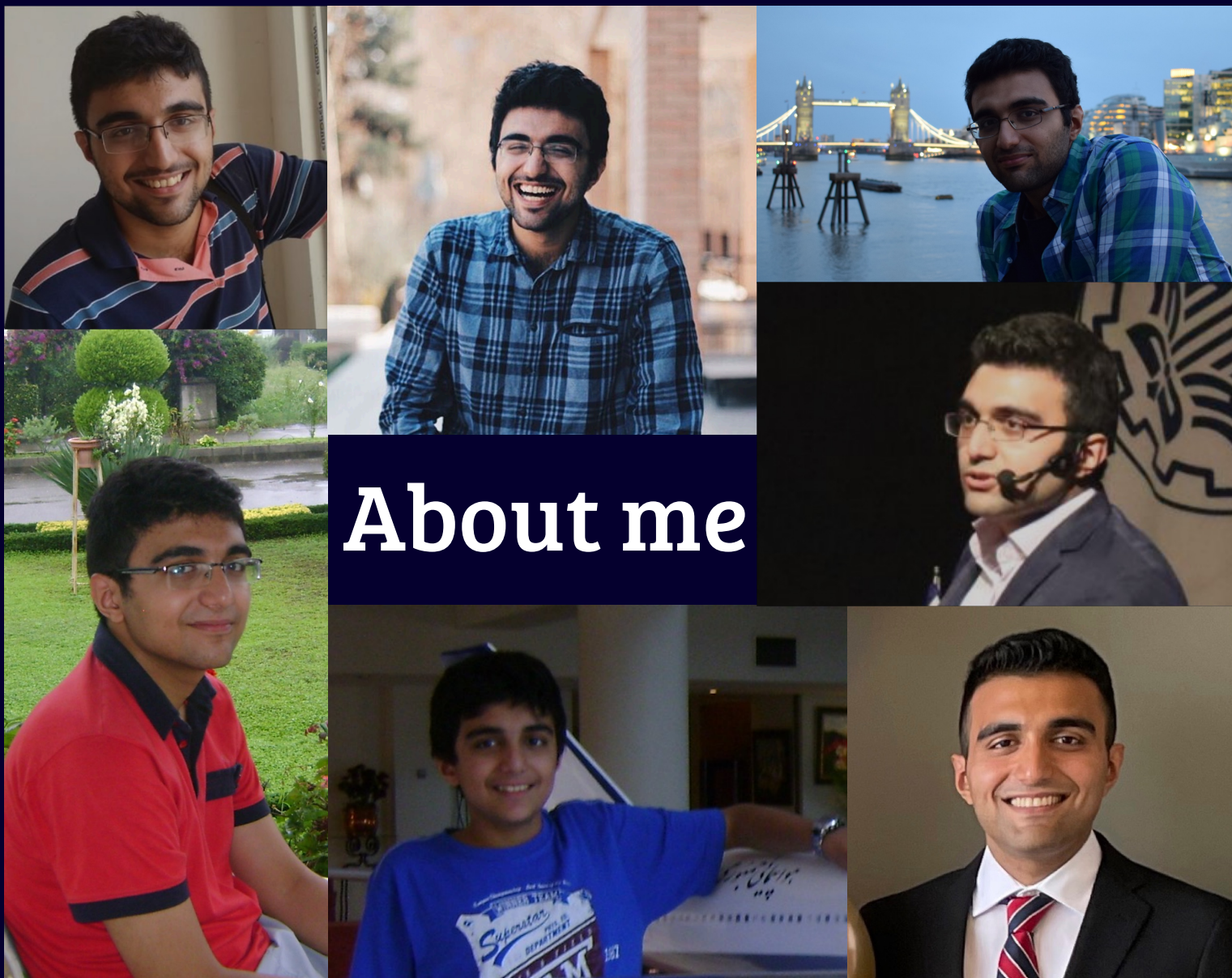




CSC309: Programming on the Web

Kianoosh Abbasi

Fall 2022



Why take a web programming course?

What is it about?

- How does **web** work?
Client/server concepts, browsers, protocols
- **Components** of a website
Server, backend engine, frontend, stylesheets
- Website **design**
Design models, frameworks, data management
- Client-side and Server-side **development**

Course assumptions

- No prior **knowledge/experience** in web development is assumed
- Requirements
 - Programming** experience & **Python** (CSC108)
 - Advanced programming & **OOP** (CSC207 & CSC148)
 - Basic **shell** & system programming (CSC209)
- Recommended Corequisite
 - Database systems** (CSC343)

What will we do?

- How does web work?
Client-server model, requests,
HTTP, browsers
- Static webpages
HTML + CSS
- Dynamic website
Backend framework: Django
- Interactive webpages
JavaScript
Single-page with React
- System administration(Optional)
Website deployment
DevOps

What will we **actually** do?

- Week 1
Course intro, web architecture,
HTML
- Week 2
CSS styling
- Week 3-4
JavaScript
jQuery
Advanced JS
- Week 5-8
Django setup
MVC design pattern
Database models & ORM
Restful APIs
- Week 9-11
Single-page applications
React
NodeJS
- Week 12
Deployment & DevOps (optional)

That's a lot, isn't it?

- Yes, it is!
- Focus on the **knowledge** and **concepts**
- Some coding at the lectures
- The rest is up to you!
 - Online materials
 - Google things often!**
 - Practice with online tutorials
 - Attend lab sessions
 - Doing the **assignments**
 - Incorporate your knowledge to the **course project**

Course delivery

- Monday, Wednesday, Friday 3-4 and 4-5
- Two sections are the same
Attend either lectures
Same TA's, assessments, etc.
- Fridays are usually tutorials
Except for Thanksgiving week or if otherwise announced

Contact points

- Course website
www.cs.toronto.edu/~kianoosh/courses/csc309
We will **not** use Quercus
- Piazza page
<https://piazza.com/utoronto.ca/fall2022/csc309>
Announcements + Q&A
- Markus page
Submissions through the git repository
- **IBS** (Interview Booking System)
Grades, interviews, mentor sessions

Contact points

- Email: csc309-2022-09@cs.toronto.edu
- Discord server
<https://discord.gg/6RgTVxYzyA>
Informal Q&A and chat
- In person office hour
Wednesdays 5:00 – 6:00
- Online office hour
Fridays 9:00 – 10:00

Assignments

- Educational questions to get you started with coding
Challenges your understanding of the concepts
Automatically-tested
- A1: HTML+CSS, A2: JavaScript, A3: Django

Assignments

- You are allowed (even **encouraged**) to Google things or check out online resources
- **BUT** all the code **MUST** be written by **yourself**
Except for the skeleton or parts that are provided by the IDE
- Assignments are **individual work**
No discussion, help, or code from other students
- Get help from TA's, labs, office hours, or online sources

Project

- A **full** (but **small**) website
Toronto Fitness Club (TFC): A website for a gym chain
- Search for studios and amenities, sign up for classes, etc.
- Individually (**not recommended**) or in groups of 2 or 3
You may team up with people from the either sections
- Two parts
Back-end: Django, Front-end: React + CSS

Project

- Each part is delivered and graded **separately**
Meetings with TA's
- Members are graded **individually**
- Regular Q&A sessions with TA's
- Teams are **allowed** to use **online codes**, libraries, or packages
Each piece of copied code must include a **reference** to its source
No uploading or sharing of code between teams
- Start looking for teammates now!

Grade breakdown

- **Assignments: 45%**
A1: 10%, A2: 15%, A3: 20%
- **Project: 55%**
PB: 25%, PF: 30%
- No final or midterm exam!

Late submissions

- **Negligible** penalty in the first hours

$$p(6) = 0.048\% \quad p(12) = 0.4\%$$

- **Small** penalty in the first day

$$p(24) = 3.35\%$$

$$p = 0.0002x^{3.06}$$

- **Significant increase** afterwards

$$p(36) = 11.57\% \quad p(48) = 27.9\% \quad p(72) = 96.49\%$$

Important heads-up!

- Designed to accommodate **unforeseen** situations
- Do **NOT** automatically push deadline in your minds!
- Most **extension** requests will **NOT** be approved
You almost have **one penalty-free day**
- Start early: assignments and projects are **big**!
- Everything must be graded **before holidays**
A very tight **hard deadline**

Schedule Overview

Also available on course website

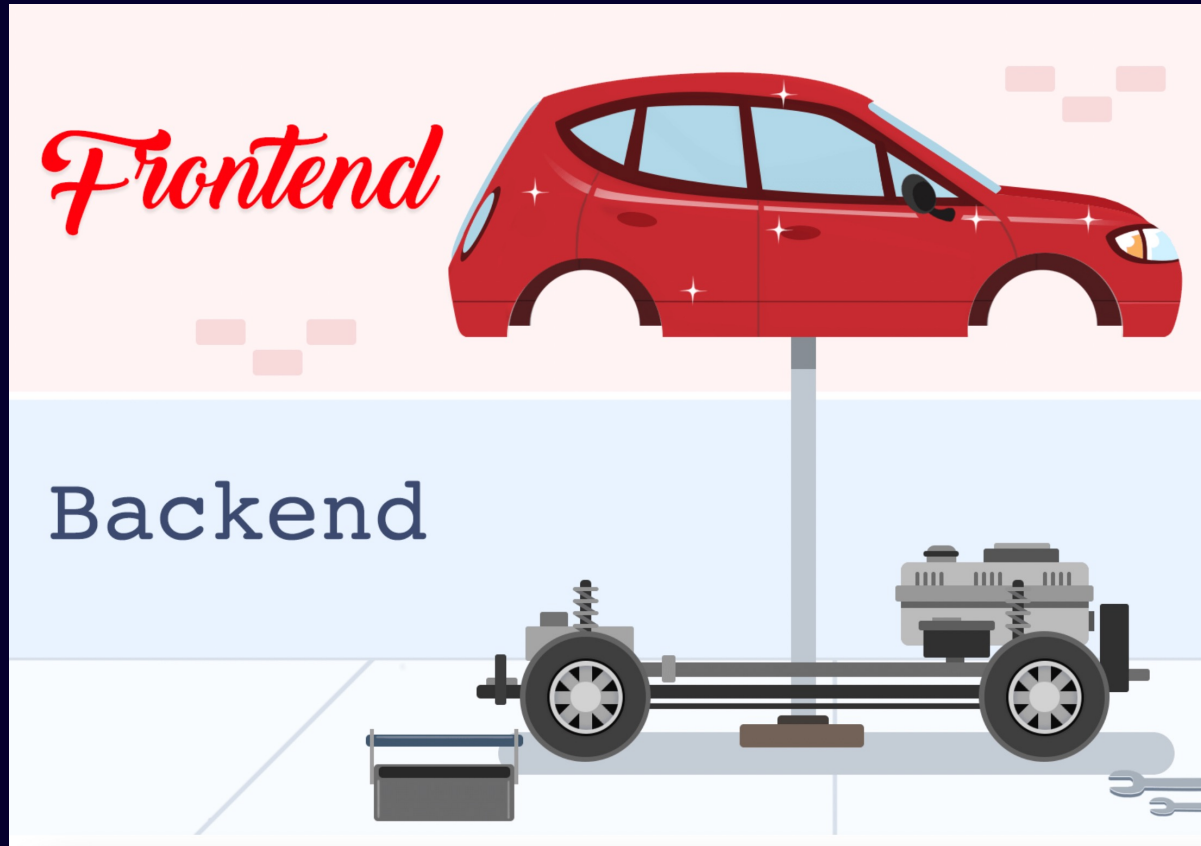
Lecture Number	Class Dates	Title	Deadlines
Week 1	Sep 12, 14	Intro + HTML	
Week 2	Sep 19, 21	CSS	
Week 3	Sep 26, 28	JavaScript #1	A1: HTML + CSS (Sunday)
Week 4	Oct 3, 5	JavaScript #2	
Week 5	Oct 12, 14 (Friday)	Django #1	A2: JavaScript (Sunday)
Week 6	Oct 17, 19	Django #2	
Week 7	Oct 24, 26	Django #3	A3: Django (Sunday)
Week 8	Oct 31, Nov 2	Django #3	
Reading week			
Week 9	Nov 14, 16	React #1	PB: Django (Thursday)
Week 10	Nov 21, 23	React #2	
Week 11	Nov 28, 30	React #3	
Week 12	Dec 5, 7	DevOps (optional)	PF: React + CSS (Thursday)

Academic integrity

- University's policy takes it **very seriously**
Violations will result in **failing** the course
- Rules
 - No **code sharing** at assignments or project
 - No **discussion** at assignments
 - No **online**/pre-written code at assignments
 - No copied code without **reference** at project

Questions?

Web development



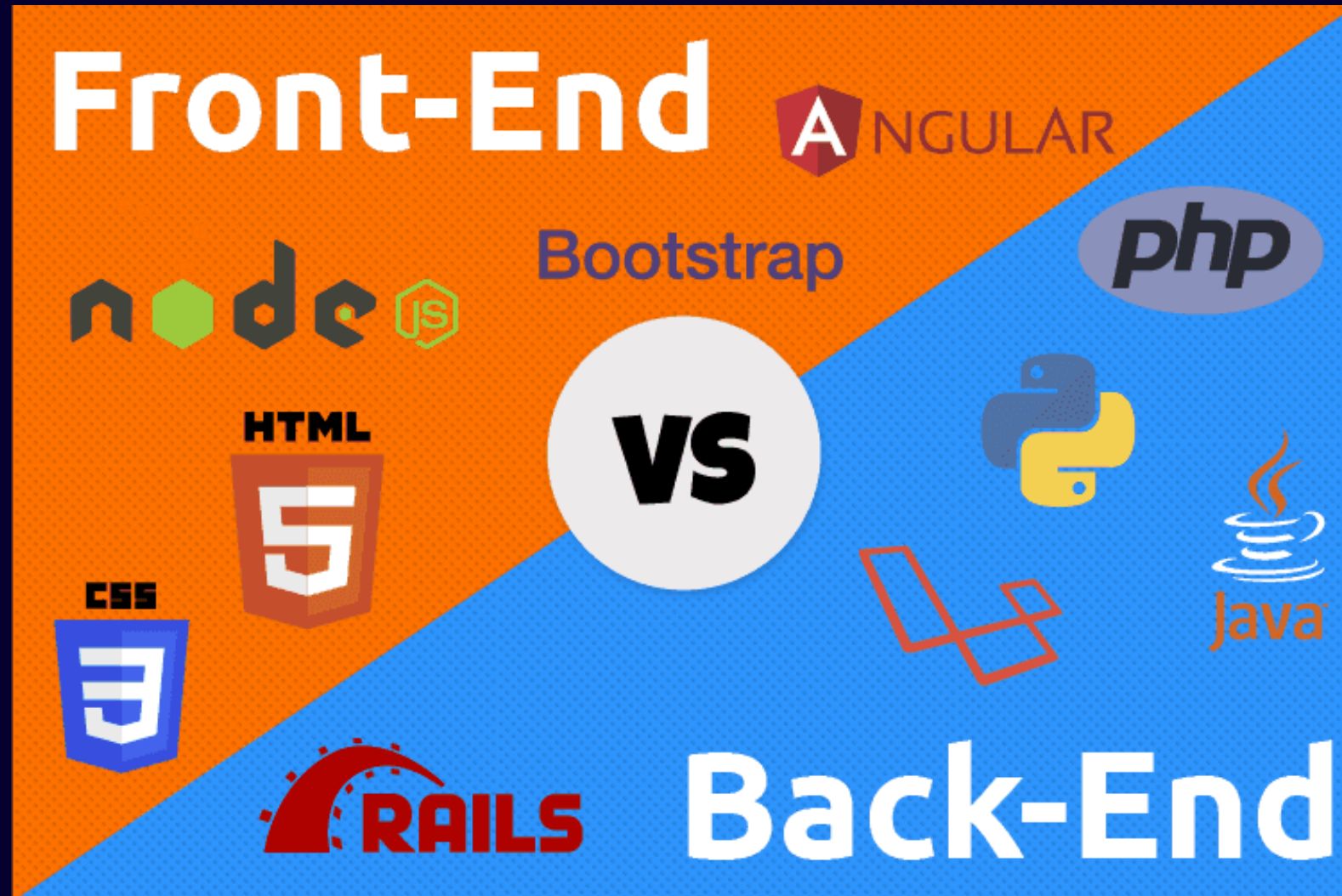
Source: blog.back4app.com

BACKEND vs FRONTEND



Source: https://www.reddit.com/r/ProgrammerHumor/comments/m187c4/backend_vs_frontend/

Web development



Source: nimapinfotech.com

Front-end development

- What user can see
 - User interface (UI)
 - User experience (UX)
- What is run on the client-side
 - HTML/CSS rendering
 - Javascript codes


How web works

← → ↻ 🔒 https://web.cs.toronto.edu

Computer Science
UNIVERSITY OF TORONTO

Home About Undergraduate Graduate Research News & Events People Contact Us

The Department of Computer Science administrative staff are available Monday to Friday, 8:45 am to 5:00 pm.
[Get the latest COVID-19 updates here.](#)



Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Filter URLs

Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms	2.56 s	5.12 s	7.68 s
200	GET	web.cs.toronto.edu	/	document	html	20.66 kB	104.90 kB	83 ms			
200	GET	use.typekit.net	WhQmG4e5xihD2s2Be7yvGea3coDHzSlayFyJfBfheKvfeCtiffHN4UJLFRbh52jhWDMR5e9ojQkwD9hwewDFe	script	js	7.24 kB	18.34 kB	3 ms			
304	GET	assets.squarespace.com	modern.js	script	js	cached	80.12 kB	19 ms			
200	GET	assets.squarespace.com	moment-js-vendor-26ddeab7fa5f90b6c8cb3-min.en-US.js	script	js	45.69 kB	251.71 kB	19 ms			
200	GET	assets.squarespace.com	ddr-resource-pack-be81d1ce004cbca505842-min.en-US.js	script	js	24.87 kB	118.94 kB	16 ms			
200	GET	assets.squarespace.com	common-vendors-stable-5f58a0e5b599c258afba7-min.en-US.js	script	js	76.61 kB	243.17 kB	22 ms			
200	GET	assets.squarespace.com	common-vendors-a15d3b6e09e0c8a937ea6-min.en-US.js	script	js	168.76 kB	585.13 kB	34 ms			
200	GET	assets.squarespace.com	common-be3203642cb72770e4c89-min.en-US.js	script	js	188.02 kB	747.99 kB	33 ms			
200	GET	assets.squarespace.com	performance-bc3576d7eca79df62e49-min.en-US.js	script	js	14.27 kB	45.70 kB	17 ms			
304	GET	ajax.googleapis.com	jquery.min.js	script	js	cached	93.54 kB	2 ms			
304	GET	ajax.googleapis.com	jquery.min.js	script	js	cached	82.40 kB	5 ms			
200	GET	www.googletagmanager.com	js?id=G-9CDDN8N95H	script	js	60.96 kB	163.12 kB	40 ms			
304	GET	code.jquery.com	jquery-3.5.0.min.js	script	js	cached	87.40 kB	17 ms			
304	GET	static1.squarespace.com	site-bundle.js	script	js	cached	57.09 kB	11 ms			
200	GET	assets.squarespace.com	style.css	stylesheet	css	2.93 kB (raced)	9.29 kB	45 ms			
304	GET	assets.squarespace.com	custom-table.js	script	js	cached	14.40 kB	40 ms			
304	GET	cdnjs.cloudflare.com	jquery.min.js	script	js	cached	87.40 kB	16 ms			
302	GET	web.cs.toronto.edu	plugin-accotabs.js	script	js	34.60 kB (raced)	124.39 kB	37 ms			
302	GET	web.cs.toronto.edu	plugin-lightbox.js	script	js	72.93 kB (raced)	228.19 kB	39 ms			

45 requests 4.75 MB / 1.38 MB transferred Finish: 6.31 s DOMContentLoaded: 951 ms load: 1.47 s

How web works

- A lot of things happen when a single **webpage** is loaded!
- Lots of HTML/CSS/JS is fetched
- All in the form of **requests** & **responses**
Browser (**client**) sends requests to one or more **servers** and receives responses

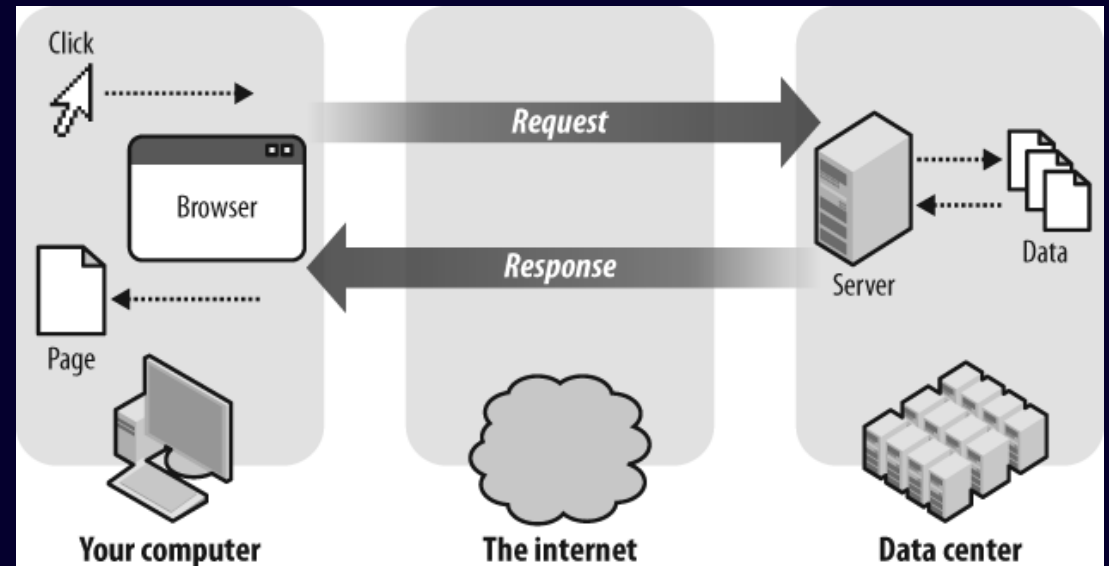
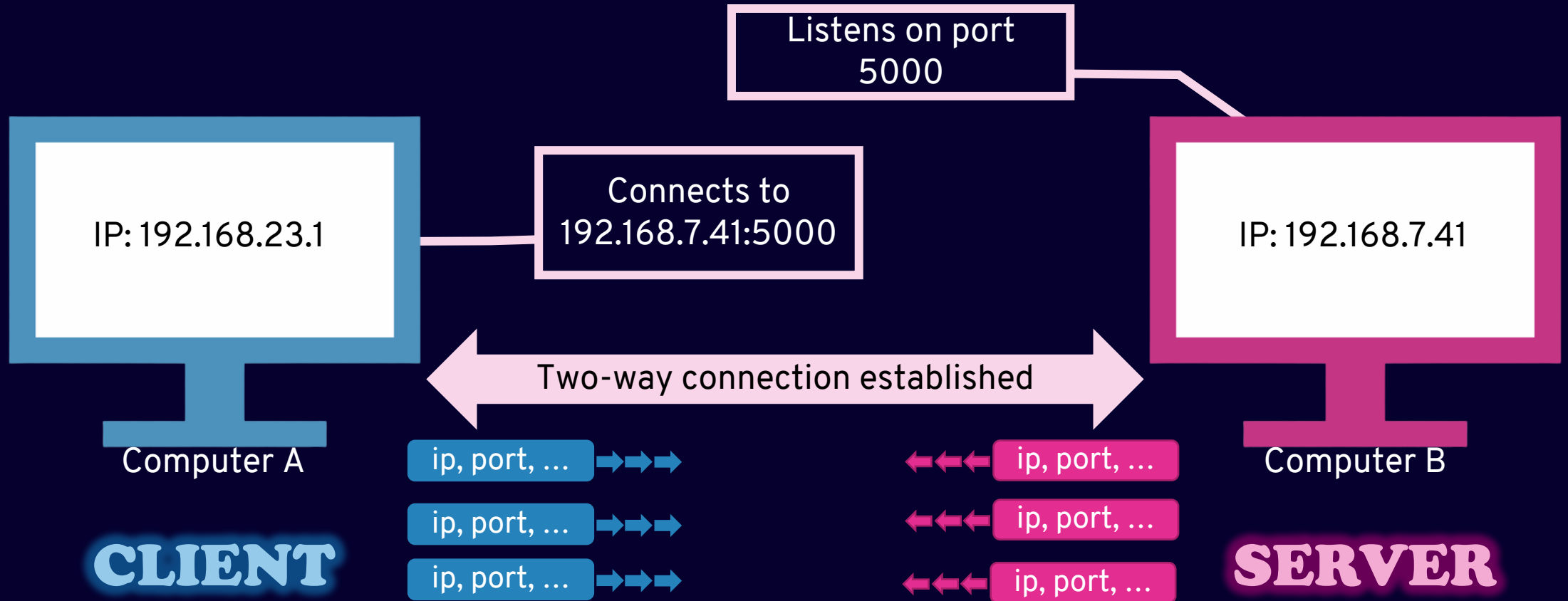


Image source: <https://medium.com/@lokeshchinni123>

How computers talk to each other?



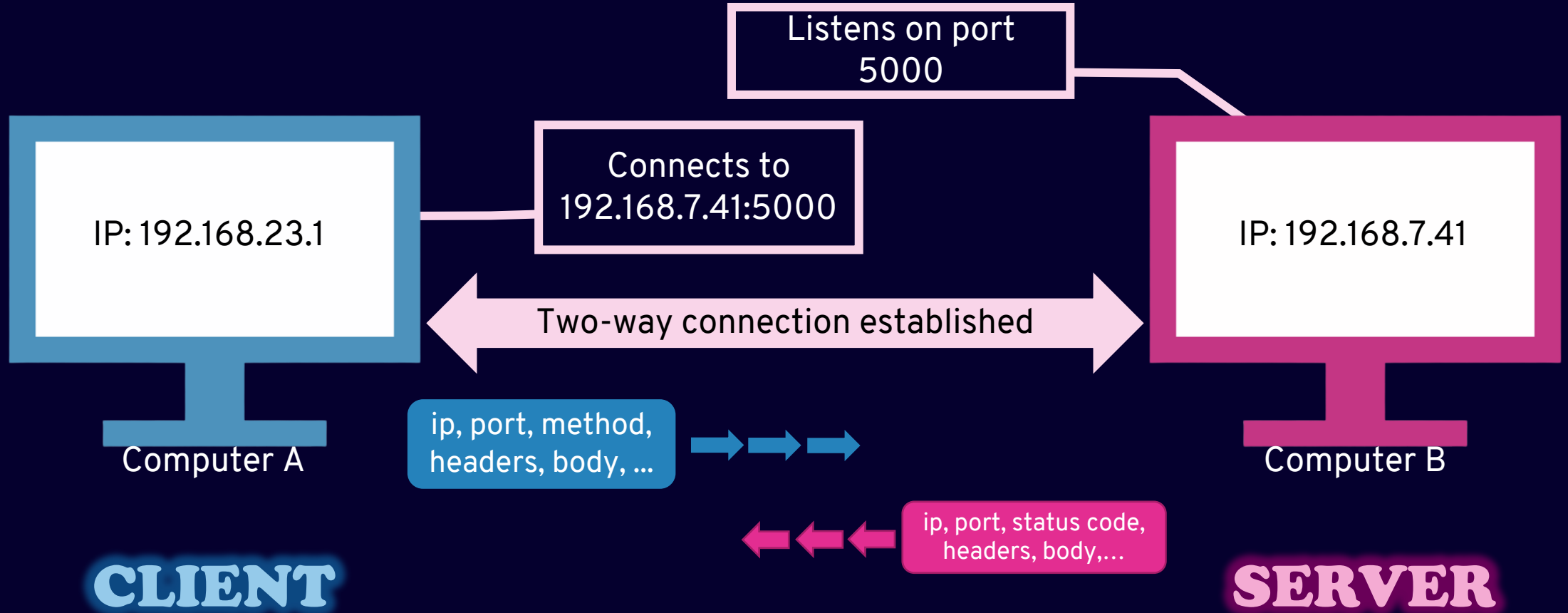
Domains

- Mapped to IP addresses
www.google.com -> 142.251.41.78
- Stored in Domain Name Servers (DNS)
- Clients first resolve the domain, then connect to the IP address
- Already knows which DNS server to talk to

Stateful vs Stateless

- Two-way open connection is **stateful**
- What the server responds depends on **previous request/responses**
- Server should keep track of **thousands** of open connections
- If connection breaks, all the state is **lost**
- A **stateless** protocol is preferred

Stateless Protocol



HyperText Transfer Protocol (HTTP)

HTTP Message

- A **string** with a special **format**
- Request a more specific target
 - Path**: /, /signup, /account/index.html, ...
 - Method**: GET, POST, PUT, ...
- Headers & Body
- Default port is **80**

HTTP Message

Requests

```
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
```

```
-12656974
(more data)
```

Responses

```
HTTP/1.1 403 Forbidden
Server: Apache
Content-Type: text/html; charset=iso-8859-1
Date: Wed, 10 Aug 2016 09:23:25 GMT
Keep-Alive: timeout=5, max=1000
Connection: Keep-Alive
Age: 3464
Date: Wed, 10 Aug 2016 09:46:25 GMT
X-Cache-Info: caching
Content-Length: 220
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML
2.0//EN">
(more data)
```

start-
line

HTTP
headers

empty
line

body

Response codes

- **Success: 200-299**
200 OK, 201 Created
- **Redirection: 300-399**
301 moved Permanently
- **Client errors: 400-499**
404 Not Found, 400 Bad Request, 403 Permission Denied
- **Server errors: 500-599**
500 Internal Server Error, 502 Bad Gateway

HTML

- A specific form of **Extensible Markup Language (XML)**
Data is annotated with nested **tags**
- HTML has specific tags for a webpage to describe what the page contains
- More on HTML later today

Web browser

- Connects, sends requests to server, and receives responses

Upon entering the **Uniform Resource Locator (URL)**

- **Renders** the response

HTML

Image

PDF

So far...

- Server **listens** on a specific port, client(s) connect to IP and port
- Stateless HTTP protocol: **Request** & **Response**
- HTTP response body can be in **HTML** format
- Browsers understand this format and **renders** accordingly

Questions?

HTML

- Focusing on the **renderer** side of a browser!
- Plain HTML files, no server/clients
- HTML file surrounded by the **<html>** tag
 <body> and <head> tags
- Tags and **elements**
- Elements can have **attributes**

HTML tags

Visit <https://www.w3schools.com/html/>

- Headings: `<h1>` to `<h6>`
- Paragraphs: `<p>`
- Links: `<a>`
Stands for anchor
- Images: ``
- Lists: `` and ``
- Tables: `<table>`
- Navigation Bar: `<nav>`
- New line: `
`

HTML attributes

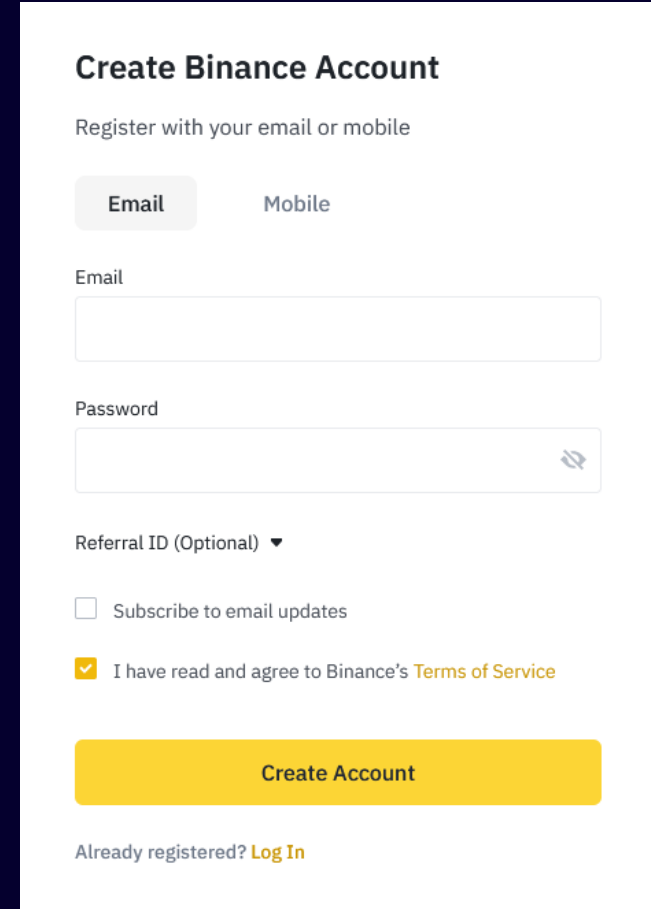
- `style` attribute: next session
- Identifiers: `id` vs `class`
- Other attributes: `src` for `` and `href` for `<a>`
- You can put any custom attribute you want

Other HTML tags

- `<div>` and ``
- Select part of document to apply specific attributes
- ``: inline organization
- `<div>`: block-level organization

Forms

- Primary way to send user data to server
- On submit, a **request** is often sent
- Comprised of many **inputs**



The image shows a 'Create Binance Account' form. At the top, it says 'Create Binance Account' and 'Register with your email or mobile'. There are two tabs: 'Email' (selected) and 'Mobile'. Below the tabs are input fields for 'Email' and 'Password'. The 'Password' field has a strength indicator icon on the right. Below the password field is a 'Referral ID (Optional)' dropdown menu. There are two checkboxes: 'Subscribe to email updates' (unchecked) and 'I have read and agree to Binance's Terms of Service' (checked). At the bottom is a large yellow 'Create Account' button. Below the button, it says 'Already registered? Log In'.

Inputs

- Text field
`<input type="text" />`
- Passwords, emails, etc.
`type="password", ...`
- Radio button
`<input type="radio" />`
- Checkbox
`<input type="checkbox" />`
- Textarea
`<textarea>`
- Submit button
`<button type="submit">`

Forms

- **Action** attribute defines the URL/path of the HTTP request
- **Method** attribute: HTTP method parameter
- Inputs: **name** and **value** attributes

GET vs POST

- GET is usually used for **queries** and **retrievals**
Google search
- The **query params** are appended to the end of the URL
Why?
- POST: sending **private user data** (name, password, etc.)

This session

- Course intro
- Client/Server model
- HTTP request/response model
- HTML tags and elements

Next session

- Adding **style** to HTML
- Basic CSS **rules**
Styles, selectors, precedence, units
- **Spacing**
Box model: margin, border, padding
- **Layout**
Positioning, flexbox, grid