');" />
```

Embedded

```
<script type="text/javascript">
    /* A JavaScript Comment */
    alert("Hello World!");
</script>
```

External

```
<head>
    <script type="text/javascript“ src="greeting.js"></script>
</head>
```

JavaScript Variables

- Declaring a variable
  - *var name;*
- Does not require specifying data types
- Can contain a value of any data type
- JavaScript automatically converts between values of different types (in many cases)
- Variable has various scopes

```
var x;       <-- a variable x is defined
var y = 0;   <-- y is defined and initialized to 0
y = 4;       <-- y is assigned the value of 4
```

Conditionals

```
var hourOfDay;   // var to hold hour of day, set it later...
var greeting;   // var to hold the greeting message.
if (hourOfDay > 4 && hourOfDay < 12){
   // if statement with condition
   greeting =  "Good Morning";
}
else if (hourOfDay >= 12 && hourOfDay < 20){
   // optional else if
   greeting =  "Good Afternoon";
}
else{ // optional else branch
   greeting = "Good Evening";
}
```

```
/* x conditional assignment */
x = (y==4) ? "y is 4" : "y is not 4";
```

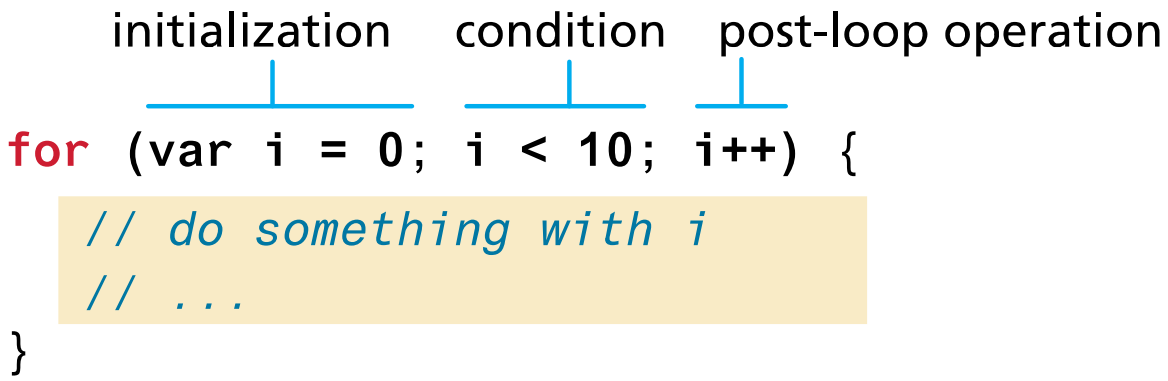
Condition — Value if true — Value if false

```
/* equivalent to */
if (y==4) {
   x = "y is 4";
}
else {
   x = "y is not 4";
}
```

Conditionals

```
switch (artType) {
    case "PT":
        output = "Painting";
        break;
    case "SC":
        output = "Sculpture";
        break;
    default:
    output = "Other";
}
```

Loops




```
var i=0; // initialise the Loop Control Variable
while(i < 10){ //test the loop control variable
    i++; //increment the loop control variable
}
```

Variable types

- Primitive types: represent simple forms of data
  - Boolean, string and number
  - Null, undefined

- Complex/reference types
  - Object (reference types)
  - Array
  - Function

Primitive Types vs. Reference Types

What is the difference between primitive types and reference types? Use the following examples to expain it to your classmate.

```
var abc = 27;

var def = ”hello”;

var foo = [45, 35, 25]

var xyz = def;

var bar = foo;

bar[0] = 200;
```

Examine the above Javascript snippet. What will be the value of foo[0] after running the script?

JavaScript – Objects

- JavaScript is different to classic OOP, which is class-based
  - It has a clear concept of ***Object*** similar to object in other OOP
  - The concept of ***Class*** is the source of confusion

- We usually start by introducing *Object*
  - An object is a collection of related data and/or functionality
  - The "data" part is referred to as "property"
  - The "functionality" part is referred to as "method"
  - In JavaScript, almost "everything" is an object
    - Most data types
    - Functions
  - The easiest way of creating an object is to use Object Literal

Object Creation using Literal

```
var objName = {
    name1: value1,
    name2: value2,
    // ...
    nameN: valueN
};
```

– Access using either of:

- objName.name1
- objName["name1"]

Object Creation using Literal

```
var person = {
        firstName:"John",
        lastName:"Doe",
        age:50,
        eyeColor:"blue“
};
```

```
var person = {
        firstName:"John",
        lastName:"Doe",
        age:50,
        eyeColor:"blue“,
        fullName : function() {
                return this.firstName + " " + this.lastName;
            }
        };
```

JavaScripts – Arrays

- Arrays are used to store multiple values in a single variable
- Object literal notation
  - var greetings = ["Good Morning", "Good Afternoon"];
- Array() constructor
  - Var greetings = new Array("Good Morning", "Good Afternoon");

- Array element is accessible with index, starting from 0, greetings[0] = “Good Morning”

- Useful methods length(), push(), reverse(), sort(),

JavaScript Functions

- **Functions** are the building blocks for modular code
  - Defined by using the reserved word **function** followed by function name and (optional) parameters

Example:

```
function subtotal (price,quantity) {
    return price * quantity;
}
```

- Call/invoke function:

```
var result = subtotal(10,2);
```

Functions – Function Expression

A function can be defined using an *anonymous function expression*

```
var calculateSubtotal = function (price,quantity) {
    return price * quantity;
};
// invokes the function
var result = calculateSubtotal(10,2);
```

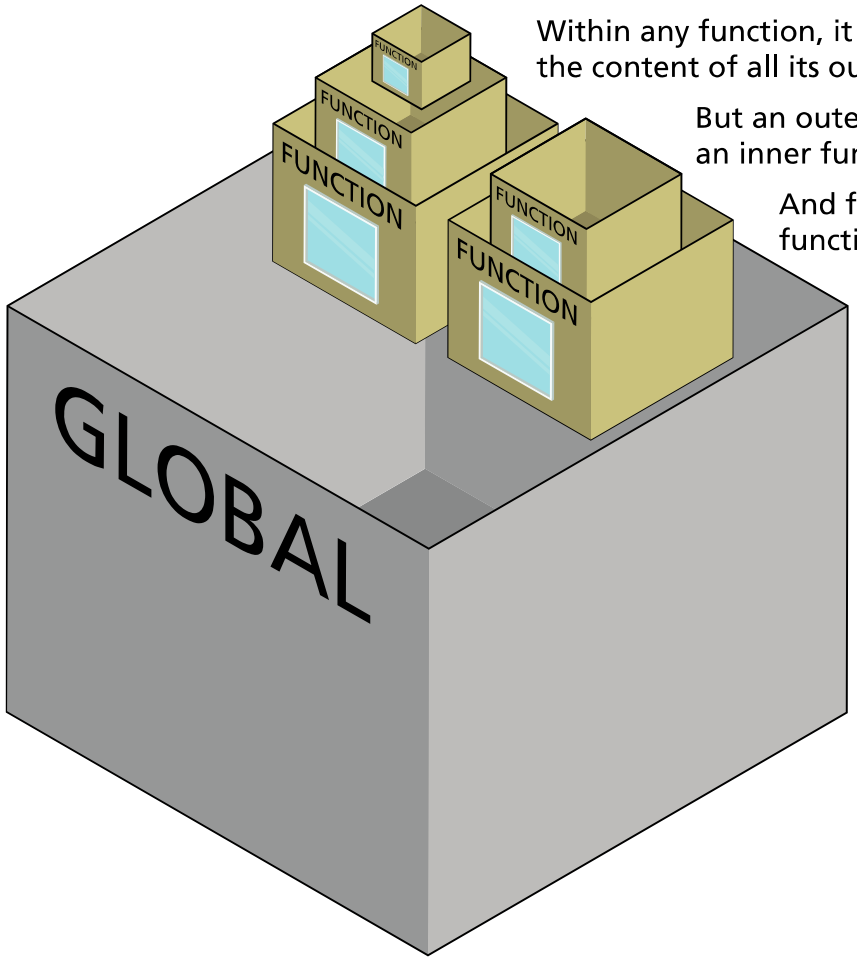
Variable Scope

- Each variable in a program has a scope
- The scope of a variable is the portion of the program in which the variable can be used
- JavaScript has function scope
  - The scope changes inside functions
- A variable declared outside a function has *global* scope
  - In the HTML context, all scripts and functions on a webpage can access it.
- Variables declared inside a function has *local* scope
  - They can only be accessed within in the function

每个变量有一个作用域， 定义在函数外的是具有全局作用域的，但是定义在函数内部的是具有局部作用域的。

JavaScript – Variable Scope

Each function is like a box
with a one-way window



Within any function, it can see out at the content of all its outer boxes

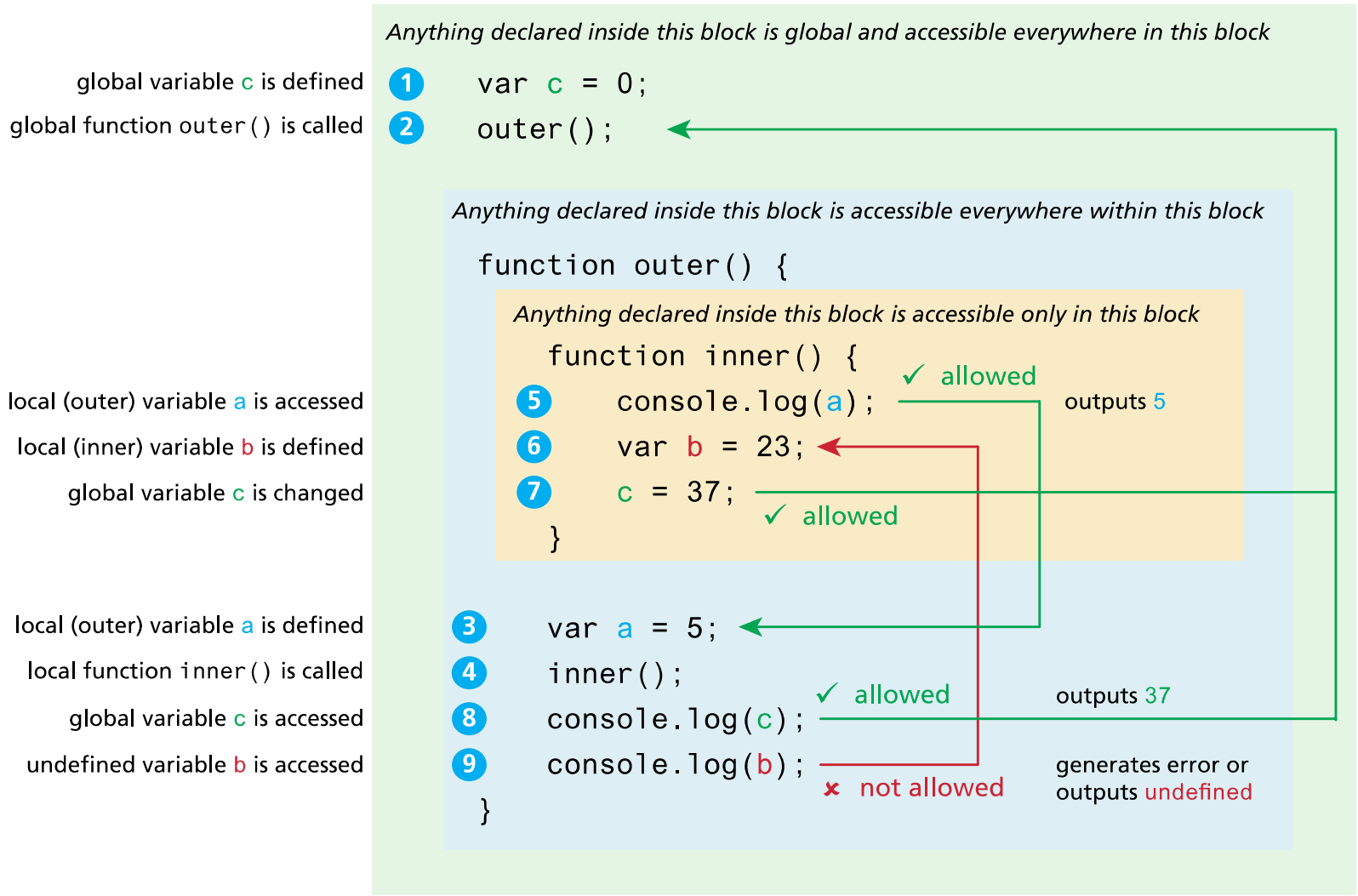
But an outer function can't look into an inner function

And functions can't see into other functions at the same level

All functions can see anything within global scope

Scope ends at global ... functions can't see outside of the global box

Variable Scope - Examples



JavaScript Output

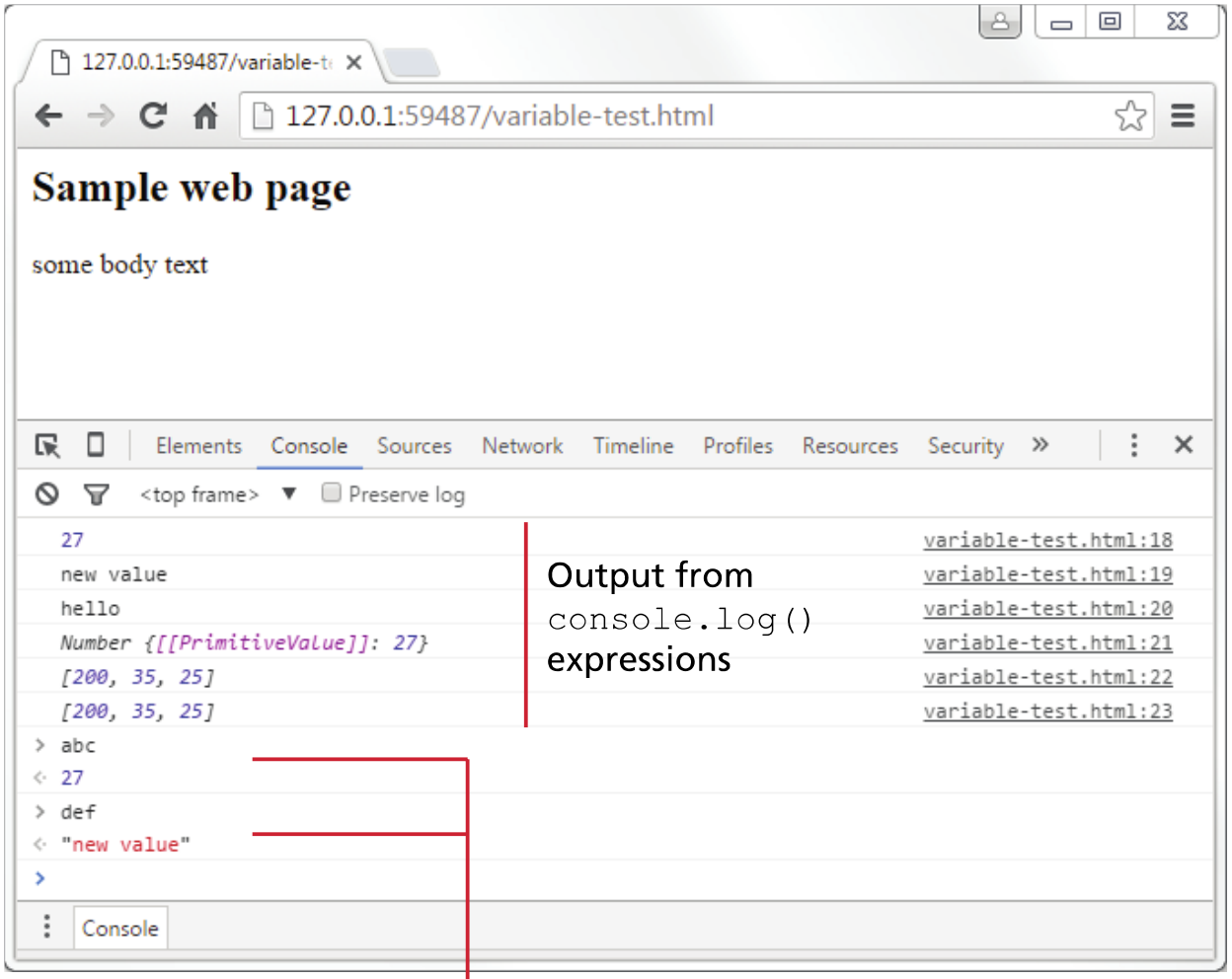
- *alert()* displays content within a pop-up box
  - *alert("Hello world");*

- *console.log()* displays content in the Browser's JavaScript console

- *document.write()* outputs the content (as mark-up) directly to the HTML document

```
var name = "COMP5347";
document.write("<h1>Title</h1>");
// this uses the concatenate operator (+)
document.write("Hello " + name + " and welcome");
```

JavaScript Output



Web page content

JavaScript console

Output from `console.log()` expressions

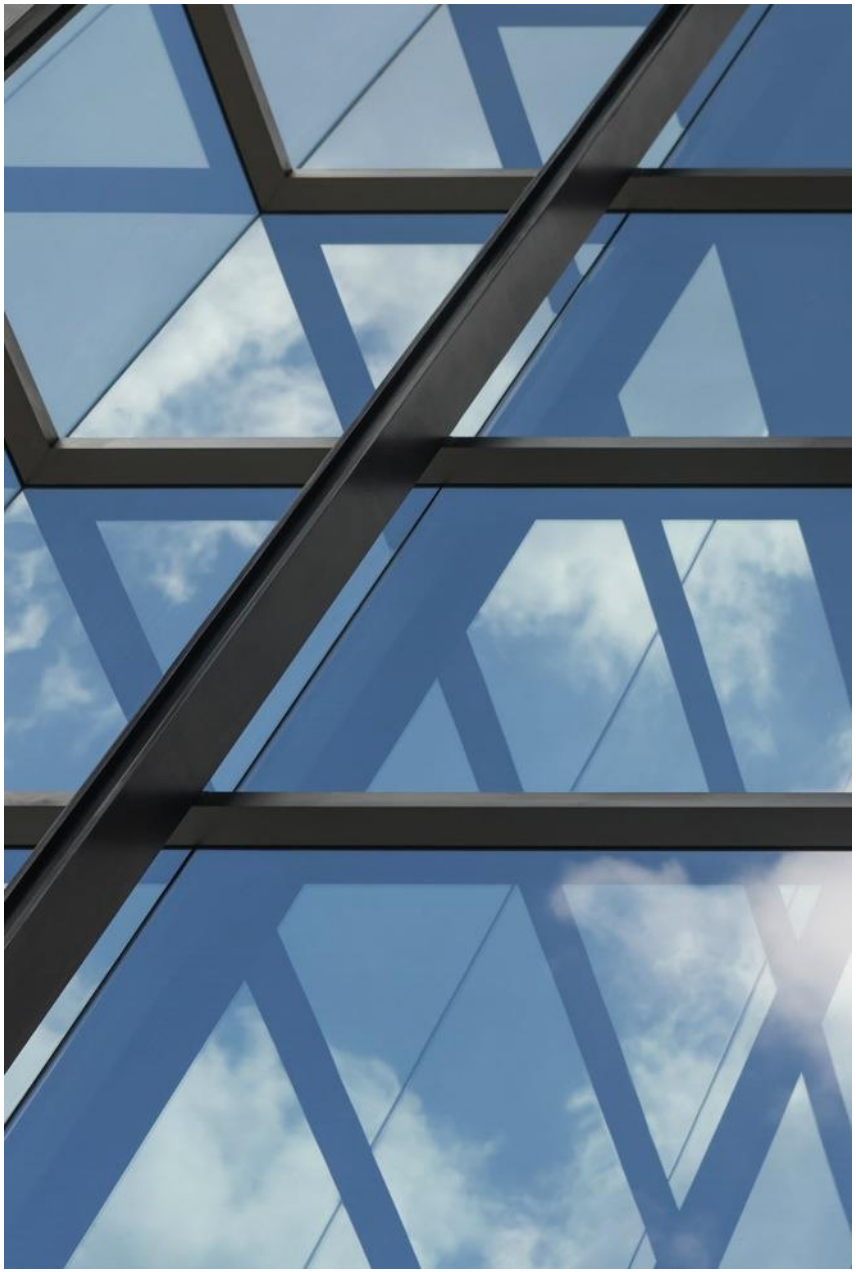
Using console interactively to query value of JavaScript variables

Week 1 Tutorial:

JavaScript (introduction)

Next week …

HTML and CSS lecture and tutorial



THE UNIVERSITY OF SYDNEY

References

1. Leon Shklar and Rich Rosen, Web application architecture. 2nd edition, Wiley, Chapter 3&7
2. Randy Connolly and Ricardo Hoar, Fundamentals of Web Development, Ch. 1 & 6
3. Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati,Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels, ***Dynamo: Amazon's Highly Available Key-Value Store***, In Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP'07), October 2007, Stevenson, Washington, USA.
4. **A Conversation with Werner Vogels**, ACM Queue, May, 2006 [http://queue.acm.org/detail.cfm?id=1142065]
5. **Facebook Architecture: Breaking it Open**, Slide 21 [https://www.slideshare.net/AditiTechnologies/facebook-architecture-breaking-it-open]
6. **Aditya Agarwal, Facebook: Science and the Social Graph, Qcon San Francisco 2008** [http://www.infoq.com/presentations/Facebook-Software-Stack]
7. **Abel Avram,** Facebook: MVC Does Not Scale, Use Flux Instead [Updated], May 15, 2014, InfoQ [http://www.infoq.com/news/2014/05/facebook-mvc-flux]