Name(s)____ Period ______ Date ____________

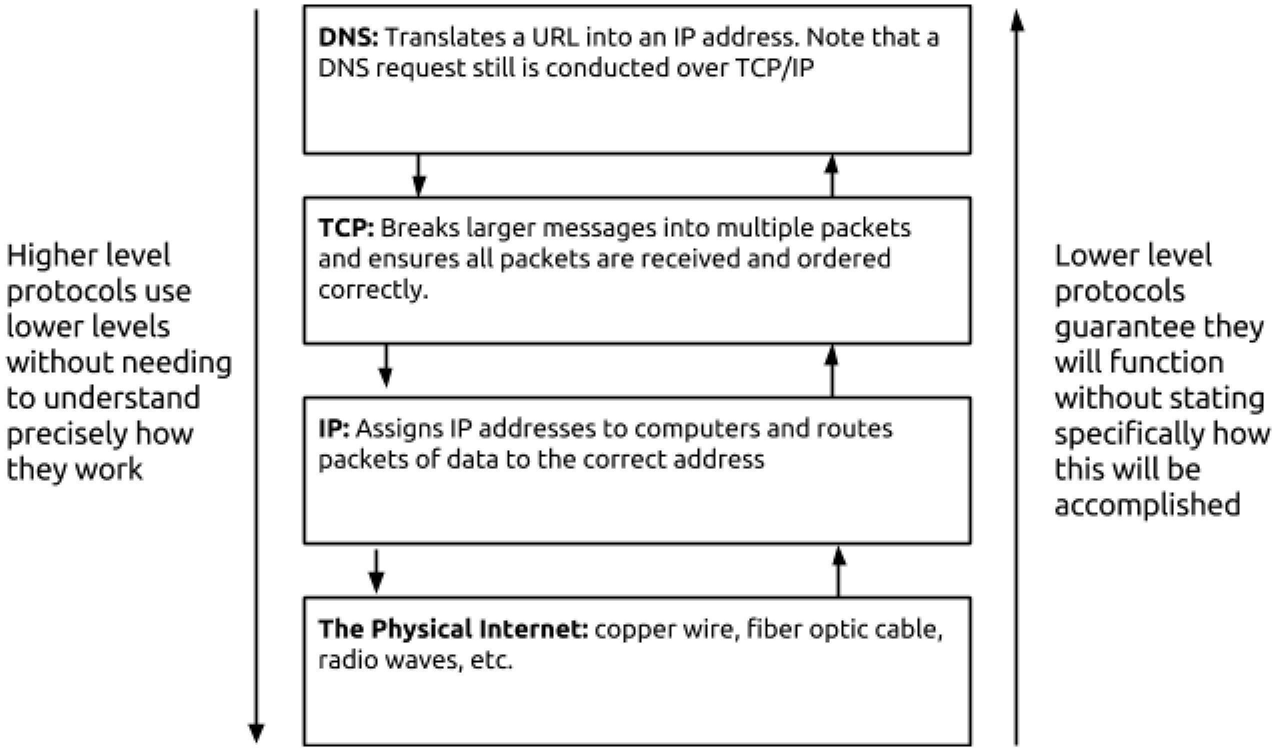
Resource - HTTP and Abstraction on the Internet

Vocabulary

- **Abstraction:** Reducing information and detail to focus on essential characteristics. It is typically possible to look at a system at many levels of abstraction, depending on how much detail is necessary to approach the challenge at hand.
- **Server:** A computer that awaits and responds to requests for data
  Example: a DNS server awaits and responds to requests for URLs to be translated to IP addresses.
- **Client:** A computer that requests data stored on a server
  Example: When you type an address into your browser, your computer is the client and it sends the request to the DNS server.

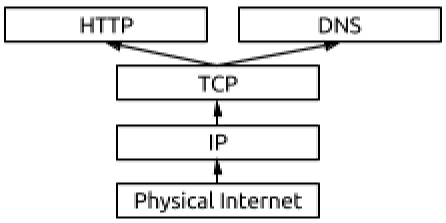
Abstraction on the Internet

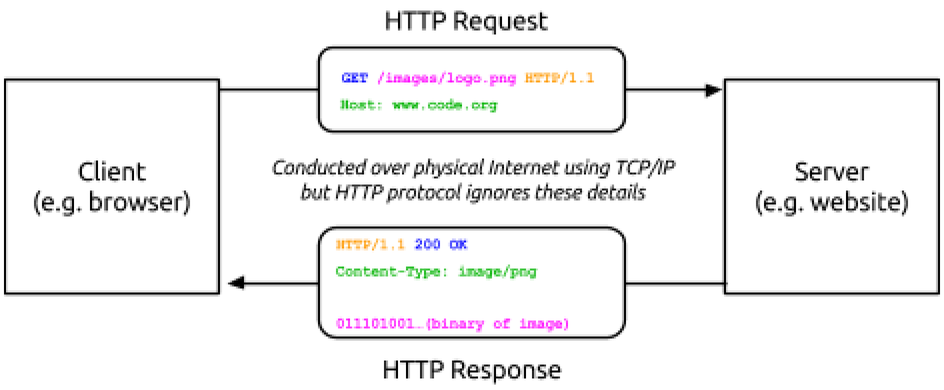
Abstraction plays a key role in the relationship between the layers of the Internet, as higher levels make use of the functionality provided by lower levels, without worrying about how they function.


Higher level protocols use lower levels without needing to understand precisely how they work
DNS: Translates a URL into an IP address. Note that a DNS request still is conducted over TCP/IP
TCP: Breaks larger messages into multiple packets and ensures all packets are received and ordered correctly.
IP: Assigns IP addresses to computers and routes packets of data to the correct address
The Physical Internet: copper wire, fiber optic cable, radio waves, etc.
Lower level protocols guarantee they will function without stating specifically how this will be accomplished

HTTP (HyperText Transfer Protocol)

HTTP is a high level protocol, that defines how users of the Internet (clients) request and receive data like web pages, images, video, audio, and files from the servers containing them. A client will send a request to the server with an identifier for a desired piece of data, and the server will attempt to respond to the request, typically by returning the information requested.




HTTP Request
Client (e.g. browser)
Conducted over physical Internet using TCP/IP but HTTP protocol ignores these details
Server (e.g. website)
HTTP Response

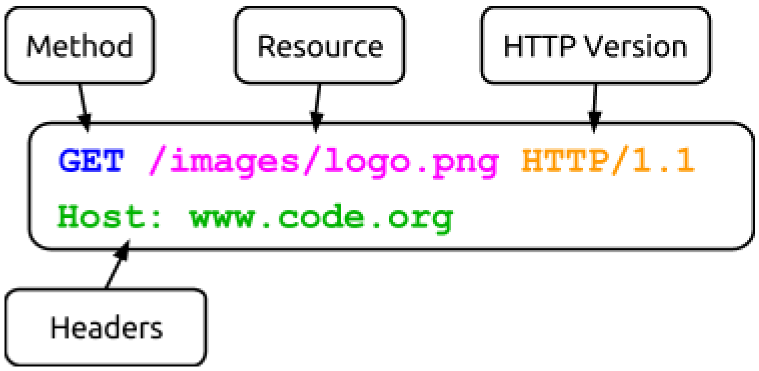
```
GET /images/logo.png HTTP/1.1
Host: www.code.org
```

```
HTTP/1.1 200 OK
Content-Type: image/png

011101001…(binary of image)
```

HTTP Requests

When you type a URL in your browser, your computer (the client) needs to “ask” the server that is storing the data and images for the web page to return its contents so your browser can display it. To do so, your computer will send an ASCII-text message called an HTTP request. Here’s what a simple HTTP request for the data of an image might look like:



```
GET /images/logo.png HTTP/1.1
Host: www.code.org
```

- **Method:** An HTTP request will begin with a method, which indicates what the client wants the server to do. The two most common methods are:

Method	Description
`GET`	Requests a specified web page or other data
`POST`	Submits some data for the server to accept or process

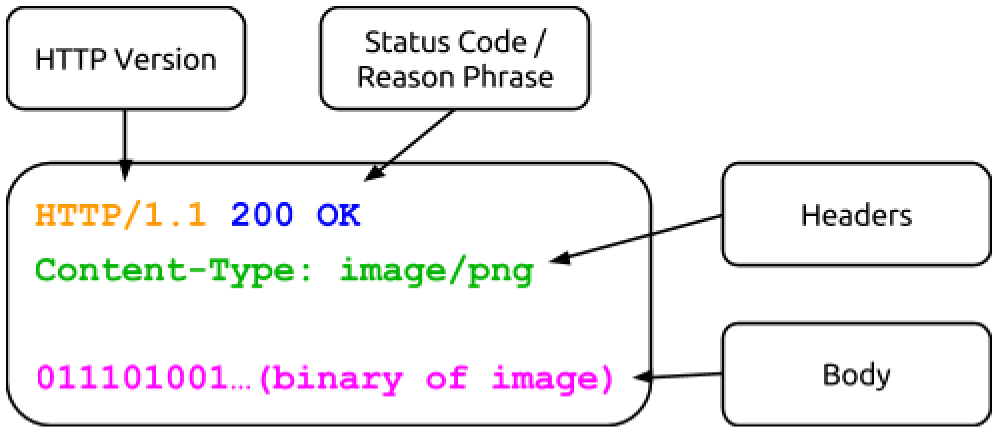
- **Resource:** The name of the resource you wish to access. In the example above, the request is for a .png image called “logo” located in the folder “images.”

- **HTTP Version**: An indication of the version of HTTP being used; in this example it is HTTP 1.1.

- **Headers:** Additional information included to help the server interpret the request. In the above example, “Host” is included, but many more can be added by placing them on additional lines.

HTTP Response

When a server receives an HTTP request it will respond with a message of its own. Once again, the response will be sent entirely in ASCII-text and must be correctly formatted. Here's a sample HTTP Response:



An HTTP Response has multiple components:

- **HTTP Version:** An indication of the version of HTTP being used; in this example it is HTTP 1.1.

- **Status Code and Reason Phrase:** A number and phrase indicating how the server responded to the request. Some common Status Codes / Reason Phrases are:

Status Code / Reason Phrase	Meaning
200 Ok	Request was handled successfully
404 Not Found	Server could not find anything matching request
301 Moved Permanently	Requested data was permanently moved
302 Moved Temporarily	Requested data was temporarily moved
500 Internal Server Error	Error by the server prevented fulfilling request

- **Headers:** Additional information about the response. The example above includes the header "Content-Type," indicating the type of content included in the response.

- **Body:** Following the headers, the response may include some data that was requested. In this example the actual binary representation of the image would be included in the response, so this response could be quite long.

HTTP as Abstraction

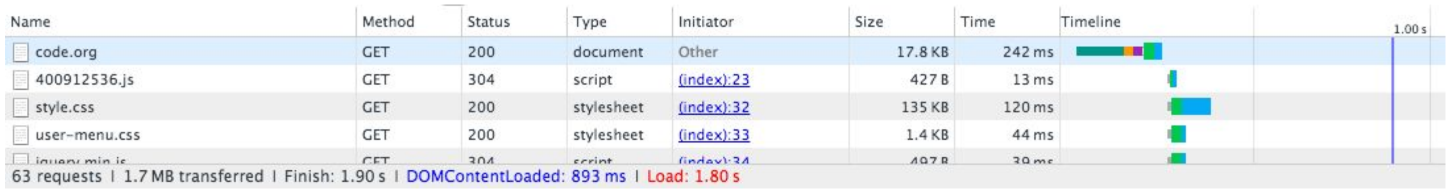
HTTP specifies how requests and responses should be formatted but does not mention how this text will actually be sent. You could conduct an HTTP exchange with a friend by writing hand-written letters and mailing them to one another, so long as your letters were formatted according to HTTP. In the real world, HTTP exchanges are conducted over TCP/IP, but this is not the concern of HTTP. It simply requires that some system be in place for these messages to be sent back and forth. In this way, HTTP uses lower-level systems abstractly, relying on their functionality without concern for the details of how that functionality is achieved.

Name(s)____ Period ______ Date ____________

Worksheet - HTTP in Action

Using the Developer Tools

Most modern browsers include a set of Developer Tools designed to provide a detailed look into your browser's activity. It is possible to monitor the traffic of HTTP requests and responses associated with a web page in real time.



Name	Method	Status	Type	Initiator	Size	Time	Timeline 1.00 s
code.org	GET	200	document	Other	17.8 KB	242 ms	
400912536.js	GET	304	script	(index):23	427 B	13 ms	
style.css	GET	200	stylesheet	(index):32	135 KB	120 ms	
user-menu.css	GET	200	stylesheet	(index):33	1.4 KB	44 ms	
jquery.min.js	GET	304	script	(index):34	497 B	39 ms	

63 requests | 1.7 MB transferred | Finish: 1.90 s | DOMContentLoaded: 893 ms | Load: 1.80 s

Use the links below to help you navigate to the Developer Tools of your browser. In Chrome, Internet Explorer and Firefox you'll need to **open the "Network" tab.**
Chrome: https://developer.chrome.com/devtools
Internet Explorer: https://msdn.microsoft.com/library/bg182326(v=vs.85)
Firefox: https://developer.mozilla.org/en-US/docs/Tools/Network_Monitor
Safari: https://developer.apple.com/safari/tools/ (look at the "Network Requests" in the "Timelines" tab.)

Seeing HTTP in Action

You will use your browser's developer tools to discover what kind of HTTP traffic is associated with visiting different types of websites. You and your partner are going to look at least 5 different types of websites:

1. http://example.com -- a very simple web page. Use this first to investigate developer tools.
2. **A "static" website** like: Wikipedia
3. **A news website** like: ESPN.com, BuzzFeed, the New York Times, etc.
4. **A streaming site** like: YouTube, or Spotify
5. **A site that accepts user input** like: twitter, facebook, email, google docs.

For each type of website below, follow these steps:

1. Monitor the HTTP traffic generated by loading the page.
2. Once the page has loaded, poke around with the other information the developer tools let you see about the data coming in. What about the protocols can you see?
3. Interact with the website by clicking links or using other functionality on the site, noting how this affects the HTTP traffic.
4. Observe other things like:
   - Total amount of data received
   - Number of HTTP requests actually generated by loading one page
   - Total time to load the page.
   - Types of data received through HTTP (it's more than just HTML)

Make notes about any questions you have or things that are interesting and worth sharing with other groups.