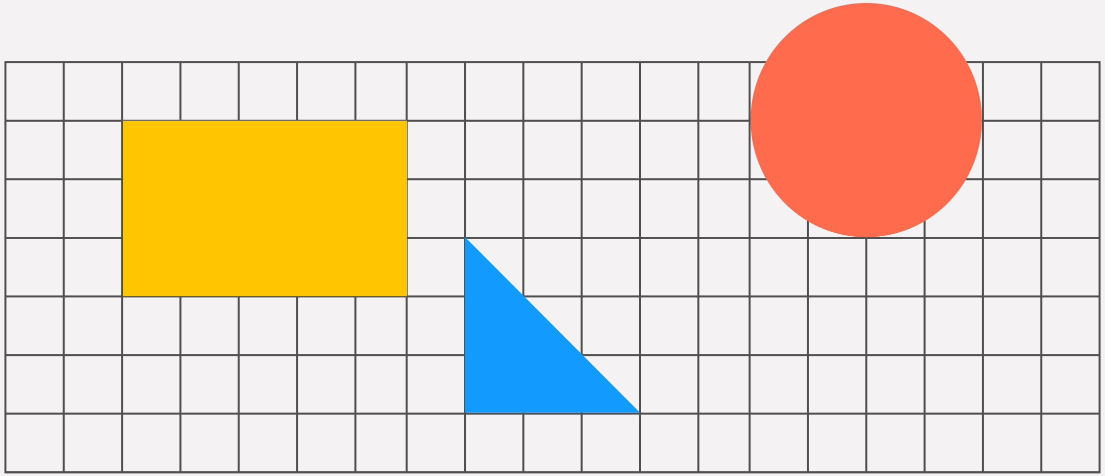
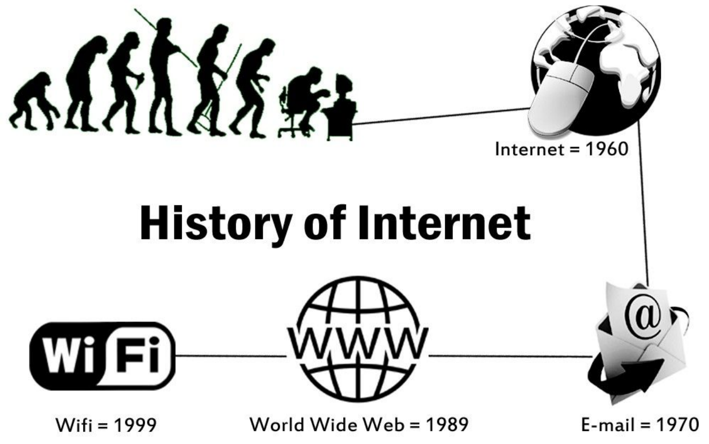
Welcome Abroad!

Ali Abrishami



Brief History of Web

- Began as a US Department of Defense network called ARPANET (1960s-70s)
- Initial services: electronic mail, file transfer
- Opened to commercial interests and most universities in late 80s
- WWW created in 1989-91 by Tim Berners-Lee
- Early web browsers released: Mosaic 1992, Netscape 1994, Internet Explorer 1995
- Amazon.com opens in 1995; Google January 1996



Source: www2014.kr

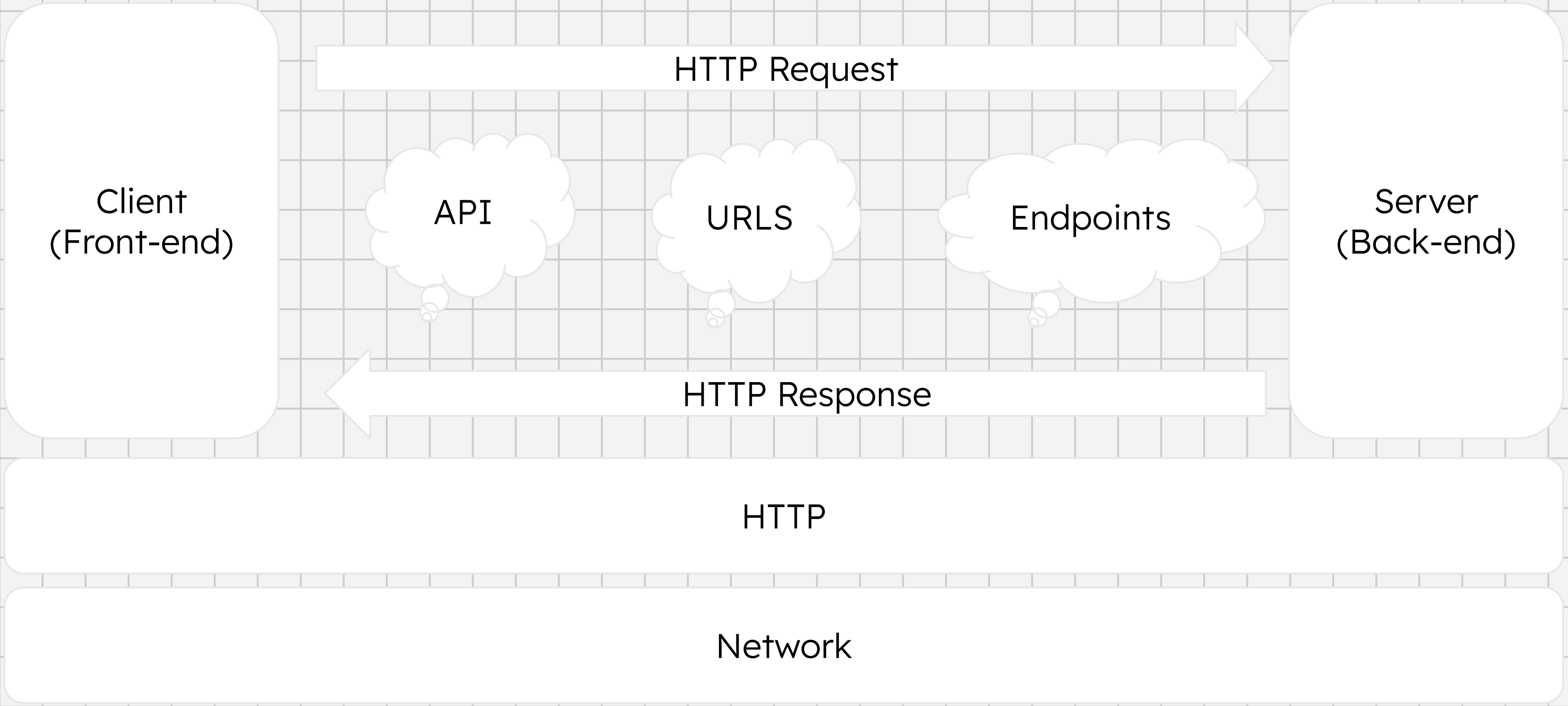
Internet vs. WWW

- The Internet: Network of networks that use the Internet protocol suite to link billions of devices worldwide
- Consists of millions of private, public, academic, business, government networks
- Networks linked together by electronic, wireless, & optical networking technologies
- Carries information resources and services, e.g. WWW

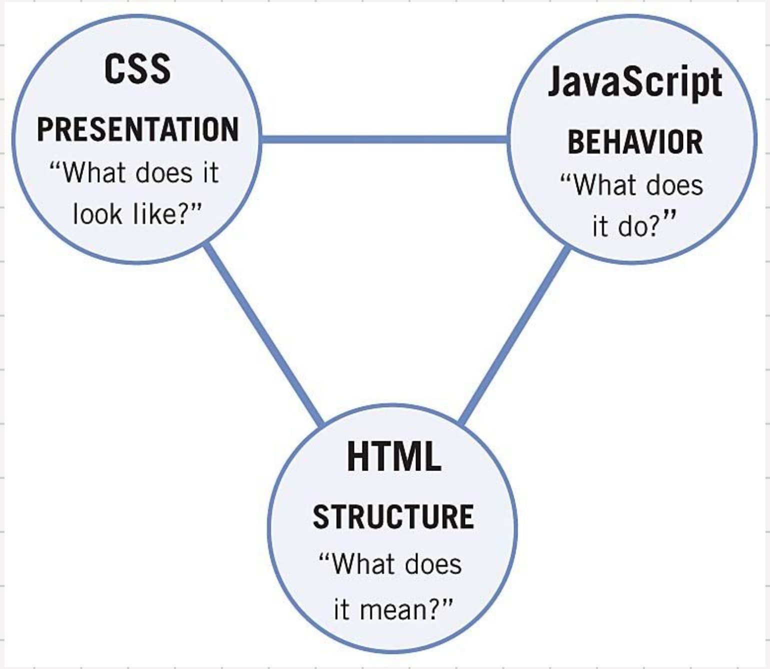
How We Use the Web Today

- Checking the weather
- Streaming movies
- Submitting requests
- The web is part of our daily life.
- Billions of people use websites every day
  - on phones
  - laptops
  - even smart fridges!

The Layers of The Web



The Holy Trinity of Web

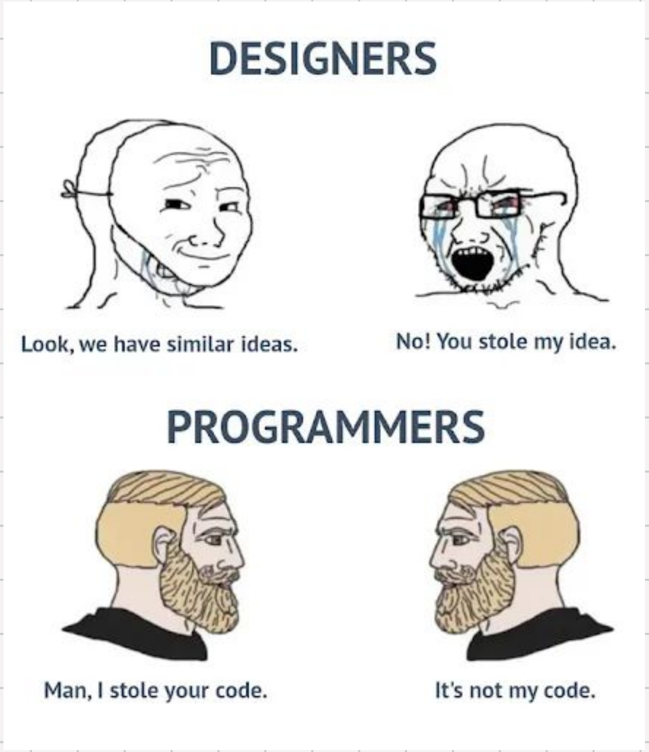


Web Architecture : A Sage

- **1990s**: **Static** Websites, **Simple HTML pages** served directly from a web server.
- **Early 2000s**: **Dynamic** Web, **Server-side rendering** with PHP, ASP, and JSP.
- **Mid 2000s**: MVC **Frameworks**, Rise of Rails, Django, and Spring MVC for **cleaner** app design.
- **2010s**: **Single-Page Apps (SPA)**, React, Angular, and Vue move **logic** to the **client** side.
- **Late 2010s**: **Serverless** & **Microservices**, Apps split into **small**, **independent services**.
- **2020s**: **Modern Hybrids**, Next.js, Remix, and Astro blend **SSR**, **SPA**, and **static generation**

Web Programming!

- It's like being an architect and a builder for the digital world.
- It's giving instructions to a computer to create experiences for people.
- It's problem-solving with creativity.



Stacks!

- A **stack** is the set of **technologies** used to build a **web app**.
- Combines **frontend**, **backend**, **database**, and **infrastructure** layers.
- Each layer handles a different part of the app (UI, logic, data, hosting).
- Choosing the right stack affects **performance**, **scalability**, and **workflow**.
- **Full-stack developers** work across **all layers** of the stack.

Just Kidding :)

- Front-End Developer: Build interactive and user-friendly web interfaces using HTML, CSS, JavaScript, and frameworks like React or Vue.
- Back-End Developer: Develop server-side logic, APIs, and database integrations using languages like Python, Node.js, Golang, or .NET.
- Full-Stack Developer: Combine front-end and back-end skills to deliver complete web solutions.
- Web DevOps & Cloud Engineer: Manage deployments, CI/CD pipelines, and scalable cloud hosting for web applications.

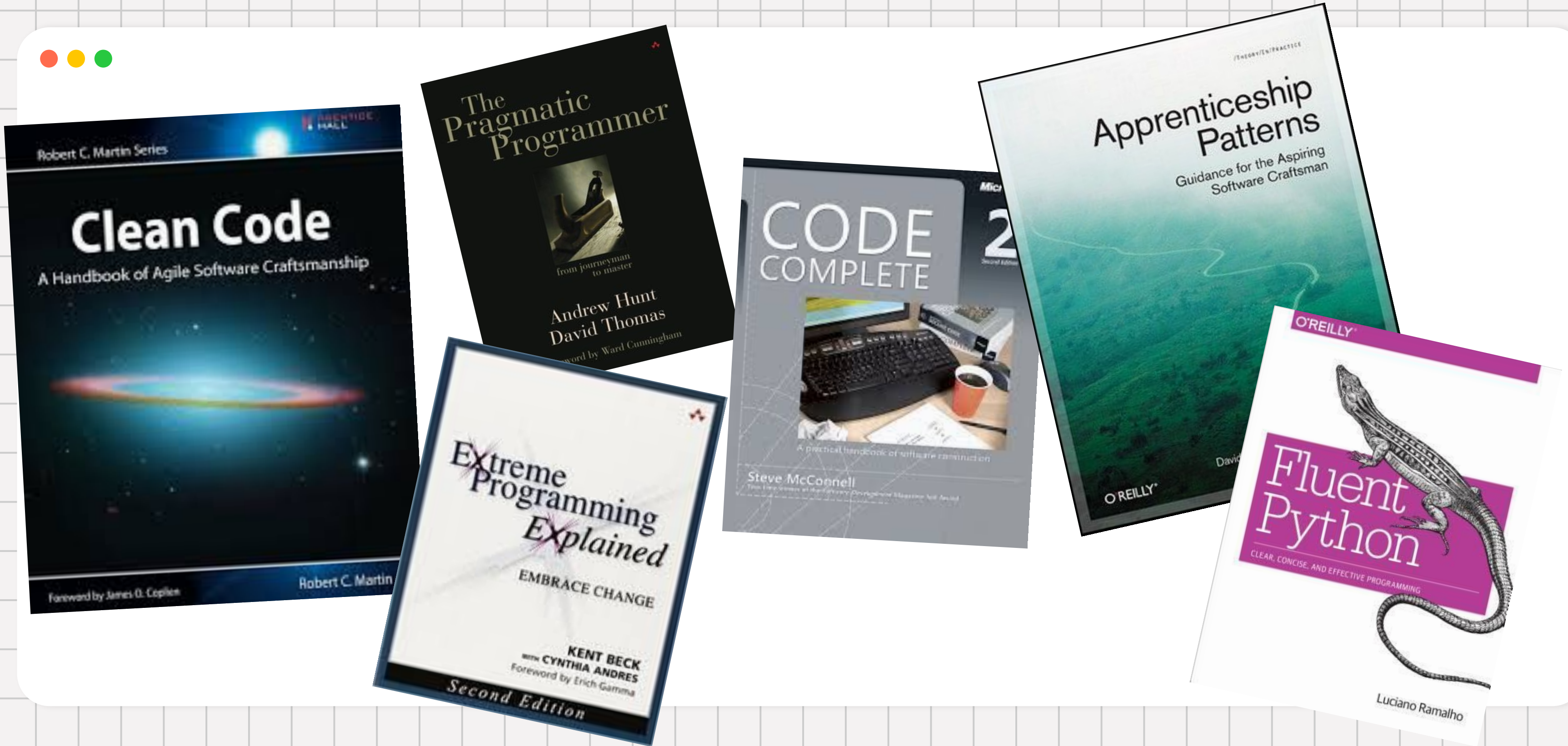
Job Opportunities

NONE!

Importance of Software Engineering

- **Clean Code Principles:** Ensures readability, consistency, and collaboration across teams.
- **Maintainable Code:** Easier debugging, feature additions, and long-term scalability.
- **Dead Code Removal:** Eliminates unused, redundant logic to reduce complexity.
- **Fail Fast Approach:** Detects errors early, improving reliability and lowering costs.

Some Important Books

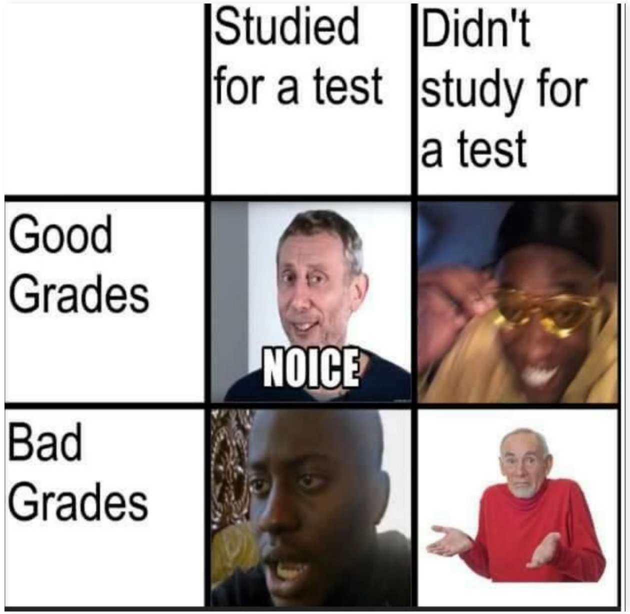


Your Turn!

QUESTIONS SO FAR?

Course Details

- Course number: 40419
- Official Title: Web Programming
- Grading:
  - Exercises: 4 pts + Extra
  - Project: 8 pts + Extra
  - Exams: 8 pts



source: reddit

The Syllabus

- Static web
    - HTML, and CSS, and JavaScript
- Web programming architecture
- Backend programming
    - Python, Django, and DRF
- Frontend programming
    - Node JS, TypeScript, and React
- DevOps
    - Docker

Course Policies

- Use of LLMs: It's free, as long as you understand the code!
- Delay policy:
  - 1st 24 hours: -1%/hr
  - Day 2: 75%
  - Day 3: 50%
- Total of 7 days of delay with no penalty.
- **https://quera.org/course/add_to_course/course/23975/**

LLM Best Practices

- **Define clear goals**: Know exactly what you want the model to achieve before prompting.
- **Prompt precisely**: Use clear, specific, and contextual instructions.
- **Iterate & refine**: Test, review, and adjust prompts or parameters for consistency.
- **Keep context short & relevant**: Provide only necessary information to reduce confusion.
- **Validate outputs**: Always fact-check and post-process model responses.
- **Respect safety & ethics**: Ensure responsible AI use
- **Note:** Thank AI after each conversation, just in case of an AI takeover :)

About The Instructor

- Instructor: Ali Abrishami
- Contact: a.abrishami110@gmail.com
- Room 601, Sunday and Tuesday
- Contact before meeting!

Teaching Assistants

- Mani Ebrahimi: `mani.ebra@gmail.com`
- Mahdi Jafari
- Seyyed Amir Mohammad Jazayeri
- Mahsa Hajirahimi
- Amirreza Khanari

Course References

- S. M. Schafer. HTML, XHTML, and CSS Bible. 5th Edition, Wiley Publishing, 2010.
- J. Forcier, P. Bissex, and W. Chun. Python Web Development with Django. Pearson Addison-Wesley, 2009.
- W. J. Chun. Core Python Applications Programming. 3rd Edition, Pearson Addison-Wesley, 2012.
- M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, and R. Stafford. Patterns of Enterprise Application Architecture. Pearson Addison-Wesley, 2003.

THIS Course References

- Jonas Schmedtmann - Front-end Development Courses - Udemy
- Django Documentation
- Django Rest Framework Documentation

Your Turn!

QUESTIONS?