Apps

what is an app?

- computer software or a program most commonly a small specific one used for mobile devices. The term app originally refered to any mobile or desktop application, but as more app stores have emerged to sell mobile apps to smartphone and tablet users, the term app has evolved to refer to small programs that can be downloaded and installed at once - techopedia
- a program package to solve a problem
- example - firefox, instagram, vs code, amazon, chrome, safari, twitter, word, terminal emulator etc
- **desktop applications** -
    - usually standalone, editors, word processors, web browser, mail etc
    - often work offline, local data storage, possible network connection
    - software development kits (SDK) - custom frameworks, OS specific
- **mobile apps** -
    - targetted at mobile platforms - phone, tablets
    - constraints -
        - limited screen space
        - user interaction (touch, audio, camera)
        - memory processing power
        - battery
    - framework -
        - OS specific (android studio, xcode)
        - cross-platform (flutter, react native)
    - Network -
        - usually network oriented
    - cocoa touch - apple specific framework
- **web apps** -
    - the **platform**
    - works across OS device, create a common base
    - heavily network oriented, mostly cant work without network, but possible
        - workarounds for offline processing
    - main focus of this course

components of an app

- storage
- computation
- presentation

example - email client

storage

- where are the emails stored?
- how are they stored on the server? file format etc

compute

- indexing of emails for fast access
- searching
- security

presentation

- display list of mails
- rendering / display of individual mails

platforms:

desktop

- keyboard, mouse, video
- desktop paradigm - folders, files, documents, etc

mobile

- touch screen
- voice, tilt, camera interfaces
- small self-containted apps

web-based

- datacenter storage - persistent
- cloud - access anywhere, multi device

embedded

- simple function, limited scope
- ex- digital camera, watches, etc

architectures

- client server
- peer to peer

client server

server

- stores data
- provides data on demand
- may perform computations

clients

- end users
- request data
- user interaction, display

network

- connects server to client
- can be local
- data pipe - no alterations

client -server model

- explicit servers
- explicit clients
- local systems -
    - both client and server are on same machine - local network comunication
    - conceptually still a networked system
- machine clients -
    - eg software / antivirus updaters
    - need not have user interaction
- variants-
    - multiple servers, single queue, multiple queues, load balancing frontends, etc
- examples:
    - email
    - databases
    - whatsapp/messaging
    - web browsing

distributed peer to peer

- no distinction between client and server
- example - torrent
- all nodes are equipotent, no one is more important
- peer to peer model, all are equivalent
- error tolerance
    - master / introducers needed
    - eleciton / re-selection of masters on failure
- shared information
- example-
    - torrent
    - blockchain - based syetems
    - IPFS(interplanetary file system), Tahoe (distributed file systems)

Notes

- CS model may be bottle-necked by high traffic at server
- P2P harder to choke
- P2P can be fast if P is near to us
- CS can fail if Server fails, P2P survives some failure
- P2P is expensive to maintain
- P2P is good for redundancy and data safety# software architecture patterns
  (TLA - three letter acronyms)
- Separation of Concerns -

```
In computer science, separation of concerns is a design principle for separating a computer program into distinct sections. Each section addresses a separate concern, a set of information that affects the code of a computer program.
```

- fundamental structure of a software and rule of creating such structures and systems such that separation of concerns divides up the software into logical parts which are independent and provide an interface to each other, making development, testing, debugging, etc easier
- the layers are loosely coupled, they dont heavily depend on each other

layered architecture

The layered architecture style is one of the most common architectural styles. The idea behind Layered Architecture is that modules or components with similar functionalities are **organized into horizontal layers**. As a result, **each layer performs a specific role** within the application.

design pattern -

a general reusable solution to a commonly occuring problem within a given context in software design

- some experienced developers notice patterns in code
- resusing those patterns can make design and development faster
- guide the design and thought process
- not only way to do things, but known good ways to do things

Example: User wanting to check mail

- **User**: want to check mail
- **Server** has email
- **Model**: store emails on server, index, ready to manipulate
- **View**: display list of emails, read individual emails ,etc
- **Controller**: sort emails, delete, archive, etc

MVC - model view controller

a very good paradigm for application to build, but thrashed recently

- Model - core data to be stored for the application - stored and models the data
    - databases, indexing for easy searching, manipulation
- View - user facing side of application - the UI and UX
    - interfaces for finding information, manipulating
- Controller - business logic - how to manipulate data - brain of program, connects model and view

origins in smalltalk language 1979

---

view need not be visual, could be audio, tactile, etc

User uses controller → to manipulate the model → that updates the view → that user sees

other design patterns:

- Model View Adapter MVA
- Model View Presenter MVP
- Model View Viewmodel MVV
- Hierarchical MVC
- Presentation Abstraction Control PAC

Each has its uses, but fundamentals are very similar

Focus on this course:

Platform: Web Based

Architecture: Client Server

Software Architecture: MVC

Important Notes:

- view layer also called presentation layer in MVP
- controller layer controls business logic of code, so can be called business layer
- in MVC, one model can have multiple views
- view may not be visual
- The controller can interact with the model and view# why the web?
- platform of choice for this course
- generic - works across platforms (OS and hardware)
- build on sound underlying principles
- worth understanding
    - constraints - what can and cannot be done (easily)
    - cost: storage, network, device sizing, datacenter

history

- telephones are circuit switched - allow A to talk to B by having a physical connection between them (complex switching network)
- physical wires tied up for duration of call even if nothing is said
- so packet switching invented - here msg/data is broken into small packets and each packet contains its metadata (src, dest, etc) and is routed through common communication channels
- wire occupied only when data to be sent
- data instead of analog voice
- usage of hub-and-spoke model instead of mesh network, data multiplexed through one or more central wires, wires across all nodes not needed
- network is neutral to type of data

- IBM SNA, Digital DECNet, Xerox Ethernet, ARPANET (Internet) etc
- As so many standards are there, we need protocols for intercommunications

protocols

- how to format packets; place them on wires; headers/checksums etc
- each network had its own protocol
- can we create inter-network?
  - how to communicate between different network protocols ?
  - or replace with a single internet protocol?
- **IP**: internet protocol - 1983
  - define headers, packet types, interpretation
  - can be carried over different underlying networks: etherenet, DECnet, PPP, SLIP
- **TCP** - Transmission Control Protocol - 1983
  - establsih a reliable communication - retry, error control, etc
  - automatically scale and adjust to network limits
  - it kind of creates a 'circuit switch' on top of a packet switch network
  - it moderates send speed etc according to link capacity
- Thus TCP/IP is used in internet
- **Domian Names** - 1985
  - use names instead of IP addresses
  - easy to remember - .com revolution still in the future
- **HyperText** - 1989
  - Text documents to be served
  - formatting hints inside document to link to other documents (hypertext)
  - by tim berners lee at CERN (switzerland)

present

- original web was limit
  - static pages
  - compliacted executable interfaces
  - limited styling
  - browser compatibility issues

NOW:

- dynamic pages - generated on the fly
- http as a transport mechanism - binary data, serialised objects, etc
- client side computation and rendering
- platform agnostic operating system
- client tracking possible by cookies and sessions

Notes

- TCP is connection oriented (a connection (virtual) is needed before communication)
- UDP is connectionless, it just sends the packets, doesn't care about reliablity
- UDP can result in loss of data
- TCP requires acknowledgement after receiving data
- TCP/IP is a **session initiation protocol**
- **ARPANET** - advanced research projects agency network
- **protocol** - a set of rules that defines how the data packets are formed and placed on wires is called protocol
- IP bridges different network protocols and and defines a standardized header for all network protocols
- internet is network of networks that connects all devices on earth to each other
- WWW uses internet to showcase webpages (https etc) to users. WWW is collection of webpages# how web works?

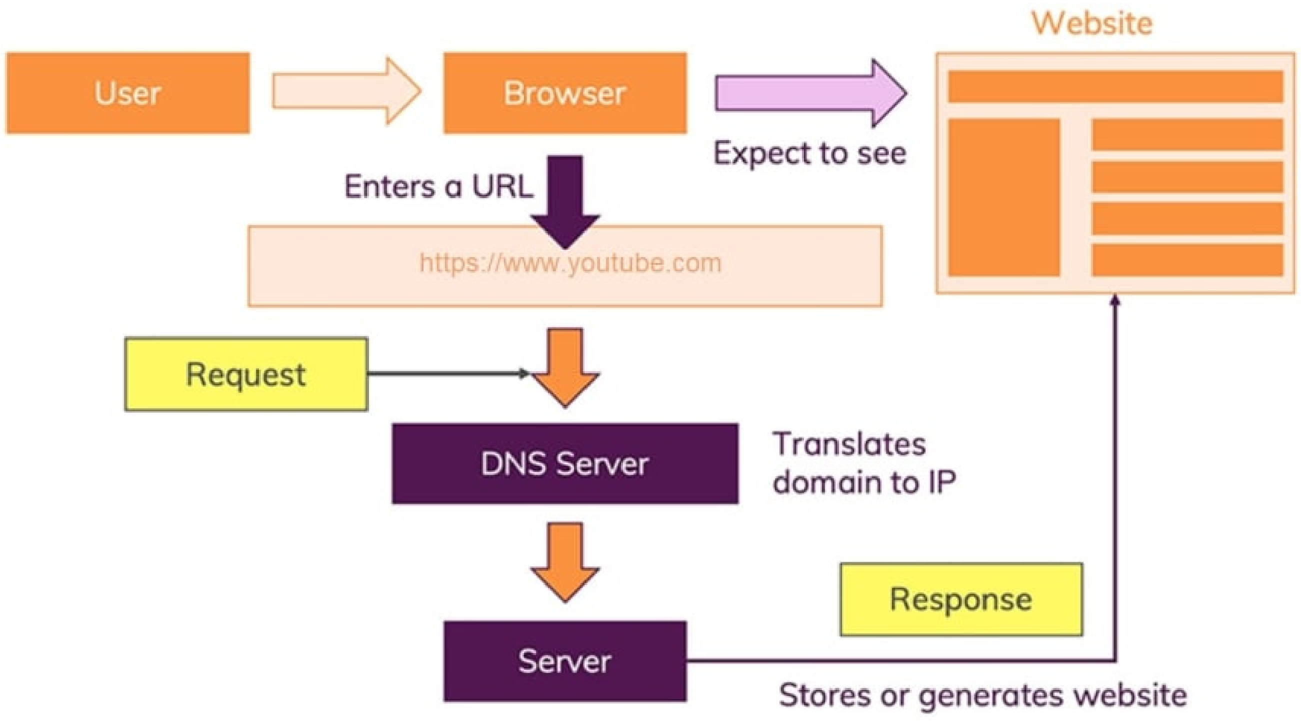
server

- any computer with a network connection
- software -
  - listen for incoming network connections on a fixed port(example 80)
  - respond in specific ways
  - opening network connnections, ports etc already known to OS
  - web server decides what to respond when some request is made, example sending a file when file is requested, rendering page on server side if required, RESTful JSON reply, etc
- protocol:
  - what should client ask server?
  - how should server respond to client?
  - HTTP

http

hyper text - regular text document that contains codes inside that indicates special funtions how to link to other documents (link → hyperlink)

hypertext transfer protocol - largely text based - client sends request, server responds with hypertext document. nowadays used for lot more than just sending HT documents.



Notes

- FTP - file transfer protocol
- HTTP is stateless protocol, it sends request and server responds as per given state
- FTP is stateful protocol - client sends a request to server and expects some response, if it doesnt get a response, it re-sends the request
- Stateless - HTTP, UDP, DNS
- Stateful - FTP, Telnet
- in stateless the C and S are loosely coupled, in stateful the server and client are tightly bound
- stateless is easier to design the server and is faster than stateful
- HTTP uses port 80
- FTP uses port 21
- Examples of web server - Apache Web Server, Nginx, Boa Webserver, FoxServ, Lighttpd, Microsoft Web Server IIS, Savant, mongoose
- Internet is interconnection of networks, connecting devices to each other.
- WWW is collection of resources on the internet, like webpages etc.
- WWW is browsed using the internet, but internet can also be used for other tasks, like IoT, FTP, etc# simplest web server

```
while true; do
        echo -e "HTTP/1.1 200 OK\n\n $(date)" | nc -l localhost 1500;
done
```

- two newlines is how HTTP 1.1 header and data are separated
- date is just a content of the http server
- netcat listens on localhost 1500 and sends the 200OK to the port
- to send request we use curl

```
curl http://localhost:1500
```

- **Note**: use open-bsd netcat, not gnu-netcat. gnu-netcat doesn't produce expected behaviour.
- The server is listening on a fixed port 1500
- On incoming request, run some code and return result
    - Standard headers to be sent as part of result
    - Output can be text or other format - MIME (Multipurpose Internet Mail Extentions)

Typical Request

```
GET / HTTP/1.1
Host: localhost:1500
User-Agent: curl/7.64.1
Accept: */*

```

- curl, wget etc simple command line utilities
- can perform full http requests
- verbose output includes all headers
- very useful for debugging
- the last line has to be empty, this demarks end of request

Notes

- Accept / means client is willing to accept any form of data (MIMEtype)
- **Loopback Devices**: a special, virtual network interface that your computer uses to communicate with itself, it is used mainly for diagnostics and troubleshooting and to connect to servers running on the local machine
    - all IPs in 127.0.0.0/8 subnet are loopback devices
        - that means, 127.0.0.1 to 127.255.255.254 all represent your computer
        - mostly 127.0.0.1 is used, and has the hostname of `localhost` mapped to it
        - 127.0.0.1 is represented as ::1 in IPv6
- 0.0.0.0 is a non-routable address. The computer doesn't try to route that address to anywhere, indicates an invalid, unknown, or inapplicable end-user address
    - it is represented in ipv6 as `::` or `::0` or `::/0`
- **CGI** - Common Gateway Interface - an interface specification that enables web servers to execute an external program, typically to process user requests. Such programs are often written in a scripting language and are commonly referred to as CGI scripts, but they may include compiled programs.# what is protocol
- Both sides agree on how to talk
- Server expects requests - nature of requests, nature of clients, types of results clients can deal with etc
- Client expects responses - ask server for something, convey what you can accept, read result and process

HTTP

- HTTP is a type of protcol, primariy text based
- requests specified as GET POST PUT etc
- headers can be used to convey acceptable response types, languages, encoding ,etc
- which host to connect to if multiple hosts on single server
- response headers also in text, conveys message type, data, cache information, status codes example 200 OK, 404 Not Found, etc
- Server errors -
    - 300 - warnings, not errors
    - 400 - user errors, wrong url etc
    - 500 - server error - example server crashes
- HTTP Actions-
    - **GET** - simple requests, queries
    - **POST**- more complex form data, large text blocks, file uploads, etc
    - **PUT** / **DELETE** - rarely used in web 1.0, extensively used in web 2.0, basic of most APIs - **REST, CRUD**

Python HTTP SERVER

- Serve files from local folder
- Understands basic HTTP requests
- Gives more detailed headers and responses
- Shows directory listing for / if /index.html is not present, else returns index.html
    - **Note**: in most servers, the index.html file is served if no particular file is asked, that is, GET/ is passed

---

Notes

Performance

- how fast can a site be?
- what limits performance
- basic observations

Latency

- Speed of light is 3e8 m/s in vacuum, 2e8 m/s in cable
- Therefore min possible latency is 5 ns / m = 5 ms/1000 km
- If data center is 2000km away, one way request takes 10 milliseconds, round trip takes 20ms
- So we are limited by 50 requests/second

Response Size

- Response = 1KB of text (headers, html, css, js, etc)
- If network connection → 100Mb/s = 100/8 MBytes/s
- Then 10,000 requests/second limit
- Google homepage is approx 150 KB

Memory

- simple HTTP server (python) consumes ~ 6mb
- multiple parallel connections can take lots of memory
- 2016 presidential debate had 2 million views on youtube, 12 TB RAM needed approx

---

Notes

- **RTT** - Round Trip Time - Time taken for round trip of request and response# Serving files via local server
- Python simple http server
- serves directory in http mode at port 8000 (changable default)
- serves index.html as / if present
- serves at 0.0.0.0 so can be accessed by any local ip, like 127.0.0.1 (called localhost), 127.126.125.124, etc
    - (any ip in range 127.0.0.1 to 127.255.255.255) will work
- for other systems in same lan, user has to know local IP (assigned by DHCP,etc) of that system and send request to that IP + port 8000
- Example 192.168.0.209:8000
- For systems outside LAN, first server needs to turn on port forwarding on router settings, then get international IP of their LAN. Then request can be sent at that IP + port# Internet Protocol

- IP
- has versions
- example IPv4 (32 bits)
- IPv6 (128 bits)

IPv4

IPv4 has 32 bits. It is represented in form of 4 octets (8 bits group)

as each octet represents 8 bits of binary data, thus it can store values from 0 to 255.

IPv4 is stored in **dotted-decimal** format, where each octet is represented in its decimal form, and octets are separated by a dot.

example:

```
192.168.0.1
```

etc.

Each octet can have only numerical values in the range `[0-255]`

IPv6

IPv6 has 128 bits. It is repesented in hexadecimal. Each hexadecimal digit represents 4 binary digits.

IPv6 has groups of 4 hexadecimal digits ($4*4=16bits$). There are 8 such groups.

$\therefore 4*4*8=128bits$

These groups are called:

- hextets
- hexadectets
- quibble
- quad-nibble

and are separated by colon `:`

Shortening of IPv6

For convenience and clarity, the representation of an IPv6 address may be shortened with the following rules.

- One or more leading zeros from any group of hexadecimal digits are removed, which is usually done to all of the leading zeros. For example, the group 0042 is converted to 42.
- Consecutive sections of zeros are replaced with two colons (::). This may only be used once in an address, as multiple use would render the address indeterminate. RFC 5952 requires that a double colon not be used to denote an omitted single section of zeros.

```
An example of application of these rules:

Initial address: 2001:0db8:0000:0000:0000:ff00:0042:8329.
After removing all leading zeros in each group: 2001:db8:0:0:0:ff00:42:8329.
After omitting consecutive sections of zeros: 2001:db8::ff00:42:8329.
```

Loopback address is `0000:0000:0000:0000:0000:0000:0000:0001` and shorted to `::1`# Port Numbers

```
In computer networking, a port is a communication endpoint. At the software level, within an operating system, a port is a logical construct that identifies a specific process or a type of network service. A port is identified for each transport protocol and address combination by a 16-bit unsigned number, known as the port number. The most common transport protocols that use port numbers are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).
```

A port number is always associated with an IP address of a host and the type of transport protocol used for communication. It completes the destination or origination network address of a message. Specific port numbers are reserved to identify specific services so that an arriving packet can be easily forwarded to a running application. For this purpose, port numbers lower than 1024 identify the historically most commonly used services and are called the well-known port numbers. Higher-numbered ports are available for general use by applications and are known as [ephemeral ports]

Number of ports

- there are **65535** ports in a computer # HTML structure
  HTML is an XML document. We denote that it is HTML using `<!DOCTYPE>` tag.

```
<!DOCTYPE HTML>
```

Then entire content of html document is put inside a `<html>` tag.

We have two outer tags, `head` and `body`

- Head contains metadata about the document
- Body contains the actual content that is rendered in the document
- Scripts can be put in both head and body

Example:

```
<!DOCTYPE HTML>
<html>
        <head>
                <title>Test Document</title>
        </head>
        <body>
```

```
            <h1>Hello World!</h1>
        </body>
</html>
```

Markup Tags

HTML is made with tags, some tags give information about the document, while some tags are helpful for marking up the document. Some examples are:

- `u` for underline
- `b` or `strong` for bold
- `i` or `em` for italics
- `a` or anchor tag for hypertexts
- `sub` for subtext
- `sup` for supertext
- `div` a division tag - no visual value but used to group parts of documents
- `p` paragraph tag, creates a new paragraph

anchor tags

- `target="_blank"` to open page in new page
    - `_self` same frame
    - `_parent` parent frame
    - `_top` topmost frame of this page
    - `framename` in the provided name of the frame

We can also link to parts inside the document using `id` attribute to any tag and then putting the id in the `href` of the `a` tag with a prefixed `#` sign.

img tag

`<img src="" />`

- src has the url of the image (absolute or relative)
- width
- height- Information Representation
- Raw data vs semantics
- lopgical structure vs styling
- html5 and CSS

Information Representation

- Computer works with only bits - 0 or 1 (binary digits)
- Numbers - binary
    - Binary Numbers → 6 = 0110
    - Two's Complement for negative numbers, eg → -6 = 1010
- Letters → A Letter to number correlation is pre-decided upon and then numbers are used

Representing Text

- ASCII
- Unicode
- UTF-8

Encoding - converting text / data into a stream of bits following some predefined conventions which can be used to decode the bits into the actual data again.
`01000001` can be :

- string of bits
- number 65 in decimal
- character A
  It depends on the context and interpretation.

ASCII

- American Standard Code for Information Interchange
- 7 bits code - 128 entities
    - a-z, A-Z, 0-9, special characters
- Only latin characters so 7 bits enough
- Didn't have any other language scripts or symbols

Unicode

- 16 bit code that has all the symbols of all the languages in the world. This had 65.536k entities (UCS-2) (2 bytes)
- 32 bits (4 bytes) code called UCS-4 that has 4 Billion+ characters, out of this only 100,000 are defined as of now

---

Notes

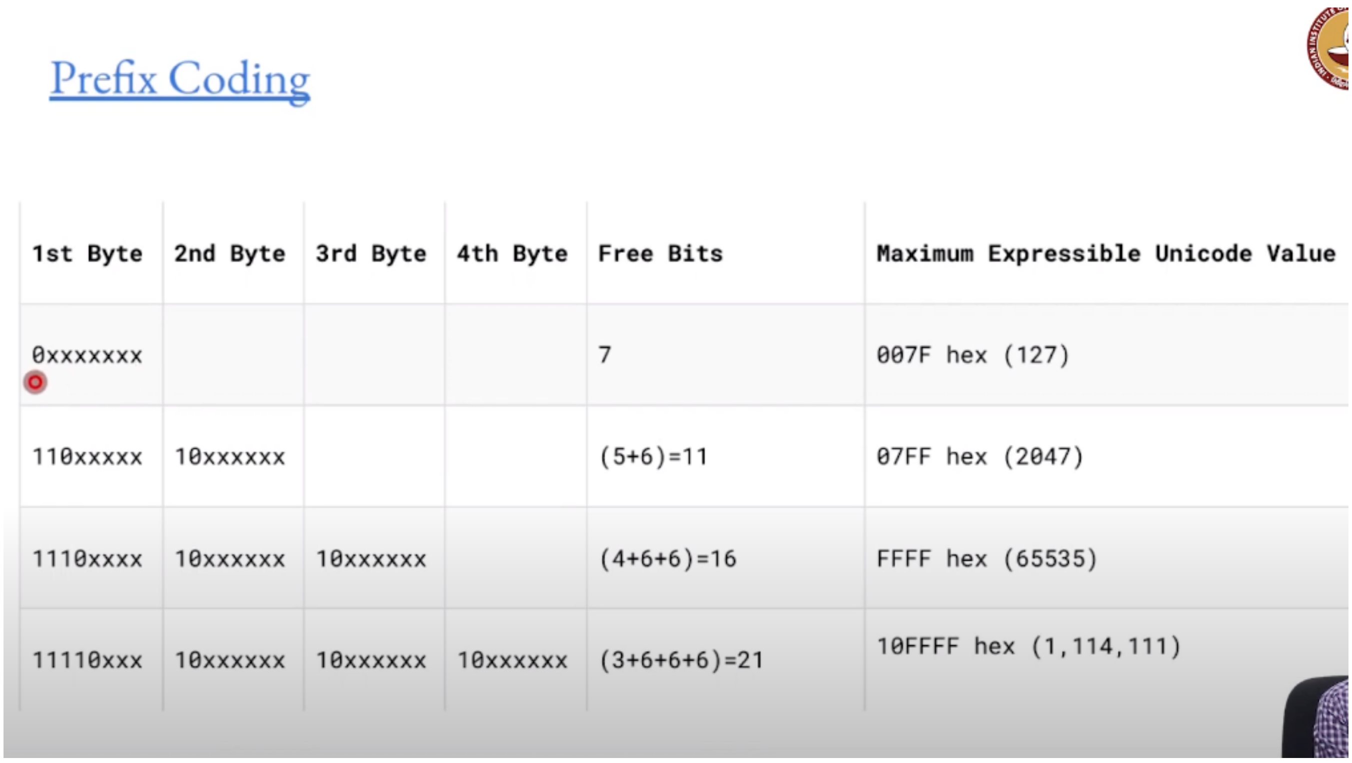
- Ascii decimal value of space = 32
- Ascii decimal value of capital letters → letter number + 64
- Ascii decimal value of small letters → letter number + 64 + 32 (n + 96)# Efficiency
- Most common language on web → English
- Should all characters be represented with same number of bits?
- Example:
    - text document with 1000 words (5000 characters approx)
    - UCS-4 encoding → $4 bytes \times 5000 characters = 20kB$
    - ASCII encoding → $1 byte \times 5000 characters = 5kB$

- Original 7 bit ASCII → $7bits \times 5000characters = 4.375kB$
- Optimal Coding based on frequency of occurance
    - 'e' is most common, then 't', 'a', 'o' , etc
    - Huffman Tree coding or similar encoding → 1-2 kB, possibly less

In general?

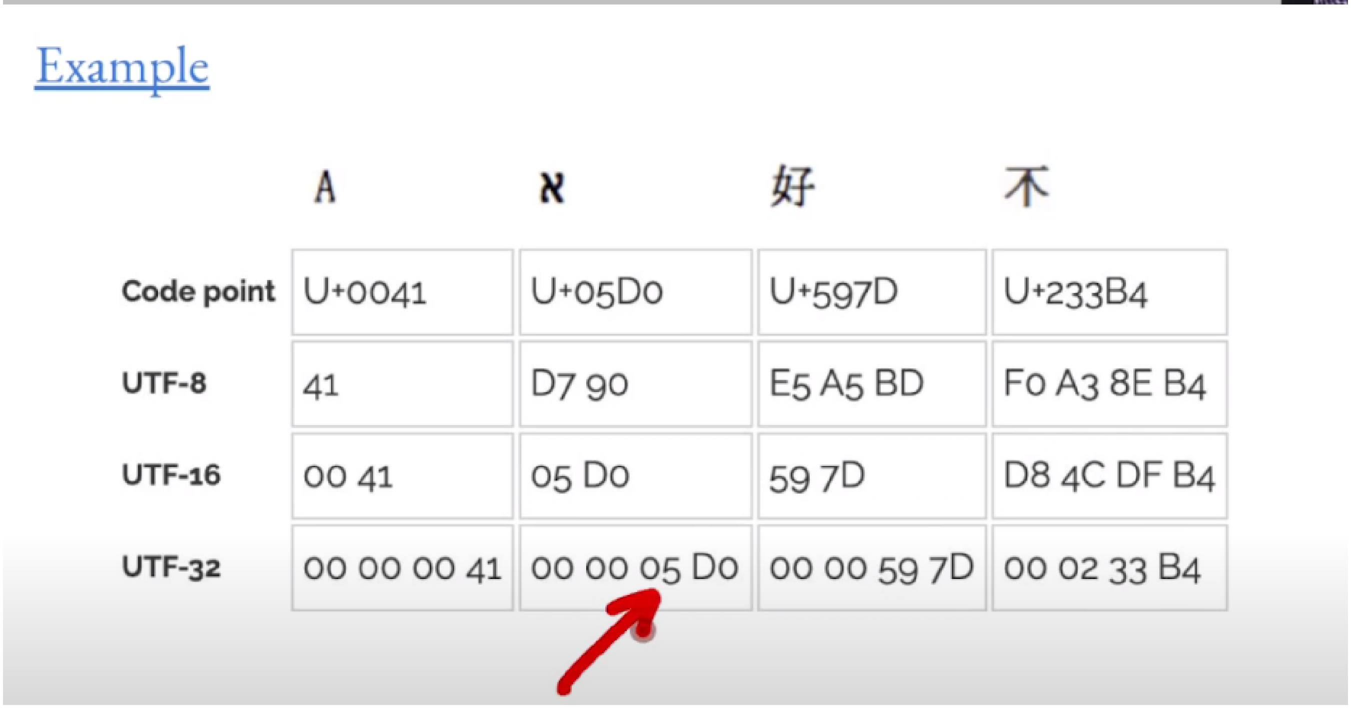
- Impossible to encode by actual character frequency as it depends on text
    - just use compression methods like 'zip' instead
- but can encoding be a good halfway point?
- example
    - use 1 byte for most common alphabets
    - group others according to frequency have 'prefix' codes to indicate

Prefix Coding



Prefix Coding

1st Byte	2nd Byte	3rd Byte	4th Byte	Free Bits	Maximum Expressible Unicode Value
0xxxxxxx				7	007F hex (127)
110xxxxx	10xxxxxx			(5+6)=11	07FF hex (2047)
1110xxxx	10xxxxxx	10xxxxxx		(4+6+6)=16	FFFF hex (65535)
11110xxx	10xxxxxx	10xxxxxx	10xxxxxx	(3+6+6+6)=21	10FFFF hex (1,114,111)



Example

	A	א	好	𣎴
Code point	U+0041	U+05D0	U+597D	U+233B4
UTF-8	41	D7 90	E5 A5 BD	F0 A3 8E B4
UTF-16	00 41	05 D0	59 7D	D8 4C DF B4
UTF-32	00 00 00 41	00 00 05 D0	00 00 59 7D	00 02 33 B4

big codepoints are stored in utf-8 using prefix-coding

- UTF-8 and UTF-16 are variable length encodings
- UTF-32 is fixed length encoding, easy to interpret, but big file size

UTF-8

- Use 8 bits for most common characters (ASCII)

- All other characters can be encoded based on prefix encoding

- More difficult for text processor-
    - First check prefix
    - Linked List through chain of prefixes possible
    - still more efficient for majority of documents
- Most common encoding used today# Markup
  A way to specify how to render the document. The style of the document, not the content.
- Content vs Meaning
- Types of Markup
- XHTML

Content

Markup is a way of using cues or codes in the regular flow of text to indicate how text should be displayed.

Markup is very useful to make the display of text clear and easy to understand.

Types of Markup

- **WYSIWYG** - what you see is what you get - directly format output and display
    - embed codes not part of regular text, specific to editor
- **Procedural**
    - Details on how to display
    - eg→ change font to large, bold, skip 2 lines, etc
- **Descriptive** - focus on what content means instead of how it looks
    - eg→ `This is a <title>, this is a <heading>, this is a <paragraph>`

Examples

- MS Word, Google Docs, etc
    - user inteface focus on appearance and not meaning
    - WYSIWYG - direct control over styling
    - often leads to complex formatting and loss of inherent meaning
- **LaTeX, HTML, nroff, groff, troff**
    - focus on meaning
    - more complex to write and edit
    - not WYSIWYG

Semantic Markup

- Content vs Presentation
- Semantics :
    - Meaning of the text
    - structure or logic of document

Notes

TeX, Nroff, Troff, Groff, PostScript → Procedural Markup as you have to mention what to do
HTML, Markdown → descriptive markup as you tell what the content is# HTML

- first used by Tim Berners Lee at CERN
- SGML → Standard Generalized Markup Language
    - Strict definitions on syntax, structure, validity
- HTML meant for browser interpretation
    - very forgiving → loose validity checks
    - best effort to display

Tags

- paired tags
- < > are used for tags
- closing tags have / before name
- Location specific tag: `<DOCTYPE>` only at top of doc
- Case insensitive
- Some self-closing tags, they have format: `<tagname/>`

Presentation vs Semantics

- strong vs b
- strong is logical markup
- b is presentational markup

History of HTML

- SGML based
    - 1989 HTML original
    - 1995 HTML 2
    - 1997 HTML 3, 4
- XML (extensible markup language) based
    - XHTML → 1997 to 2010
- HTML5

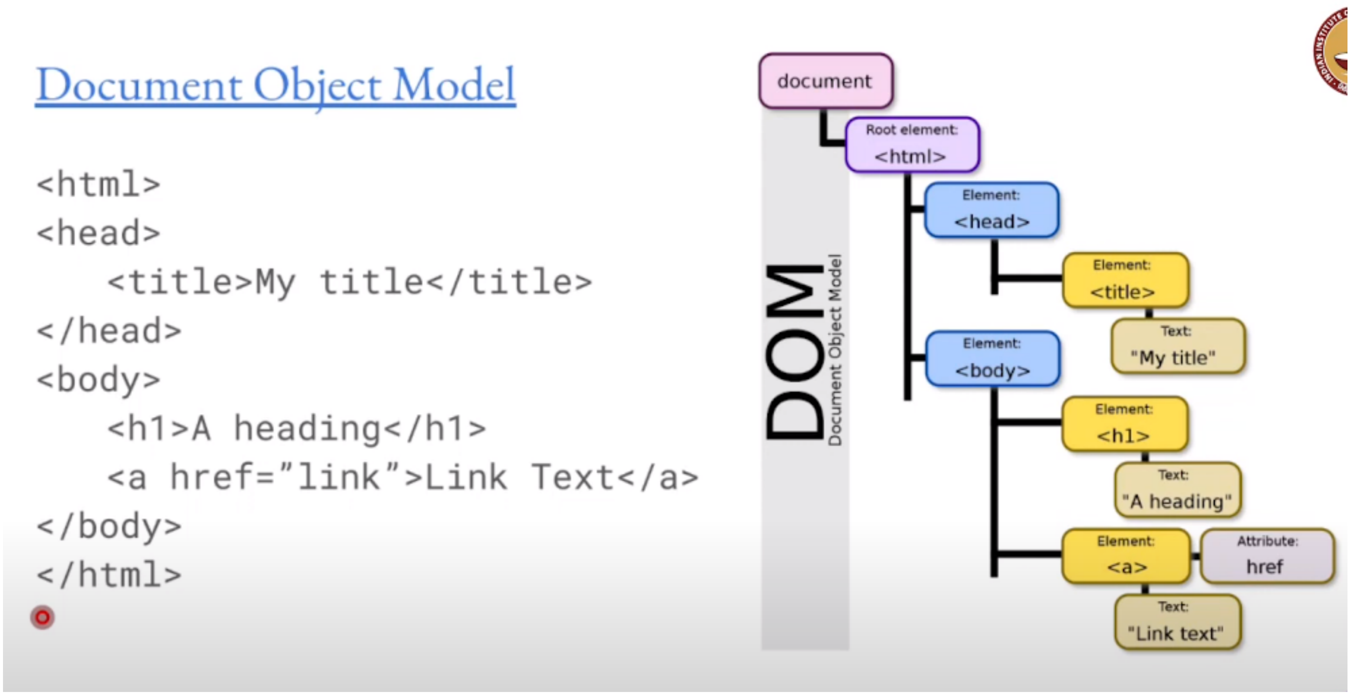
- first release 2008
- W3C recommendation → 2014

HTML5

- block elements `<div>`
- Inline elements `<span>`
- Logical Elements `<nav>`, `<footer>`
- Media: `<audio>`, `<video>`
- Remove 'presentation only' tags like
  - `<center>`
  - `<font>`

Document Object Model (DOM)

The nested markup code gives rise to a tree of document objects.



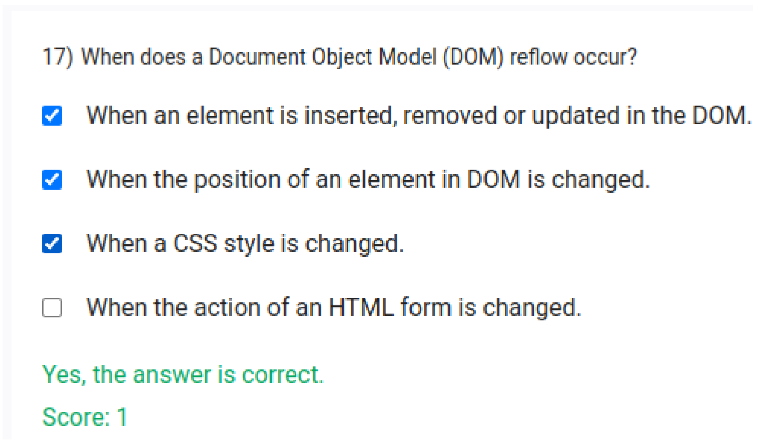
Document Object Model

```
<html>
<head>
    <title>My title</title>
</head>
<body>
    <h1>A heading</h1>
    <a href="link">Link Text</a>
</body>
</html>
```

- Tree structure representing the logical layout of the document
- Direct manipulation of the tree is possible
- Appilcation programming interfaces APIs
  - canvas
  - offline
  - web storage
  - drag and drop
- Javascript primary means of manipulating
- CSS used for styling

Notes

- List (like this) formed using `ul` / `ol` tags
- `ul` → unordered (bulleted lists)
- `ol` → ordered (numbered lists)
- the style of the list (which bullet / numbering system to use) can be changed
- Regardless of style of list, each list item is marked using `li` tag
- Checkbox → `<input type="checkbox" />`
- Text field → `<input type="text" />`
- Alt attribute of `img` tag is shown when image cant be loaded, or for screen readers
- Horizontal line (rule) can be created using `hr` tag
- `controls` attribute in audio or video tag is used to show UI controls to play/pause/ change volume etc
- `&copy;` used to show copyright symbol
- A reflow on an element recomputes the dimensions and position of the element, and it also triggers further reflows on that element's children, ancestors and elements that appear after it in the DOM. Then it calls a final repaint.

- 
17) When does a Document Object Model (DOM) reflow occur?

- [x] When an element is inserted, removed or updated in the DOM.
- [x] When the position of an element in DOM is changed.
- [x] When a CSS style is changed.
- [ ] When the action of an HTML form is changed.

Yes, the answer is correct.
Score: 1

- ## Markup vs Style

- Markup tells the logical structure of the document
- Style tells how the document should look

Separation of Styling

- Style hints in separate blocks
- separate files included
- Themes possible
- Style sheets - specify presentation information
- Cascading Style Sheets (CSS) → allow multiple definitions, latest takes precedence

---

Notes

- global selector in CSS `*` selects all tags
- color can be provided by:
  - color name eg white, black, tomato
  - hexcode eg, `#ffffff`
  - rgb or rgba functions
- comments in CSS use`/* ... */` syntax
- Responsive Website → Adapt to changes in screen sizes
- CSS precedence → Inline > Internal > Extrenal# Inline CSS
- Directly add style to a tag (scoped)
- Example:

```
<h1 style="color: blue;text-align: center;">A heading</h1>
```

Internal CSS

- Embed inside `<head>` tag
- example:

```
<style>
        body{
                background-color: linen;
        }
        h1{
                color: maroon;
                margin-left: 40px;
        }
</style>
```

this will apply the style to all h1 tags in body

External CSS

- Extract common content for reuse
- Multiple CSS files can be included
- Latest definition of style takes precedence

Responsive Design

- Mobile and Tablets have smaller screens
- different form factors
- adapt to screen - respond
- CSS control styling - HTML controls content

Bootstrap

- CSS framework, originated from twitter

- standard styles for various components
    - buttons
    - forms
    - icons
- mobile first: highly responsive layout

Javascript

- interpretted language brought into the browser
- not really related to java in any way - formally ECMAscript
- programming ability inside website
- not part of the core presentation requirements

Notes

- CSS shorthand properties are properties which combine multiple properties into one. the value of the properties takes multiple space separated values corresponding to each of the properties.
    - example → border, margin, padding
- For conflicting styles, the order in which the CSS files are loaded, the CSS styles are defined are all important
- HTML attribute order is not importantThe `<thead>` tag is used to group header content in an HTML table.

The `<thead>` element is used in conjunction with the `<tbody>` and `<tfoot>` elements to specify each part of a table (header, body, footer).

Browsers can use these elements to enable scrolling of the table body independently of the header and footer. Also, when printing a large table that spans multiple pages, these elements can enable the table header and footer to be printed at the top and bottom of each page.

Note: The `<thead>` element must have one or more `<tr>` tags inside.

The `<thead>` tag must be used in the following context: As a child of a `<table>` element, after any `<caption>` and `<colgroup>` elements, and before any `<tbody>`, `<tfoot>`, and `<tr>` elements.

Tip: The `<thead>`, `<tbody>`, and`<tfoot>`elements will not affect the layout of the table by default. However, you can use CSS to style these elements (see example below)!

Style

- `nth-child(even)` - only even child elements
  example:

```
tr:nth-child(even){
        background-color: lightgray;
}
```

to make table rows background-color alternatingtag → using name of tag
class → using dot .
id → using hash #

Relational Selectors

- children (direct child)
- descendant (child of child of ....)

Descendant

```
form input {
        ...
}
```

all input descendants of form are selected

Child

```
form > input {
        ...
}
```

only direct children are selected

Pseudo Class Selectors

Based on state / structure of HTML

- hover → when mouse is hovered
- nth-child(...) → select the child which evaluate true to the expression ...

Example:

```
form > input:hover{}
```

Pseudo Element Selectors

- use double colon ::

- Example:

```
form p::first-letter{}
```

apply style to first letter of each p inside form

```
input[x="y"]{

}
```

apply style to all input tags who has attribute x with value y
`<input x="y">`

References

Pseudo-Classes
Pseudo-Elements
https://getbootstrap.com/

Model - View - Controller

- Model - Stores data and how to access it
- View → display data, UI
- Controller - Connect m and v, control data flow

MVC origins in smalltalk-80 language from xerox PARC

Separation of responsibilities - abstraction → roots in OO GUI development

Running Example: Student Gradebook

Input Data

for model →

- student list
- course list
- student-course marks

Views:

- marks for individual students
- summary of course
- histograms

Controller:

- add new students
- add new courses
- modify marks

Notes:

- Smalltalk-80 is dynamically typed, OO, programming language
- View responsible for user interaction with application# Views
- User interface
- User interaction

User interface

- screen
- audio
- vibration (haptics)

User interaction

- keyboard / mouse
- touchscreen
- spoken voice
- custom buttons
- determined by hardware constraints
- different target devices possible
- user-agent information useful to identify context **!**
- may not be under designer control

Types of Views

- Fully static
- Partly Dynamic (Wikipedia)
- Mostly Dynamic (Amazon)

Output

- HTML - most commonly used - direct rendering
- Dynamic images
- JSON/XML - machine readable

" View is any representation useful to another entity (human/machine)"# User interface design

- design for interaction with user
- goal -
  - simple - easy for user to understand and use
  - efficient - user achieves goal with minimum effort
- aesthetics - what looks good
- accessibility

Systematic Process

- functionality requirement gathering - what is needed
- User and Task Analysis - user preference, tasks needs
- Prototyping - wireframes, mockups
- testing - user acceptance, usablity, accessibility# Guidelines / Heuristics
- Jakob Nielsen's Heuristics for design
- website

10 Guidelines:

#1: Visibility of system status

The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.

#2: Match between system and the real world

The design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.

#3: User control and freedom

Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process.

#4: Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions.

#5: Error prevention

Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions, or check for them and present users with a confirmation option before they commit to the action.

```
There are two types of errors: slips and mistakes. Slips are unconscious errors caused by inattention. Mistakes are conscious errors based on a mismatch between the user’s mental model and the design.
```

#6: Recognition rather than recall

Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design (e.g. field labels or menu items) should be visible or easily retrievable when needed.

#7: Flexibility and efficiency of use

Shortcuts — hidden from novice users — may speed up the interaction for the expert user such that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

#8: Aesthetic and minimalist design

Interfaces should not contain information which is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.

#9: Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.

#10: Help and documentation

It’s best if the system doesn’t need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks.

General Principles:

- Consistency
- Simple and minimal steps
- Simple language
- minimal and aesthetically pleasing# Tools

Wireframes

- Visual guides to represent structure of web page
- information design
- navigation design
- user interface design

lorem ipsum:

fake latin text that is only meant as a text placeholder to show how the text content would look without distracting the design by seeming to be actual text.

tools for wireframes:

- LucidChart
- Adobe XD
- Figma

Programmatic HTML generation: PyHTML

- Composable functions - each function generates a specific output
- Example:
    - to generate a h1 heading: function should return
    - `<h1>text of heading</h1>`

Example of pyhtml:

```
import pyhtml as h
t = h.html(
        h.head(
                h.title('Test Page')
        ),
        h.body(
                h.h1('This is a title'),
                h.div('This is some text'),
                h.div(h.h2('inside title'),
                        h.p('some text in a paragraph'))
        )
)
print(t.render())
```

More complex HTML:

```
def f_table(ctx):
        return (tr(
                td(cell) for cell in row
                ) for row in ctx['table'])
```

Templates:

- Standard template text
- Placeholders / variables
- basic /very limited programmability
- examples:
    - python inbuilt string templates - good for simple tasks
    - jinja2 - used by flask
    - genshi
    - mako

Python string template

```
from string import Template

t = Template('$name is the $job of $company')
s = t.substitute(name='Tim Cook', job='CEO', company='Apple Inc.')
print(s)
```

prints "Tim Cook is the CEO of Apple Inc."

Jinja

- ties in closely with flask
- template functionality with detailed API
- templates can generate any output, not just HTML

Example:

```
from jinja2 import Template
t = Template("Hello {{ something }}!")
print(t.render(something="World"))

t = Template("My favourite numbers: {% for n in range(1,10) %} {{ n }} " "{% endfor %}")
print(t.render())
```

will print:

```
Hello World!
My favourite numbers: 1 2 3 4 5 6 7 8 9
```

Accessibility

- Various forms of disability or impairment
    - Vision
    - Speech
    - Touch
    - Sensor-Motor
- Can a page be accessed by people with impairments?
- How can the accessibility of a page be improved?

World Wide Consortium (W3C) - accessibility guidelines

- guidelines

Standards

Interplay between many components of a page

- Web content: HTML, images, scripts, etc
- User-agents: desktop browser, mobile browser, speech-oriented browser, assistive devices
- Authoring tools: text editor, word processor, compiler

Principle - Perceivable

- Provide text alternatives for non-text content
- Provide captions and other alternatives for multimedia
- Create content that can be presented in different ways, including by assistive technologies, without losing meaning.
- Make it easier for users to see and hear content.

Principle - Operable

- Make all functionality available from the keyboard
- give users enough time to read and use content
- do not use content that causes seizures or physical reactions
- help users navigate and find content
- make it easier to use inputs other than keyboard

Principle - Understandable

- Make text readable and understandable
- Make content appear and operate in predictable ways
- Help users avoid and correct mistakes

Principle - Robust

- Maximize compatibility with current and future user tools.

Examples:

- Use aria-describedby attribute

Notes

- {{ }} are variable interpolation
- {% %} are blocks
- {# #} are comments in jinja2```
  if **name** == '**main**':
  main()

```
only runs main if its run directly (not importted)

# string formatter
```

string = "hello {name}"
string.format(name="Sayan"))

```

```

string = "hello {}"
string.format("Sayan"))

```
## specifiers:
- `+` for showing sign of number always
`a="this is {p:+}"
- `d` for decimal value
```

```
- `x` for hexadecimal value# Persistent Storage
Example: Gradebook
- students: ID, name, address
- courses: ID, name, department, year
- StudentCourse Relationship - which students are registered for which courses

## Spreadsheets
- arbitrary data organized into rows and columns
- operations defined on cells or ranges
- multiple inter-linked sheets within single spreadsheet

## Relationships?
- student - course ?
- separate entry with full details - student name, id, address, course id, name, department, etc ? NO
        - redundant
- Create another table joining students and courses
        - only ID required
        - relation specified with keys# Memory Data Structures
- lists
- tuples
- dictionaries
- objects

---
## KEYS
used to uniquely identify elements
- data entry errors less likely
- duplicates not a problem - unique key
---
## OBJECTS
```

class Student:
idnext = 0 # Class Variable
def **init**(self, name):
self.name = name
self.id = Student.idnext
Student.idnext = Student.idnext + 1

```
- auto initialise ID to ensure unique
- functions to set/get values

---
# PERSISTENT STORAGE?
- in memory data structures lost when server shut down or restarted
- - save to disk? structured data?
        - python pickle module (serialising data)
        - csv - comma separated values
        - tsv - tab separated values
- Essentially same as spreadsheets - limited flexibility
---
## ADVANTAGES AND DISADVANTAGES OF SPREADSHEETS

### ADVANTAGES:
- natuarally represent tabular data
- extension, adding fields easy
- separate sheet for relationships

### DISADVANTAGES
- lookups, cross-referencing harder than dedicated database
- stored preocedures - limited functionality
- atomic operations - no clear definition

---
# RELATIONAL DATABASES
- Data strored in tabular format
        - columns of tables: fields(name, address, departements, etc)
        - row of tables: individual entries (student1, student2, etc)

---
## Unstructured databases (NoSQL)
- easily add/change fields
- arbitrary data
- noSQL
        - mongoDB
        - couchDB
- Flexible, but potential loss of validation# Relationships
- joining two tables together using their unique ids -> expresses relationships between them

## Types of Relationships
- **One to one**
        - one student has one roll number
        - one roll number uniquely identifies one student
        - example: assign unique message-ID to each email in inbox
- **One to many ( many to one)**
        - one student stays in only one hostel
        - one hostel has many students
        - example: save emails in folders, one email is in only one folder, but one folder has multiple emails
- **Many to many**
        - one student can register for many courses
        - one course can have many students
        - example: assign labels to emails, one email can have multiple labels, and vice versa
```

```
---
# Diagrams
- Entity Relationship (ER) diagram
- Unified Modeling Language (UML)
- Class relation diagram

## Crow foot notation -
[https://vertabelo.com/blog/crow-s-foot-notation/](https://vertabelo.com/blog/crow-s-foot-notation/)
# SQL
## KEY
- **Primary Key** - important for fast access on large databases, unique attribute which cannot be null
- **Foreign Key** - connect to different table - Relationships
## Queries
- Retrieve data from database
- eg- find all students with name beginning with A
- find all courses offered in 2021
## Structured Query Language (SQL)
- english like, but structured
- quite verbose
- specific mathematical operations -
        - inner join
        - outer join
- Example inner join:
- ![[Pasted image 20220603113939.png]]
- ![[Pasted image 20220603113954.png]]

## Cartesian Product
- N entries in table 1
- M entries in table 2
- M x N combinations - filter on them
- Powerful SQL Queries can be constructed
## Example - find students in Calculus
- find id number for course
- look up studentcourses table to find all entries with this coures id
- look up students to find names of students with those ids
```

select s.name
from Students s
join StudentsCourses sc ON s.IDNumber = sc.studentID
join Courses c ON c.ID = sc.courseID
where c.name = 'Calculus'

```
---
# Notes
- Single line comment in SQL: `-- this is a comment`
- ![[Pasted image 20220603115558.png]]
- **TRUNCATE** command deletes all data from table, but schema is preserved
- **DROP** command drops the entire table along with data and schema
- **DROP** cannot be rolled back
- **NOSQL** databases dont have to adhere to ACID properties
- `LIKE` keyword used to check likeness of strings in SQL, example:
- ![[Pasted image 20220603120001.png]]
- here `%` refers to multiple characters, `_` refers to single character
- to find highest or lowest of something, sort by attribute, then limit 1
- ![[Pasted image 20220603120457.png]]
- ![[Pasted image 20220603120531.png]]
- VALID JOINS-
        - INNER JOIN
        - FULL JOIN
        - LEFT JOIN
        - RIGHT JOIN
```