



# ASYNCHRONOUS NETWORKING

Client-Server Model + AJAX

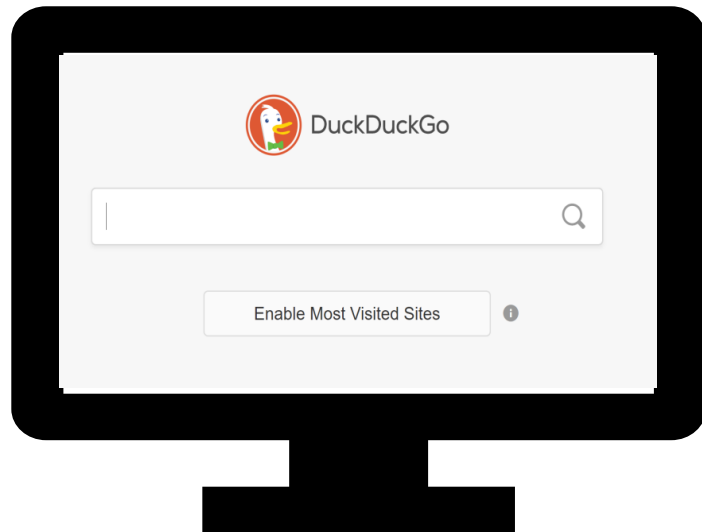
# OVERVIEW

- Client-Server Model + AJAX
- Concurrency & JS
- Networking with XMLHttpRequest()
- Networking with Promises & fetch()
- Networking with async/await & fetch()

# CLIENT-SERVER INTERACTION

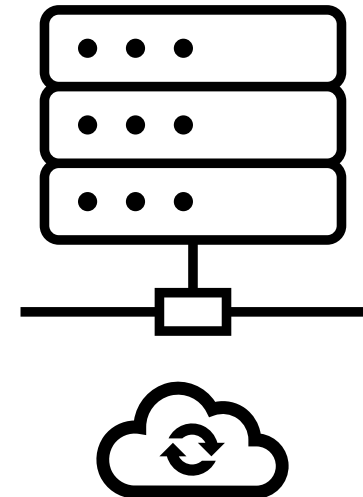
## Client

Smart phones, laptops, etc.



## Server

Racks of machines in datacentres

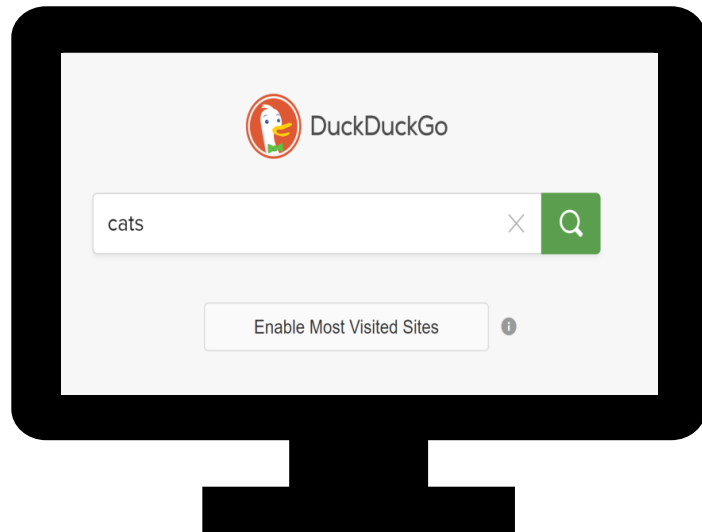


# CLIENT-SERVER INTERACTION

## Client

Makes a request to a server

e.g. Search for websites that contain 'cats'

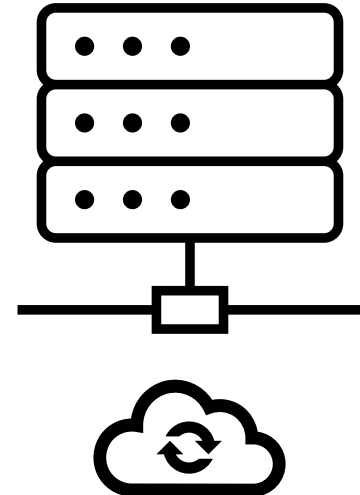


## Request

<https://duckduckgo.com/ac/?q=cats>

## Server

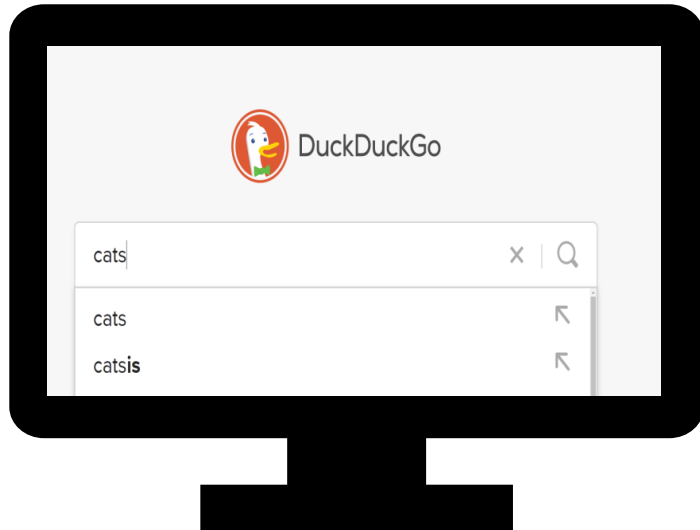
Waiting for requests



# CLIENT-SERVER INTERACTION

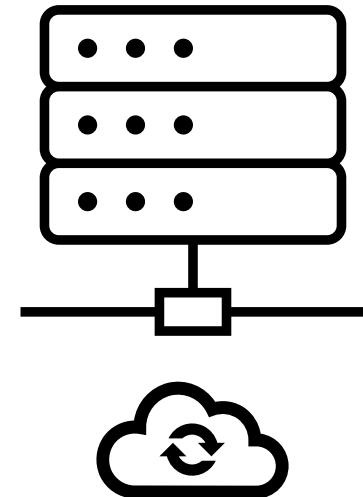
## Client

Receives payload as a server response  
Renders the HTML & CSS, executes the JS



## Server

Validates arguments, executes API, returns data



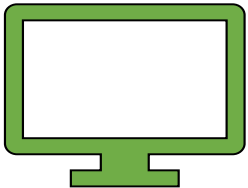
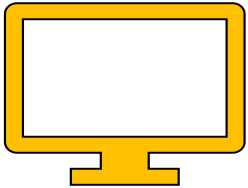
## Response

`{"autocomplete": ["cats", "catsis"]}`



# CLIENT-SERVER REALITY

Webapp Clients



Response



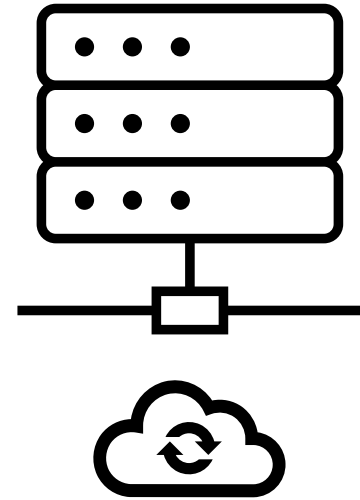
Response



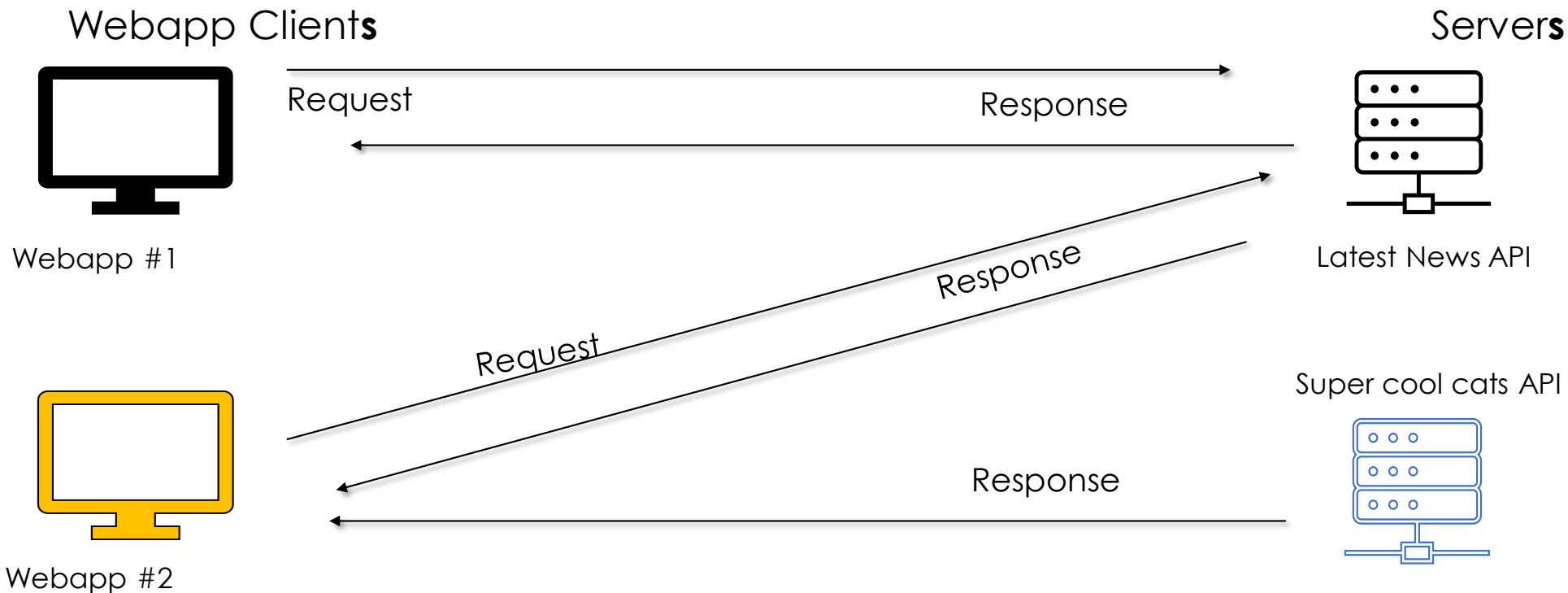
Response



Server



# CLIENT-SERVER REALITY<sup>2</sup>



# LATENCY & THROUGHPUT: THE PROBLEM

## Client-side

- Interactivity + responsiveness = low **latency** is most important
- Page loads ideally fast + once
- Losing your place in a page due to reload is bad UX

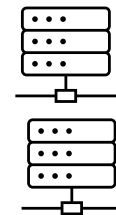
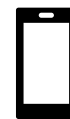
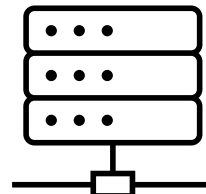
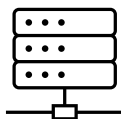
## Server-side

- Need to handle at least thousands of requests
- Resources (time, power, # of servers) are expensive
- Maximising **throughput** is most important



# EVOLUTION OF THE CLIENT-SERVER

## A BRIEF HISTORY



### 1990s & early 2000s

- Clients = majority desktop web browsers
- On-site servers
- Era of “static webpages” i.e. **server-side rendering (SSR)**
- Scale by upgrading server

### Mid-2000s – Early 2010s

- Majority of clients still desktop browsers
- JS adding much more interactivity client-side
- Still scale server vertically

### ~2010s – Present

- iPhone ushers in the smart phone era
- Clients now IoT and mobile
- Almost all rendering is client-side
- Cloud server-hosting is dominant
- Scale servers horizontally

# LATENCY & THROUGHPUT: SOLUTIONS?

## Client-side

- Leave presentation strictly to the frontend
- Initial page load contains minimal necessary data
- Fetch data **asynchronously** as needed ("lazy-loading")

## Server-side

- Scale to handling millions of requests per second by **asynchronous** event processing
- Whilst waiting for I/O, serve another request => never idle
- Increase load-balancing by adding more servers

# Asynchronous Javascript And XML\*

- AJAX = set of techniques to create asynchronous webapps.
- Not any one particular technology
- Allows offline apps, smooth UI/UX, modular frontend & backend

\*JSON is used more nowadays rather than XML. So, Ajax?  
Doesn't quite roll off the tongue as well.

## DEFINING AJAX

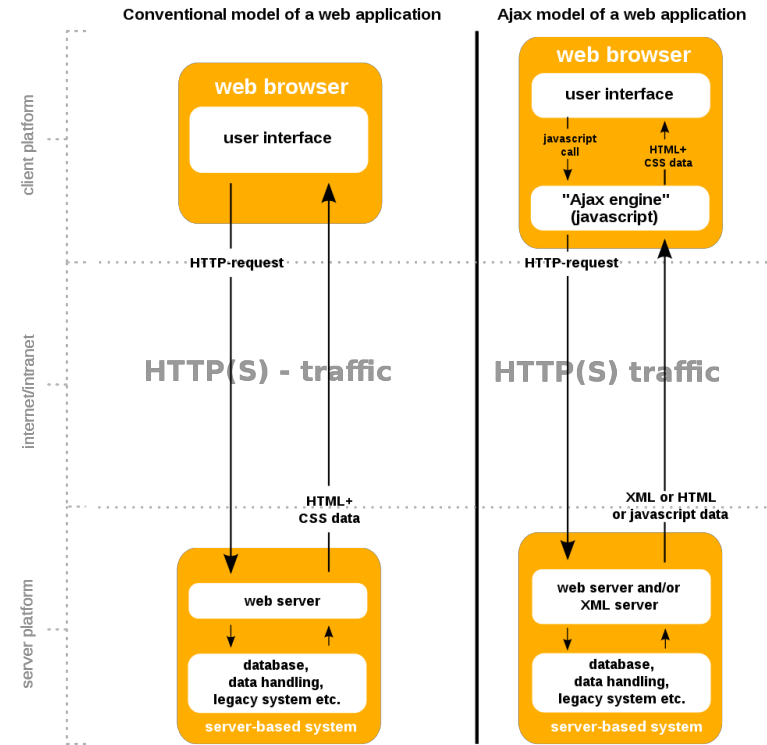


Image Credit: [Wikipedia](https://en.wikipedia.org/wiki/Ajax_(programming))

# SUMMARY

- Today:
  - Client-Server Model
  - AJAX
- Coming Up Next:
  - Concurrency & JS