The Internet and the Web

Introducing web development

CPIT-405 Web Applications

Khalid Alharbi, PhD

Last updated: Spring 25

Table of Contents

Course Website

https://cpit405.gitlab.io/ ↗

- Check the course website every day! (I'm serious)
- To make it easy for you to track topics and schedule:
    - Go to the calendar
    - Always do the readings before the class
    - The current/active week is highlighted differently
- Lecture slides are also listed on the course website
    - The slides are rendered directly in the browser
    - Slides also include code examples through interactive live code editors
    - They are just websites, so no special software is needed.

Course Road Map

1) Web Technologies
- The Internet and the web
- Layer 7 (application layer)
- Web servers
- HTTP Protocol
  - Status codes
- Client Server architecture
- Client

4) DOM
- DOM Tree elements
- Traversing the DOM tree
- Create and remove DOM elements in JS
- Change the style of an element in JS
- document.getElementById
- document.getElementsByTagName
- document.getElementsByName
- document.querySelector()
- document.querySelectorAll()

2) HTML and CSS
- HTML elements
- Internal Bookmark links
- HTML forms

- CSS selectors and properties
- CSS units
- CSS Box Model
- Inline vs Block level elements
- Positioning
- Layouts

5) Ajax and JSON
- JSON.stringify()
- JSON.parse()
- XMLHttpRequest
- Fetch API with promises
- Fetch API with async/await

3) JavaScript
- Variables
  -> (var, let, const)
- Iterations
  -> for loop (for, forEach(), for... in, for... of)
  -> while loops (while, do...while)
- Conditional statements:
  -> if, if...else
  -> switch...case
- functions: declaration and invocation
- Callback functions
- Objects
- Arrays
- Date
- Math
- String
- Mouse and Keyboard Events
  -> Registering an event in HTML
  -> Registering and event in JS
  -> .addEventListener() vs. .on...()
  -> Event bubbling and event.stopPropagation()
- Cookies
  -> create and read a cookie in JS

6) React
- create-react-app
- JSX syntax
- React components
- Handling events
- Fetching data in React
- Props
- Hooks: useState()

7) PHP
- Data types
- Arrays
- local vs global variables
- Super-globals
- String concatenation
- Conditional statements
- Iterations
- echo
- Working with forms
- Form validation: client side vs server side
- Connecting to a database using mysqli and pdo
- Performing CRUD operations: insert, select, update, and deleted
- Prepared statements: create, bind, and execute
- Protection against SQL injection

Course Syllabus

https://cpit405.gitlab.io/syllabus/ ↗

- Read it! It serves as the contract between us.
- Be prepared, there will be a quiz on the syllabus in the next lecture.

Teaching Philosophy

- Use MS Teams for communication and assignment submissions

- I'm here for you!

  - I want you to participate, interrupt me when you have a question, and ask me to slow down if I'm going too fast!

  - Every question is important

  - Help each other

  - Web development can be incredibly rewarding and engaging

  - Need help after class? DM me on Teams

- I will be coding in the lectures

  - You will be asked to apply what you've learned in the class

- Coding is like cooking. You won't learn by just watching, learn by doing!

Late Policy, Use of AI tools, and Special Accommodations

- Assignments submitted late will incur a 25% penalty.

- You may submit an assignment or a lab activity up to one week late.

  - after that the submission will not be graded and you'll receive 0 points for it.

  - **Exception**: Every student gets three free late passes, allowing you to submit a maximum of 3 assignments up to 1 week past the due date without penalty.

- If you discover that you cannot attend the midterm or final exam, you will need to provide me with relevant documentation ASAP before the midterm or final exam to make other arrangements.

- Read the syllabus statement on the use of generative AI tools ↗

- Disability Accommodations:

  - Please be open in communicating any requests for accommodations due to disability no later than the second week of the semester.

The Internet and the Web

What is the Internet?

- The Internet is a large interconnected computer networks using the Internet protocol suite (TCP/IP).

- It's a network of networks that enables communication and data exchange between devices

- The Internet is the infrastructure for global communication

    - Internet Applications:

        - Email

        - File Transfer Protocol (FTP)

        - Instant messaging

        - Streaming services (audio/video)

        - DNS (Domain Name System) servers

        - and many more

Internet History

- In the 1960s, computers were large and require physical access to execute tasks.
- The U.S. Defense Department funded projects to enable communication in response to the Soviet Union's advanced projects.
- The US Department of Defense created an agency called the ARPANET (Advanced Research Projects Agency Network), the network that ultimately evolved into what we now know as the Internet.
- **In the 1970s**, a new communications protocol was established called Transfer Control Protocol/Internetwork Protocol (TCP/IP)
- TCP/IP was adopted as the protocol standard for ARPANET (the predecessor to the Internet) in 1983.
- TCP/IP enabled different kinds of computers on different networks to communicate with each other.
- A psychologist and computer scientist named Joseph Licklider, created a network designed for communication during the Cold War.
- **1962**: The concept of a network of interconnected computers was first proposed by J.C.R. Licklider.

Internet and Web History Timeline

- **1969**: ARPANET, the precursor to the Internet, was launched by the U.S. Department of Defense.
- **1974**: The term "Internet" was first used to refer to a global network of networks.
- **Late 1970s**: TCP/IP, the foundational protocol of the Internet was developed.
- **January 1, 1983**: is often considered the official birthday of the Internet.
- **1989**: The World Wide Web, a system of interlinked hypertext documents, was invented by Tim Berners-Lee.
- **1993**: The first web browser, Mosaic, was released, making the Internet more accessible to the public.
- **late 1990s and early 2000s**: The dot-com boom led to rapid growth and commercialization of the Internet.
- **mid-2000s**: The advent of social media changed the way people use the Internet.
- **Today**: The Internet is an integral part of everyday life, used for communication, commerce, entertainment, and more.

What is the web?

- The World Wide Web (WWW or the web) is an information system that enables the sharing of documents and other web content over the Internet using the Hypetext Transfer protocol (HTTP).
- Openness and accessibility: The web provides user-friendly access to information on the internet
  - Relies on web browsers for display
  - Offers content in various formats (text, images, videos)
  - Uses hyperlinks for navigation
- Web applications
  - News websites
  - Shopping websites,
  - Social media platforms (e.g., Facebook, Reddit, TikTok etc.)
  - Productivity tools (e.g., Google Docs, Sheets, and Slides)
  - Learning platforms
  - and many more

Protocols of the Web

- Protocols dictate how browsers and servers talk, enabling data transfer and website rendering
- It's very important for web developers to understand web related protocols and the inner workings of the web and internet
- Understanding protocols and how the web works will help you build efficient, secure, and usable web applications

The OSI model

Layer	Name	Description	Protocols
7	Application	Top Level APIs and provides functions for applications.	FTP, HTTP, SMTP
6	Presentation	De/Encryption, Encoding, String representation	AFP, MIME
5	Session	Build and control sessions	X.225, SCP
4	Transport	End-to-end connections, reliability and flow control	TCP, UDP
3	Network	Connection model, host addressing, and message forwarding	IPv4/v6; RIP; QoS
2	Data Link	Physical addressing, Logical link control (error & flow control)	ARP, PPP
1	Physical	Physical networking used to send and receive signals	USB, Bluetooth

The TCP/IP Suite

Name	Description	Protocols
Application Layer	provides direct service to users, which include Telnet (remote login), FTP (file transfer), and SMTP (electronic mail delivery).	FTP, HTTP, SMTP
Transport Layers	provides a reliable end-to-end communication or data transfer for applications.	TCP and UDP
Internet Layer	provides the addressing or the routing of a sent data to its destination	IP, ARP, ICMP
Link Layer	provides interface to networking hardware such as the ethernet or gateways.	Ethernet, PPP,

OSI model and TCP/IP Suite

- The OSI model is a theoretical, conceptual, seven-layer reference model.

- While the OSI model is valuable for educational purposes and understanding network principles, it's not directly implemented in real-world networks.

- The TCP/IP suite is a concrete implementation of some of the concepts outlined in the conceptual framework defined in the OSI model.

- TCP/IP is protocol specific. It defines specific protocols for each layer.

- TCP/IP is the dominant protocol suite used for internet communication, including web browsing, email, and online gaming

The Hypertext Transfer Protocol (HTTP)

- The Hypertext Transfer Protocol (HTTP) is the foundation of communication between web browsers and servers on the World Wide Web.
- An HTTP request is what a web browser uses to ask for a resource or an information necessary for rendering a web page.
    - This means the server does not store any information about the client between different requests.
- HTTP is a "stateless" protocol, which means that each HTTP request runs independent of any other requests.
- A typical HTTP request contains:
    - HTTP version type
    - URL
    - HTTP method
    - HTTP request headers
    - Optional HTTP body.

HTTP Version

- HTTP Version (e.g., HTTP/1.1, HTTP/2, HTTP/3) specifies the version of the Hypertext Transfer Protocol being used.
- **HTTP/1.1**: Relies on a single TCP connection per request, leading to sequential communication.
  - **Limited Multiplexing**: Can handle multiple requests on a single connection, but not efficiently.
- **HTTP/2**: It extended the usage of persistent connections by multiplexing many concurrent requests/responses through a single TCP/IP connection.
  - Released in 2015 to address the limitations of HTTP/1.1.
  - Uses a more efficient binary format for data transfer compared to plain text in HTTP/1.1.
  - Header Compression: Reduces data transmission by compressing request and response headers.
- **HTTP/3**: It uses QUIC, a multiplexed transport protocol based on UDP, instead of TCP.
  - HTTP/3 was introduced in 2022 and is implemented as standard in all major Web browsers.
  - It aims to improve security with built-in encryption.
  - It addresses issues related to performance when a smartphone switches from one network to another

URL and URI

- URL stands for Uniform Resource Locator. It's a reference (an address) to a resource on the internet.

  - For example, "https://www.google.com" is a URL that specifies the location of Google's website and uses the HTTPS protocol to access it.

- URI (Uniform Resource Identifier): A broader concept that includes URLs and other identifiers used to name resources.

  - For example, "mailto:example@example.com" is a URI because it identifies an email address as a resource but it does not specify the location of a resource on the internet.

URL Components

- **Protocol**: Examples: http, https, ftp, and file.

- **Domain Name**: human-readable name (e.g., example.com)

- **Port**: the port number on the server to which the client should connect. For example, 80 for HTTP and 443 for HTTPS as in https://google.com:443.

- **Path**: The specific resource path on the server. For example: https://www.google.com/favicon.ico

- **Query String**: This is used to include additional data to be sent to the server and starts with `?` . For example, https://www.google.com/search?q=cat

- **Fragment**: This is used to specify a specific HTML element id in the web page followed by a hash (#). For example, https://en.wikipedia.org/wiki/World_Wide_Web#History

HTTP Methods (I)

- HTTP method is a verb used in an HTTP request to indicate the action to be performed by the server on the identified resource URL.

    - GET, POST, PUT, PATH, DELETE, OPTIONS, HEAD

- We will use the JSONPlaceholder API ↗ to demonstrate HTTP methods and working with APIs using Postman ↗ .

    - It is a simple/fake RESTful API that developers can use for testing and learning how to consume APIs.

    - It provides endpoints for typical HTTP methods like GET, POST, PUT, and DELETE.

HTTP Methods (II)

Method	Description	JSONPlaceholder API Example
GET	Retrieves data from a resource.	`GET https://jsonplaceholder.typicode.com/posts/1`
POST	Creates a new resource.	`POST https://jsonplaceholder.typicode.com/posts` . The body should contain the new resource.
PUT	Updates an existing resource completely.	`PUT https://jsonplaceholder.typicode.com/posts/1` . The body should contain the updated resource fields.
DELETE	Deletes a resource.	`DELETE https://jsonplaceholder.typicode.com/posts/1`
PATCH	Updates an existing resource partially.	`PATCH https://jsonplaceholder.typicode.com/posts/1` . The body should contain the updated resource fields.

HTTP headers

- HTTP request headers are extra lines of data included in a request by the client (such as a web browser)
- HTTP response headers are extra lines of data included in a response by the server.
- Their purpose is to provide additional information for context and instructions enabling the server or client to understand the request or response better.
- Common HTTP headers:
  - `Authorization` : Contains the credentials to authenticate a user agent with a server.
  - `Content-Type` : Indicates the media type of the resource in the body of the request.
  - `User-Agent` : Contains information about application type, operating system of the requesting browser of the user.
  - `Cookie` : Contains stored HTTP cookies previously sent by the server with the Set-Cookie header.

HTTP body

- The HTTP body is the part of an HTTP request or response that contains the actual data to be sent or received

- In a request, the body may contain data that the client wants to send to the server. Example: login form data.

- In a response, the body may contain data that the server has sent to the client. Example: the HTML of the requested page.

- The format of the data in the body can vary. It could be text, JSON, XML, binary data like an image, or any other format that the client and server agree to use.

- Not all HTTP requests have a body. It depends on the API server and how they accept a request or respond to a request. The Content-Type header is often used to indicate the format of the data in the body. Example: `image/jpeg` and `application/json`

HTTP Status Codes (I)

- An HTTP status code is a server's response to a client's request sent over the HTTP protocol.
- These status codes are three-digit numbers where the first digit defines the class of the response.
- **1xx (Informational)**: The request was received, and the process is continuing.
- **2xx (Successful)**: The request was successfully received, understood, and accepted. Example: 200 OK.
- **3xx (Redirection)**: Further action must be taken to complete the request. Example: 301 Moved Permanently.
- **4xx (Client Error)**: The request contains bad syntax or cannot be fulfilled. Example: 404 Not Found.
- **5xx (Server Error)**: The server failed to fulfill an apparently valid request. Example: 500 Internal Server Error.

HTTP Status Codes (II)

Status Code	Class	Description	Example
200	Successful	OK	Generic successful response
201	Successful	Created	Successfully created a new item in the app or database
204	Successful	No Content	Successful but there is no additional content to send in the response payload body.
301	Redirection	Moved Permanently	A resource moved from 'public/images' to 'static/images' permanently
302	Redirection	Moved Temporarily	A resource moved from 'public/images' to 'static/images' temporarily and the response is being redirected to the temporarily path

HTTP Status Codes (III)

Status Code	Class	Description	Example
400	Client Error	Bad Request	The request is missing some required data (e.g., missing URL params, header or body)
401	Client Error	Unauthorized typically for failure in authentication	Tried to access a resource that requires authentication
403	Client Error	Forbidden. The server understood the request, but it refuses to authorize it.	Tried to access a resource that you'r not authorized to see due to application specific logic (e.g., not admin)
404	Client Error	Not Found	Tried to access a resource that does not exist on the server

HTTP Status Codes (IV)

Status Code	Class	Description	Example
500	Server Error	Internal Server Error	The application crashes in a runtime error
502	Server Error	Bad Gateway.	When making a request to the web application, which in turn makes another request to the third-party API. If the third-party API is down or returns an invalid response, the web application might return a 502.
503	Server Error	Service Unavailable	The server is temporarily unable to handle a request during heavy traffic or an internal server not responding and result in a timeout.

HTTP RestFULL API Demo

Let's see a demo of working with a RESTful API that uses HTTP requests to GET, PUT, POST, and DELETE data using Postman.

How the Internet and the Web work?

Demystifying the Internet and the Web through Hind's Shoe Shopping Journey

Hind's Story (I)

Hind would like to purchase shoes from Amazon.

1. **URL (Uniform Resource Locator)**: She opens her web browser (e.g., Chrome).
   - In the address bar, she types the URL, *https://amazon.com*.
   - URL (Uniform Resource Locator) is the human-readable address.
2. **DNS (Domain Name System)**: The browser uses DNS behind the scenes.
   - The browser sends a DNS query to a DNS server, which is often via UDP on port 53.
   - DNS acts like the phone book for websites, translating the domain in the URL into a numerical IP address.
   - The DNS server responds with the IP address of Amazon's web server.
   - Now the browser knows where to send its HTTP/HTTPS request to fetch the home page of the online store Hind trying to access.

Hind's Story (II)

3. **Establishing a TCP connection**: The browser establishes a TCP connection with Amazon's web server to the resolved IP address on port 443 (default for HTTPS).

   - TCP/IP ensures reliable data transmission by breaking information into packets and reassembling them correctly at the other end.

   - The browser initiates the TCP session with Amazon's **web server** using three-way handshake (SYN, SYN-ACK, ACK).

     - The web server is the system program running on Amazon's physical server to deliver content to users.

   - At this stage, the connection exists at the TCP layer,

Hind's Story (III)

4. **HTTPS (TLS Handshake)**: Since the website uses https, the browser establishes a secure connection with the web server using TLS (Transport Layer Security).

- HTTPS = HTTP over TLS.

- TLS handshake takes place between the web server and client (web browser).

- The browser initiates the TLS handshake by sending a request to the server.

- The server responds with a message that includes protocol version, session ID, and digital certificate, which contains the server's public key.

- The browser verifies the server's certificate with the **certificate authority (CA)** to ensure it's valid and trustworthy.

- If the CA is valid, the browser generates a session key and encrypts it using the server's public key from the certificate.

- The server uses its private key to decrypt the session key sent by the browser.

- The server and the browser now have a shared symmetric encryption key which they will use to encrypt and decrypt data exchanged during the secure session.

Hind's Story (IV)

5. **HTTP Request**:

- HTTP (Hypertext Transfer Protocol) is a protocol used for transmitting hypertext (i.e., "web pages")
- The browser sends an HTTP GET request to the server, asking for the webpage associated with the URL (homepage).
  - An HTTP request includes an HTTP method (verb) which tells the server what kind of request it is (e.g., GET, POST, PUT, DELETE, etc.).
  - The request may include additional headers (e.g., `User-Agent`) to provide the server with additional information about the client making the request.

Hind's Story (V)

6. **HTTP Response**:

- The web server processes the request and responds with an HTTP status code, headers, and body payload with the appropriate HTML, CSS, and JavaScript files.
    - The HTML provides the structure of the page
    - The CSS styles the page
    - The JavaScript makes the page interactive.
    - The browser then starts rendering the payload into a webpage.
    - Now, Hind can interact with the web page and buy shoes 👠 .

Hind's Story (VI)

7. **HTTP Requests**

- As Hind interacts with the webpage (clicking on different links), her browser sends additional HTTP requests to the server, which responds back with the appropriate response headers and payload.
- Hind's browser may keep the TCP connection open to improve performance and reduce the overhead of establishing a new TCP and TLS handshakes.
- If the browser or the server is idle for a certain period of time, either side may decide to close the TCP/TLS connection to save resources.

Static and Dynamic Web Sites

Static Websites

- Static websites deliver the same content to every user, exactly as it's stored. Example: our course website: https://cpit405.gitlab.io.
- They are typically built using HTML, CSS, and JavaScript without any server-side processing for content generation.
- They are faster and cheaper to host because they require less server resources.
- They are best suited for content that doesn't change frequently.

Dynamic Websites

- Dynamic websites generate content in real-time based on various factors like user input, time, location, etc. Example: news websites
- They use server-side languages like PHP, Java, Node.js, Python, etc. to generate HTML content.
- They are more complex in terms of content creation and personalization.
- They require more server resources and are more complex to host and manage.
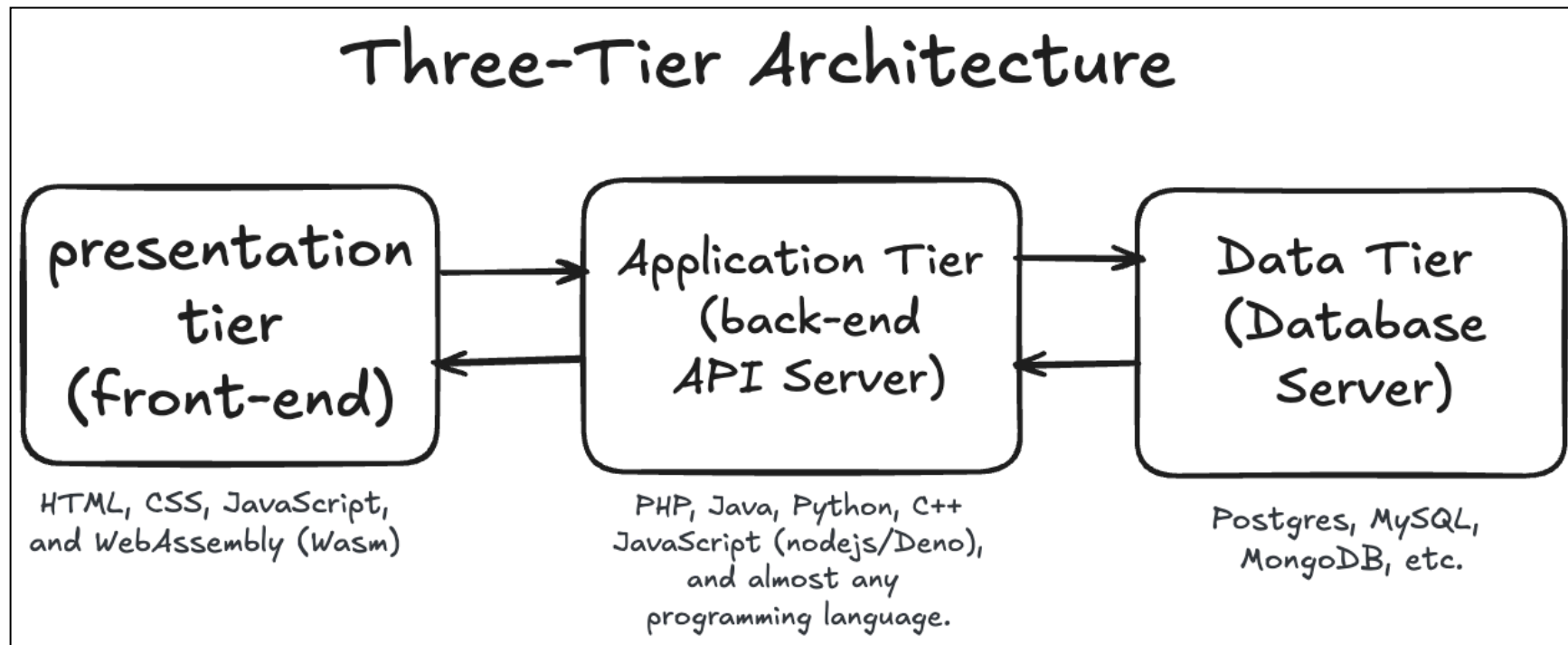
Web Application Architecture

- Web Application Architecture refers to the interaction between web applications, databases, and middleware systems (e.g., API servers).
- It provides the organization that makes up all the interactions between the components of a web application.
- Components of Web Application Architecture
  - **Client-side (Frontend)**: This involves using languages like HTML, CSS, and JavaScript to create a responsive interface.
  - **Server-side (Backend)**: This involves processing requests, performing operations on the database server or file system, and sending responses. Languages like Python, Java, and PHP are often used here.
  - **Database/Datastore**: This is where data is persistent. It can be SQL database server (like MySQL, PostgreSQL) or NoSQL database server (like MongoDB, Firebase).

Commonly used Web Application Architecture
The three-tier architecture

- The three-tier architecture is a well-established software application architecture that organizes applications into three tiers:

    - **Presentation Tier (front-end)**: This is the front-end layer of the architecture, which is visible to the user. It's typically built using HTML, CSS, and JavaScript.

    - **Application Tier (back-end)**: This is the middle layer that handles the business logic of the application. It's typically built using server-side languages like PHP, Java, Node.js, Python, etc.

    - **Data Tier**: This is the data store layer that stores and retrieves data. This layer interacts with the database using CRUD operations (Create, Read, Update, Delete). It often involves a database management system like MySQL, PostgreSQL, or MongoDB.

The three-tier architecture

Three-Tier Architecture

presentation tier (front-end)

HTML, CSS, JavaScript, and WebAssembly (Wasm)

Application Tier (back-end API Server)

PHP, Java, Python, C++ JavaScript (nodejs/Deno), and almost any programming language.

Data Tier (Database Server)

Postgres, MySQL, MongoDB, etc.

Web Technologies

- **HTML**: The markup language for creating web pages. It provides the structure of a webpage.
- **CSS**: A stylesheet language used for styling of a webpage.
- **JavaScript**: A high-level, interpreted programming language provides an interactive web experience.
- **HTTP/HTTPS**: The protocol over which data is sent between your browser and the website that you're connected to.
- **JSON/XML**: Data formats used for exchanging data between the client and the server. JSON is modern and considered the defacto data exchange format for the web.

Web Tools

- **Text Editor/IDE**: Code editors like Visual Studio Code for writing and editing code.
- **Version Control Systems**: Tools like Git for tracking changes in source code during web development.
- **Web Browsers Dev Tools**: All modern web browsers come equipped with a robust set of tools for developers. These tools offer a variety of functionalities, including the ability to examine the HTML, CSS, and JavaScript currently loaded on a page, as well as getting performance metrics.
    - Examples: Chrome Dev tools and Firefox DevTools
- **Web Servers**: Tools like VS Code Live Server for running a local development server, and Apache and Nginx for serving production ready web applications.
- **Deployment Platforms**: GitHub pages for hosting static web pages, AWS, Google Cloud, Microsoft Azure, or Heroku for deploying dynamic web applications.

Frontend vs Backend vs Full Stack Development

- **Frontend Development**: Involves creating the user interface with HTML, CSS, and JavaScript.

- **Backend Development**: Handles server-side programming and database management.

- **Full Stack Development**: Combines both frontend and backend development skills.

Wrap up and what's next

- We have seen how the Internet and the Web work

- We have learned about the following topics:

  - Course Overview

  - Introduction to the Internet

  - The Web and the Internet

  - Protocols of the Web

  - Static and Dynamic Web Sites

  - Web Application Architecture

  - Web Technologies and Tools

  - Frontend, backend, and full stack web development

- Coming up next: Use HTML to structure your web pages 🔗 ! 🤩