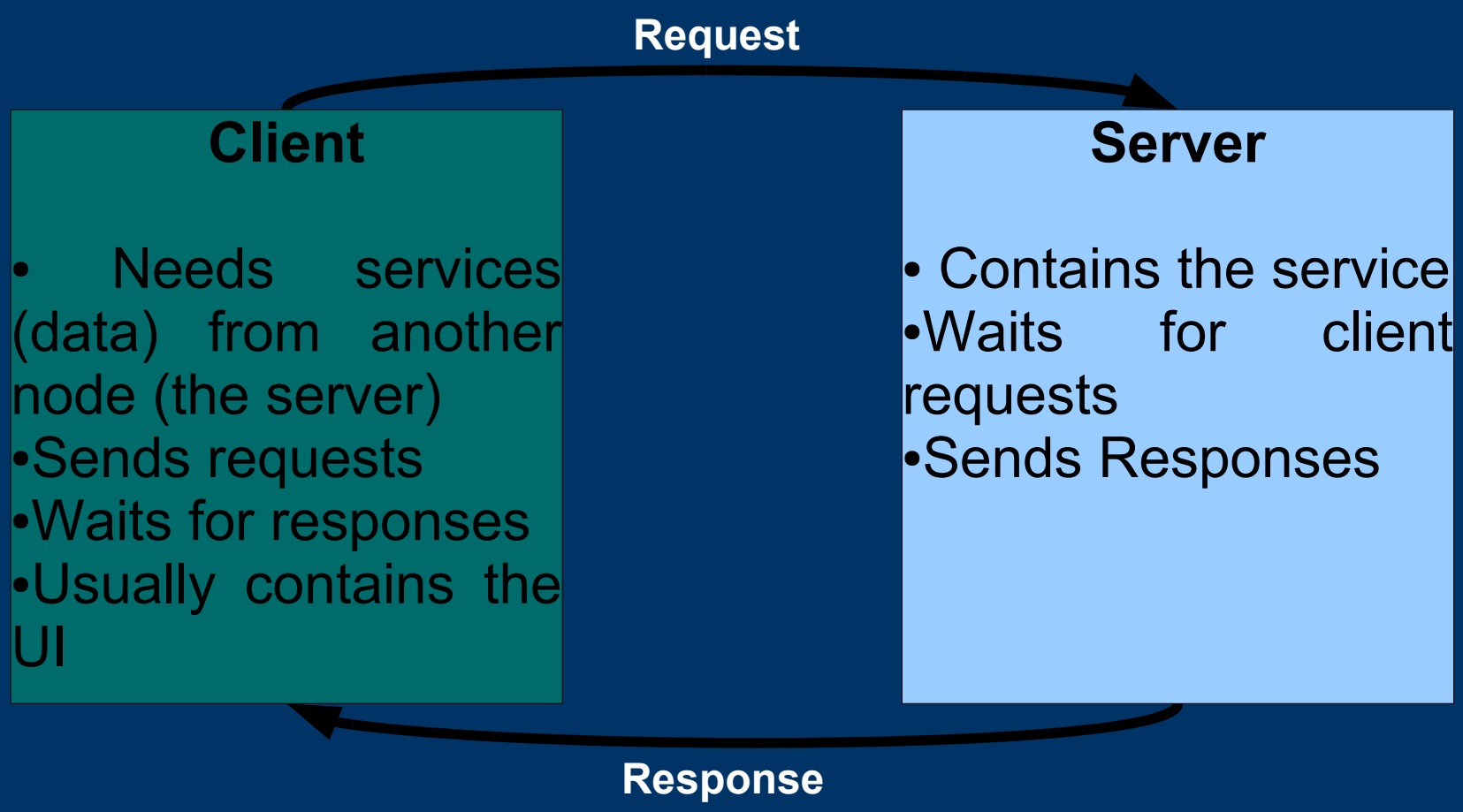
Web Interaction: HTTP

Hypertext Transfer Protocol

Client Server Architecture Request Response Model



Client – Server Apps in Web

- Client: User Agent a.k.a. The Web Browser
  - The first Web Browser created by Tim Berners-Lee (WorldWideWeb)
  - The software we use to **surf** the web.

Client-Server Apps in Web

- Server: The Web Server
  - Also known as HTTP Server.
  - Server that hosts web content.
  - Most web servers can be configured to handle **dynamic content**
    - Usually, 'attach' a programming language and related tools to the server and you're ready to do web programming.

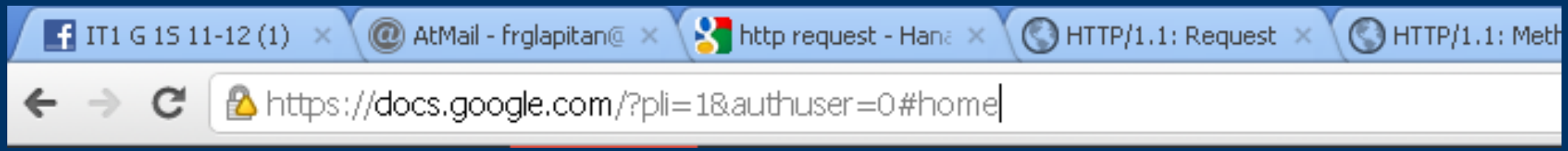
Browser – Server Interaction

- **PROTOCOL**
  - System of that two or more communicating parties follow when they 'talk' with each other.
  - For Web: HTTP, Hypertext Transfer Protocol
  - When Accessing a web page, the browser makes an HTTP Request (message) and sends to the server.
  - Server creates one or more HTTP Responses as a response to request.
  - HTTP is a “text-based” protocol.

The HTTP Request

When does a browser makes a request?

- **When user types a URL on the address bar of the browser.**



The HTTP Request

When does a browser makes a request?

- **When additional resources are linked to a resource.**

The HTTP Request

When does a browser makes a request?

- **When the user clicks on a hyperlink (and the hyperlink refers to a Web Resource).**

The HTTP Request

When does a browser makes a request?

- **As a response to a** *request redirect*.

The HTTP Request

When does a browser makes a request?

- **User Submits a Web Form.**

The HTTP Request

- HTTP Request
  - The message that browsers send to web servers.
  - Contains information about the request, **kind** of request, where the request comes from, type of browser used, etc...

```
Request          = Request-Line
                   *(( general-header
                       | request-header
                       | entity-header ) CRLF)
                   CRLF

                   [ message-body ]
```

Parts of the Request

- Request Line
  - The First Line of the Request
  - It tells us what kind of HTTP Request it is (Method)
  - The URI (resource) being requested
  - Protocol Version.

```
Request-Line= Method SP Request-URI SP HTTP-Version CRLF
```

Parts of the Request Line : Kinds of Requests

```
Method  = "OPTIONS"
        | "GET"
        | "HEAD"
        | "POST"
        | "PUT"
        | "DELETE"
        | "TRACE"
        | "CONNECT"
        | extension-method

extension-method = token
```

Kinds of Requests

- OPTIONS - Request for information about communication options available at the service provider (server) and/or options regarding a resource.
- GET – retrieve whatever information is identified by the Request-URI.
- HEAD – identical to GET but only gets metainformation about the request.
- POST – covers different functions
  - Annotation of existing resources
  - “posting” a message
  - Providing data for a data handling process
  - Extending a database

Kinds of Requests

- PUT – store supplied entity at the server under supplied URI
- DELETE – removes resource identified by request-URI
- TRACE – remote, application-layer loop back of the message (for debugging)
- CONNECT – for use with a proxy.

Parts of the Request Line: Request URI & HTTP Version

- Request-URI: URI of the resource, may be relative* or absolute
- HTTP Version
  - HTTP/0.9
  - HTTP/1.0
  - HTTP/1.1

Examples

```
GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1

OPTIONS * HTTP/1.1

DELETE  /pub/WWW/TheProject.html HTTP/1.1
Host: www.w3.org

TRACE * HTTP/1.1
```

Headers

- From the second line up to the empty line before the message body are called **headers**
- **Three** types of headers
  - **General Headers:** Common headers also used in responses, e.g. Date, Host, etc..
  - **Request Headers:** Information about the Client
  - **Entity Headers:** Information about the content of the request message body (e.g. When transferring files via upload).

```
Header-Name: Value CRLF
```

General Headers

```
general-header = Cache-Control
               | Connection
               | Date
               | Pragma
               | Trailer
               | Transfer-Encoding
               | Upgrade
               | Via
               | Warning
```

Request Headers

```
request-header = Accept
               | Accept-Charset
               | Accept-Encoding
               | Accept-Language
               | Authorization
               | Expect
               | From
               | Host
               | If-Match
               | If-Modified-Since
               | If-None-Match
               | If-Range
               | If-Unmodified-Since
               | Max-Forwards
               | Proxy-Authorization
               | Range
               | Referer
               | TE
               | User-Agent
```

Entity Headers

```
entity-header  = Allow
               | Content-Encoding
               | Content-Language
               | Content-Length
               | Content-Location
               | Content-MD5
               | Content-Range
               | Content-Type
               | Expires
               | Last-Modified
               | extension-header

extension-header = message-header
```

Message Body

- Message Body = Entity Body encoded as per Encoding specified by entity headers.
  - Instances when there is a message body:
    - When submitting a POST request (i.e. form data is encoded in the body).
    - When submitting a form with a file attachment (i.e. file upload).

Example

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/
    jpeg, image/pjpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE
    5.01; Windows NT)
Host: hypothetical.ora.com
Connection: Keep-Alive
```

HTTP Response

- Also a text encoded message from web server to browser.
- One request may trigger multiple responses depending on size of response.

```
Response        = Status-Line
                  *(( general-header
                   | response-header
                   | entity-header ) CRLF)
                  CRLF
                  [ message-body ]
```

Parts of HTTP Response: Status-Line

- Status line tells us
  - HTTP Version
  - The Status Code and Phrase, which tells us if the request was successfully processed (or not)

```
Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
```

Status Code and Reason Phrase

- Status Code: A 3-digit number corresponding to a response type.
- Reason Phrase: Describes the response or result of processing the request.
- Status Codes grouped according to kind of response, grouping determined by the first digit.

Status Code and Reason Phrase

Status Codes

- **1xx**: Informational - Request received, continuing process

- **2xx**: Success - The action was successfully received, understood, and accepted

- **3xx**: Redirection - Further action must be taken in order to complete the request

- **4xx**: Client Error - The request contains bad syntax or cannot be fulfilled

- **5xx**: Server Error - The server failed to fulfill an apparently valid request

Assignment:

MEMORIZE
41 or so
Status Codes and Corresponding Phrases

Response Header

- Contains information about the server.

```
response-header = Accept-Ranges
                | Age
                | ETag
                | Location
                | Proxy-Authenticate
                | Retry-After
                | Server
                | Vary
                | WWW-Authenticate
```

Response Example

```
HTTP/1.1 200 OK
Date: Mon, 06 Dec 1999 20:54:26 GMT
Server: Apache/1.3.6 (Unix)
Last-Modified: Fri, 04 Oct 1996 14:06:11 GMT
ETag: "2f5cd-964-381e1bd6"
Accept-Ranges: bytes
Content-length: 327
Connection: close
Content-type: text/html

<title>Sample Homepage</title>
<img src="/images/oreilly_mast.gif">
<h1>Welcome</h2>
Hi there, this is a simple web page.  Granted, it may not
be as elegant as some other web pages you've seen on the net,
 but there are
some common qualities:

<ul>
  <li> An image,
  <li> Text,
  <li> and a <a href="/example2.html"> hyperlink. </a>
</ul>
```

HTTP Communication