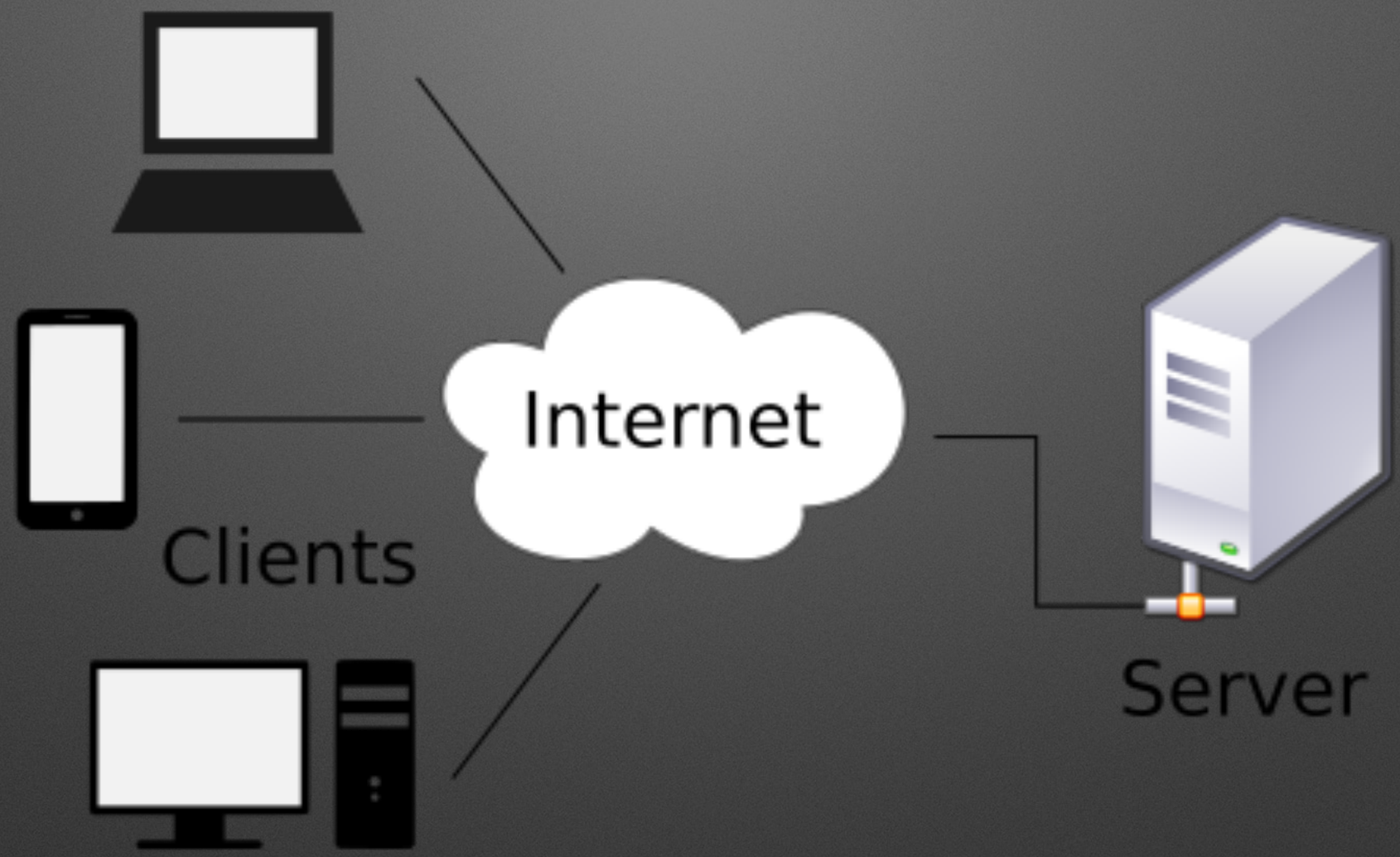
# Web Development Crash-Course: Building a Simple Single-Page Webapp

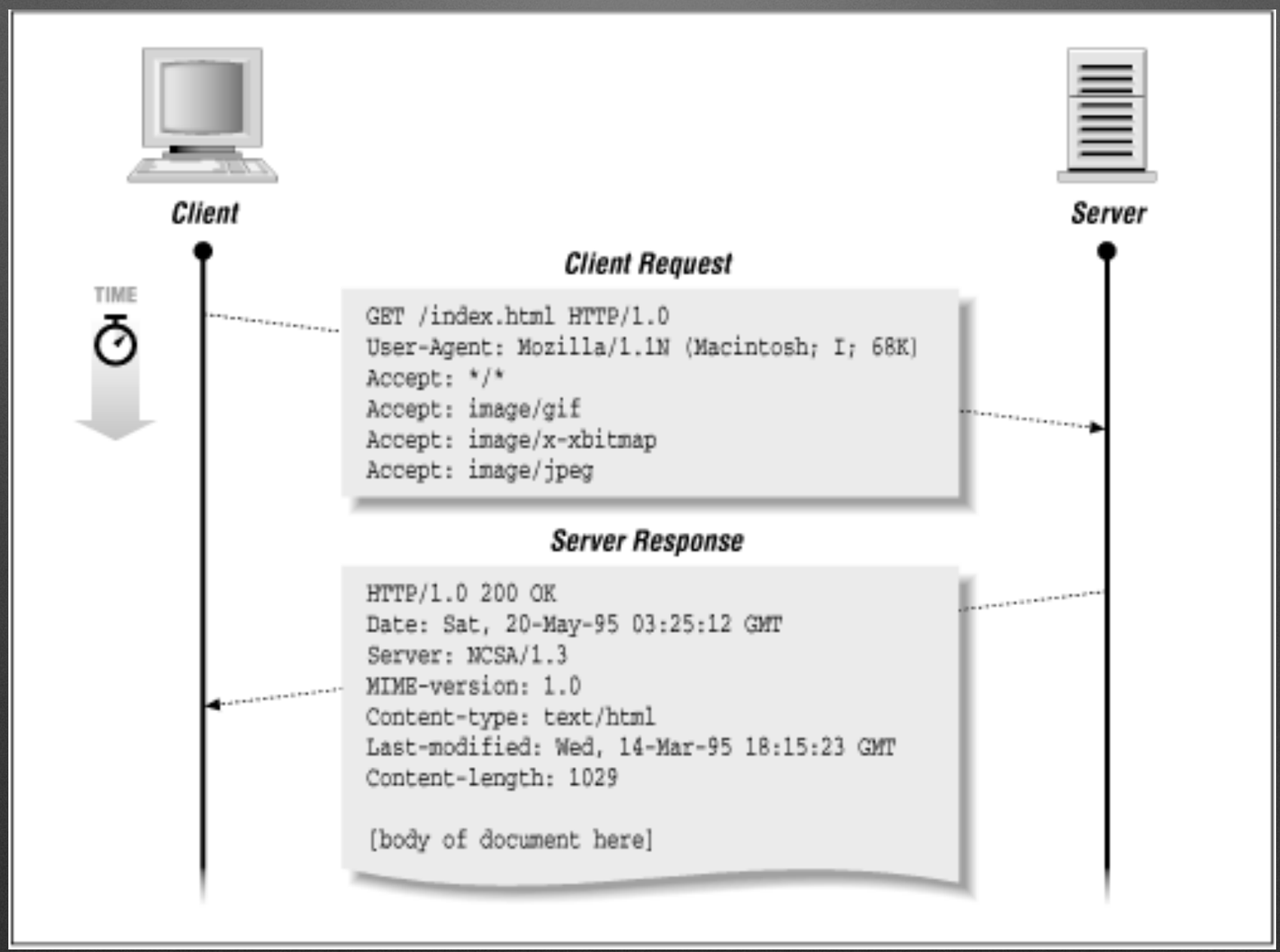
# The Internet

- A giant collection of files stored on different computers around the world
- Web browsers are user interfaces that request the transfer of these files and then render the files
- A webpage is just a text file with some fancy formatting that allows it to be rendered a certain way
- A bunch of application protocols/services make everything work: IP (Internet Protocol), TCP (Transfer Control Protocol), HTTP (Hypertext Transfer Protocol), and more

# Client-Server Model



# HTTP

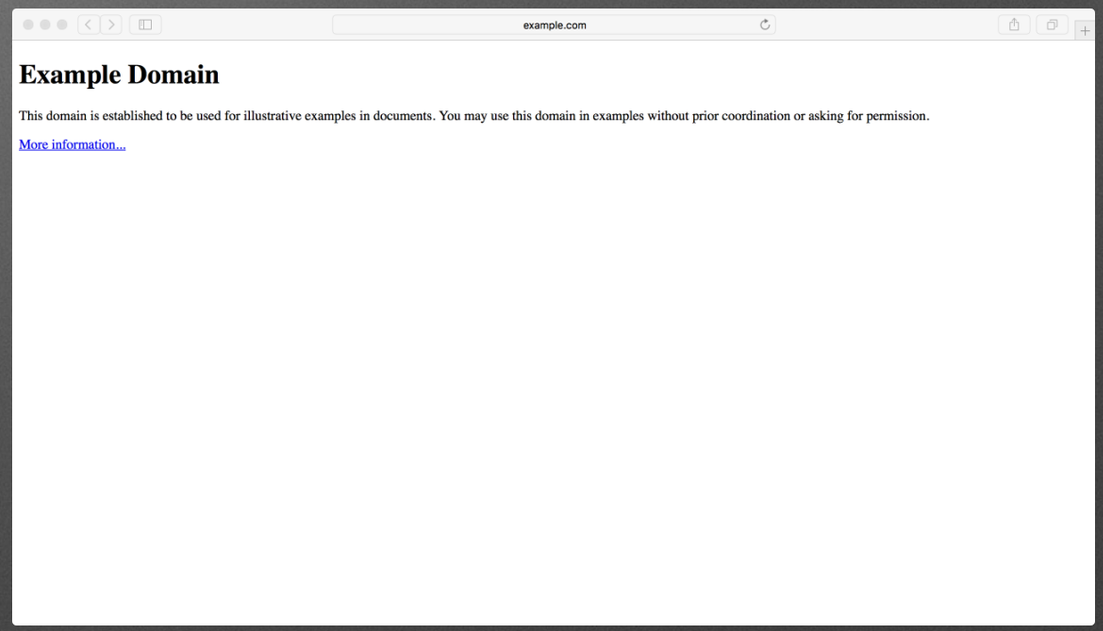

Client
Server
TIME
Client Request
GET /index.html HTTP/1.0
User-Agent: Mozilla/1.1N (Macintosh; I; 68K)
Accept: */*
Accept: image/gif
Accept: image/x-xbitmap
Accept: image/jpeg
Server Response
HTTP/1.0 200 OK
Date: Sat, 20-May-95 03:25:12 GMT
Server: NCSA/1.3
MIME-version: 1.0
Content-type: text/html
Last-modified: Wed, 14-Mar-95 18:15:23 GMT
Content-length: 1029
[body of document here]

# Core Technologies of Webpage

- HTML (Hypertext Markup Language) - defines the core components and structure of a page
- CSS (Cascading Style Sheet) - defines how the HTML components look
- JS (JavaScript) - programming language that defines how interacting with the webpage changes the way it looks; makes the web page dynamic
- Web server - program that receives HTTP requests and responds to them appropriately
- DB (Databases) - store data needed for long term (e.g. user login data)
- HTML/CSS/JS operate on the client machine => “front-end”
- Web server/DB operate on the server machine => “back-end”

# Rendered HTML

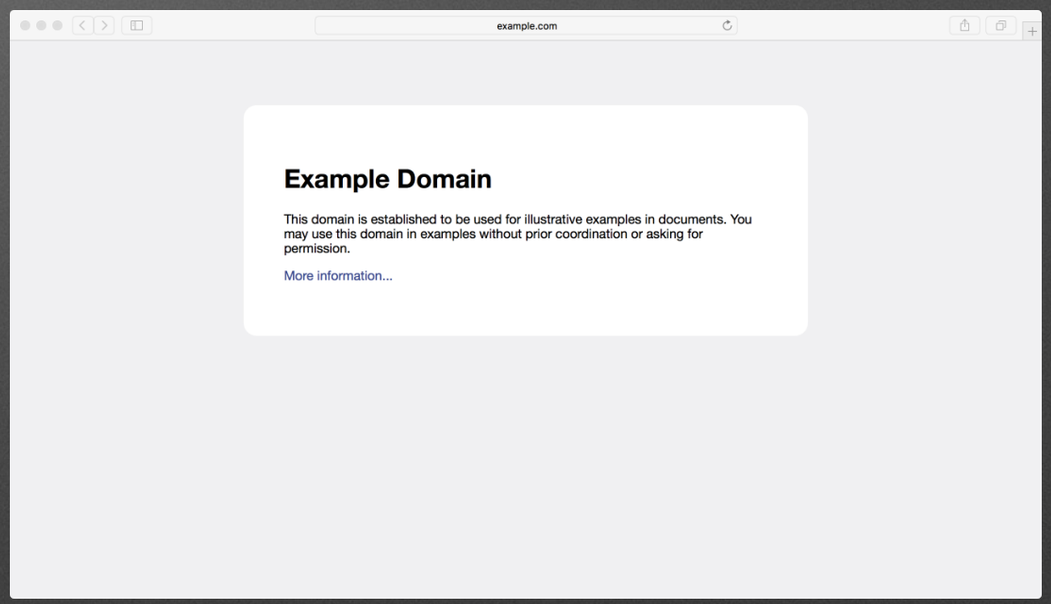
```
<!doctype html>
<html>
  <head>
      <title>Example Domain</title>
      <meta charset="utf-8" />
      <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1" />
  </head>
  <body>
    <div>
        <h1>Example Domain</h1>
        <p>
          This domain is established to be used for illustrative examples in
          documents. You may use this domain in examples without prior
          coordination or asking for permission.
        </p>
        <p>
          <a href="http://www.iana.org/domains/example">More information...</a>
        </p>
    </div>
  </body>
</html>
```



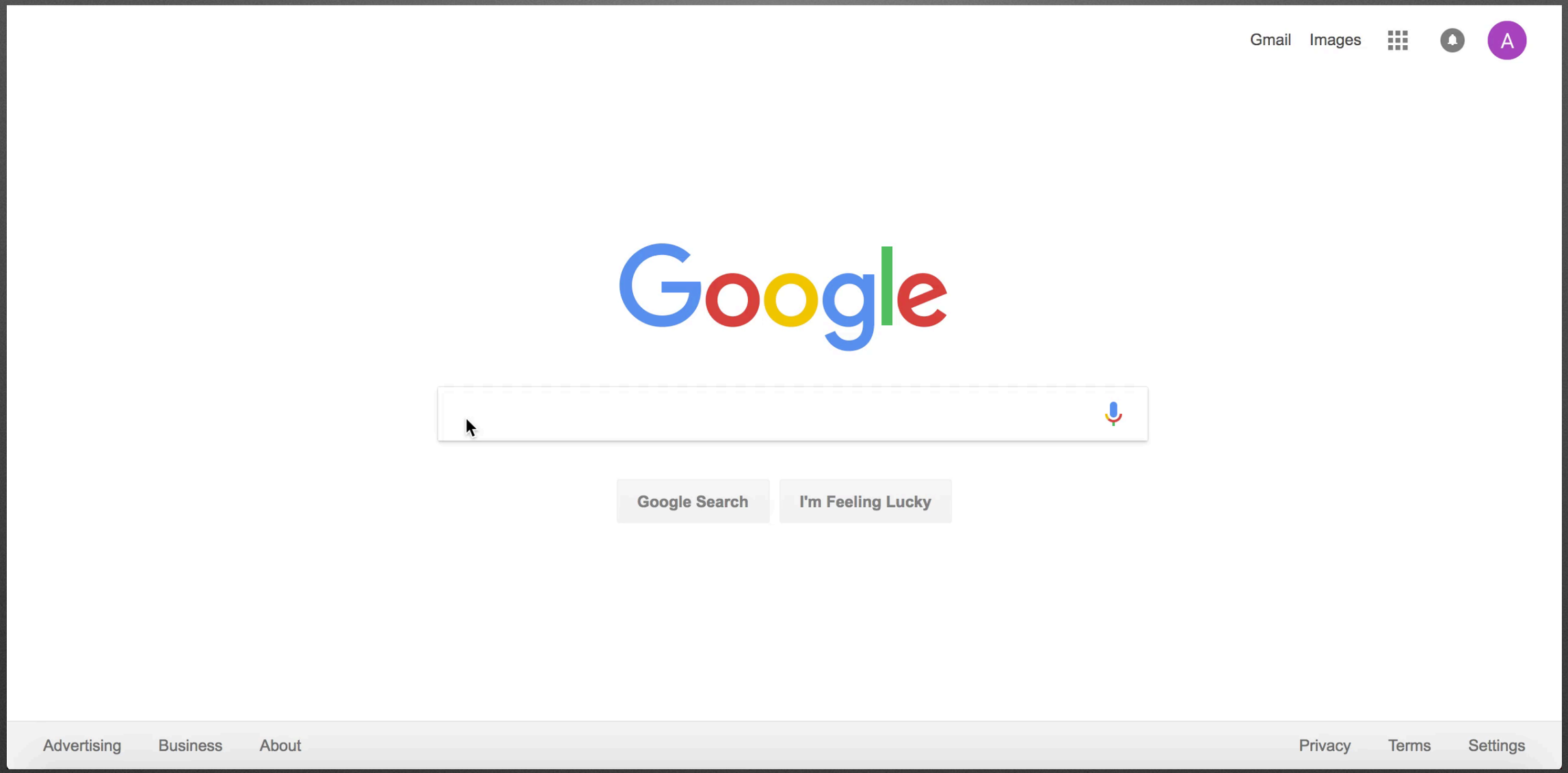
# Rendered HTML + CSS

```html
<!doctype html>
<html>
  <head>
      <title>Example Domain</title>
      <meta charset="utf-8" />
      <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1" />
  </head>
  <body>
    <div>
        <h1>Example Domain</h1>
        <p>
          This domain is established to be used for illustrative examples in
          documents. You may use this domain in examples without prior
          coordination or asking for permission.
        </p>
        <p>
          <a href="http://www.iana.org/domains/example">More information...</a>
        </p>
    </div>
  </body>
</html>
```

```css
body {
  background-color: #f0f0f2;
  margin: 0;
  padding: 0;
  font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
}

div {
  width: 600px;
  margin: 5em auto;
  padding: 50px;
  background-color: #fff;
  border-radius: 1em;
}

a:link, a:visited {
  color: #38488f;
  text-decoration: none;
}

@media (max-width: 700px) {
    body {
      background-color: #fff;
    }
    div {
      width: auto;
      margin: 0 auto;
      border-radius: 0;
      padding: 1em;
    }
}
```



# JS in Action


Gmail
Images
A
Google
Google Search
I'm Feeling Lucky
Advertising
Business
About
Privacy
Terms
Settings

# Creating a Website

- Create HTML/CSS/JS pages
- Create HTTP server that can properly handle requests for the webpages
- Set up a database system to store long-term data
- Building many of these technologies constitutes “re-inventing the wheel” => frameworks!

# Frameworks

- Set of code files that facilitate the process of implementing a website
- Everyone uses them because they make life better
- Many frameworks for both front-end and back-end; each have their strengths and weaknesses
- Back-end frameworks provide an implemented web-server, database management tools, etc.
- Front-end frameworks generally provide organized ways of building user interfaces

# Web Development Engineering Challenges

- Like computer science in general, all about weighing pros and cons
- Designing app and determine its needs
- Choosing frameworks that best suits these needs
- Structuring and optimizing code so that it plays to the strengths of your design

# Let's dive into Flask and ReactJS!

# Flask

- Popular back-end framework built in Python
- Used by some major companies like LinkedIn and Pinterest
- Called a "micro-framework" because it's lightweight and doesn't rely on too many external libraries
- Very extensible and you can easily plug in additional third party libraries for more functionality

# React

- Super popular front-end framework (i.e Javascript) developed by engineers at Facebook
- Used by some major companies like Airbnb, Netflix, and... Facebook
- Allows you to almost entirely avoid writing HTML as you define everything in JS
- Core idea is to build Object-like components, which can be reused and configured

# Let's dive into some code!

# Resources

- More about Flask - Flask documentation
- More about React - React documentation
- React Debugging - React Developer Tools
- Version control - Git documentation
- Client-side navigation - ReactRouter documentation
- Server-side rendering - python-react documentation
- Database plugin for Flask - Flask-SQLAlchemy documentation
- Deploying your web app - Guide to Heroku Flask deployment

# Thanks for listening!