

1b. Course Overview

CSCI 2541 Database Systems & Team Projects

Wood

Last time...

Structure that
is **independent**
of the
underlying file
formats

1

Queries to
flexibly read,
update, and
delete
information

2

Transactions
that provide
guarantees
about **multi-**
user
consistency

3

...this time.

Queries & Data Independence

Queries to
flexibly read,
update, and delete
information

What this means....

A user of a relational database system should be able to use the database without knowing precisely how data is stored, e.g.

```
SELECT Name, Reservation  
FROM Customers  
WHERE Name= 'Cat Meadows'
```

The above “query” does not need to know how the data in Customers is stored. Why should you need to worry about that?!

How to define and use a database

Queries to
flexibly read,
update, and delete
information

Data **Definition** Language (**DDL**) to specify database schema

- What data, and how it is organized (**logical level**)

Data **Manipulation** Language (**DML**) allows users to access or manipulate data as organized by data model

- procedural DMLs: require user to specify **what data and how** to get it
- non-procedural DMLs: require user to specify what data is needed **without** specifying how to get it.

Often, one language provides both features (e.g., **SQL**)

Relational DB Query Languages

Queries to
flexibly read,
update, and delete
information

Formal query languages:

- Relational algebra,
- Relational Calculus,
- Why study formal languages?

Commercial query language: **SQL**

SQL: “descendent” of SEQUEL; mostly relational algebra and some aspects of relational calculus

- has procedural and non-procedural aspects
- Has DDL and DML components

What is SQL?

Queries to
flexibly read,
update, and delete
information

SQL is **not** a specific database software

It is a standardized query language

- Defines how to create a database schema and issue read/write queries

DBMS
MySQL
SQLite

DBMS software must implement the SQL **standard**

- Plus some of their own extensions
- None actually follow the official ANSI SQL standard precisely...

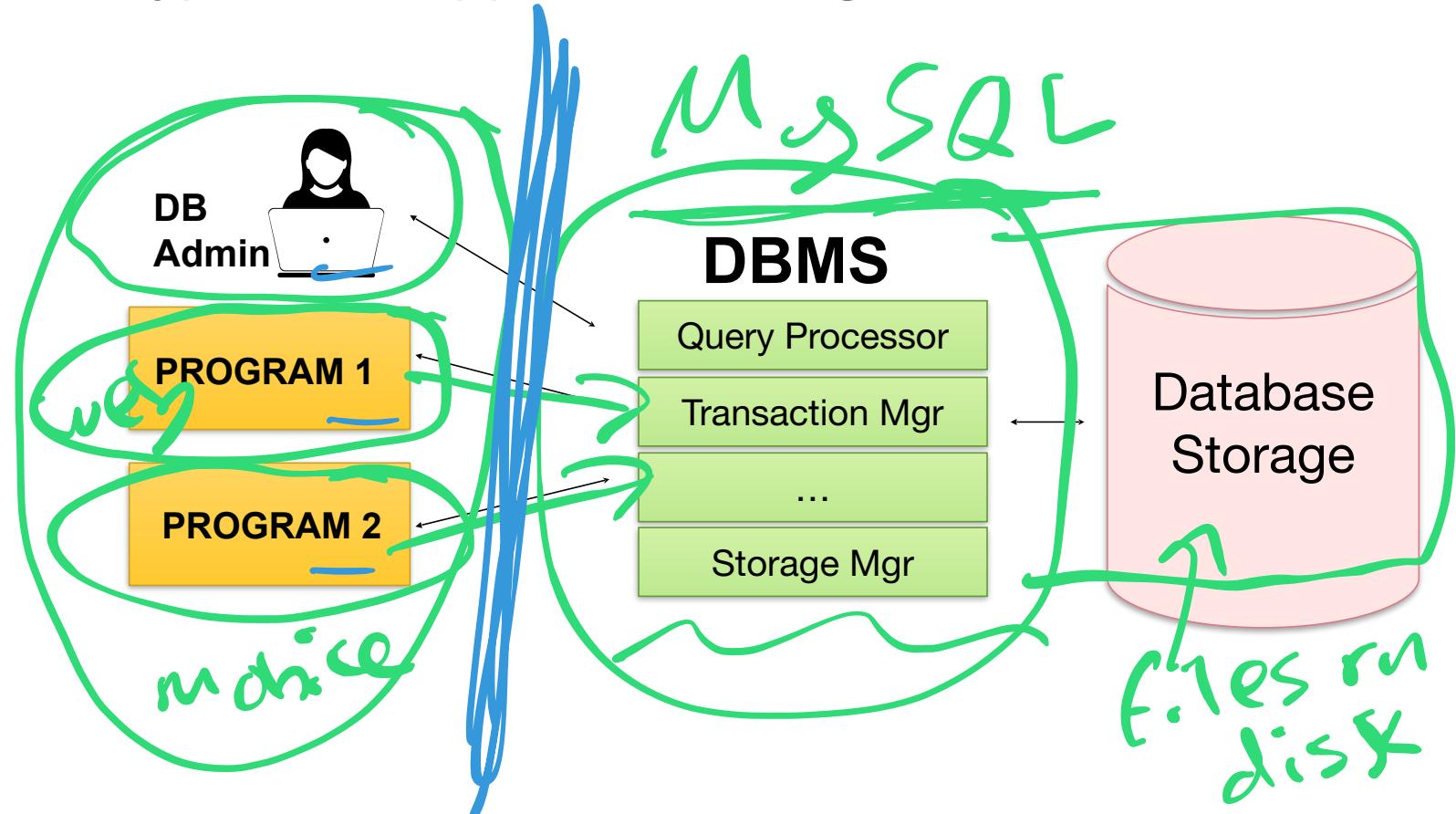
Good news: SQL queries are “cross platform” and will work on many different database systems

Confusing? Curious?

Connecting to a DB

Transactions that provide multi-user consistency

A typical DB Application design



The data abstraction is provided by the DBMS

- Separation b/w Logical and Physical, Query language parsing, multi-user, etc.

A database management system provides efficient, convenient, and safe multi-user storage and access to massive amounts of persistent data

- **Efficient & Convenient** - Able to handle large data sets, complex queries without searching all files and data items, easy to write queries
- **Scalability** – Large/huge data
- **Persistence & Safety** - Data exists after program execution completes, handles loss of power
- **Multi-user** - More than one user can access and update data at the same time while preserving consistency....concept of transactions

Components of a DBMS

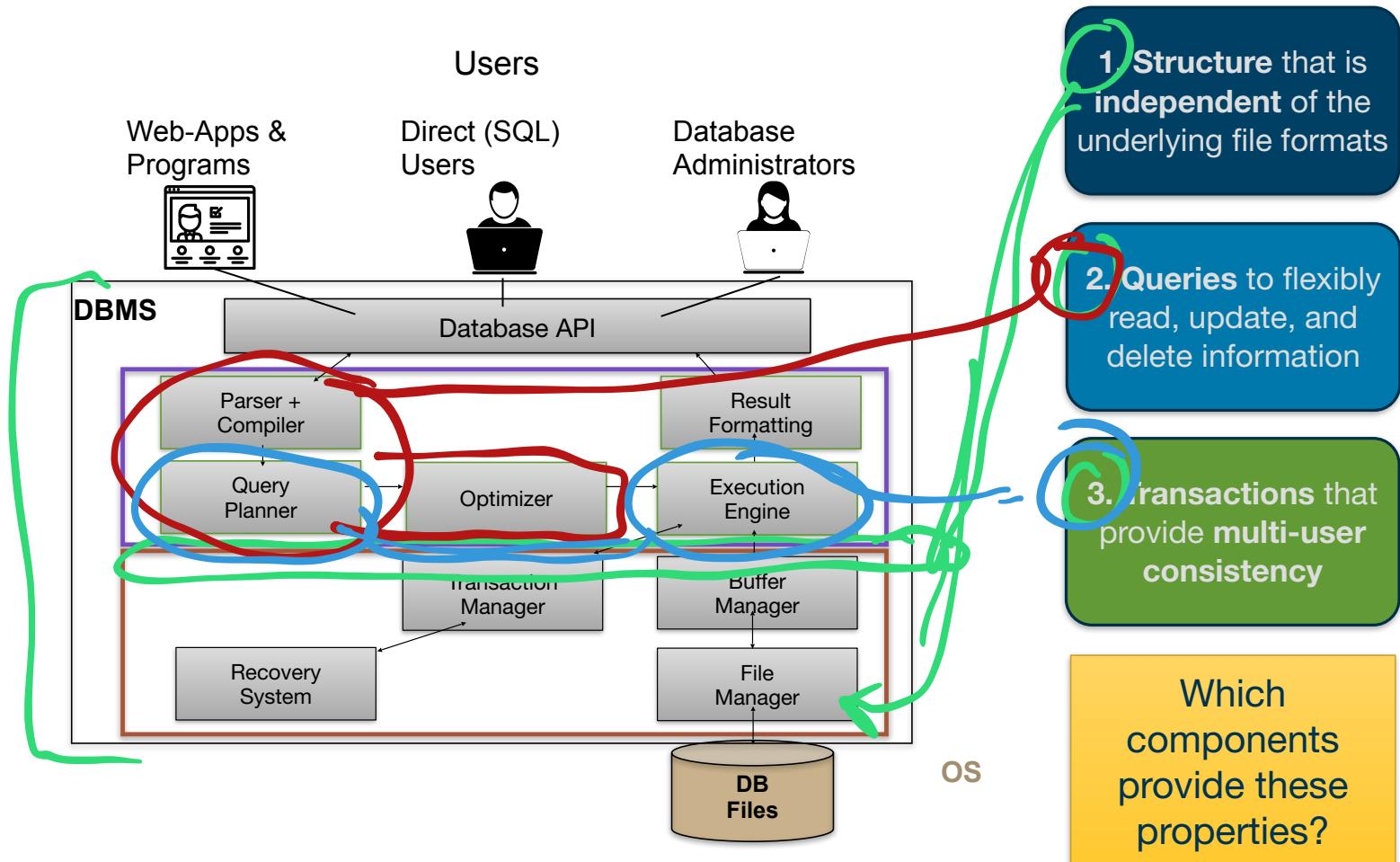
Transactions that provide multi-user consistency

A database management system provides efficient, convenient, and safe multi-user storage and access to massive amounts of persistent data.

A DBMS is a complicated software system containing many components:

- **Query processor** - translates user/application queries into low-level data manipulations
 - Sub-components: query parser, query optimizer
- **Storage manager** - maintains storage information including memory allocation, buffer management, and file storage
 - Sub-components: buffer manager, file manager
- **Transaction manager** - performs scheduling of operations and implements concurrency control algorithms
- You will learn more about storage management and concurrency in the Operating Systems course... enjoy!

DBMS Architecture: Complete Picture



This course is about Database Design...

Focus is on design of databases

- Working at the **logical level**

Internals of DBMS is not the focus in this course

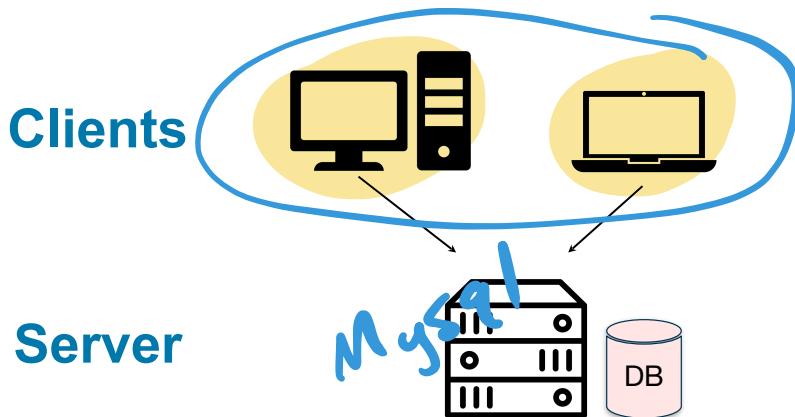
- BUT we will touch upon a few key concepts that make DBMS' work
- DBMS design brings together several key concepts from Computer Science
 - Languages, Compilers/translation, Algorithms, Data structures, Operating systems...

Database System Architectures

There are several different database architectures:

- **Embedded architecture** - DB files and DBMS processing occurs at the clients (e.g. Microsoft Access or SQLite)
 - You might work with this in Systems Programming 3410
- **DB client-server architecture** - dedicated machine running DBMS accessed by remote clients (e.g. MS SQL Server, MySQL, Postgres)
- **Multi-Tier client-server architecture** - DBMS is bottom tier, second tier is an application server containing business logic, top tier is clients (e.g. Web browser-Apache/Tomcat-Oracle)
 - i.e., a LAMP Stack

DB Client-Server Architecture



Tier 1: Client Application

- User Interface
- Business and Data Logic

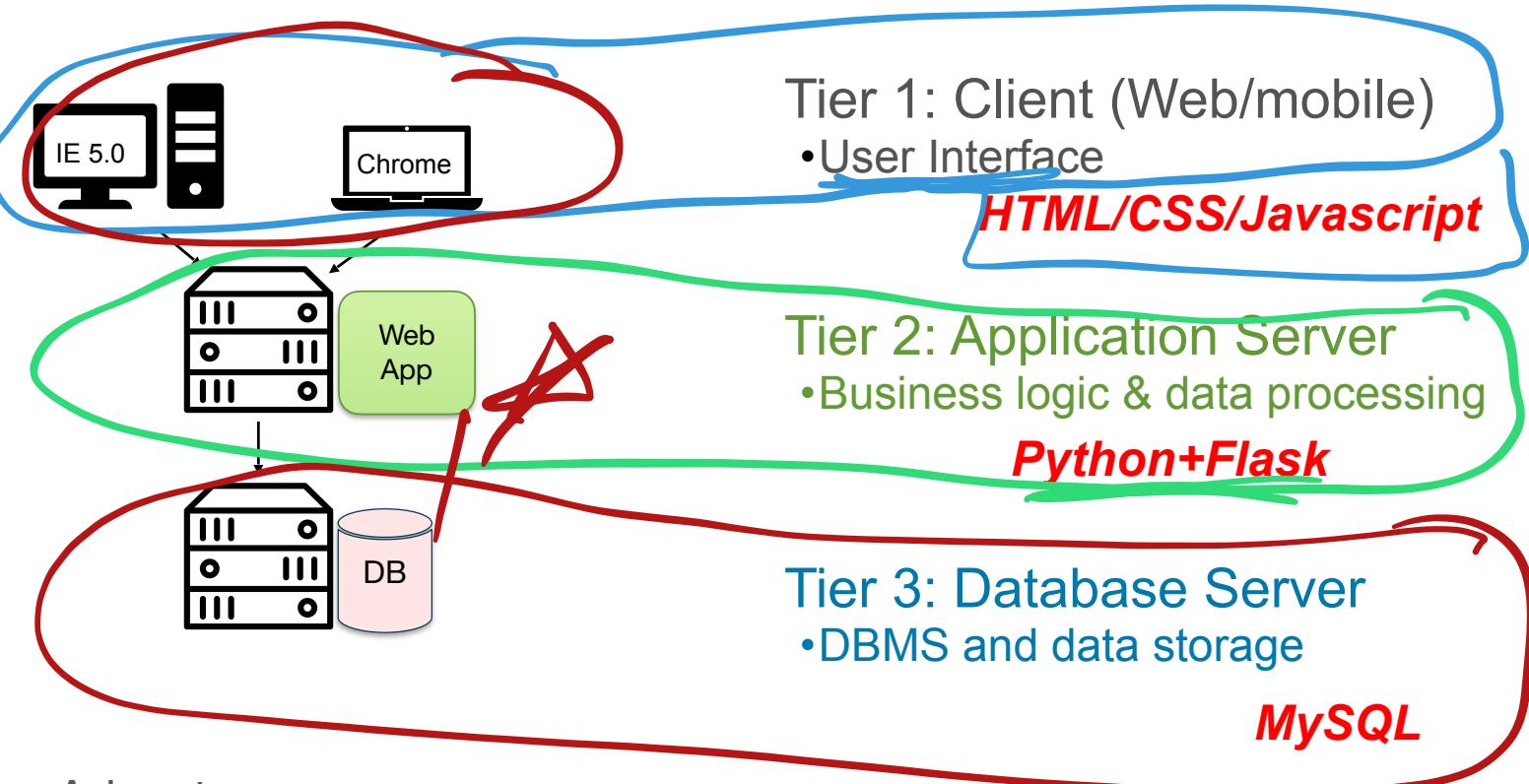
Tier 2: Database Server

- Server-side validation
- Data storage/management

Advantages:

- Only one copy of DBMS software on dedicated machine
- Increased performance
- Easier to maintain consistency and manage concurrency

Three-Tier Client-Server Architecture – our approach



Advantages:

- Reduced client administration and cost using thin web clients.
- Easy to scale architecture and perform load balancing.

Course Requirements & Logistics...

Course Objectives

Relational database theory and design

- Concepts of data storage and retrieval

Fluency in SQL and web front- and back-end development

- Working with relational database systems: MySQL
- Python to develop DB connected apps

Software integration experience and team S/W development experience

- Design and deploy a large database application
- Full stack (web) development

Brief introduction to NoSQL database models

Late Policy

All group assignments must be submitted by the deadline; no late work accepted

- Applies to most labs and team projects
- Find a way to get it done; partners will report if work was evenly divided and you will lose points if you did not do your share

For individual assignments you may ask for an extension if needed

- I give the first extension for any reason...
- But you must recognize that submitting late places an extra burden on the staff and disrupts our ability to give feedback to other students

Async Attendance

We prefer you attend lectures/labs “live”

- If you cannot do this, contact me this week
 - Once we are back on campus, we will have a way for quarantined students to participate remotely

You need to find a way to stay engaged!

- Asking or answering questions in class is the easiest
 - Post questions to slack under each class's thread
 - Look for bonus activities in #engage

Course Requirements: Grading

Exam (midterm): 22.5%

- Based on lectures and labs. Around Week 6-7

Homework, Lab Assignments: 25%

- Programming and written homework
- In-Lab/Class exercises given out during class and equivalent to a “quiz”

Team Project: 37.5%

- Phase 1 (15%) + Phase 2 (22.5%)
- No final exam BUT final project demos are required
- To pass project, your demos have to work...NO broken websites!

Team Contribution: 10%

- How well do you work with others in labs and projects

Engagement: 5%

- Participation in class or online with a variety of bonus opportunities
- Usually limited to 2 points per week; Need 15 points for full credit

Grades curved (and scaled as percentage of highest score in class)

Approximate grading method after curving and scaling

A- to A: 90-100%
B- to B+: 80 to < 90
C- to C+: 70 to < 80
D-: >60

The Project

A significant part of your grade for the course is a large database systems project

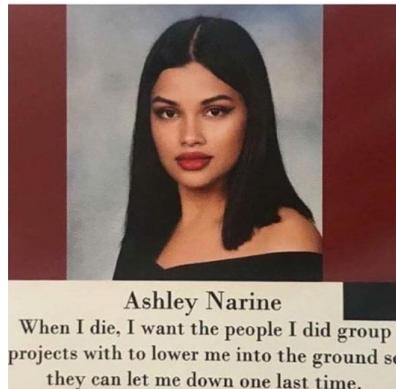
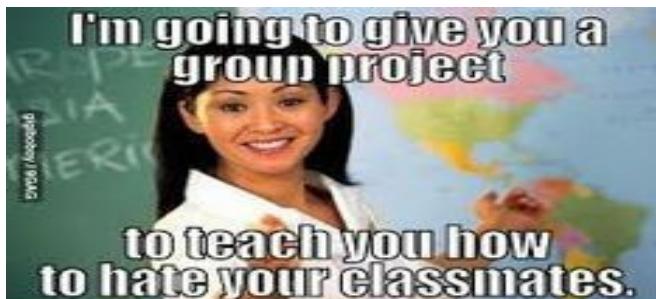
In the project you will design & implement a database system

- Full stack development:
 - Front End (HTML/CSS & optional Javascript)
 - Application server – in Python
 - DBMS backend – MySQL
- All the above are useful (high demand) skills
 - Note that limited background will be given on web programming (seek help from TAs)

The project will involve working in teams of 2 to 4

- Larger teams must develop projects with more features

Why do we have team projects



You make half of the assignment and I'll make the other half and then we'll join them together



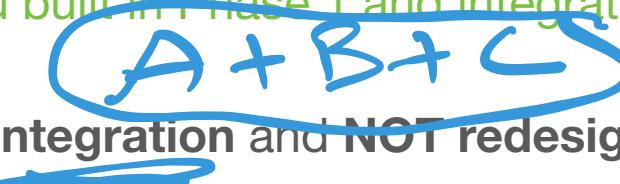
Real World: Teamwork and S/W development in teams is the default!

- Communication
- Collaboration
- Conflict resolution
- Using tools to enable collaborative SW dev

Team Project: Requirements & Expectations

Project broken into 2 phases:

- **Phase 1:** teams to build an application assigned to your team

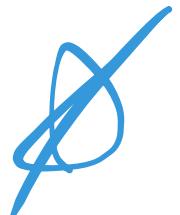
- **Phase 2:** Work in new teams to integrate different applications and produce the final project
 - Take what you built in Phase 1 and integrate with systems built by others....

 - This requires **integration** and **NOT** redesign

You HAVE to deliver a working project...else Zero

Teamwork Assessment...part of your grade!

You have to work in teams

- Each team member required to ‘produce’ equitable
- Teamwork will be assessed...
 - Not all team members may get the same grade on the project!
 - You must bring teamwork issues to attention of the instructor
 - If you do not contribute your fair portion of Phase 1, then you may be required to complete Phase 2 **on your own**



The second half of the course will have one session (lecture or a lab) dedicated to teamwork check-ins each week

- Instruction team will meet with each team, and assess if the weekly deliverables are being met by each team member

If you cannot commit time each week to working on the team project then please drop the course!



Academic Integrity Policy

No collaboration on homework/programming assignments

- Including external resources, tutors, online help
- Okay to clarify questions for a classmate
- Okay to share a 1-2 line code snippet example
- **Not okay** to share solutions or solution code

No collaboration between teams on team projects

- within team each team member must have clear role -- i.e., clearly partitioned tasks for each team member

Most semesters, at least one student **fails my class because of cheating. Several students have been suspended from the university; none returned.**

~~1:55pm~~

HTML/CSS Lab Time!

(Break first?)

General Format and Expectations

- Hands on practical experience on the techniques described in lecture.
 - Usually intro slides, guided coding, then a group lab activity
- Lab exercises are generally due 24 hours after lab
 - Extra time this week since you are figuring things out
- Queries and code submitted are expected to work. If it doesn't run you will not get credit.

HTML & CSS

HTML

Hypertext



Hypertext

A document containing links to other locations or content in a page

Markup Language

VS Programming
Larry

Markup Language

A human readable language system that uses tags to write and format the elements in a document.

HTML =
Hypertext + Markup Language

HTML

A language that uses tags and attributes to define the content (Links, Text, and images) of a webpage.

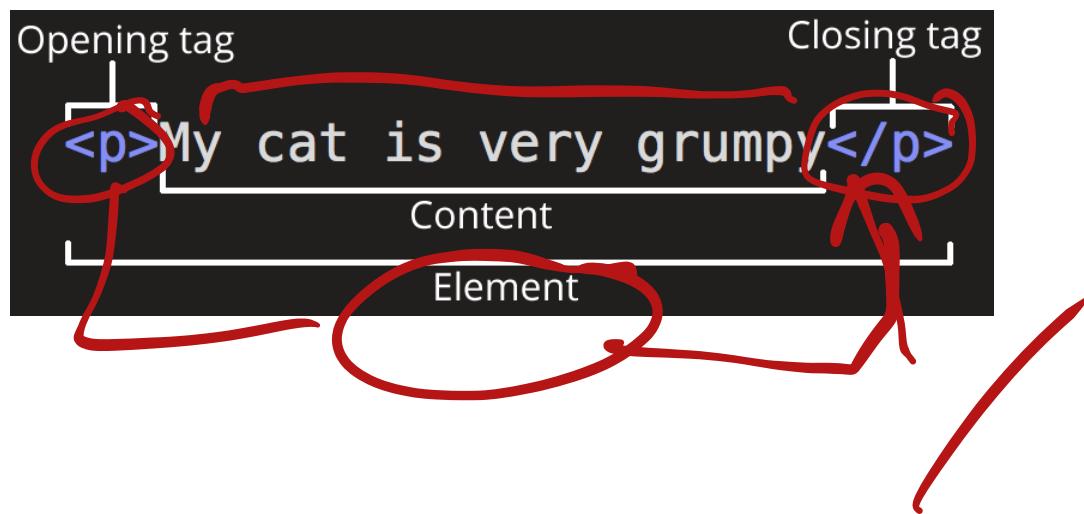
What does HTML actually do?

- Invented in 1989 as a way to create web pages for the internet.
- Uses HTML tags and attributes to define documents.
- **Tags** are used to create elements on a page and are signified by an opening tag <> and a closing tag </>.

```
<p>Hello this is my paragraph </p>
```

- **Attributes** are used to describe the characteristics of an HTML element in greater detail.

- ```
<p align="center"> Hello this is my paragraph. </p>
```



# A basic html webpage

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

XML, JSON

---

## My First Heading

My first paragraph.

**But wait, how do we make it  
look good?**

# CSS: Cascading Style Sheets

# CSS: Cascading Style Sheets

- CSS is the language of design.
- It's what controls the color, textures, and layout of a web page
- Use it to control how elements are displayed on a page both in location and in how they look.

# The old, awful way

---

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

My First Heading

My first paragraph.

# Abstraction: The Key Concept of CS

- Abstraction allows us to separate out components so they aren't tightly tied to each other
  - Java Virtual Machine separates code from underlying HW so you can run same program on any machine
  - DBMS separates physical implementation of data storage/indexing from the logical schema/query interface
  - HTML and CSS separates content and style
- Why is this so powerful and useful???

# Separating Style from Content

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<link rel="stylesheet" type="text/css" href="styles.css" >
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

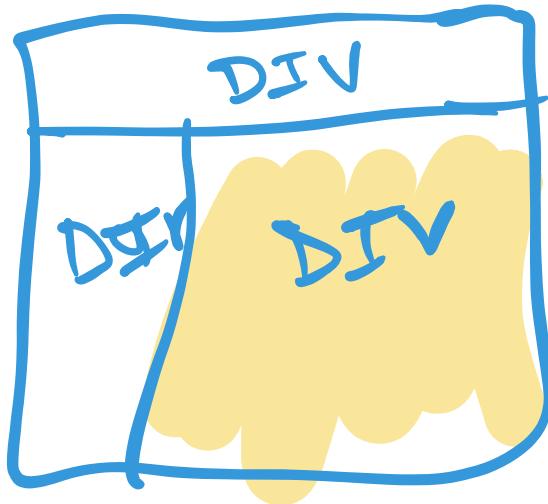
# Styling an html page

styles.css

```
body{
background-color: grey;
font-size: large;
color: red;
}

p{
background-color: lightgrey;
font-size: medium;
color: blue;
padding: 20pt;
}
```

DIV



```
selector{
property: prop-value; <-
Declaration
}
```

# Styling an html page

```
body{
 background-color: grey;
 font-size: large;
 color: red;
}

p{
 background-color: lightgrey;
 font-size: medium;
 color: blue;
 padding: 20pt;
}
```

My First Heading

My first paragraph.

# Classes vs ID's

- You can specify CSS styling based on tags, classes, and ids.
- Add an id to a tag if you want to be able to style that specific element only:
  - `<p id="style_only_this_one"> </p>`

uniq!

- `<p class="style_all_paragraphs"></p>`

# Classes vs ID's

- You can specify CSS styling based on tags, classes, and ids.
- Add an id to a tag if you want to be able to style that specific element only:
  - `<p id="style_only_this_one"></p>`
- Add a class to a tag if you want to style multiple elements on a page:
  - `<p class="style_all_paragraphs"></p>`

Use id's to style a specific element that appears only once and classes to style elements that appear repeatedly.

# Styling Classes and Id's

*background-image:*

```
p{
background-color: lightgrey;
font-size: medium;
color: blue;
padding: 20pt;
}

#style_only_this_one{
background-color: lightgrey;
font-size: medium;
color: blue;
padding: 20pt;
}

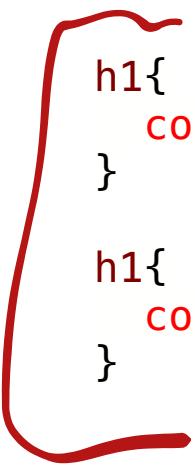
.style_all_paragraphs{
background-color: lightgrey;
font-size: medium;
color: blue;
padding: 20pt;
}
```

*ID*

*Class*

# CSS Inheritance

- CSS is called cascading because of inheritance.
- When multiple rules conflict with each other, styles cascade downwards thus applying only the last rule.



```
h1{
 color: red;
}

h1{
 color: blue;
}
```

**What color will h1 elements be on the page?**

# CSS Inheritance

- CSS is called cascading because of inheritance.
- When multiple rules conflict with each other, styles cascade downwards thus applying only the last rule.

```
h1{
 color: red;
}
```

```
h1{
 color: blue;
}
```

Because of inheritance only the last rule is applied. The heading is blue.

# CSS Specificity

- CSS rules with more specific selectors override CSS rules with less specific selectors regardless of order.
- The rules of specificity are as follows:
  - The least specific is an element tag: <p>
  - Using a ~~class~~ will override an element tag style: .myHeader
  - Using an ~~id~~ will override both a class and an element tag style: #myTitle
  - Using an in-line style on a tag will override anything else (generally should avoid this since it breaks abstraction)

# Lab Activities

- **Pre-Lab1 Hello World Wide Web:** basics of HTML tags, nesting, validation and DOM
  - not graded, due today for participation point
- **Lab1 Practice Student Roster:** try to make a page with a list of student names
  - **Team A:** not graded, see what you can figure out!
- **Lab1 Practice Lots of Tags:** walk through of most common HTML tags and CSS
  - not graded, follow along as we discuss
- **Lab1 Student Bios:** work in a small group to make a website with CSS styling
  - **Team B:** DUE Tuesday 1/18 at 11:59PM (usually will be a shorter deadline)
- **HW1 Python Training & Exercises:** work on your own to learn the basic syntax of python
  - DUE Tuesday 1/18 at 11:59PM

# Replit Lab Process

- Join the breakout room with your team number
- Say hello to your partner(s)
- ONE member should click the project on repl and CREATE a new team
- The other members should JOIN the team created by the first member (look for their repl username)
- Work on the lab together!
  - Be nice! Let everyone contribute!
- Click Submit when you are done
  - You can edit and submit again until the deadline

# Teams

Abdullah A. - Team A: 2 - Team B: 1  
Abel S. - Team A: 1 - Team B: 1  
Adham P. - Team A: 3 - Team B: 2  
Adil B. - Team A: 1 - Team B: 1  
Berk G. - Team A: 3 - Team B: 2  
Bradford S. - Team A: 18 - Team B: 9  
Chang P. - Team A: 20 - Team B: 10  
Claes B. - Team A: 18 - Team B: 9  
Clare J. - Team A: 17 - Team B: 9  
Colin R. - Team A: 4 - Team B: 2  
Damien V. - Team A: 4 - Team B: 2  
Dania A. - Team A: 12 - Team B: 6  
Daniel M. - Team A: 5 - Team B: 3  
Ethan C. - Team A: 5 - Team B: 3  
Evan F. - Team A: 6 - Team B: 3  
Hamza K. - Team A: 19 - Team B: 10  
Jackson C. - Team A: 7 - Team B: 4  
Jacob R. - Team A: 6 - Team B: 3  
Jay S. - Team A: 7 - Team B: 4  
Jiyan A. - Team A: 19 - Team B: 10  
Jonathan N. - Team A: 8 - Team B: 4  
Josh R. - Team A: 9 - Team B: 5  
Josie L. - Team A: 13 - Team B: 7  
Karl S. - Team A: 8 - Team B: 4  
Kate H. - Team A: 14 - Team B: 7

Kayla B. - Team A: 13 - Team B: 7  
Leo P. - Team A: 23 - Team B: 12  
Manue A. - Team A: 17 - Team B: 9  
Marcus L. - Team A: 20 - Team B: 10  
Maria G. - Team A: 12 - Team B: 6  
Marlee A. - Team A: 14 - Team B: 7  
Matthew C. - Team A: 21 - Team B: 11  
Matthew G. - Team A: 21 - Team B: 11  
Mikaela N. - Team A: 22 - Team B: 11  
Oury B. - Team A: 22 - Team B: 11  
Owen A. - Team A: 11 - Team B: 6  
Pravin K. - Team A: 9 - Team B: 5  
Rhonda Z. - Team A: 23 - Team B: 12  
Ryah C. - Team A: 15 - Team B: 8  
Sam K. - Team A: 15 - Team B: 8  
Sarah J. - Team A: 16 - Team B: 8  
Selman E. - Team A: 10 - Team B: 5  
Sidra H. - Team A: 16 - Team B: 8  
Steven Y. - Team A: 10 - Team B: 5  
Surya T. - Team A: 24 - Team B: 12  
Tanim K. - Team A: 11 - Team B: 6  
Timothy L. - Team A: 24 - Team B: 12  
Vinay M. - Team A: 24 - Team B: 12  
Viraj P. - Team A: 2 - Team B: 1

# Attributions

These slides are adapted from materials made by  
Prof. Bhagi Narahari

## Image attribution:



Created by Wilson Joseph  
from Noun Project



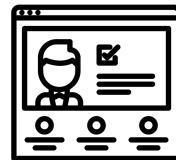
Created by Ditta  
from Noun Project



Created by Gregor Cremer  
from Noun Project



Created by Wilson Joseph  
from Noun Project



Created by lastspark  
from Noun Project



Created by Dawid Sobolewski  
from Noun Project



Created by Wilson Joseph  
from Noun Project



Created by Nataša  
from Noun Project



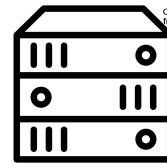
Created by srujanreddy  
from Noun Project



Created by jinu.mallick9  
from Noun Project



Created by jinu.mallick9  
from Noun Project



Created by Senthil Agra  
from Noun Project



Created by Yvesse Akans  
from Noun Project