What is the Internet?

Question

Here is a typical software engineering interview question:

"What happens when you type google.com into your browser's address box and press enter?"

Reference: https://github.com/alex/what-happens-when

Answer

A bird's-eye view gives insight into the "full stack"

- G key is pressed
- The "enter" key bottoms out
- Browser parses the URL
- Browser looks up the DNS
- Browser makes a connection
- Request and response
- Browser parses response
- Browser renders page

Learning objectives

- Understand and explain what happens when you type google.com into your browser's address box and press enter
- Gain an appreciation of complexity hidden beneath the hood of everyday actions
- Learn important terms and concepts in "full stack" web development, which we will come up again in more depth later

The "g" key is pressed

What you see:

- Autocomplete

What you don't see:

- Lookups for suggestions

The "enter" key bottoms out

What you see:

- Things start to happen. Your browser might blank out, or start refreshing
- Progress bars, spinners
- "X" instead of refresh symbol ↻

The "enter" key bottoms out (cont.)

What you don't see:

- State of the browser changes. Goes into this in-between loading state.
- The `WM_KEYDOWN` or `KeyDown` event
- The keycode `13`, which stands for the "enter" key, is transmitted to the computer (USB, PS/2 interface, or a "virtualized" keyboard, like from a touchscreen)

Browser parses the URL

- Browser looks for the `protocol` "http", the `hostname` "google.com" and the `resource` "/".
- Browser decides if this is a URL or a search term. Browsers will default to searching if what's entered in the address bar isn't a valid URL.

Browser parses the URL (cont.)

- Checks if website needs to be redirected via the HTTP Strict Transport Security (HSTS) list.
- Converts non-ASCII Unicode characters in hostname. Only certain characters are valid for hostnames, `a-z`, `A-Z`, `0-9`, `-`, and `.`.

Browser looks up the DNS

- First, it checks the `cache`.
- If it can't find it in the cache, it'll `gethostbyname`, which first checks the `hosts` file (OS dependent).
- If it can't find the host in the `hosts` file, it will look for it in the DNS server, which is determined by the network (router, modem, etc.)

Browser makes a connection

- Now that it has the IP address, it will open a `socket`
- What's important here is the IP address and the `port`, for example a router is typically configured to `192.168.1.1:80`. The IP address is `192.168.1.1`, and `80` is the port number.
- Another example of an IP address is `127.0.0.1` which is typically your `localhost`.

Request

- If the protocol is secure (HTTPS), it will do a TLS (Transport Layer Security) handshake
- For both HTTP and HTTPS, it will do an HTTP `GET` request
- For each `request`, there is a `response`

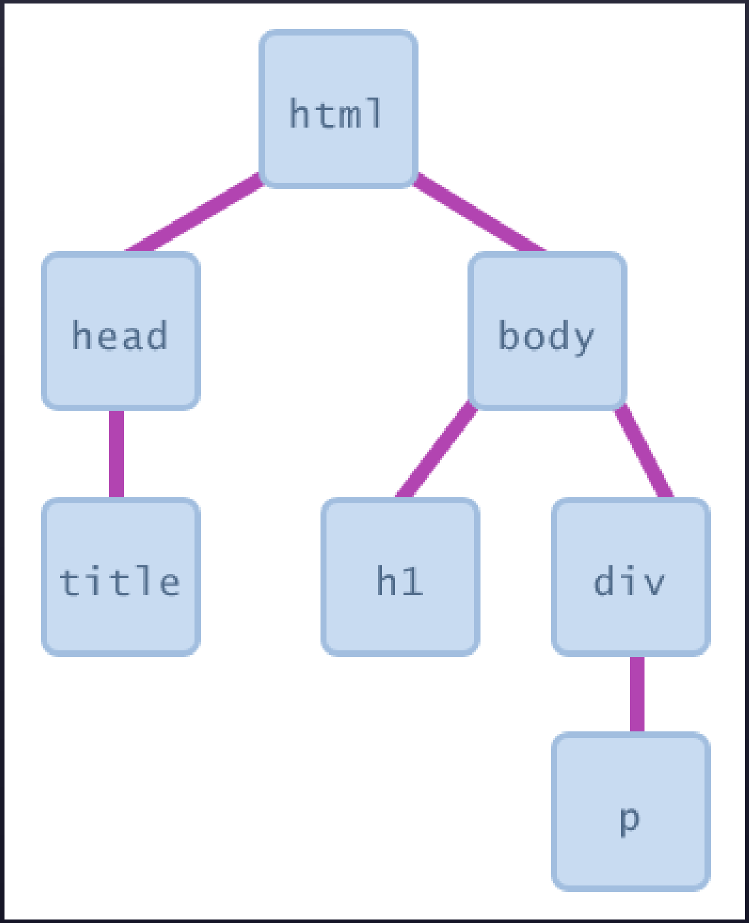
Response

- A typical response is a `200 OK`, which means everything was good
- Other responses include `301 Moved Permanently`, which is a redirect, `404 Not Found`, and `500 Internal Server Error`.
- The first digit usually signifies what kind of code it is (e.g. `2xx`, `3xx`, `4xx`, or `5xx`).

Browser parses resources

- The response comes along with `resources`, which include all the data that a browser needs to render a page: HTML, CSS, JavaScript, images, etc.
- Browser has to parse each of these resources to construct a `tree` known as the `DOM` (document object model).

Browser parses resources (cont.)



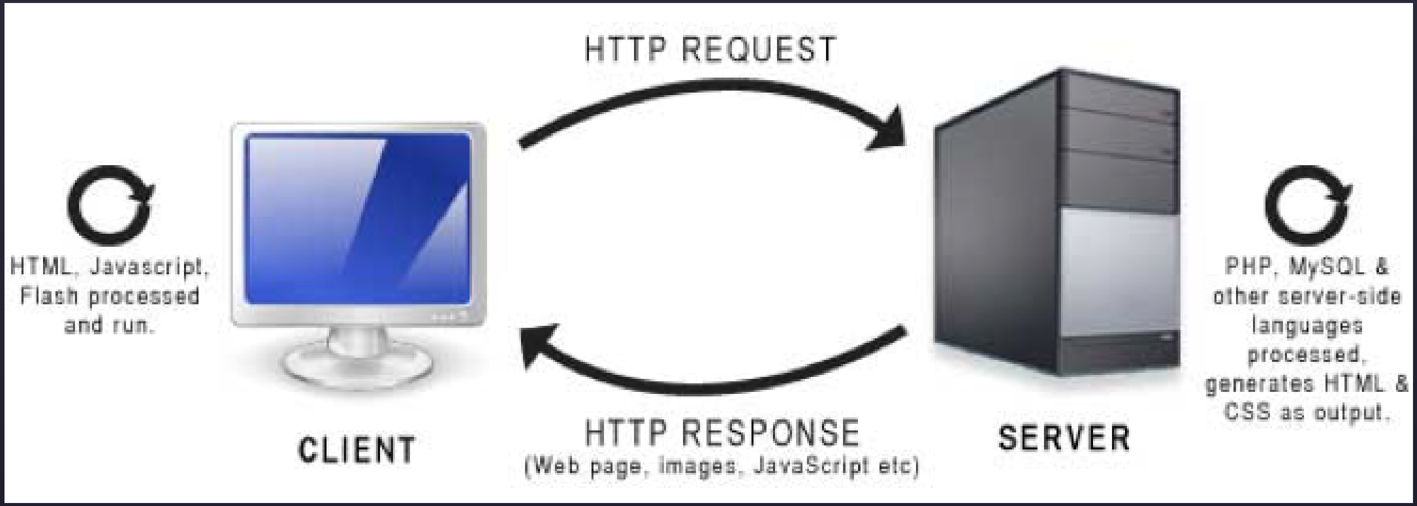
Browser parses resources (cont.)

- The DOM represents each of the HTML `elements` on a page and their hierarchical relationships. Each element also comes with a set of `attributes`.
- An element is an individual component, whereas an attribute is a modifier specific to that individual component (e.g. `<p></p>` is an element and `<p class="title"></p>` is a `p` element with a class attribute set to `title`)

Browser renders page

- It goes through each of the nodes in the DOM tree and calculates the sizes of them on the page
- These calculations are dynamically determined by a bunch of factors: CSS box model (margins, borders, padding), viewport, CSS position (float, absolute, relative), etc.
- Does actual graphic rendering of page (often involves GPU commands through Direct3D or OpenGL)

Summary


HTTP REQUEST
HTML, Javascript, Flash processed and run.
CLIENT
HTTP RESPONSE
(Web page, images, JavaScript etc)
SERVER
PHP, MySQL & other server-side languages processed, generates HTML & CSS as output.

- A ton of things happen when you type google.com into your browser's address box and press enter

Summary (cont.)

- Many moving parts, interdependent pieces of a larger system
- Most of this is abstracted away in different layers, which allows you to think of only the problem at hand

Next: The layer we're going to start is at the `command line`. This is how developers manipulate the files that are needed to run do any of the things we just mentioned

Reference: https://github.com/alex/what-happens-when