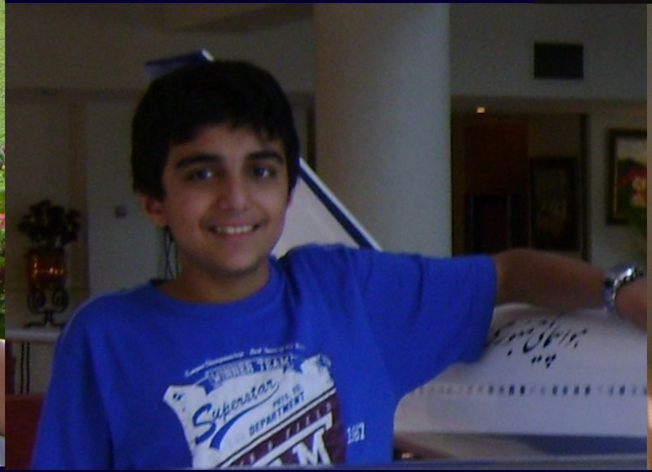
CSC309: Programming on the Web

Kianoosh Abbasi

Winter 2022

About me



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)

- 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 3-6
  Django setup
  MVC design pattern
  Database models & ORM
  Restful APIs
- Week 7-8
  JavaScript
  jQuery
  Advanced JS
- 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

- Thursdays 9-11, Fridays 1-3
- Two sections are the same
  Attend either lectures
  Same TA's, assessments, etc.
- Online until Jan 31st
  Afterwards, who knows!

Contact points

- Course website
  www.cs.toronto.edu/~kianoosh/courses/csc309

- Quercus page
  will not be used a lot

- Piazza page
  https://piazza.com/utoronto.ca/winter2022/csc309
  Announcements + Q&A

- Markus page
  Will be set up in about a week

Contact points

- Email me at kianoosh@cs.toronto.edu

- Head TA: Han Xian Xu Huang
  hanxianxu.huang@utoronto.ca

- Discord server
  https://discord.gg/VGFQN62EeY
  Informal Q&A and chat

Labs

- One hour per week
  - Starting from next week
- Discussing last week's lecture in detail
  - Tutorial
  - TA Office hour
- You can attend any of the sessions

Instructor's office hours

- In person office hour
  Thursdays 11:00 - 12:00
- Online office hour
  Mondays 12:00 - 1:00
- Or by appointment

Assignments

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

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
  Restify: Social media for restaurants
- Follow, like, comment on restaurants, posts, menus
- Individually (not recommended) or in groups of 2 or 3
  You may team up with people from the either sections
- Three phases
  P1: HTML + CSS, P2: Django, P3: React

Project

- Each phase 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!
  - P1 deadline as soon as Feb 11th

Grade breakdown

- Assignments: 40%

  A1: 10%, A2: 15%, A3: 15%

- Project: 60%

  P1: 15%, P2: 20%, P3: 25%

- No final or midterm exam!

Schedule Overview

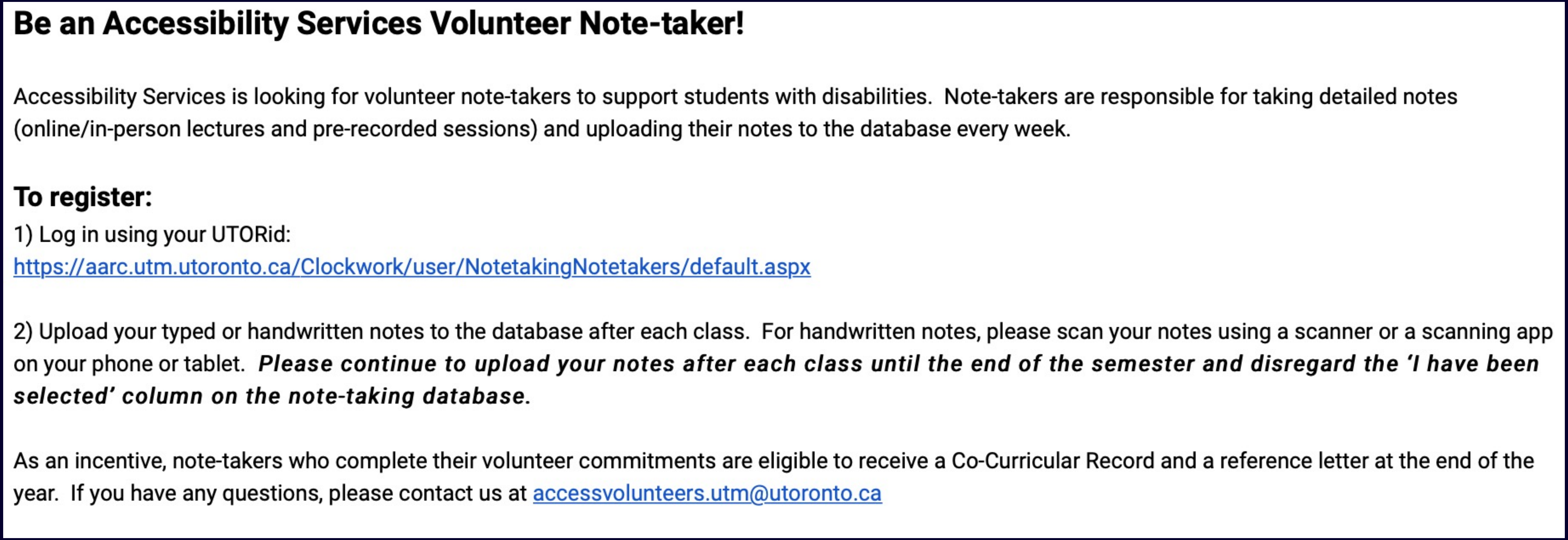
Also available on course website

Lecture Number	Class Dates	Title	Deadlines
Week 1	Jan 13-14	Intro + HTML	
Week 2	Jan 20-21	CSS	
Week 3	Jan 27-28	Django #1	A1: HTML + CSS
Week 4	Feb 3-4	Django #2	
Week 5	Feb 10-11	Django #3	P1: HTML + CSS
Week 6	Feb 17-18	Django #4	
Reading week			A2: Django
Week 7	Mar 3-4	JavaScript #1	
Week 8	Mar 10-11	JavaScript #2	P2: Django
Week 9	Mar 17-18	React #1	
Week 10	Mar 24-25	React #2	A3: JavaScript
Week 11	Mar 31-Apr 1	React #3	
Week 12	Apr 8-9	DevOps (optional)	P3: React

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

Note-taker requests



Be an Accessibility Services Volunteer Note-taker!

Accessibility Services is looking for volunteer note-takers to support students with disabilities. Note-takers are responsible for taking detailed notes (online/in-person lectures and pre-recorded sessions) and uploading their notes to the database every week.

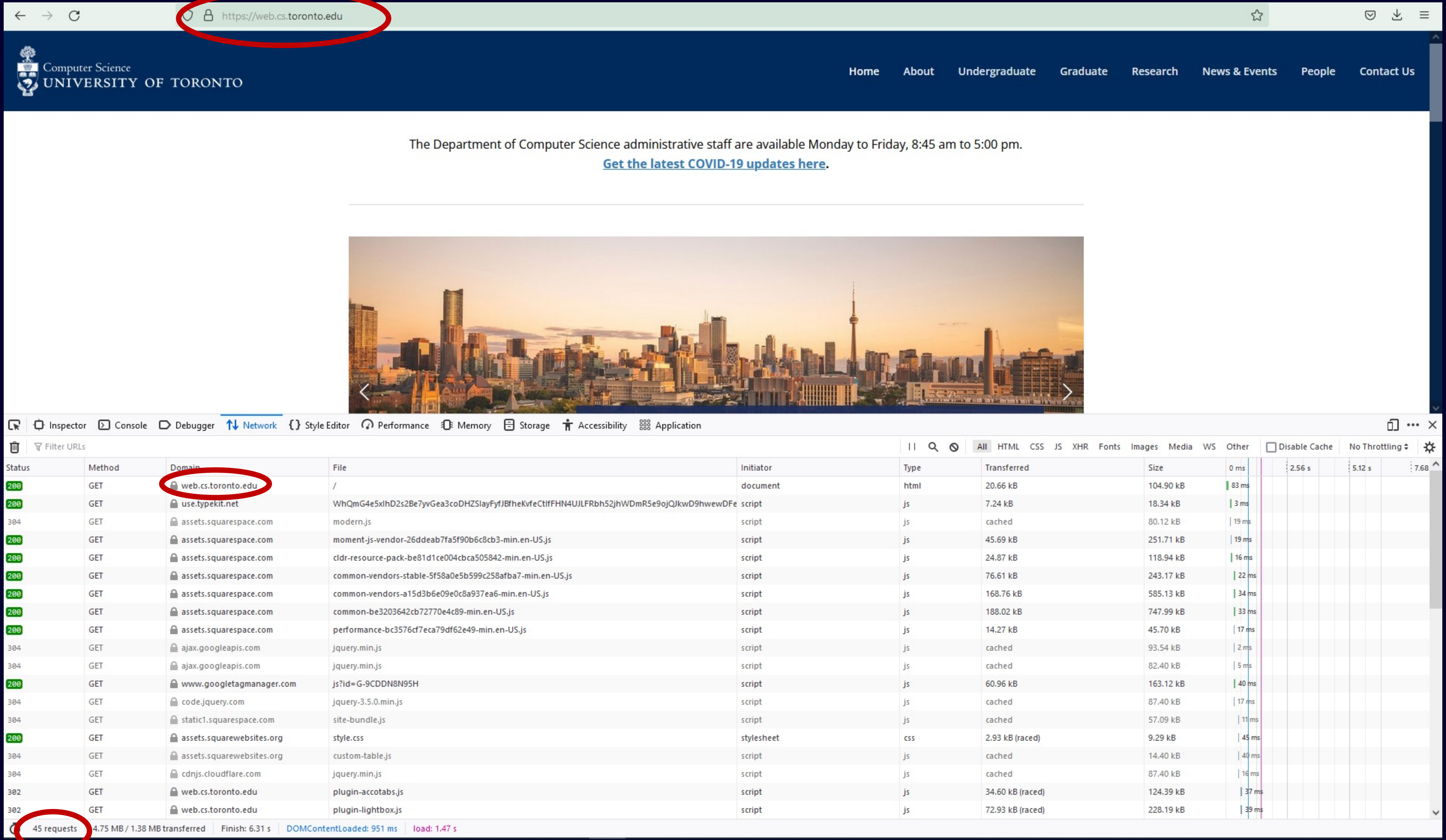
To register:
1) Log in using your UTORid:
https://aarc.utm.utoronto.ca/Clockwork/user/NotetakingNotetakers/default.aspx

2) Upload your typed or handwritten notes to the database after each class. For handwritten notes, please scan your notes using a scanner or a scanning app on your phone or tablet. ***Please continue to upload your notes after each class until the end of the semester and disregard the 'I have been selected' column on the note-taking database.***

As an incentive, note-takers who complete their volunteer commitments are eligible to receive a Co-Curricular Record and a reference letter at the end of the year. If you have any questions, please contact us at accessvolunteers.utm@utoronto.ca

Questions?

How web works


https://web.cs.toronto.edu

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.

Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms
200	GET	web.cs.toronto.edu	/	document	html	20.66 kB	104.90 kB	83 ms
200	GET	use.typekit.net	WhQmG4e5xlhD2s2Be7yvGea3coDHZSlayFyfJBfheKvfeCtlfFHN4UJLFRbh52jhWDmR5e9ojQJkwD9hwewDFe	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	cldr-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-bc3576cf7eca79df62e49-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.squarewebsites.org	style.css	stylesheet	css	2.93 kB (raced)	9.29 kB	45 ms
304	GET	assets.squarewebsites.org	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

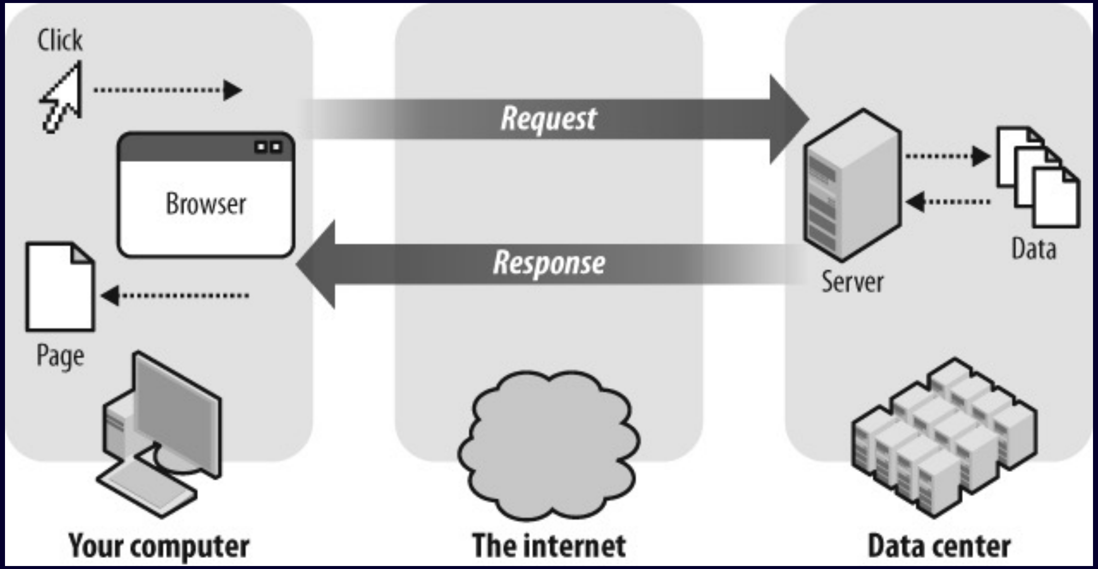
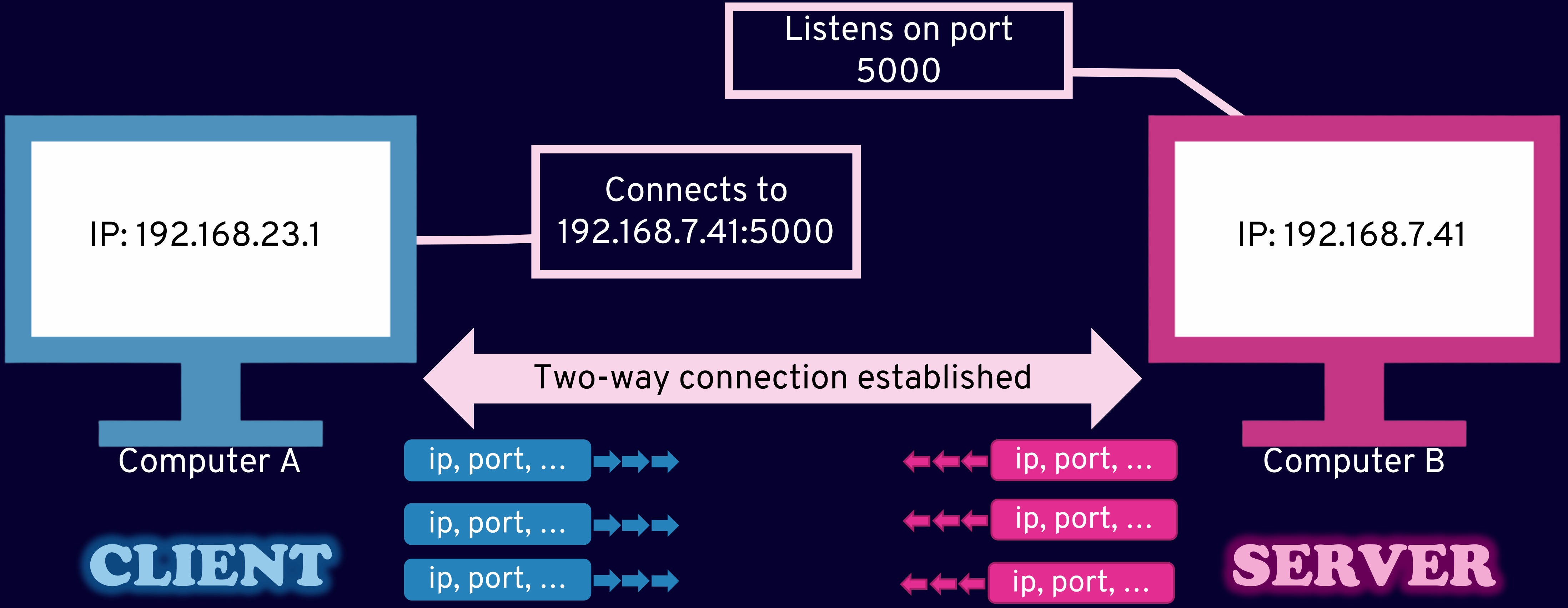


Image source: https://medium.com/@lokeshchinni123

How computers talk to each other?



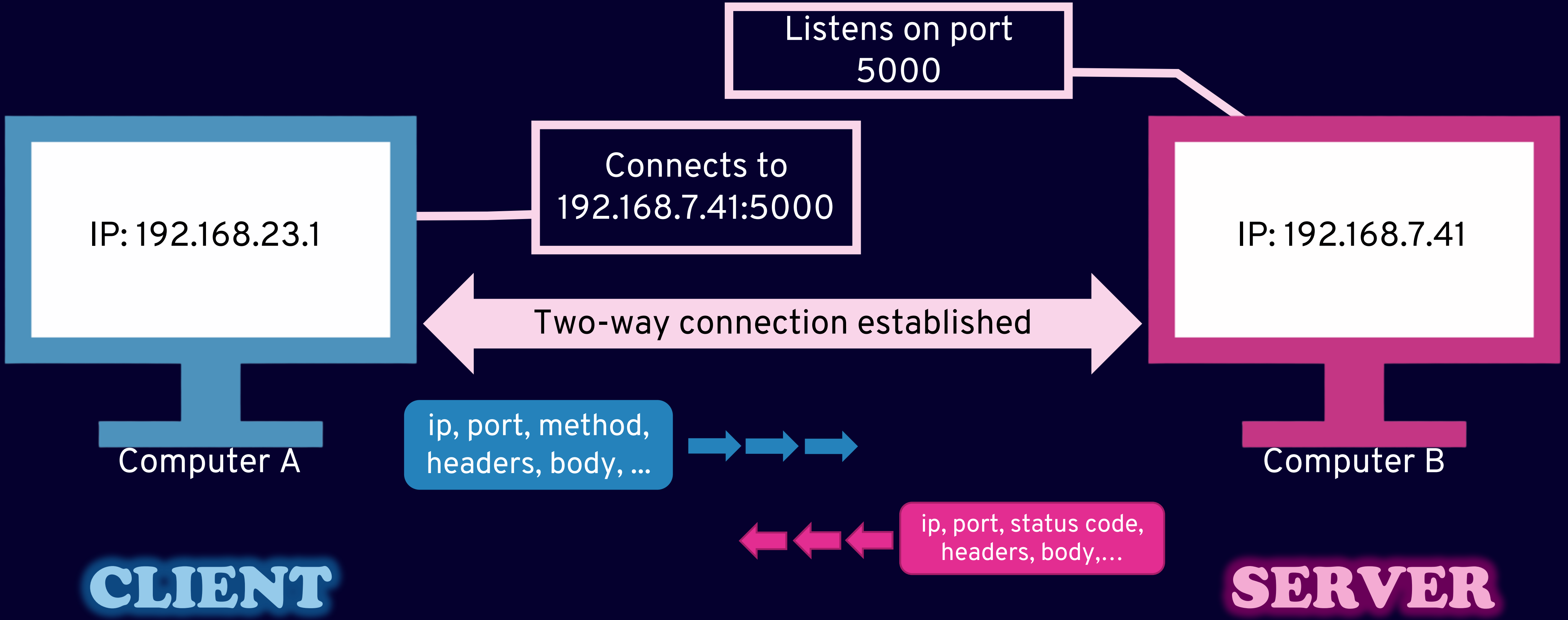
Domains

- 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


Listens on port 5000
IP: 192.168.23.1
Connects to 192.168.7.41:5000
IP: 192.168.7.41
Two-way connection established
Computer A
ip, port, method, headers, body, ...
Computer B
ip, port, status code, headers, body,...
CLIENT
SERVER

HTTP Message

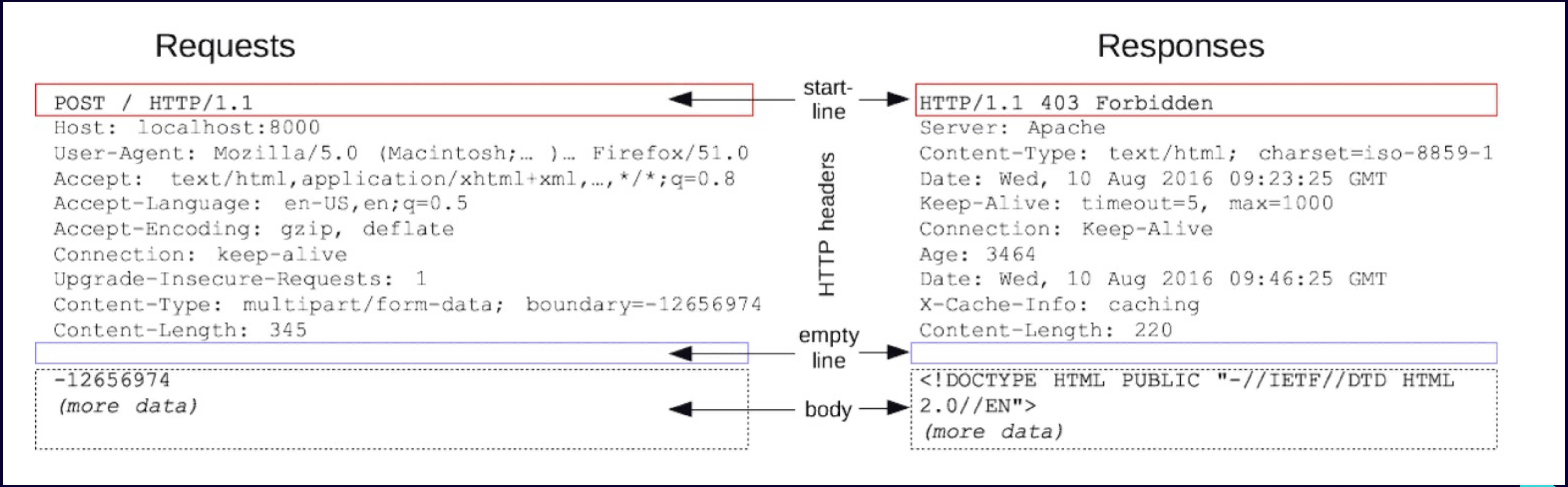
- A string with a special format

- Request a more specific target
  Path: /, /signup, /account/index.html, …
  Method: GET, POST, PUT, …

- Headers & Body

- Default port is 80

HTTP Message



```
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)
```

```
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)
```

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: <img />
- Lists: <ol> and <ul>
- Tables: <table>
- Navigation Bar: <nav>
- New line: <br />

HTML attributes

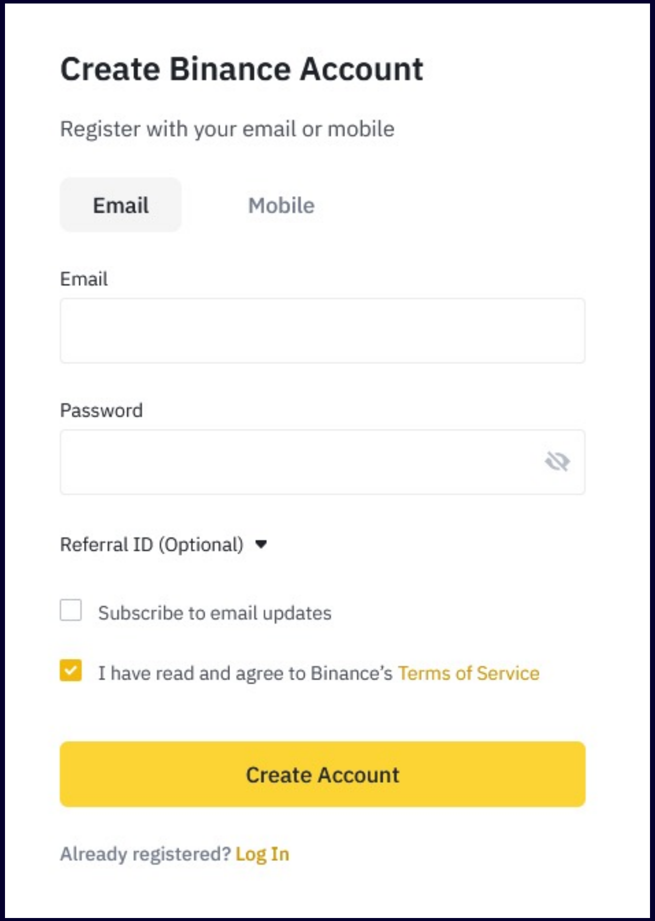
- style attribute: next session
- Identifiers: id vs class
- Other attributes: src for <img /> and href for <a>
- You can put any custom attribute you want

Other HTML tags

- <div> and <span>
- Select part of document to apply specific attributes
- <span>: 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


Create Binance Account

Register with your email or mobile

Email

Password

Referral ID (Optional) ▾

☐ Subscribe to email updates

☑ I have read and agree to Binance's Terms of Service

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

Final notes

- Join the Piazza page
  https://piazza.com/utoronto.ca/winter2022/csc309
- Start looking for teammates
- Take a look at Assignment 1
  Deadline in 2 weeks
- Attend labs
- Practice using online resources