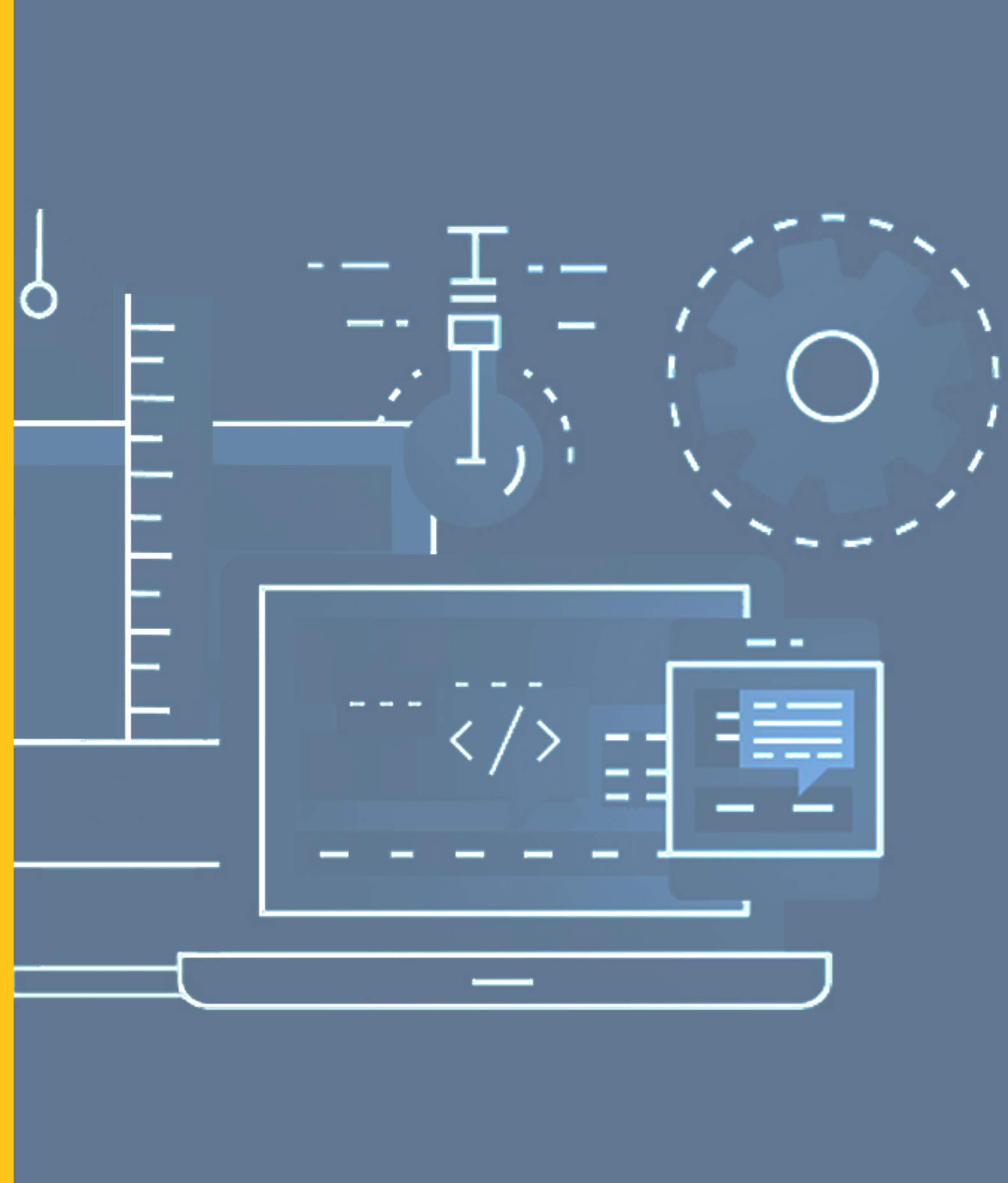


SHELLY GRAHAM, 07/27/2020

# WEB DEV 4

## SUMMER 2020

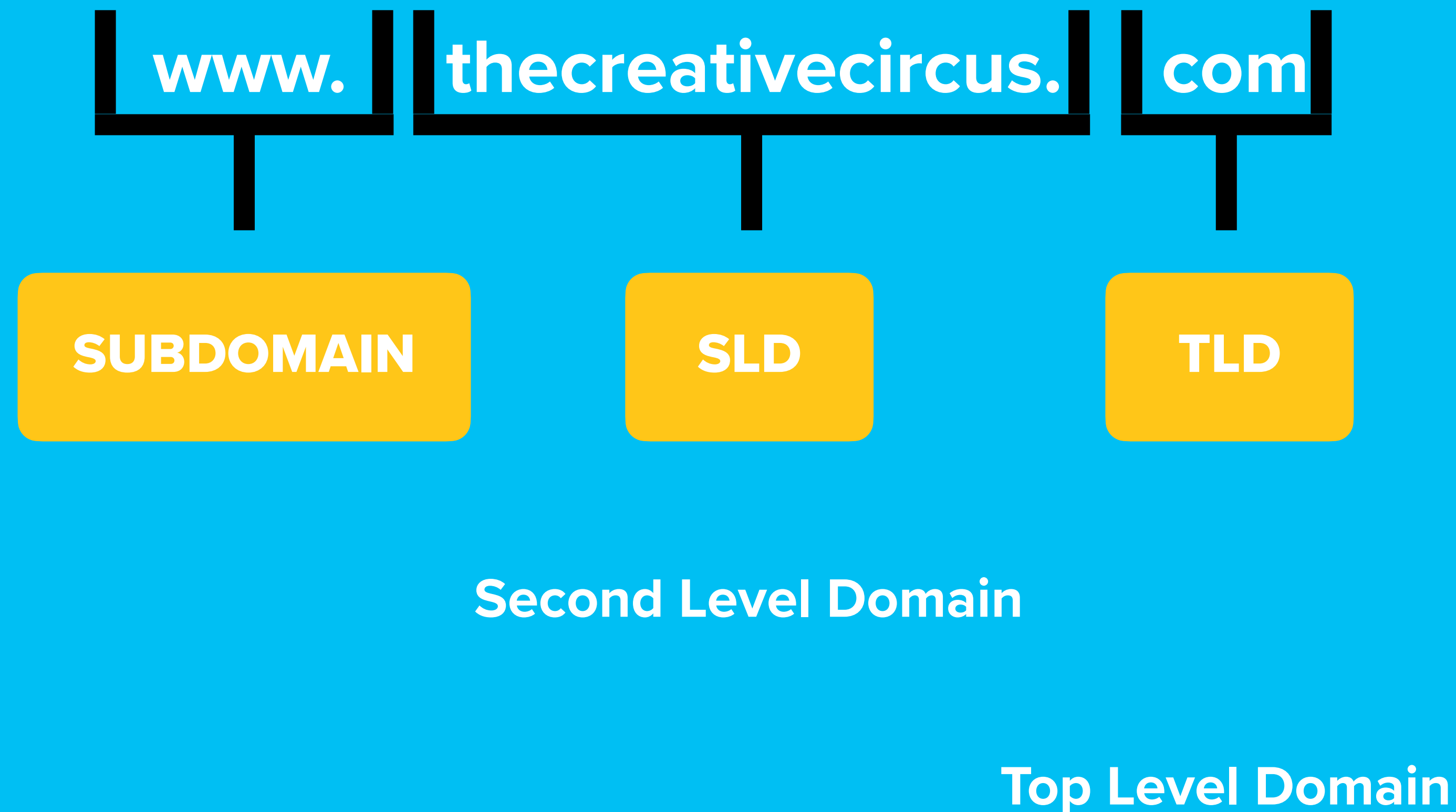
Week 5: Web Servers



# WEEK 5: WEB SERVERS

# SUBDOMAINS

- Extensions/subset of a registered domain name
- No need to buy a new domain name
- Use them to send visitors to a different/specific web address
- To divide content into different sections



# SUBDOMAINS EXAMPLES

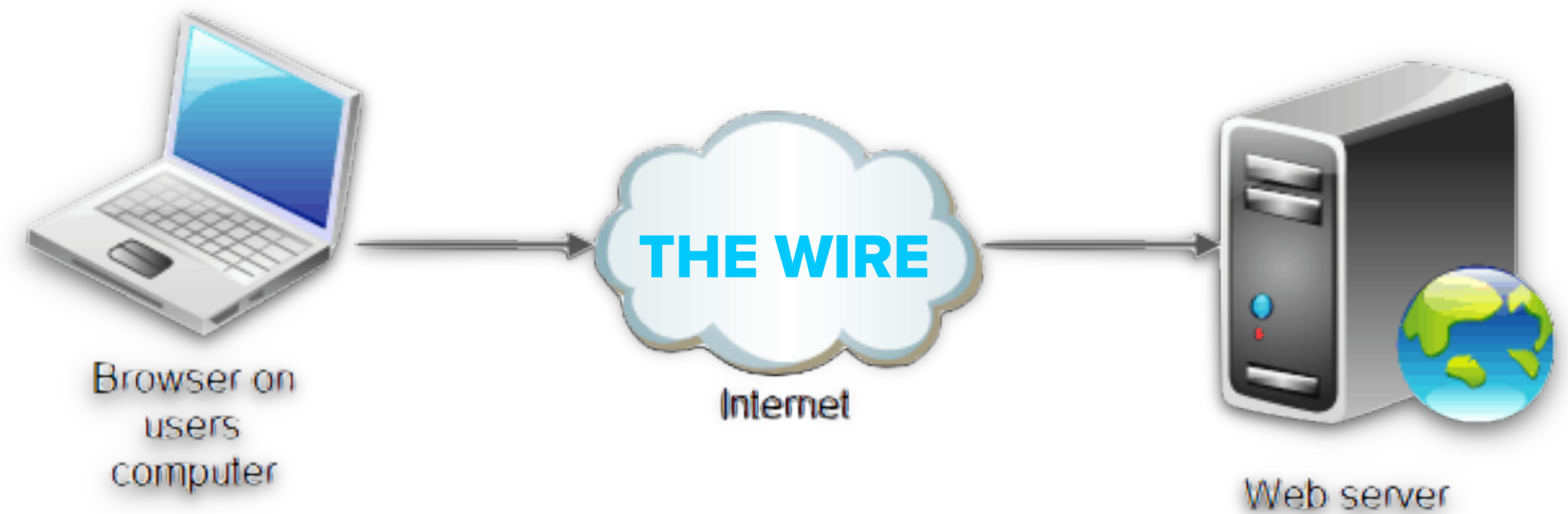
- Mobile version of website
- If global website, different web content for different countries
- Separate nice-to-have content from crucial content informant
- It indexes quickly, ideally benefits of good SLD standing with Google Index



# WEB SERVERS

# WHAT IS A WEB SERVER?

- Software that serves web content (websites, PDFs, Apps...)
- Uses the HTTP(s) protocol to send data
- Serves static and dynamic content
- Web 1.0 = static websites
- Web 2.0 = dynamic websites you can interact and therefore change
- Fun fact: JS files are static files!



# HOW DOES A WEB SERVER WORK?

- If the client requests a website, the server will process the request:
  1. Establish a connection between client and server through TCP = Transmission Control Protocol (standard that defines how to establish and maintain a network connection)
  2. Depending on request, server will locate data on server, update data before it's being displayed, log into database on server, etc.
  3. Respond with displaying the response result = website

# TYPES OF REQUESTS WE CAN MAKE TO SERVER

## HTTP Methods:

- **GET** = Receive unaltered data from server/database
- **POST** = Add new data to server/database
- **PUT** = Update existing data on server/database
- **DELETE** = Delete data from server/database

```
GET /api/customers
GET /api/customers/1
PUT /api/customers/1
DELETE /api/customers/1
POST /api/customers
```



# STRUCTURE OF HTTP REQUEST

- Start line: Method + request target (URL) —> GET/index.html
- Headers: see picture below
- Body: Usually empty in HTTP requests
- Detailed info: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

```
POST / HTTP/1.1
```

```
Host: localhost:8000
```

```
User-Agent: Mozilla/5.0 (Macintosh;... )... Firefox/51.0
```

```
Accept: text/html,application/xhtml+xml,...,*/*;q=0.8
```

```
Accept-Language: en-US,en;q=0.5
```

```
Accept-Encoding: gzip, deflate
```

```
Connection: keep-alive
```

```
Upgrade-Insecure-Requests: 1
```

```
Content-Type: multipart/form-data; boundary=-12656974
```

```
Content-Length: 345
```

Request headers

General headers

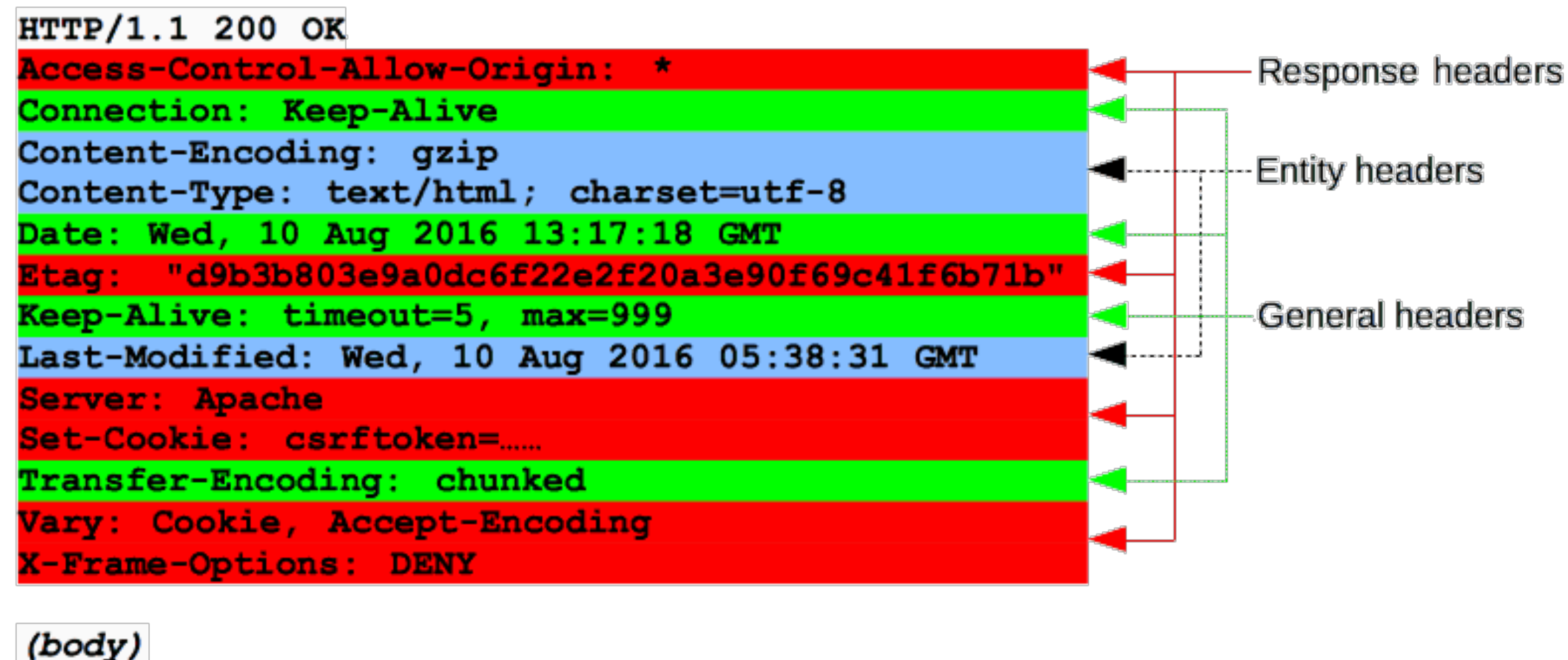
Entity headers

# SERVER RESPONSE

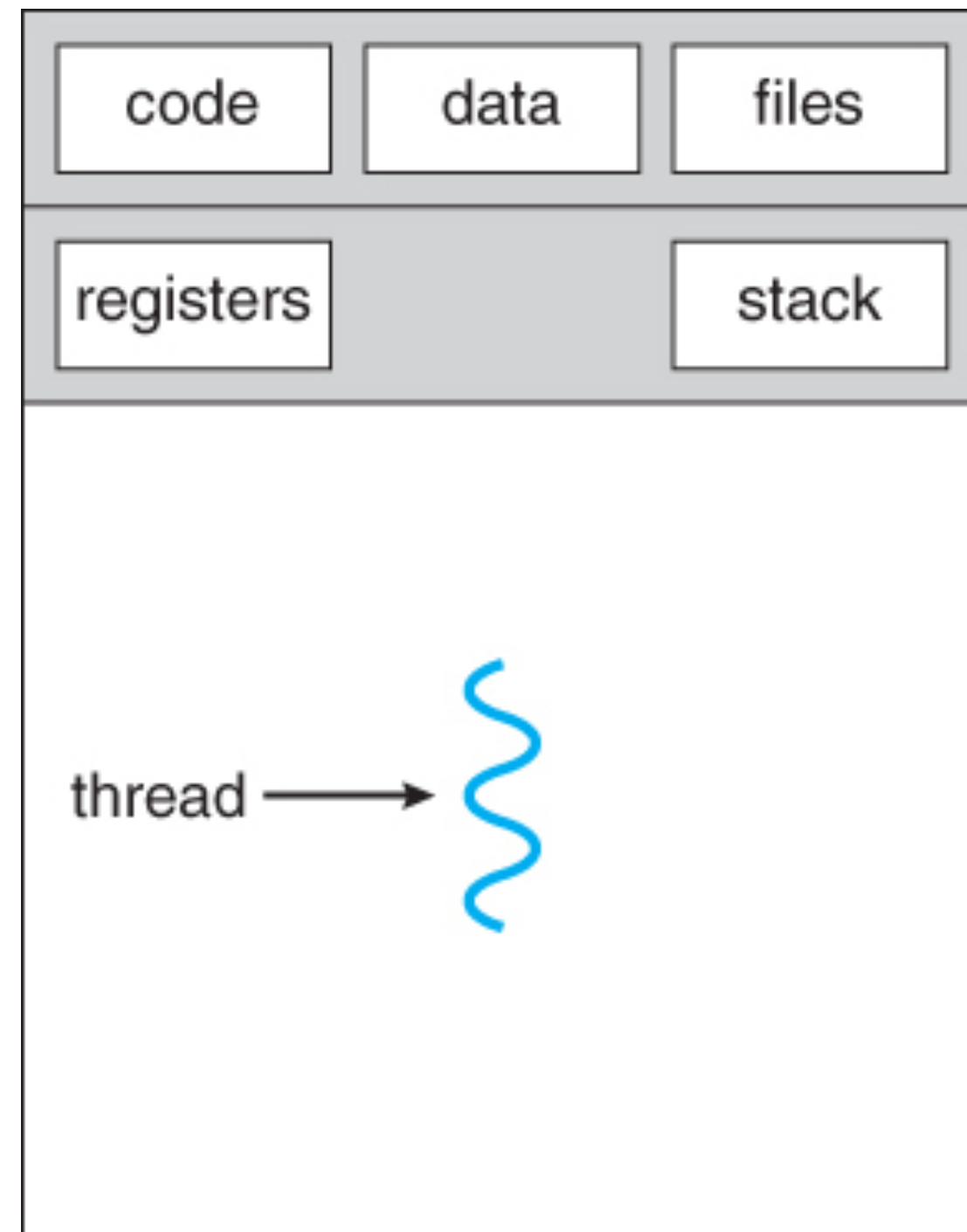
- Provides us with lots of information
- Status codes:
- 1xx = informational
- 2xx = success —> 200 = success
- 3xx = redirection
- 4xx = Client Error —> 400 = Bad request, 403 = forbidden, 404 = not found
- 5xx = Server Error —> 500 = internal server error
- Details: <https://www.restapitutorial.com/httpstatuscodes.html>
- Server also makes memory of the client who requested data = Socket

# STRUCTURE OF HTTP RESPONSE (FROM SERVER)

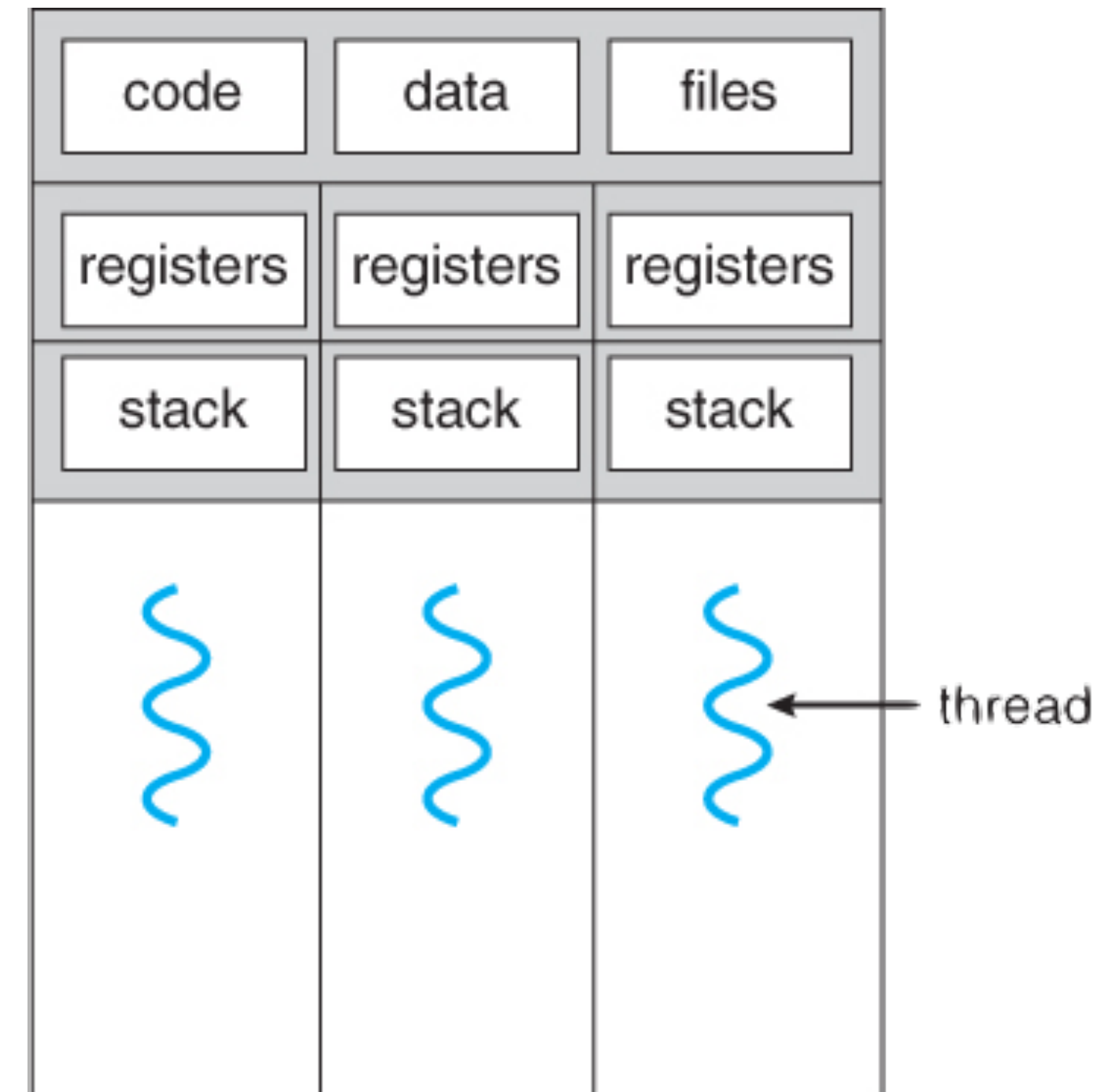
- Status line: Status code + Status text —> e.g. 404 - not found!
- Headers: see picture below
- Body: The content/data itself
- Detailed info: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>



# DIFFERENT TYPES OF SERVERS



single-threaded process



multithreaded process

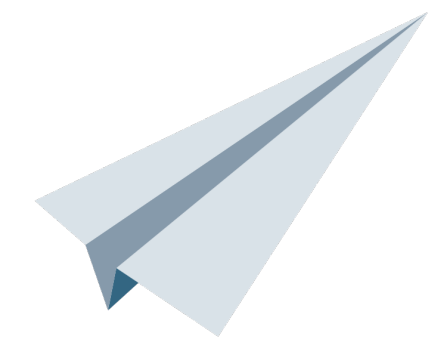
# SINGLE-THREAD SERVERS

## Blocking Single-thread server:

- Server can only serve 1 client at a time = synchronous
- Will make all other requests wait until initial request has been served
- Some servers create a new socket for every single client BUT takes of lots of memory on server
- Often uses PHP —> easy scripting language, good for beginners
- Old but stable technology: Great support but slow to adapt
- Many CMS (Wordpress, Drupal, Joomla) and Frameworks (Yii, Laravel, Code Ignitor, Cake PHP) use it



# Apache



# LIGHTTPD

fly light.

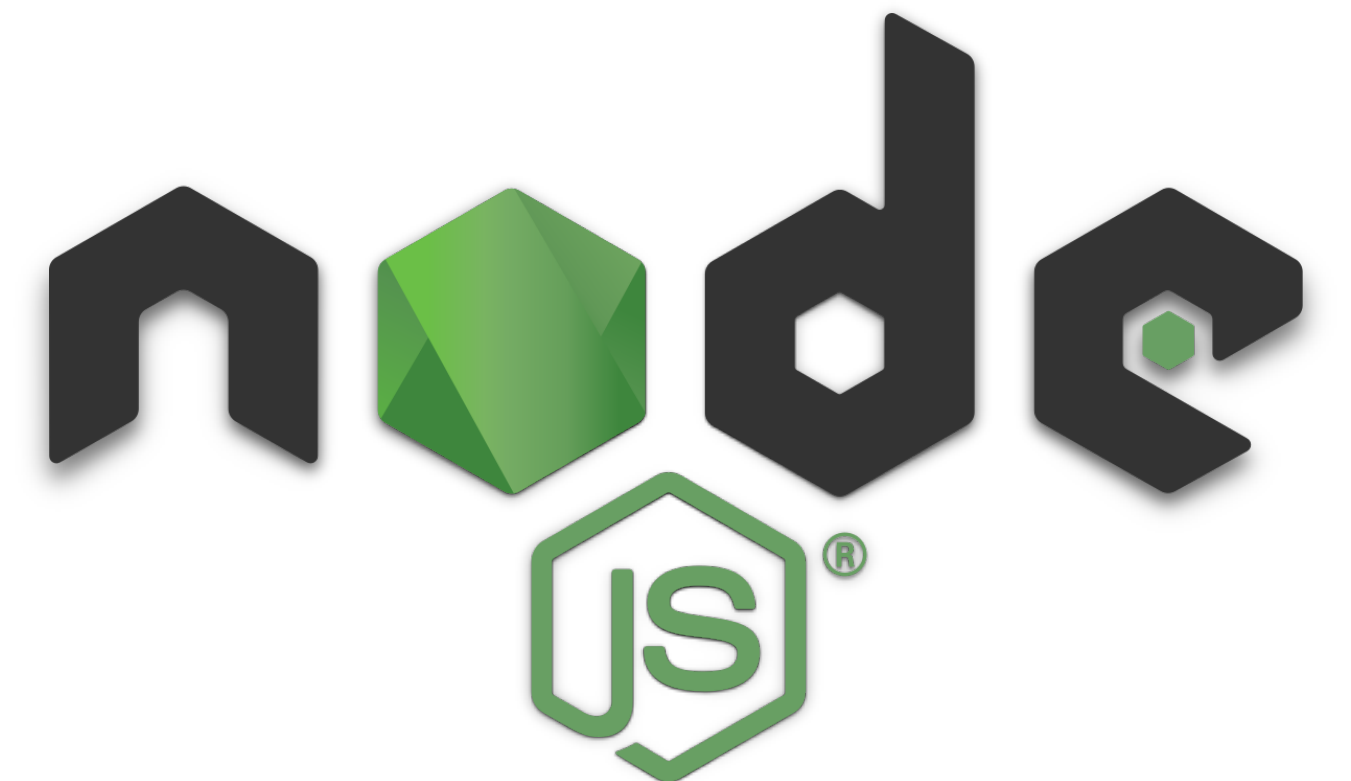




# MULTI-THREAD SERVERS

## Multi-thread server:

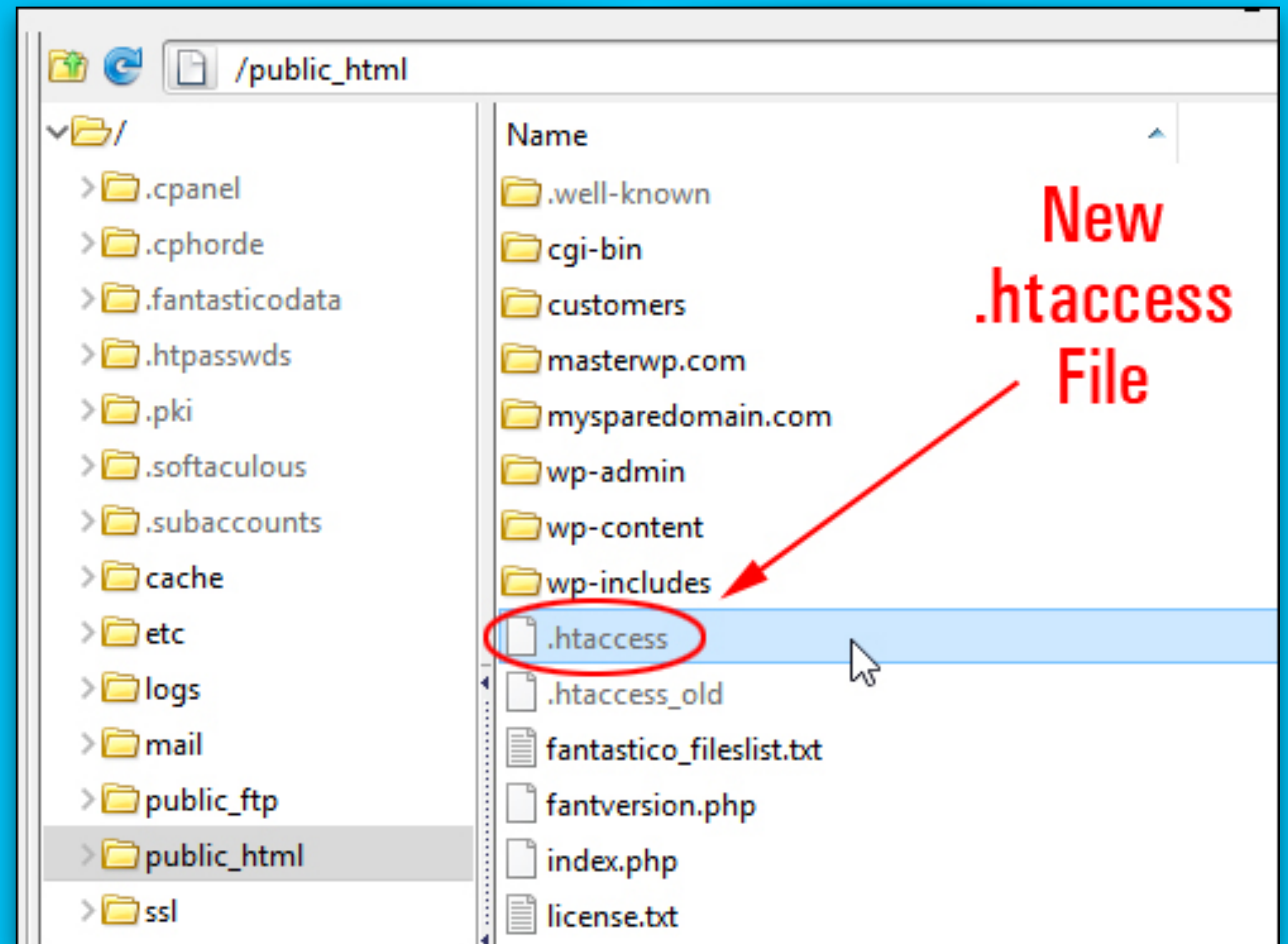
- Server can server thousands of threads at once = asynchronous
- Perfect for real time communication on the internet = web sockets
- Code written in JS! Which lets you be a full-stack developer with only 1 language!
- Relatively new concept but has large community that continuously adds modules to use
- Companies that use Node.JS:  
Netflix, LinkedIn, Walmart, Trello, Uber, PayPal, Ebay, NASA...



# HOW TO CHANGE YOUR HOSTING PREFERENCES: .HTACCESS FILE

# .HTACCESS FILE

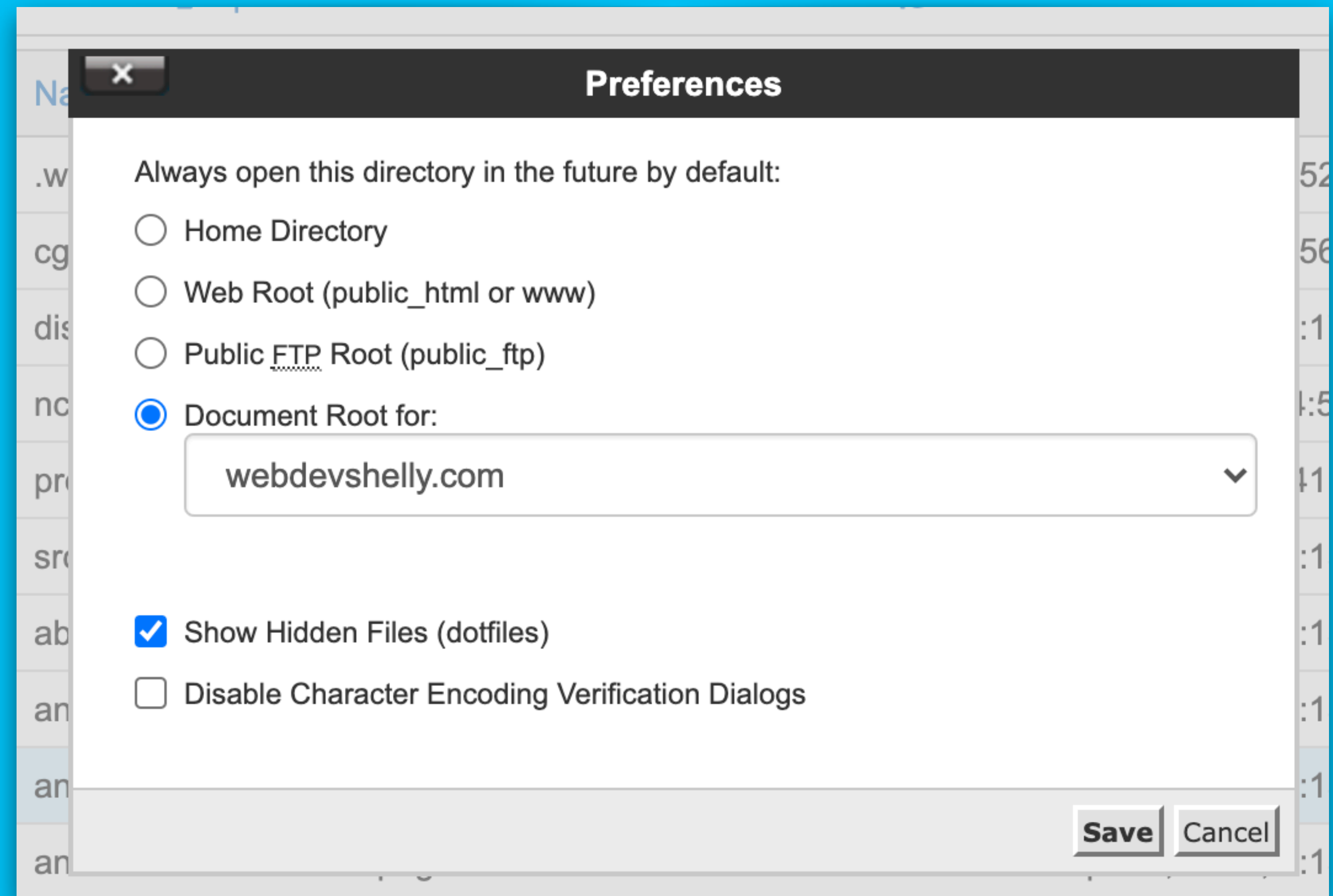
- Short for Hypertext Access
- Configuration file used by apache-based web servers
- Lives at the root (top level of website) of your web server
- **VERY** powerful file that can alter the behavior of your web server
- If you plan to edit it, **ALWAYS** make a copy before you start
- Only 1 .htaccess file per project





# .HTACCESS FILE

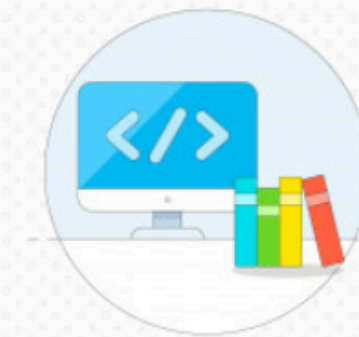
- You can view .htaccess file with your FTP server
- On Linux-based machine (Macs), files that start with a dot are invisible —> show hidden files with **COMMAND + SHIFT + DOT**
- Cyberduck: **COMMAND + SHIFT + R**
- Transmit: **COMMAND + SHIFT + B**
- Show .htaccess file in cPanel through **Settings > Show Hidden Files**



# .HTACCESS FILE RESOURCES

- Good starting point: HTML5 Boilerplate .htaccess file, well maintained and regularly updated:
- <https://github.com/h5bp/html5-boilerplate/blob/master/dist/.htaccess>
- Weaver Tips: <https://weaver.tips/search?q=htaccess>
- Always turn off directory access!

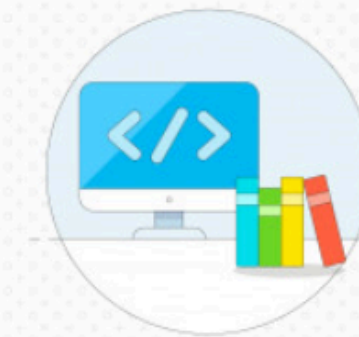
## Search results for: htaccess



### Turn off Directory Listing in .htaccess file

Prevent users from browsing your web hosting environment like a collection of files and folders. Simply add the following... [Read more.](#)

htaccess



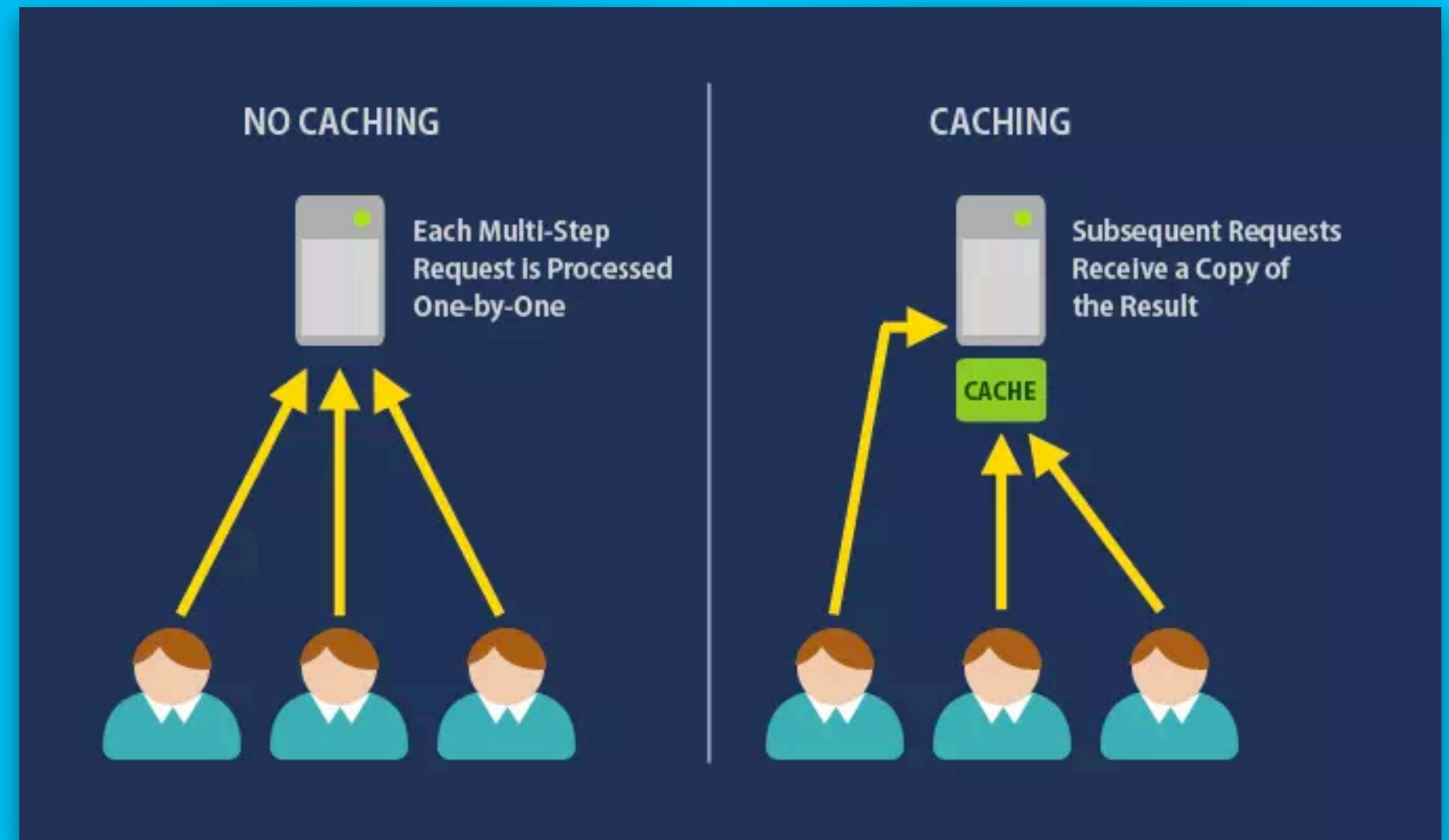
### Force www

To force users to the www version of your site, simply add the following to your .htaccess... [Read more.](#)

htaccess redirect

# CACHING

- Caching means saving mostly static responses from web servers for future use
- Static: images, CSS files, JS files
- Reduces network activity (bandwidth) between client and server & helps with website speed



# TYPES OF CACHE

## Browser Cache

- Sets aside a section of your computer's hard disk to store representations that you've seen
- Especially useful when users hit the “back” button

## Proxy Cache

- Like browser cache but on a larger scale, also known as shared cache
- Serve thousands of users; large corporations and ISPs often set them up on their firewalls
- Proxy cache are out of network, requests have to be routed to them

# IS CACHE GOOD OR BAD?

- One of the most misunderstood technologies on the Internet
- It can help your website perform better (by storing static content)
- It can also prevent users from seeing the current/most recent updates made to your website
- Many companies spend \$\$\$ setting up farms of servers around the world to replicate their content to make access as fast as possible
- Caches do the same for you, and they're even closer to the end user. Best of all, you don't have to pay for them.



# HOW A BROWSER DETERMINES CACHE CONTENT:

- If the response's headers tell the cache not to keep it, it won't
- If the request is secure (HTTPS) it won't be cached
- A cached representation is considered *fresh* (that is, able to be sent to a client without checking with the origin server) if:
  - It has an expiry time or other age-controlling header set, and is still within the fresh period, or
  - If the cache has seen the representation recently, and it was modified relatively long ago.
- Fresh representations are served directly from the cache, without checking with the origin server

# CACHE BUSTING



**Request:** style.css



**Response:** style.css



**ORIGINAL FILE**



**SERVER**



**Request:** style.v2.css



**Response:** style.v2.css



**UPDATED FILE**



# WHAT IS CACHE BUSTING?

- AKA Fingerprinting
- Updating the file names or paths to your files to force the browser to reload your site/certain files from the server
- Helps to serve latest content
- 3 ways to do this:
  1. Query Strings
  2. File Name Versioning
  3. File Path Versioning
- Great resource: <https://css-tricks.com/strategies-for-cache-busting-css/>



# CACHE BUSTING

- Use program to do the work for you:
  - Gulp Cache Bust <https://www.npmjs.com/package/gulp-cache-bust>
  - Gulp Buster <https://www.npmjs.com/package/gulp-buster>
- In your .htaccess file
- Write your own rule with Node.js

**QUESTIONS?**

**FIN**