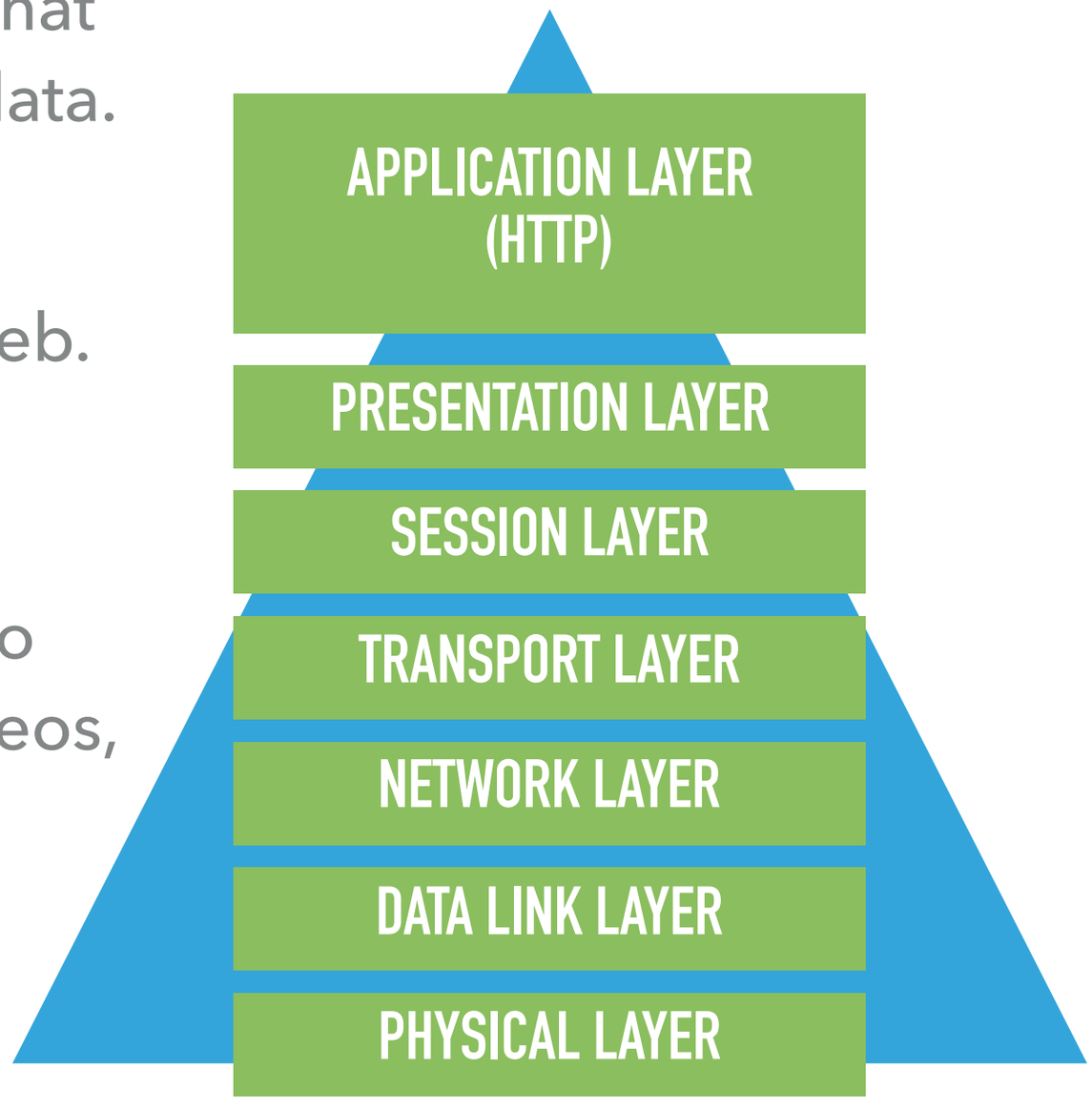
JRS CODING SCHOOL

HTTP

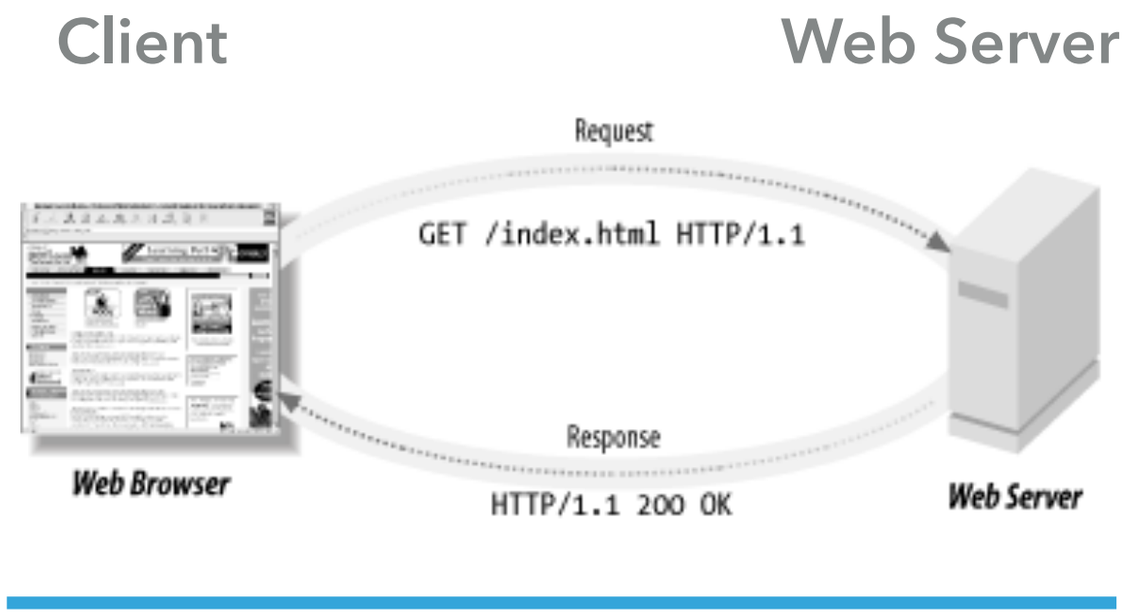
WHAT IS HTTP?

- HTTP = **H**ypertext **T**ransfer **P**rotocol
- HTTP is an application layer protocol that allows web based apps to exchange data.
- HTTP is the foundation of data communication for the World Wide Web.
- It's based on the TCP/IP protocol.
- Two computers can use this protocol to communicate. Send data, images, videos, text, documents.



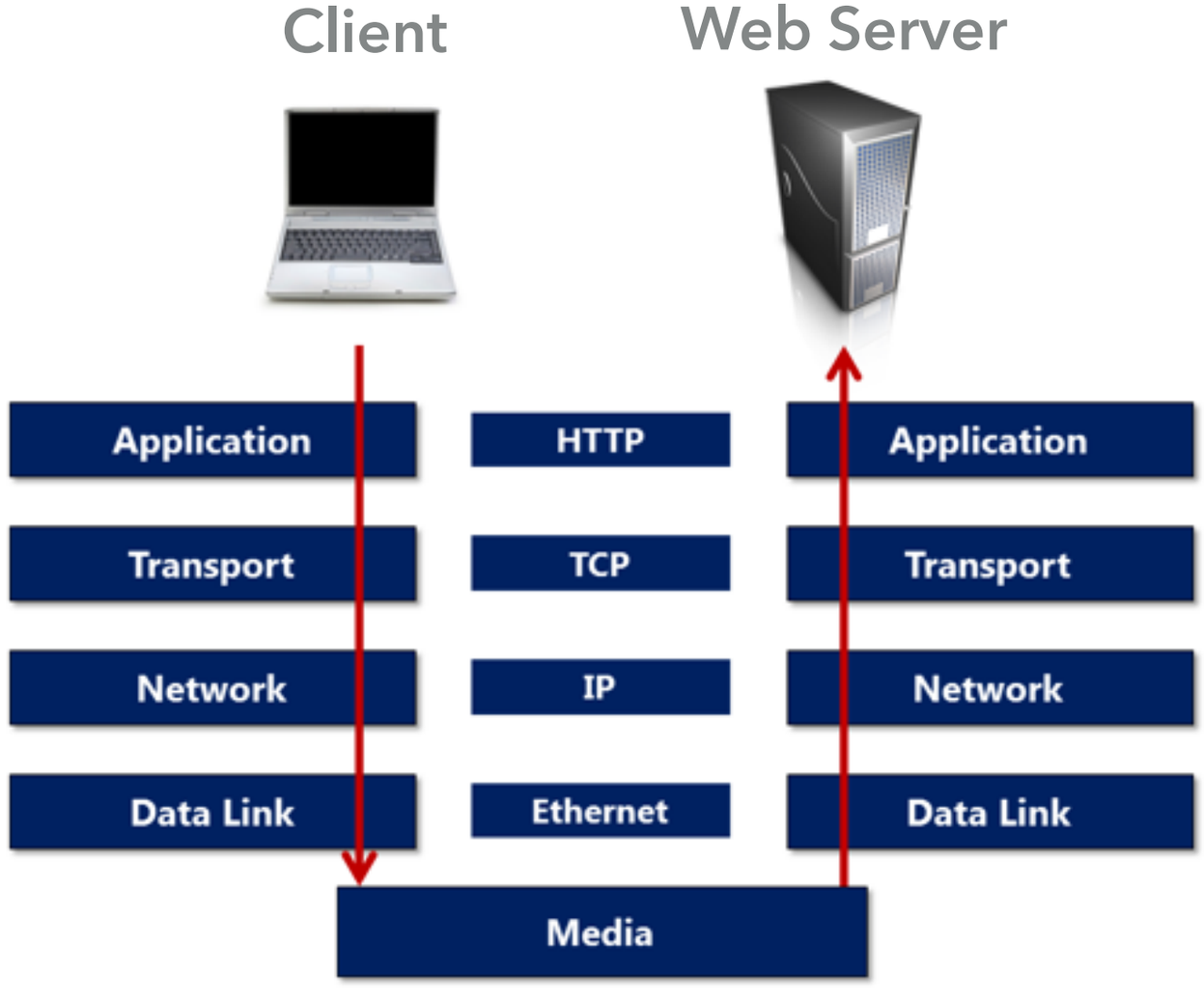
PROTOCOL LAYERS

- An HTTP exchange takes place over a TCP/IP socket. The client (your web browser, for example) opens a socket, connects to the HTTP server via the port number the server is listening to, and issues a command. The command is routed to the server via the internet. The server receives the command and does something with it.



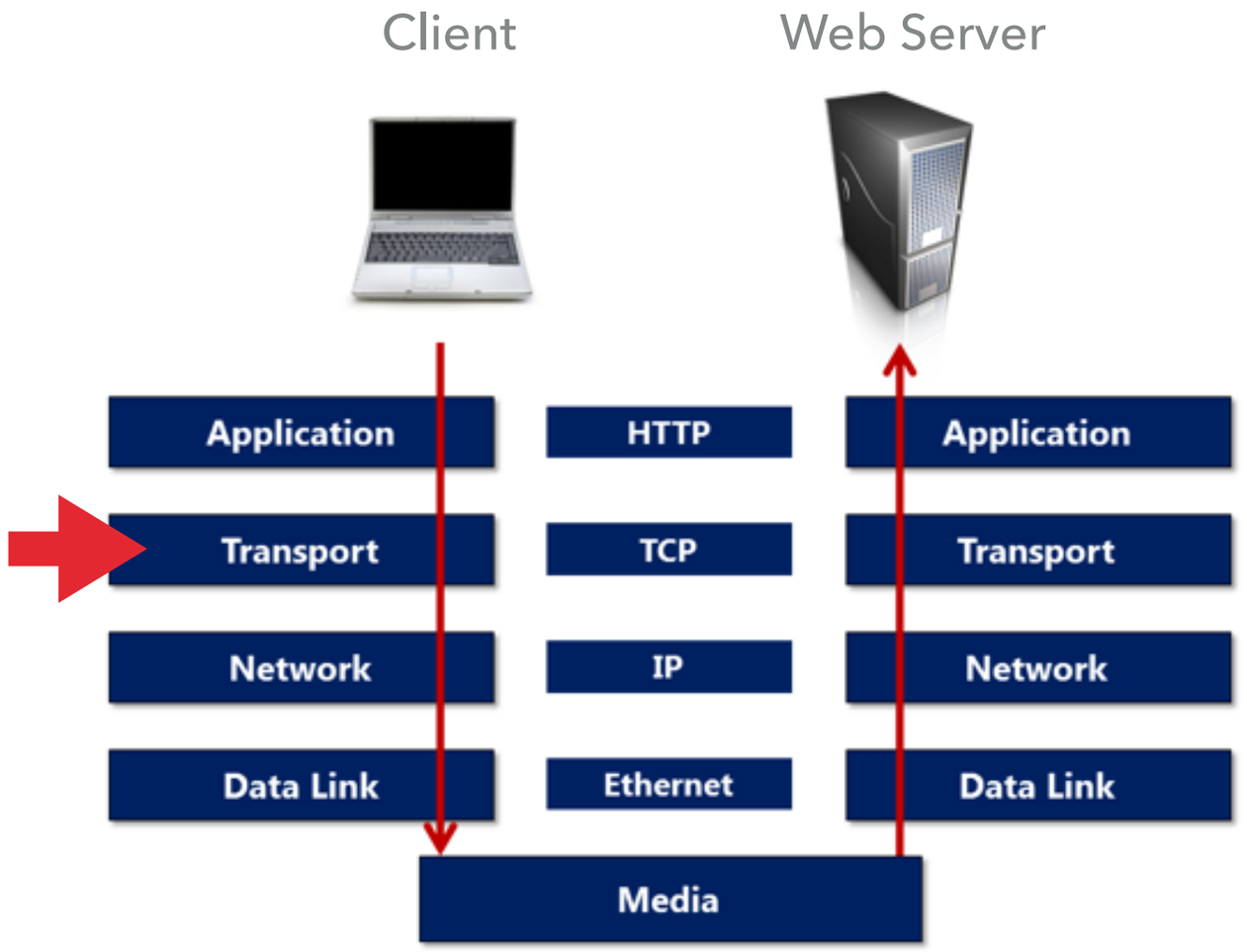
- A message from a web browser has to travel down a series of layers, and when it arrives at the web server it travels up through a series of layers to reach the web service process.

- The protocol layers below HTTP define how the messages actually cross the network and reach the server.



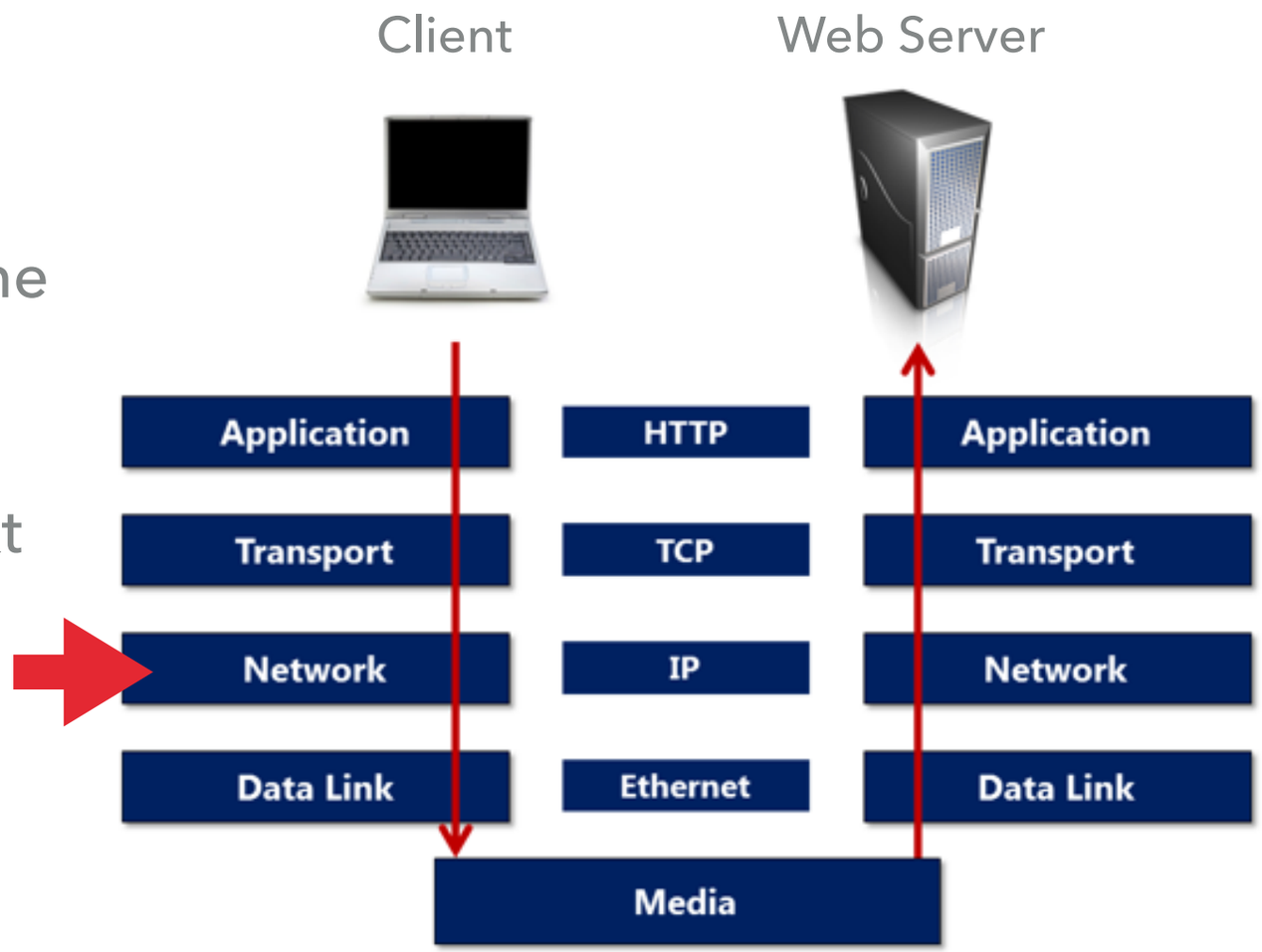
PROTOCOL LAYERS: TRANSPORT LAYER (TCP)

- TCP (Transmission Control Protocol).
- When a user types a URL into the browser, the browser opens a TCP socket by specifying the web server address and port number, then starts writing data into the socket.
- The TCP layer accepts the data and ensures the data gets delivered to the server without getting lost or duplicated. You don't have to worry about it!
- At this point, we are still on the client machine



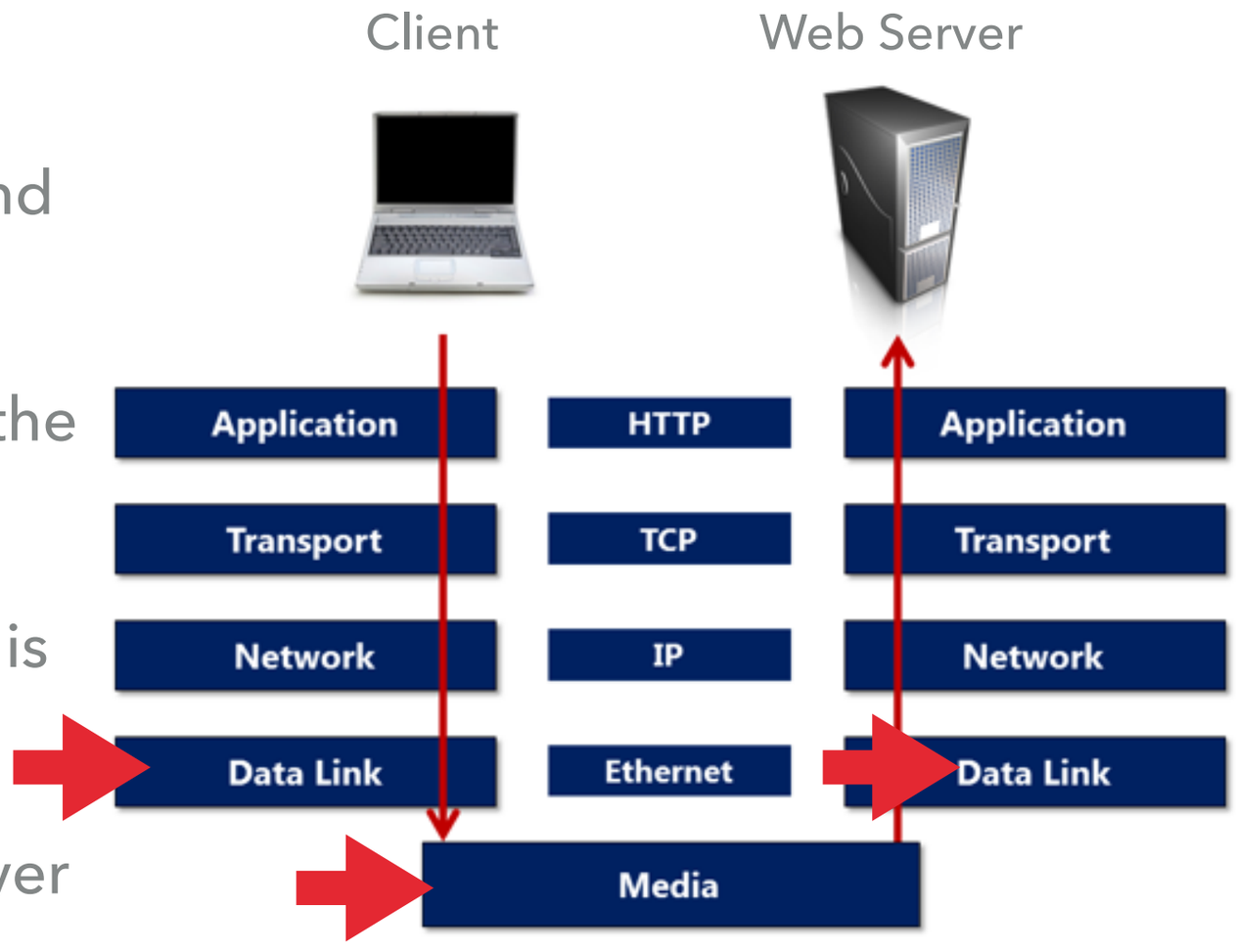
PROTOCOL LAYERS: NETWORK LAYER (IP)

- After TCP at the transport layer comes the internet protocol (IP) as a network layer protocol.
- IP is responsible for taking pieces of information and moving them through the various switches, routers, gateways, repeaters, and other devices that move information from one network to the next and all around the world.
- To deliver data IP requires computers to have an address (the famous IP address. Ex: 208.192.32.40).
- At this point, we're still inside the client computer. Eventually these IP packets have to travel over a piece of wire, a fiber optic cable, a wireless network, or a satellite link.



PROTOCOL LAYERS: DATA LINK LAYER

- Eventually these IP packets have to travel over a piece of wire, a fiber optic cable, a wireless network, or a satellite link.
- Ethernet packets consist of 1's and 0's and other electrical signals.
- Signals travel over the media and enter the web server via a network card.
- Once we hit the web server, the process is reversed.
- Once on the web server, the data link layer delivers packets to the IP layer, which hands over data to TCP, which can reassemble the data into the original HTTP message sent by the client and push it into the web server process which eventually results in a response back to the client.



A CONVERSATION BETWEEN TWO COMPUTERS USING THE HTTP PROTOCOL

REQUEST AND RESPONSE

REQUEST RESPONSE CYCLE

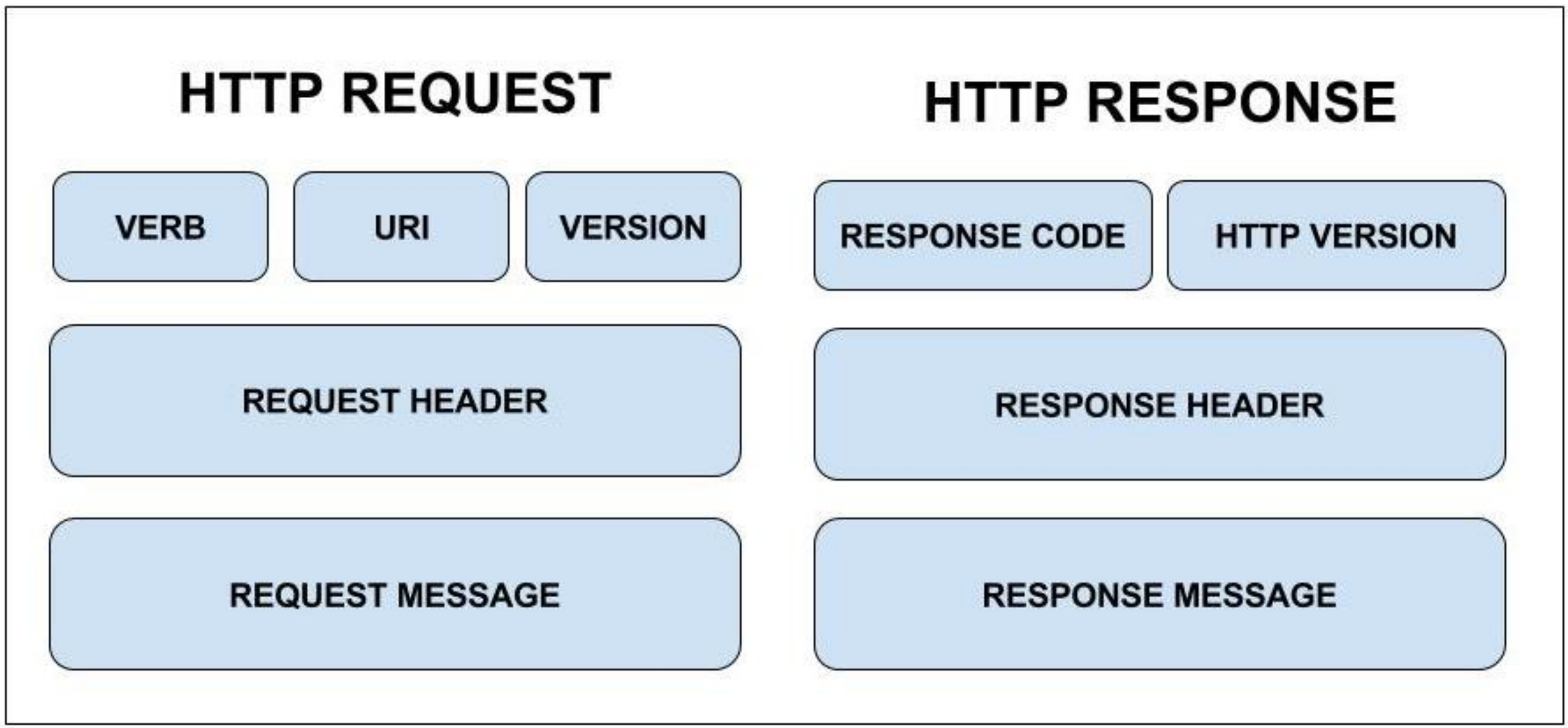
- The conversation between two computers takes the form of a request and a response.

- The user types in a url in the browser or clicks a link on a web page.
- Your applications will make a request to a web server (programmatically), too!
- The client establishes a connection to a web server and sends a request to a web server consists of an HTTP message.
- Protocols “underneath” HTTP help to establish the connection and reliably transmit the messages back and forth.
- The web server on the other side processes the request, prepares a response, established a connection to the client, sends back an HTTP response with a response code and a message, and then disconnects.

REQUEST AND RESPONSE CYCLE IS CONNECTIONLESS

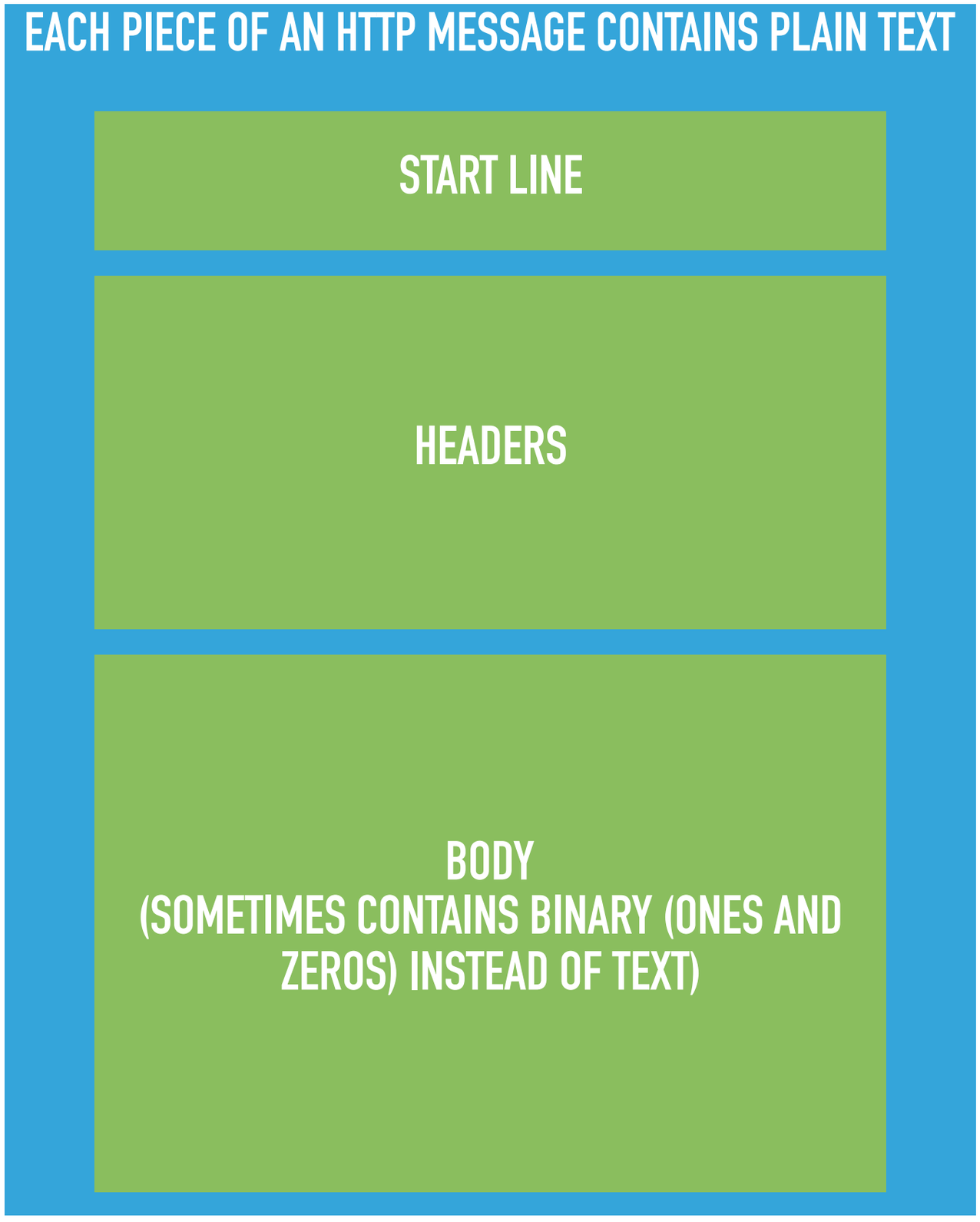
- HTTP is connectionless.
- A client makes a request to a server.
- The client disconnects after making the request.
- The client waits for a response.
- The server processes the request.
- The server re-establishes the connection with the client.
- The server sends a response back.

USER REQUESTS AN HTML "HOME" PAGE



HTTP MESSAGES

HTTP MESSAGE BREAKDOWN



HTTP MESSAGE BREAKDOWN: "GET" REQUEST MESSAGE

- The info within each section varies depending on whether the message is a **request** or a **response**.

START LINE

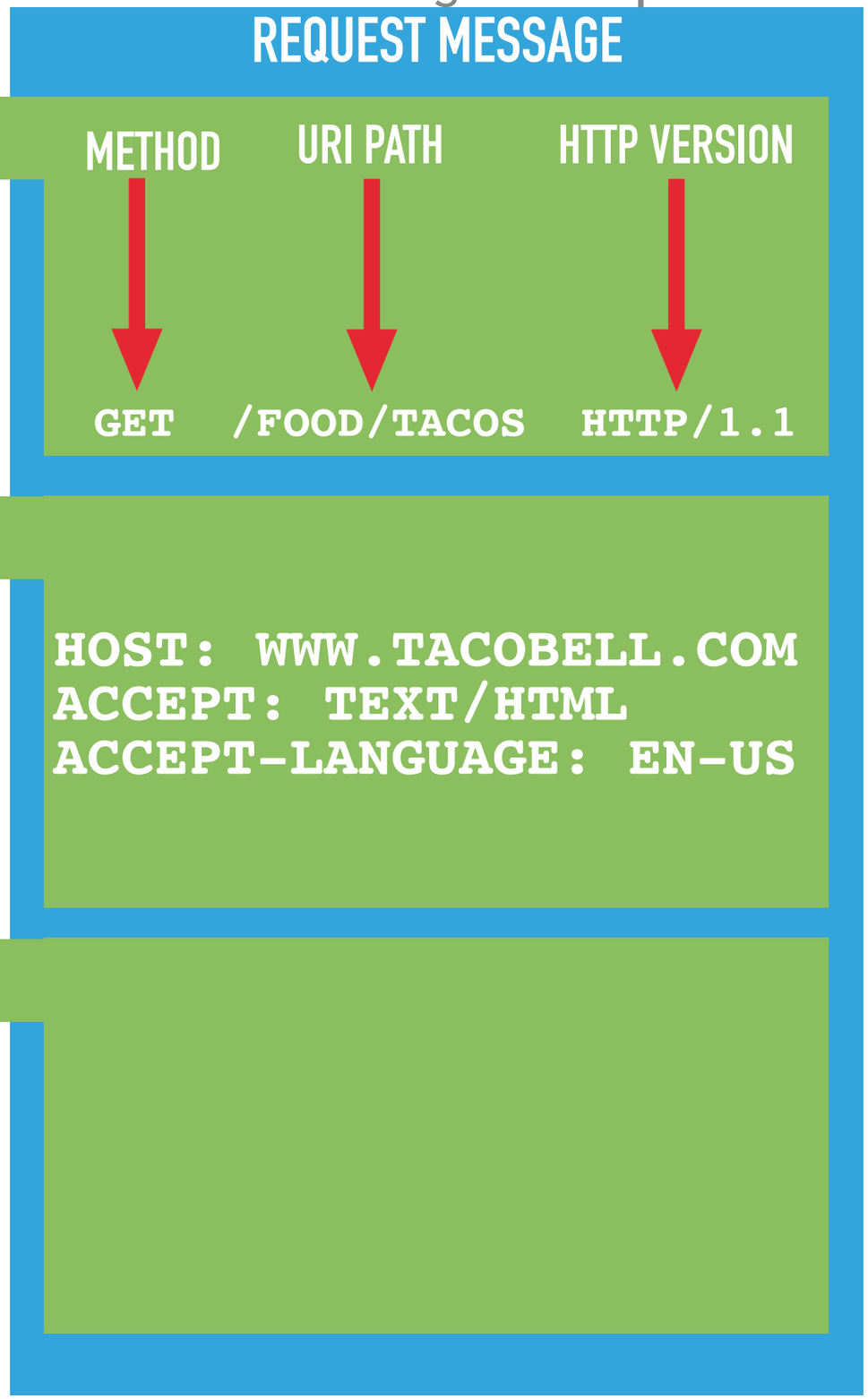
- Method tells the server what to do
  - POST: Create the resource
  - GET: Give me the resource so I can display it
  - PUT: Replace the resource
  - DELETE: Remove the resource

HEADERS

- HOST: The name of server where we send the request
- ACCEPT: What type of information we are requesting
- ACCEPT: The accept value is a MIME type.

BODY

- GET or DELETE: body is not required
- POST or PUT: body is required



HTTP MESSAGE BREAKDOWN: RESPONSE MESSAGE

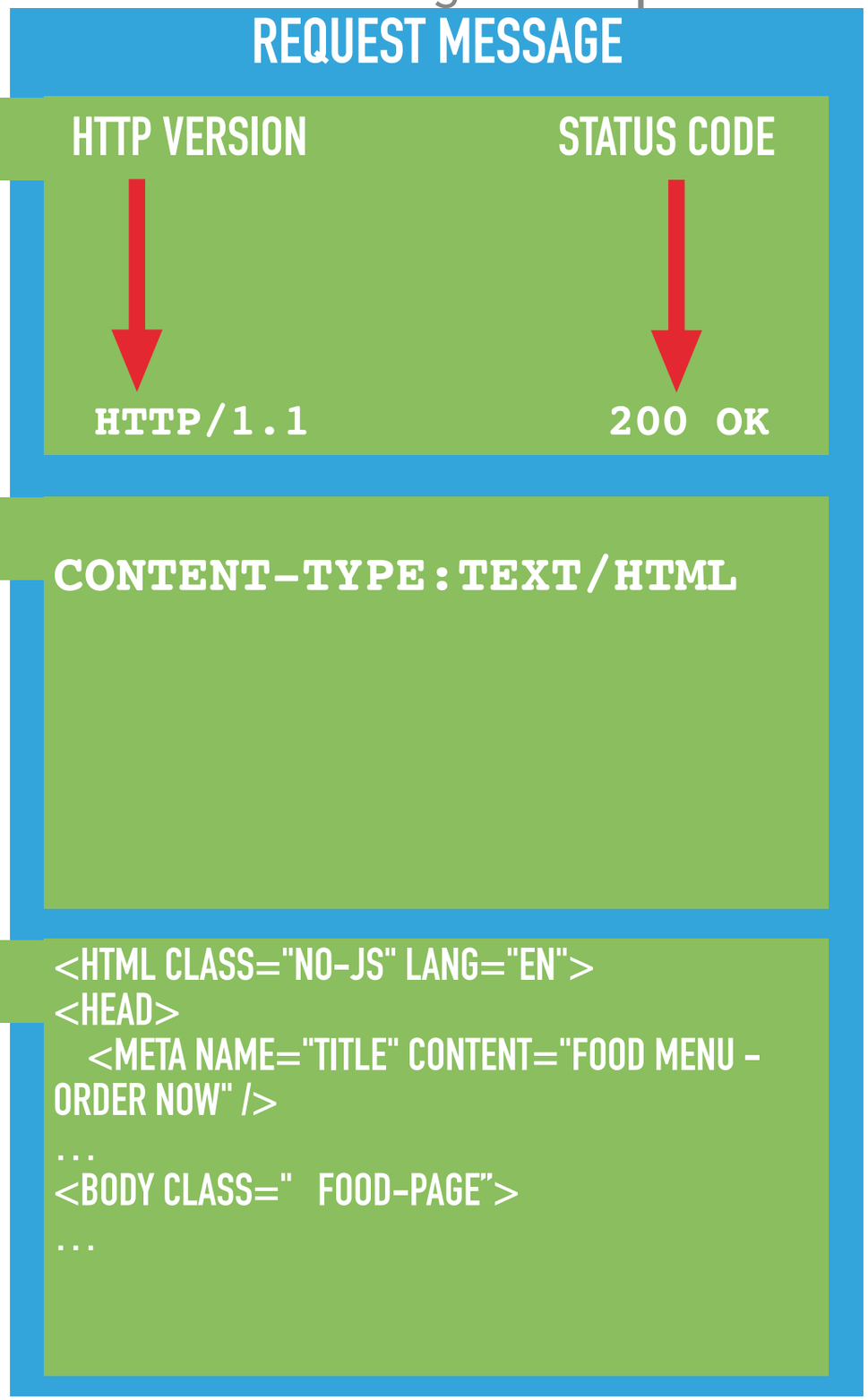
- The info within each section varies depending on whether the message is a **request** or a **response**.

START LINE

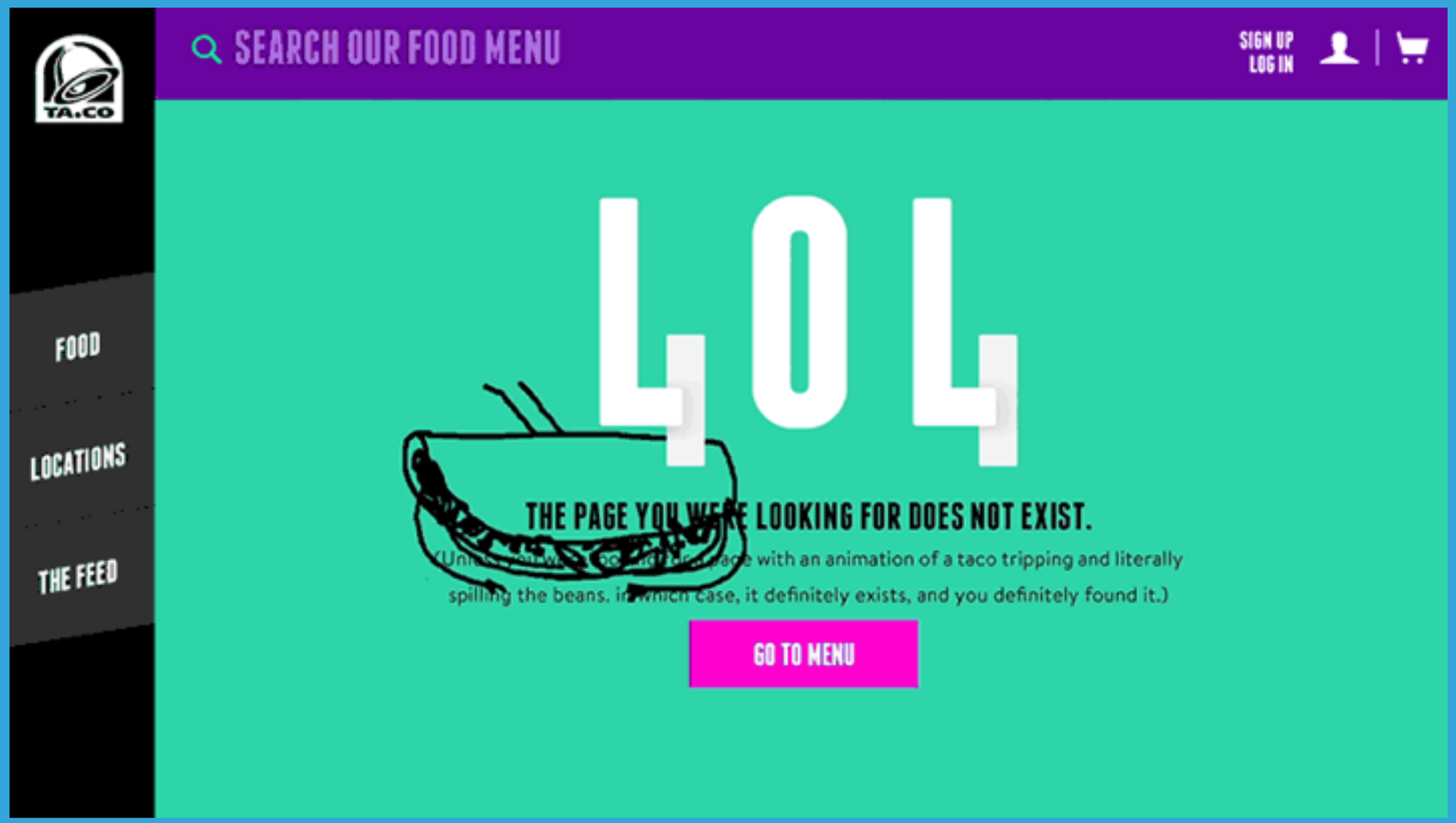
- Common status codes:
  - 200: OK (SUCCESS)
  - 201: CREATED
  - 404: NOT FOUND

HEADERS

BODY



HTTP RESOURCES



HTTP RESOURCES . . . THE BREAKDOWN

- Open a web browser and https://www.tacobell.com/
- The browser understands this and makes an HTTP request to a web server named www.tacobell.com.
- https://www.tacobell.com/ is a *resource locator*. It's known as a URL or **U**niform **R**esource **L**ocator. It represents a unique *resource* on the web.
- A resource is something you desire from a web server. An image, web page, video, or data representation of something in your application.
- There are billions of resources available to you on the internet. Each one has a unique URL.
  - https://www.tacobell.com/food/tacos
  - https://www.tacobell.com/food/tacos/fiery-doritos-locos-tacos-supreme
  - https://www.tacobell.com/food/tacos/crunchy-taco-supreme
  - https://www.tacobell.com/food/burritos
  - https://www.tacobell.com/food/burritos/beefy-potato-rito
  - https://www.tacobell.com/food/burritos/combo-burrito

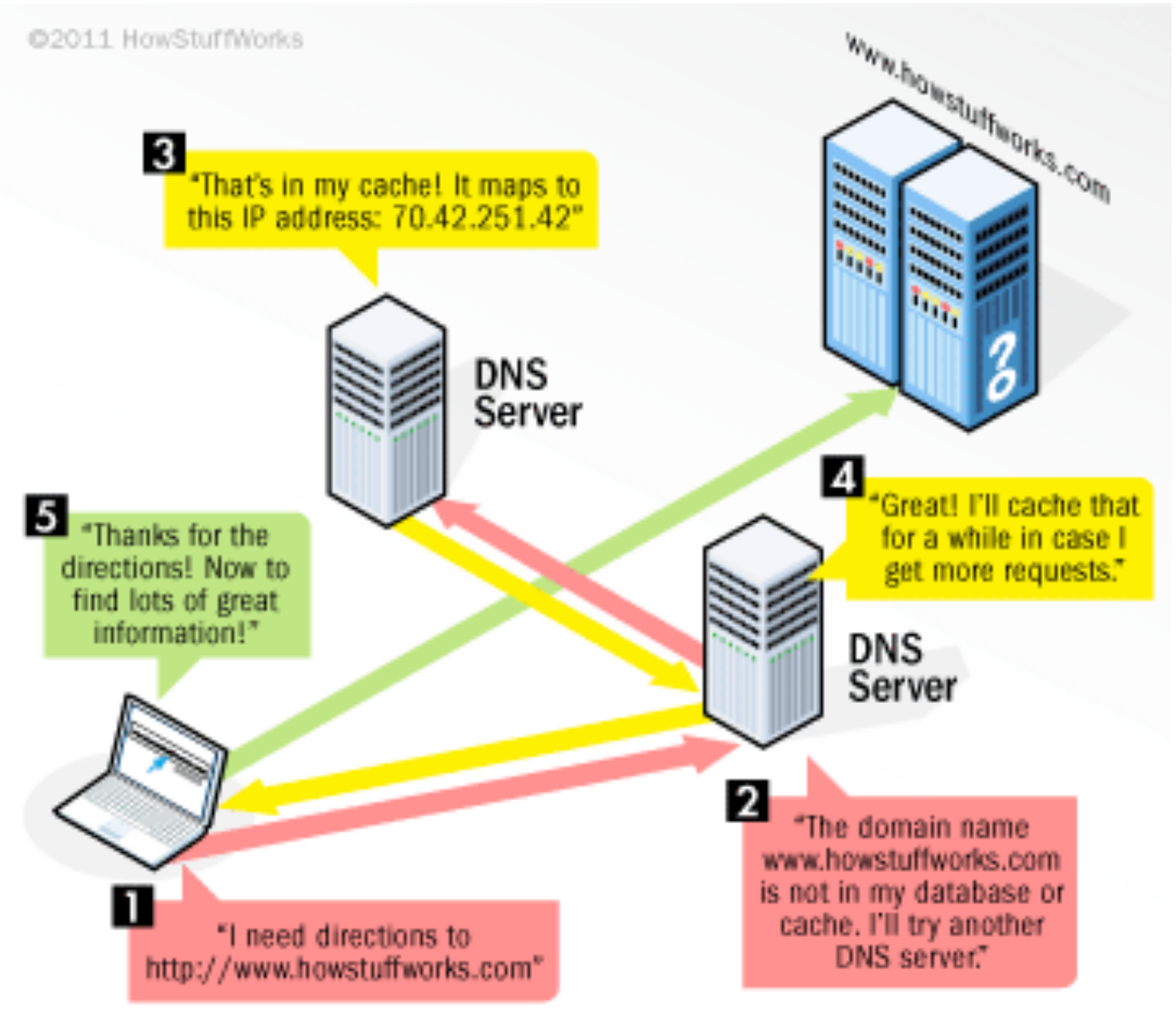
HTTP RESOURCES: URLS HAVE 3 PARTS – 1) URL SCHEME

- https://www.tacobell.com/

- URL Scheme: the part before the // describes the protocol to access a particular resource
  - Examples:
    - http: - tells the browser to use the http protocol
    - https: - tells the browser to use the secure http protocol
    - ftp: - tells the browser to use the file transfer protocol
    - mailto: tells the browser this is an email address

HTTP RESOURCES: URLS HAVE 3 PARTS – 2) HOST NAME

- https://www.tacobell.com/food/tacos
- Every thing after the scheme is particular to that protocol
- *www.tacobell.com* is the host name. This tells the browser which computer on the internet is hosting the resource.
- Domain Name System (DNS) translates www.tacobell.com into a network address.
- DNS Servers exist all across the globe.
- To deliver data the internet protocol requires computers to have an address (the famous IP address. Ex: 208.192.32.40).
- As an alternative, you can specify the host portion using an IP address instead of the friendly host name.

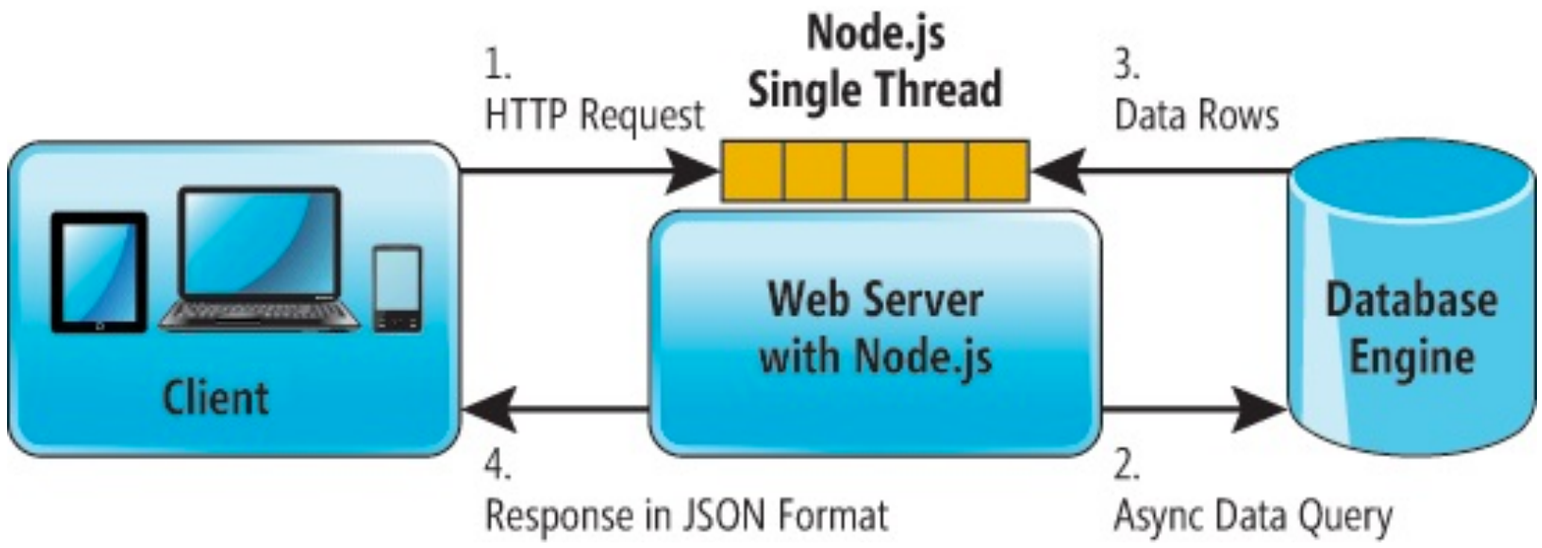


HTTP RESOURCES: URLS HAVE 3 PARTS – 3) PATH

- https://www.tacobell.com/**food/tacos/fiery-doritos-locos-tacos-supreme**
- **/food/tacos/fiery-doritos-locos-tacos-supreme** is the URL path. The www.tacobell.com host should recognize what specific resource (taco) is requested by this path and respond appropriately.
- The path refers to a resource, such as JSON data, image, html, video…
- More often than not the path refers to something dynamic.

- An application running on a web server will take the request and build a resources using content stored in a database.

NODE.JS WEB SERVER PROCESSES HTTP REQUESTS



A STATELESS PROTOCOL

HTTP doesn't remember anything about the previous request

HTTP PROTOCOL

- HTTP is a statless protocol.
- “Stateless" means the server doesn't remember any information about a query after it has been responded to.
- HTTP is amnesiac; if you send two commands to an HTTP server, the first command should not in any way affect the second command.

URLS AND API RESOURCE DESIGN: BEST DESIGN PRACTICE

- https://www.yourapi.com/animals/cats/breeds
- https://www.yourapi.com/animals/cats/breeds/siamese
- https://www.yourapi.com/animals/cats/breeds/tabby
- Keep URLs simple
- 2 base URLs per resource
  - The first URL is for a collection
    - https://www.yourapi.com/animals/cats/breeds
  - The second is for a specific element in the collection.
    - https://www.yourapi.com/animals/cats/breeds/siamese
  - For API resources, keep verbs out of your base URLs.