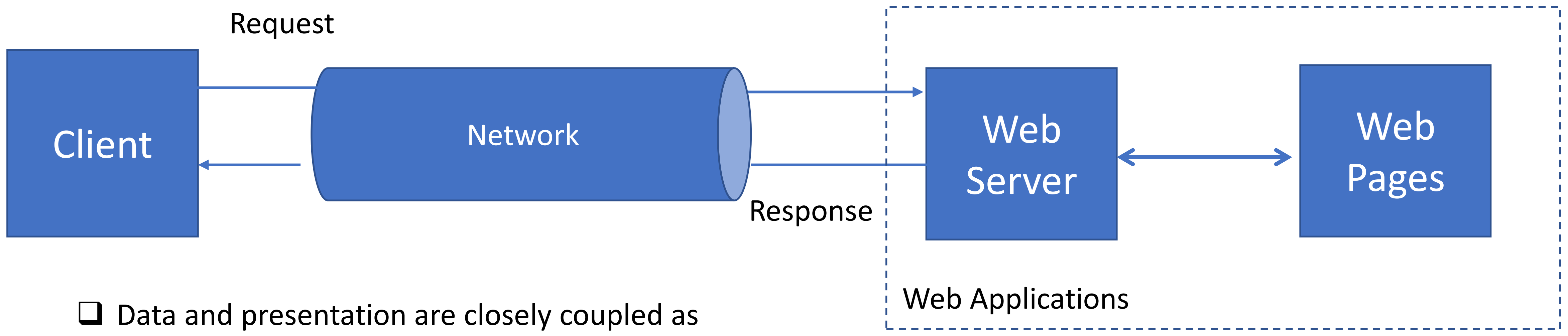
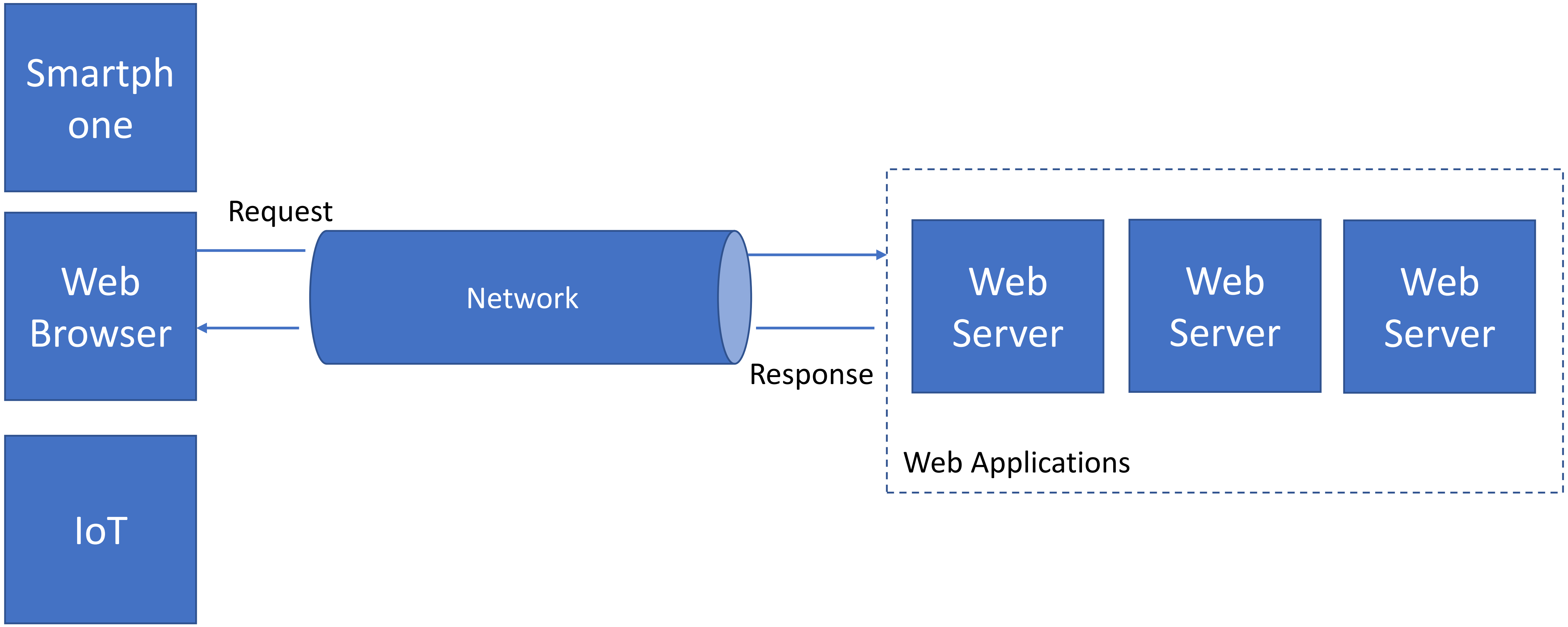
HTTP Revisited

Chandreyee Chowdhury

Web App Architecture-web 1.0



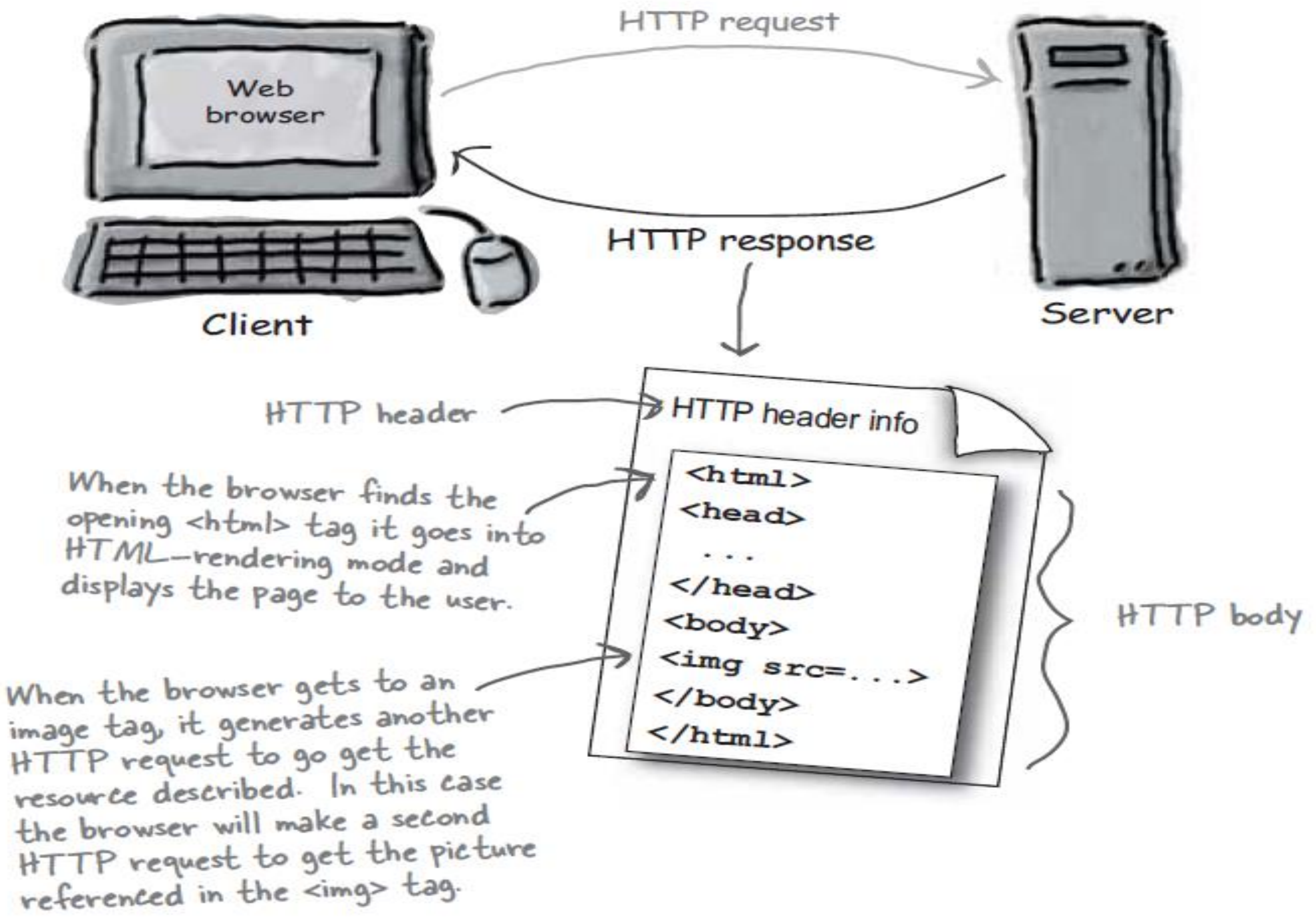
- Data and presentation are closely coupled as web pages are static

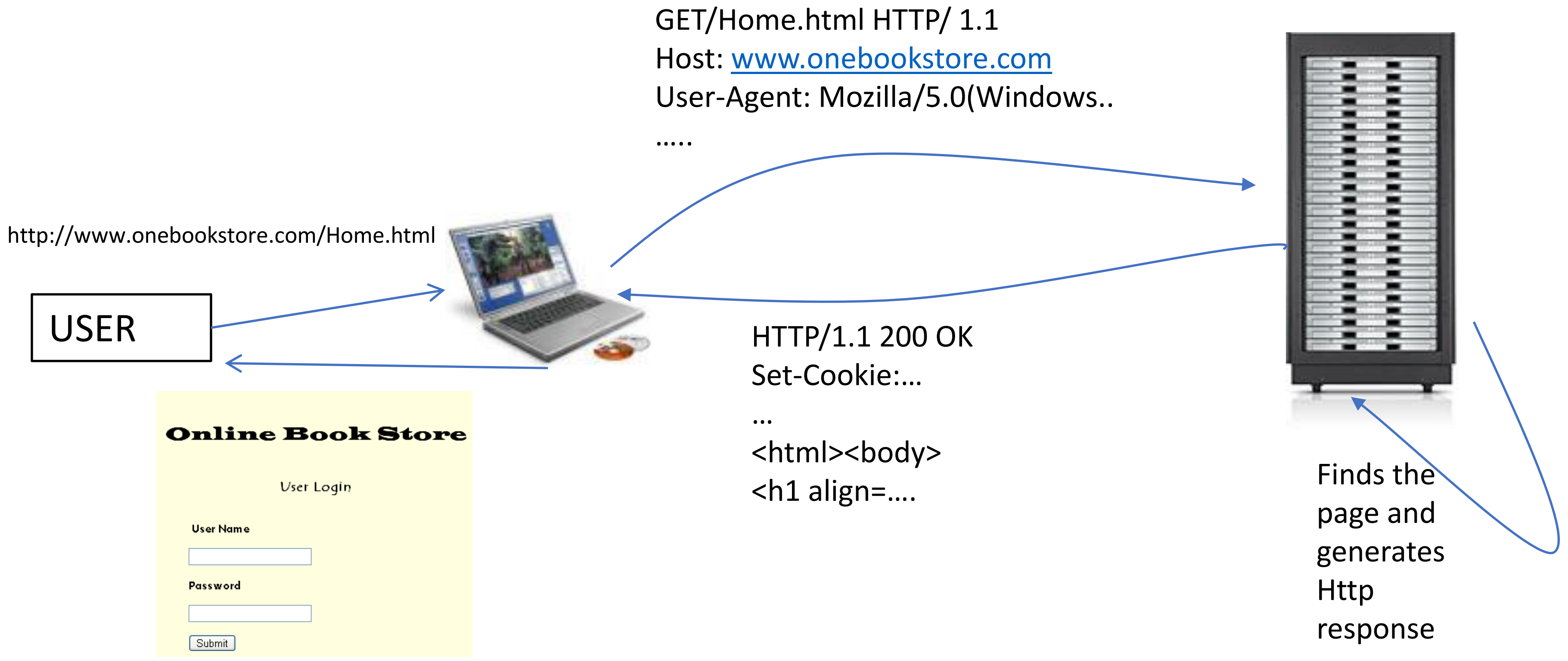

Smartphone
Web Browser
IoT
Request
Network
Response
Web Server
Web Server
Web Server
Web Applications

Why HTTP

- It provides a uniform interface to access the resources and services from a web server or cloud
- Reuse infrastructure for ubiquity
  - Web is ubiquitous
- Reusing
  - Application frameworks and libraries
  - Load balancing infrastructure for different applications including interacting with cloud
  - Session handling
- Distribute requests throughout the servers

HTTP and HTML

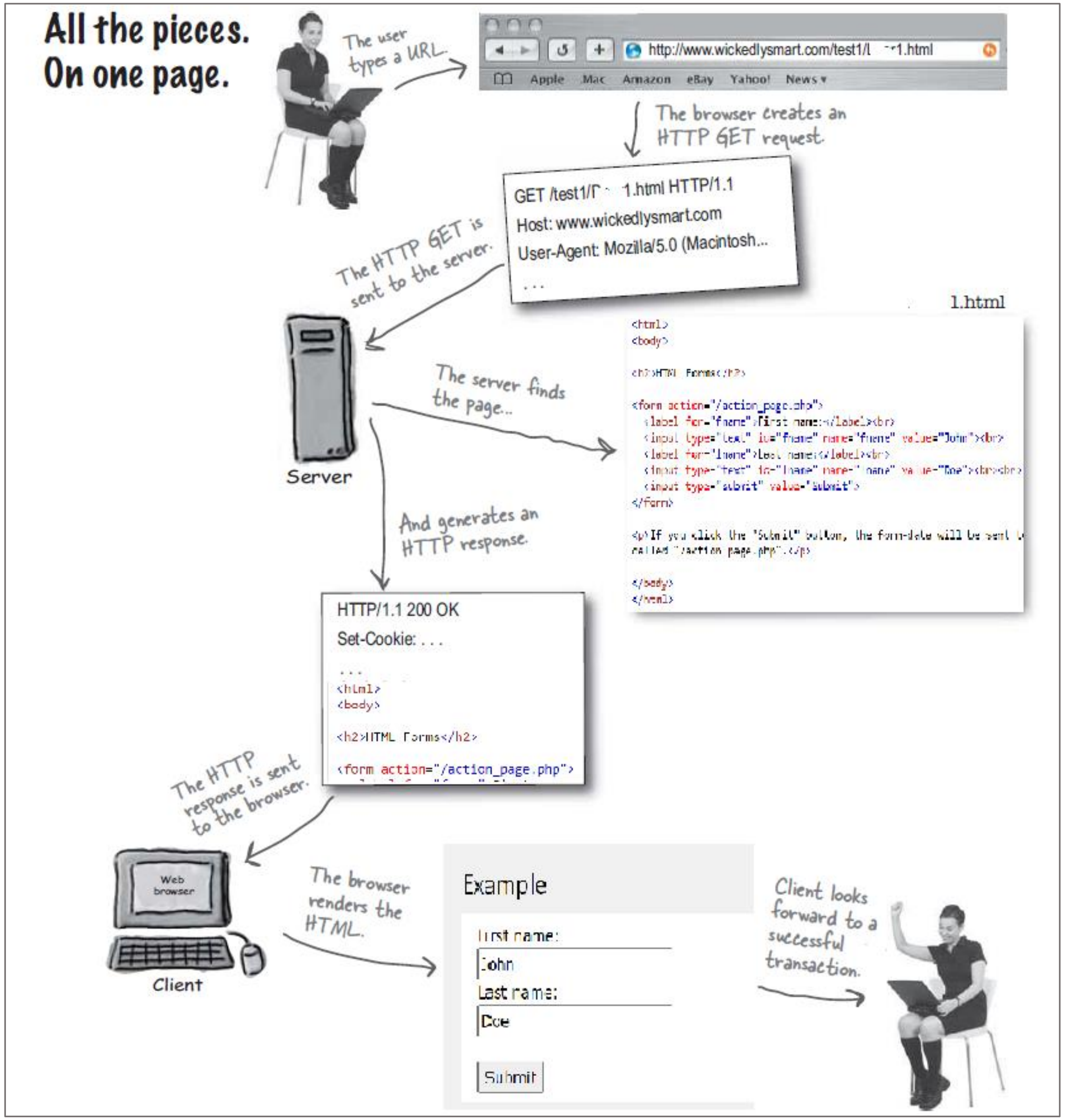

HTTP request
Web browser
HTTP response
Client
Server
HTTP header
HTTP header info
<html>
<head>
...
</head>
<body>
<img src=...>
</body>
</html>
HTTP body
When the browser finds the opening <html> tag it goes into HTML-rendering mode and displays the page to the user.
When the browser gets to an image tag, it generates another HTTP request to go get the resource described. In this case the browser will make a second HTTP request to get the picture referenced in the <img> tag.


http://www.onebookstore.com/Home.html
USER
Online Book Store
User Login
User Name
Password
Submit
Finds the page and generates Http response

```
GET/Home.html HTTP/ 1.1
Host: www.onebookstore.com
User-Agent: Mozilla/5.0(Windows..
.....
```

```
HTTP/1.1 200 OK
Set-Cookie:...
...
<html><body>
<h1 align=....
```

HTTP Request Response


All the pieces.
On one page.
The user types a URL.
http://www.wickedlysmart.com/test1/[illegible]1.html
Apple
Mac
Amazon
eBay
Yahoo!
News
The browser creates an HTTP GET request.
The HTTP GET is sent to the server.
Server
The server finds the page...
1.html
And generates an HTTP response.
The HTTP response is sent to the browser.
Web browser
Client
The browser renders the HTML.
Example
First name:
John
Last name:
Doe
Submit
Client looks forward to a successful transaction.

```
GET /test1/[illegible]1.html HTTP/1.1
Host: www.wickedlysmart.com
User-Agent: Mozilla/5.0 (Macintosh...
. . .
```

1.html

```
<html>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>

<p>If you click the "Submit" button, the form-data will be sent t
called "/action_page.php".</p>

</body>
</html>
```

```
HTTP/1.1 200 OK
Set-Cookie: . . .
. . .
<html>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php">
```

Http Request

- ***Every request has a method and a resource path***
- ***HTTP GET***
  - The total amount of characters in a GET is really limited (depending on the server)
  - The data you send with the GET is appended to the URL up in the browser bar, so whatever you send is exposed
  - Because of this, the user can bookmark a form submission if you use GET
- HTTP POST
  - The data is included in the request body
  - More data can be sent
  - General purpose sending of data

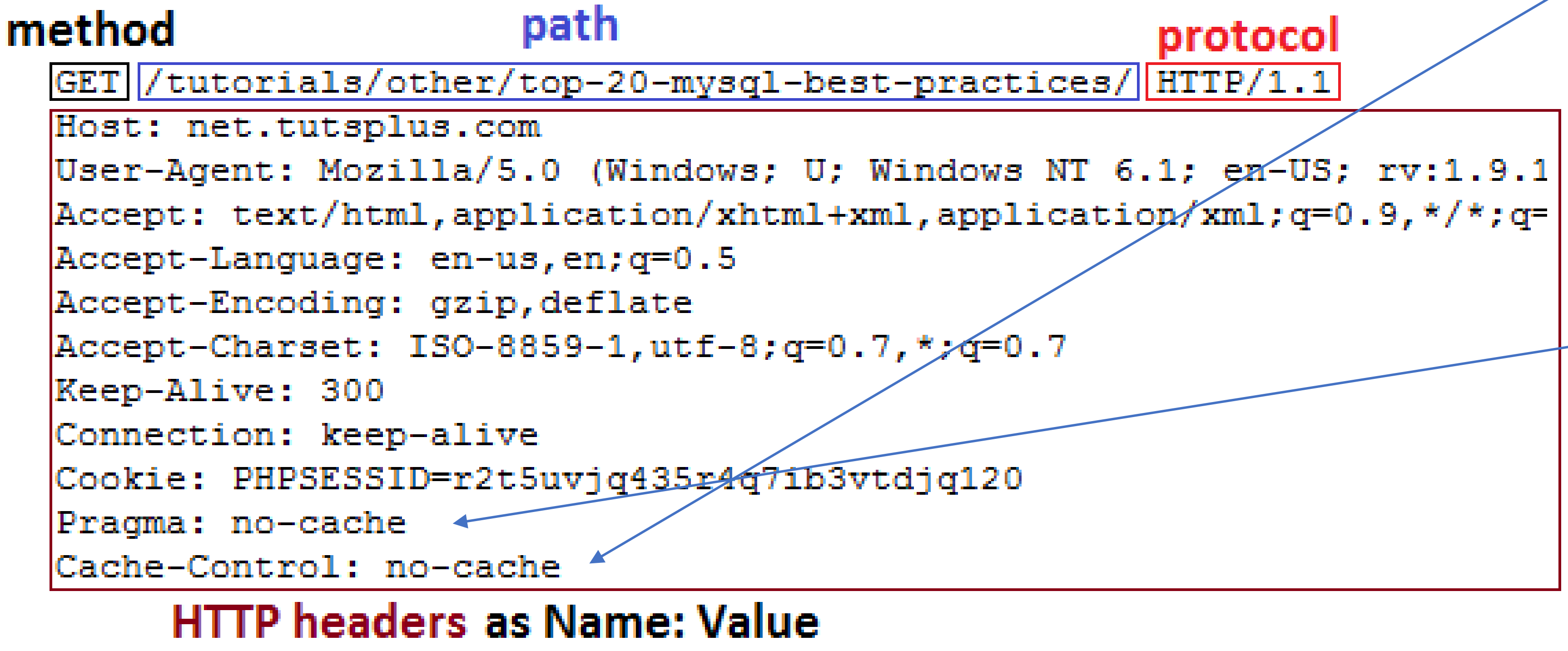
Http Methods

- Put
  - Asking the server to store some Data
- Delete
  - Remove some information from the server

GET POST PUT DELETE	PATH + Resource

Request line

- GET/com/Kolkata/Home.html HTTP/ 1.1
- Method
- <path+resource>



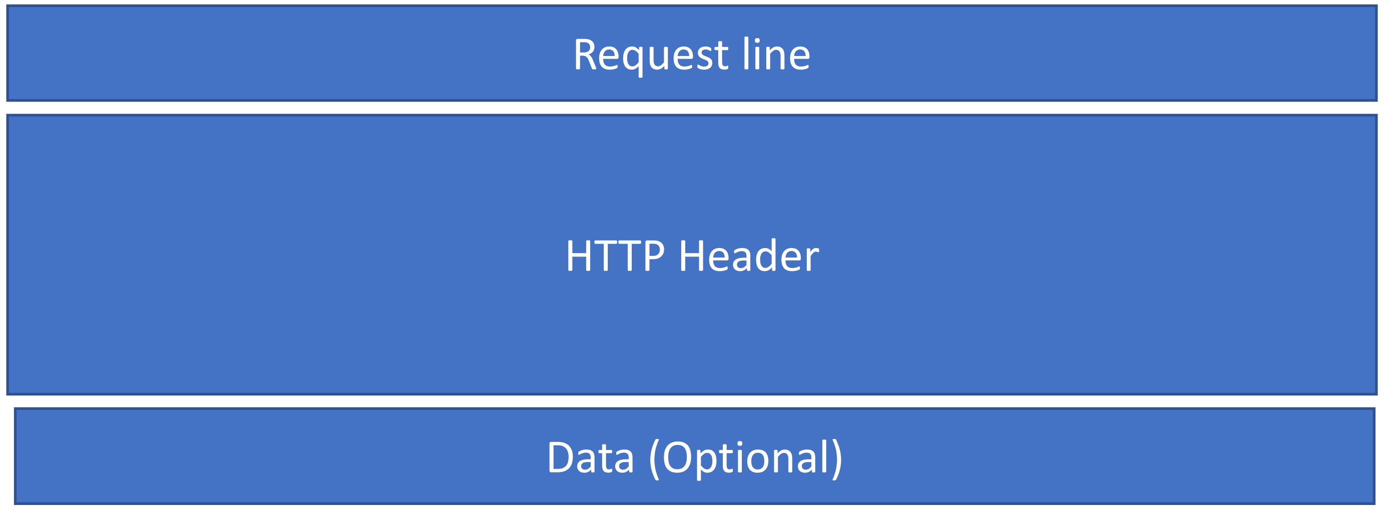
```
GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1
Host: net.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
Pragma: no-cache
Cache-Control: no-cache
```

The response may be stored by *any* cache, even if the response is normally non-cacheable. However, the stored response MUST *always* go through validation with the origin server first before using it

The server MUST NOT use a cached copy when responding to such a request.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

HTTP Headers



- Headers are meta information but body of a HTTP message contains pure data
  - These are the extra information that the client is giving the server to help it complete that Request.
- If message body is sent without the header then the server may process the request
  - May not send the response in the expected format
- When body of a message is missing when it was required, the relevant information would not be processed

Uniform Resource Locator

https://wishnet.in/home/login.html

- The way resources are identified is called a URL
- `http://<host name>:<port number>/path/resource?key1=value1&key2=value2`
- Using the query parameters we can pass extra information about a specific aspect of a resource to the server
- URL encoding encodes any character that is not allowed in the query param spec
- For dynamically constructed URLs with data, it is better to encode all URLs as data may not follow the spec
- It is good to provide the correct file extension in the encoded URLs but not a requirement

Image/jpg

Image/png

Text/plain

Text/html

Data Types

- The way data is stored in the server and the way it is sent in the body may differ
  - MIME type allows this adaptation
- A media type (also known as a Multipurpose Internet Mail Extensions or MIME type) indicates the nature and format of a document, file, or assortment of bytes. MIME types are defined and standardized in IETF's RFC 6838
- There should be some way of interpreting the type of data sent in body
  - Image data
- The type represents the general category into which the data type falls, such as video or text.
  - The subtype identifies the exact kind of data of the specified type
- All of these different MIME types are identifiers for a well known format for the data in the body of either a request or a response.
  - Based on the MIME type the data will be processed
- MIME types are changed between client and server

Content-Type: multipart/form-data

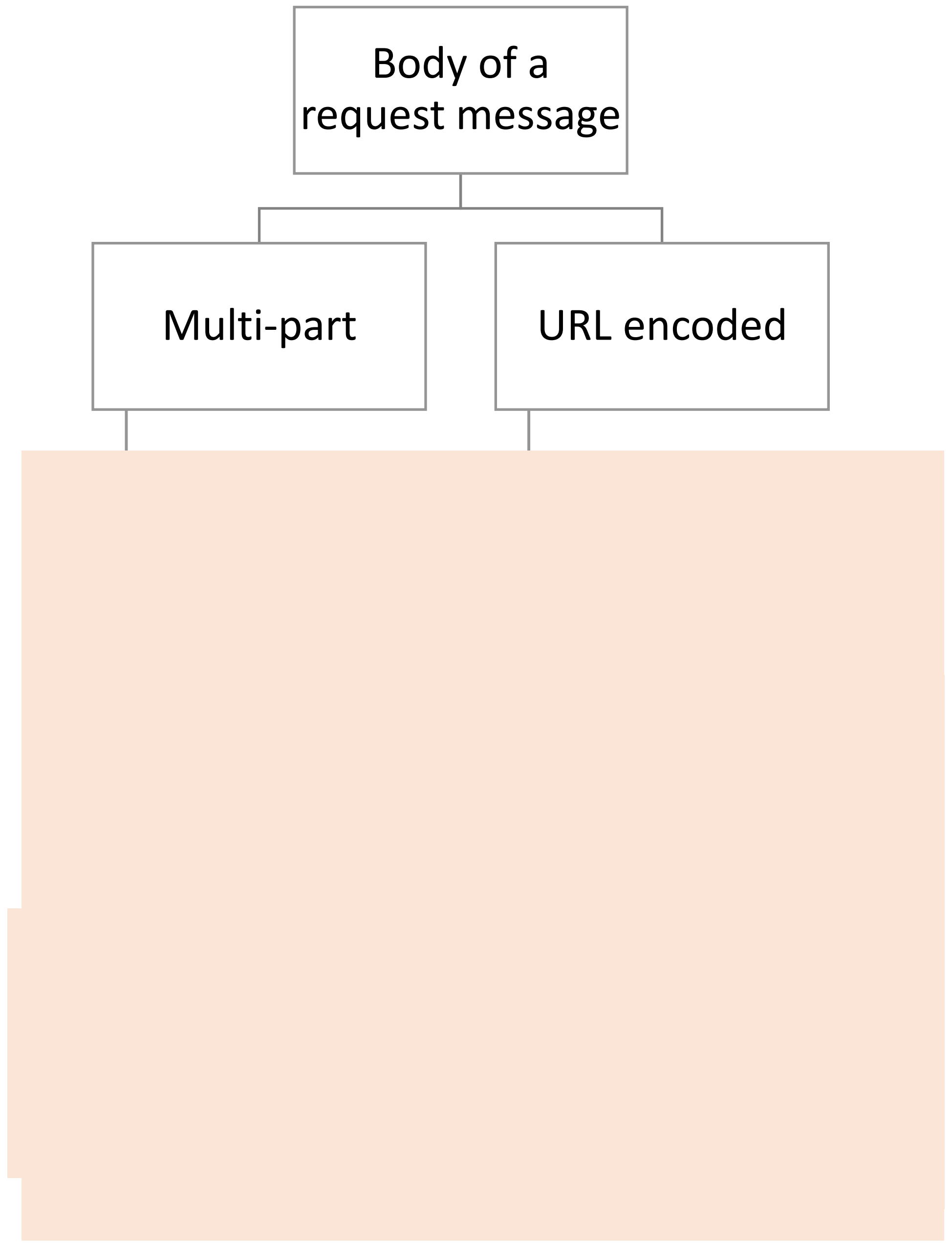
- These are written as content types
- Multipart types indicate a category of document broken into pieces, often with different MIME types

```html
<form action="/SelectCoffeeType.html" method="post" enctype="multipart/form-data">
   <input type="text" name="description" value="some text">
   <input type="file" name="myFile">
   <button type="submit">Submit</button>
</form>
```

```
POST /foo HTTP/1.1
Content-Length: 68137
Content-Type: multipart/form-data; boundary=---------------------------974767299852498929531610575

-----------------------------974767299852498929531610575
Content-Disposition: form-data; name="description"

some text
-----------------------------974767299852498929531610575
Content-Disposition: form-data; name="myFile"; filename="foo.txt"
Content-Type: text/plain

(content of the uploaded file foo.txt)
-----------------------------974767299852498929531610575--
```

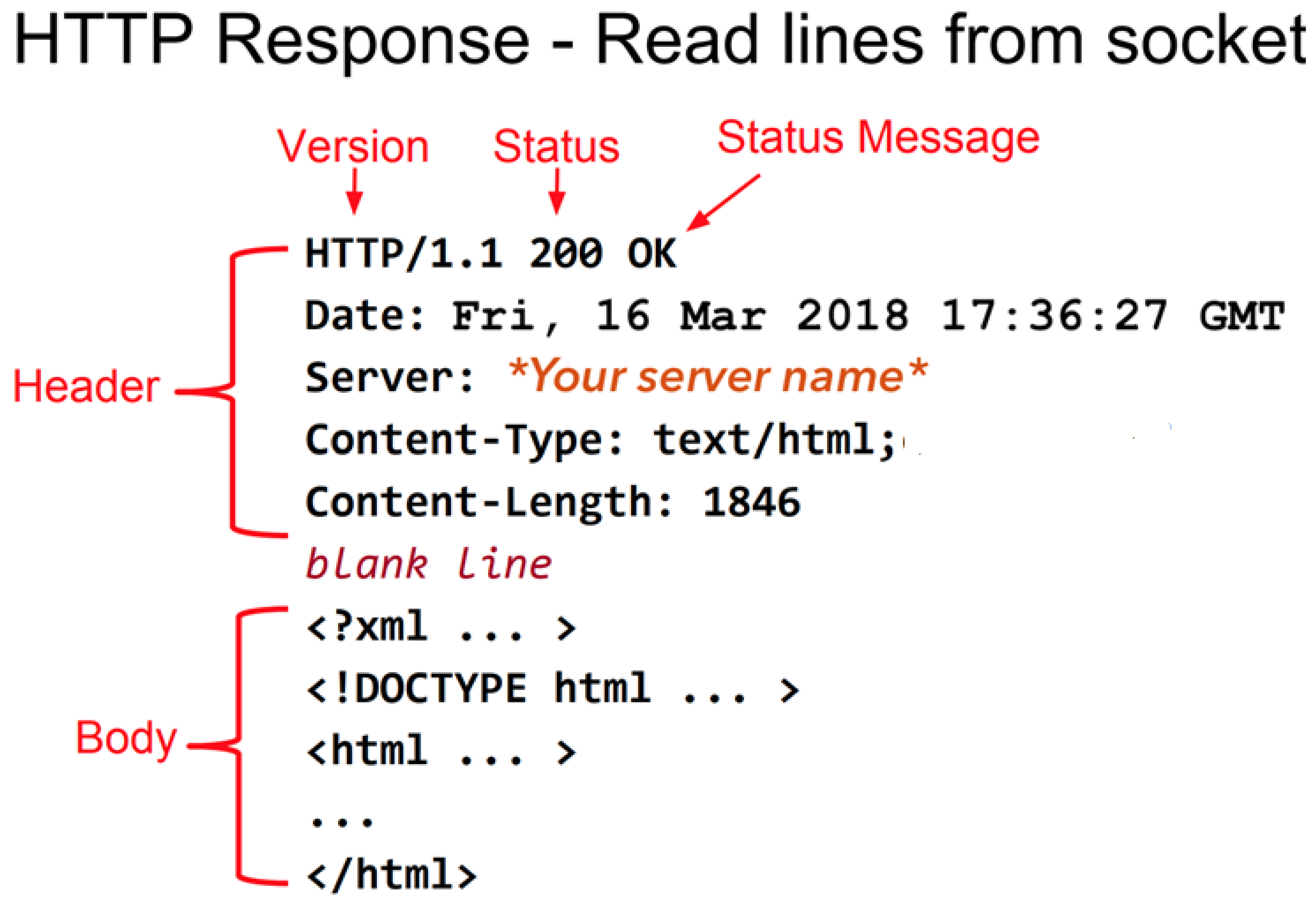
Request Body Encoding

- These are written as content types



HTTP Response

- We cant be present at the sever to check what has happened
- 1XX
- 2XX
- 3XX
- 4XX
- 5XX

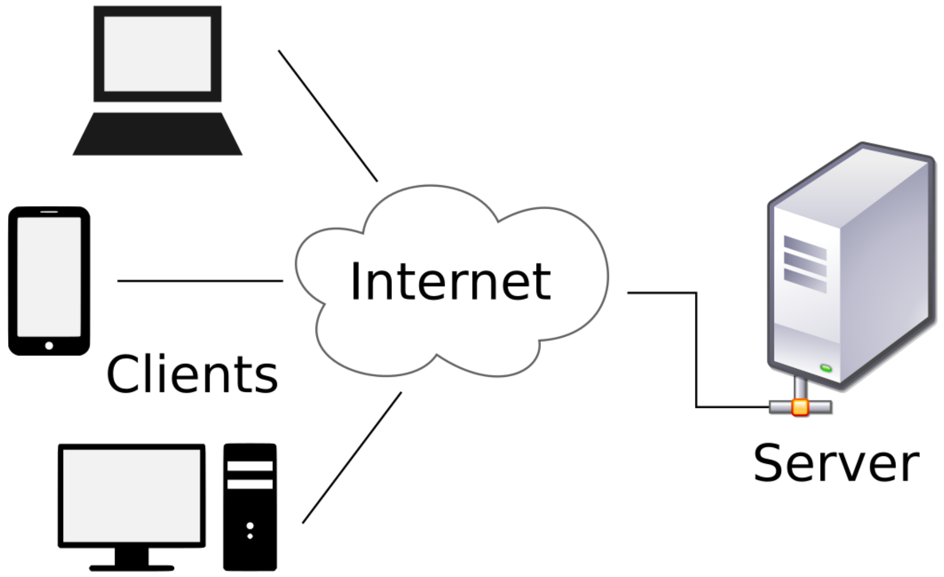


```
HTTP/1.1 200 OK
Date: Fri, 16 Mar 2018 17:36:27 GMT
Server: *Your server name*
Content-Type: text/html;
Content-Length: 1846
blank line
<?xml ... >
<!DOCTYPE html ... >
<html ... >
...
</html>
```

Response Codes

- 1XX- informal continuing process
- 2XX- successful
  - 200 means the client can assume that the server has successfully handled the request
- 3XX-redirection
  - Resend the request as the requested resource may have been moved
- 4XX- client error
  - Requested resource not found
  - Problem in request formatting
- 5XX-server error
  - The response body may contain the detailing of the error
- Depending on the response code and the MIME type, the body of the response is processed

HTTP is client driven

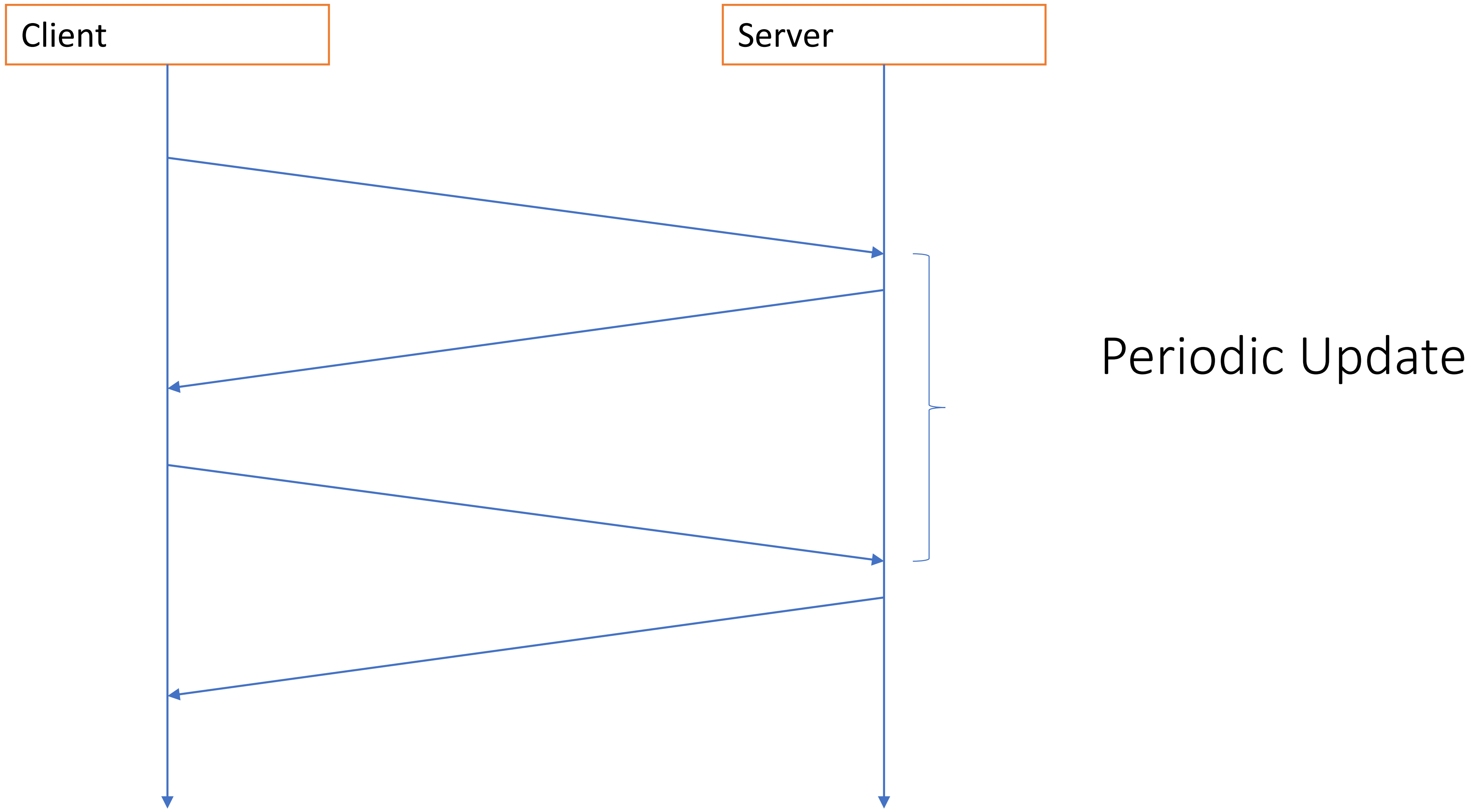


- If client 2 updates important data that client 1 just pulled in, unless client 1 makes a second request for an update the server cannot notify client 1

- Server is not able to push data as and when needed

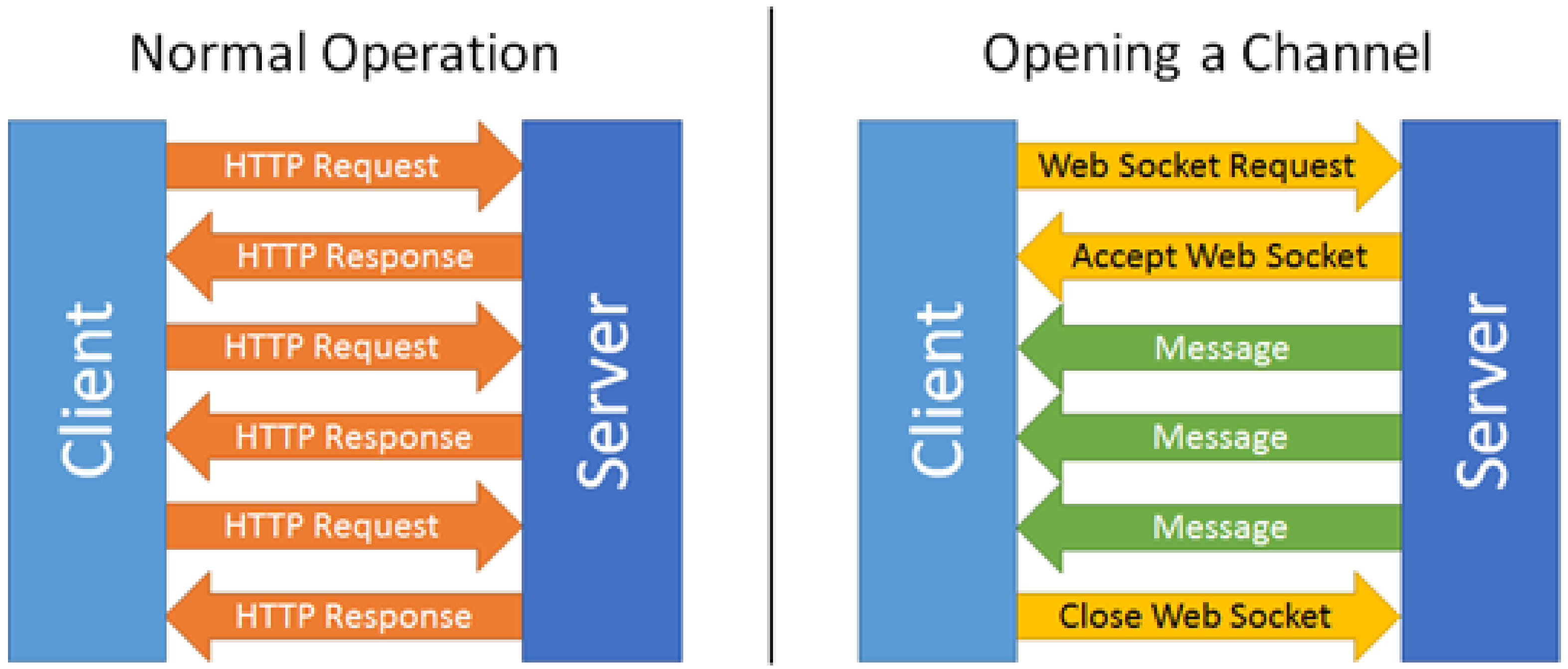
Pushing Data

- Manual- User may click on “refresh” when (s)he has decided to pull the data
- Every time the window is opened data is pulled from the server
- Periodic update
  - Overhead on the server processes as it needs to process the request even when no update is to be notified
  - Polling
  - Exponential backoff



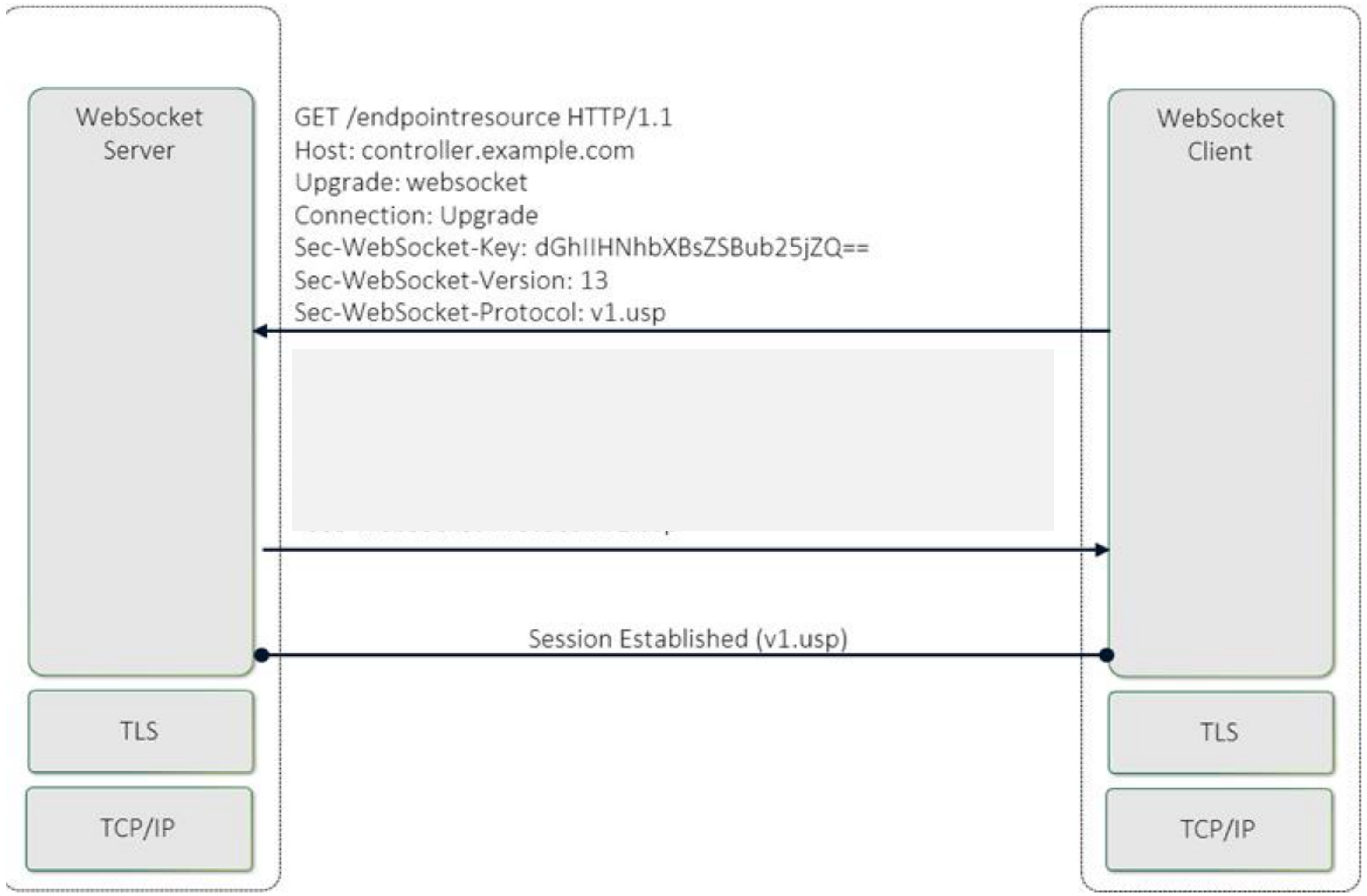
https://dzone.com/articles/thoughts-on-server-sent-events-http2-and-envoy-1

WebSockets



- communication protocol which features bi-directional, full-duplex communication over a persistent TCP connection
- Any party can push data anytime
- Single TCP connection for full duplex traffic
- Message transfer on websockets does not require all parts of HTTP to be sent (header, URL, content type, body etc.)
- Simply send binary messages or some other format back and forth in a server
- By default, port 80 is used
- Port 443 is used for connection tunneled over the TLS

Web Socket HTTP compatibility



GET /endpointresource HTTP/1.1
Host: controller.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
Sec-WebSocket-Protocol: v1.usp

Web Socket Advantages

- Stateful connection
- Message overhead of polling is more than web socket
- STOMP- Simple Text Oriented Messaging Protocol

https://spring.io/guides/gs/messaging-stomp-websocket/

S. El Mimouni and M. Bouhdadi, "Formal modeling of the Simple Text Oriented Messaging Protocol using Event-B method," 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), 2015, pp. 1-4, doi: 10.1109/AICCSA.2015.7507170.

Web Socket Problems

- Web sockets enable a server to push data only if the client is connected
- Web sockets are difficult to synchronize with more clients
- Keeping an open connection can have substantial resource impact
- For shared hosting servers, web socket is not a scalable option
- http responses can be cached by browser or by proxies
    - There is no such built-in mechanism for requests sent via webSockets