Understanding how websites work, a glimpse into the request or response lifecycle

What is a browser?

- A web browser is a program that asks a website for information
    - This “ask” is called a request
    - When the request is completed, the website returns an answer, called a “response”
    - The response has data in the form of HTML, CSS, JavaScript, images and more
    - And then your computer downloads all these little files, and your browser puts them together and “reads” them
    - Lastly, your browser will display, or “render”, the final output
    - There are 4 major browsers
        - Chrome
        - Firefox
        - Safari
        - Edge
    - But there are hundreds, if not thousands, of smaller lesser-known web browsers available

What does your browser do?

- Browsers collect information, and displays it
    - Your browser’s job is to request information from a website, download the files it needs, and piece them all together in a nice and interactive manner
- Not all browsers are the same
    - What works on one browser may not work on other browsers
    - So if you ever stumble on a website that seems broken in Chrome, try opening it in Firefox
    - And vice-versa
    - (or Safari or Edge)
- It communicates
    - Your browser is responsible for the communication/transactions between your computer and the website
    - It’s like picking up the phone and dialling your best friend
    - You need to wait for them to answer the phone and say “Hello?”

What are web requests?

- Requests ask for information
    - Web requests are when you ask for information from a website
    - You’re simply asking for the files you need to display the website
    - Often this is called a “handshake”
    - How it works
        - It starts off by going to a website, like google.com
        - Your browser will then ask google.com for the data and information it needs to display the page nicely
        - This whole process is called a request

- Request sizes
    - A request can be any size
    - Google.com is a small request, there's not much on there
    - But if you load up instagram.com, every image will be a new request
    - Or on facebook.com, every comment you make, every like you click, those are all requests
- Request types
    - Sometimes a request comes in the form of asking for information
    - But it can also be in the form of "asking to save" information
    - Technically, these are called POST requests
    - There's also DELETE request to delete information
    - And PUT/PATCH requests to update information
    - A normal "handshake" is a GET request
    - This all gets into RESTful API's
- Requests speeds
    - Everything on the internet has to travel through the internet
    - Smaller requests (and responses) mean less bandwidth, which means faster "serve" time
    - It's like this
        - How long does it take to download a 1gb file vs. downloading a 1kb file?
        - One is almost instant, the other can take several hours
    - Requests work the same way

The request lifecycle

- How it begins
    - When you ask for information from a website, you're actually doing a lot more than that
    - When you ask https://kalob.io for information, there's a domain name mapping behind it
    - Kalob.io actually points to an IP address, and that IP address connects to my server
- How it ends
    - Then the server accepts my request, understand what it's supposed to output, and sends HTML, CSS, JavaScript, images, etc. back in a "response"
    - Lastly the browser downloads these files and saves them on your computer (often called a "cache"), pieces them together, and renders the content

What are server responses?

- A "response" happens when a server responds to your web request
    - A response happens after a request is made
    - Responses can be any format but typically it's HTML/CSS/JavaScript
    - In cases where there's an API, an application programming interface, the requests are usually smaller and come in XML or JSON format
    - These formats are much smaller and are pretty close to being "plain text", that means there's not much interpretation needed to display the output
    - Often JavaScript or a server programming language will handle these API responses
    - An example

        - When you load up Instagram.com you're making a request to the website to load the initial data
        - As you scroll down it loads more dynamically so that first request can be small and fast
        - That second request when you "load more" is an API endpoint that probably returns JSON or XML (both are more lightweight than pure HTML)
        - Your browser reads the JSON response, JavaScript interprets it and says "oh I need to display more photos" and dynamically creates your HTML structure for you and adds it to the page
- Assets like images
    - Lastly, each image is likely its own request
    - So, the browser creates a new request for each image file to download, the server sends a response with the image data, and again your browser displays that image
    - Though there are ways to bundle responses together for a faster one-time payload response

Interpreting HTML, CSS, and JavaScript

- Most common response data
    - When you make a request and get a response from a server, you're likely going to get HTML, CSS and JavaScript
    - Your browser's job is to download each of those files
    - But the browser has "engines"
    - It has rules for HTML, different rules for CSS and different rules for JavaScript
- Engines
    - If your browser thinks it's opening a JavaScript file, it'll use the JavaScript "engine" to read and interpret it
    - Likewise with CSS and HTML
    - Once all the files are downloaded (as best as it can) it will piece them all together and display your web page
- JavaScript takes over
    - After the initial response is rendered (displayed) in your browser, most of the time JavaScript takes over
    - JavaScript can tell the browser to make more requests on the fly by dynamically adding links to images and other assets to on your page, or by directly communicating with the browser and creating a new request

Viewing your requests and responses

- You'll need a modern web browser