Midterm 1 Notes

Topics Covered:

- Front End Development
- Git (GitHub)
- Unix Commands
- HTML
- CSS
- Java Script

Front-End Development Notes

How the Internet Works

- **Misconception:** The internet is not a "cloud."
- **Reality:** It's a global network of cables and computers.
    - Example: Computer in Seattle can directly connect with one in Spain.
- **Servers:** Some computers stay online 24/7 to serve data (websites, files).

Process of Visiting a Website

1. You type a URL (e.g., `google.com`).
2. Your **ISP** receives the request and forwards it to a **DNS server**.
3. **DNS:** Acts like a phonebook – returns the IP address of the website.
    - Every device on the internet has an **IP address**.
4. Browser sends request to that IP via the **Internet Backbone** (submarine cables).
5. Server responds with website data.

Website Files

- **HTML:** Structure of a webpage. Think of it as the house's frame.
- **CSS:** Styling. Adds paint, furniture, and design to the house.
- **JavaScript:** Behavior. Adds wiring, lights, and interactivity.

Extra Exam Notes

- **ISP** = Internet Service Provider (who you pay for internet).
- **Internet Backbone** = Physical network of undersea fiber-optic cables.
- Browsers (Chrome, Firefox, Safari) all **render HTML, CSS, JS**.

Git & GitHub Notes

What is Version Control?

- **Definition:** System to manage changes to files/projects.
- **Uses:**
  - Work in teams without interference.
  - Roll back to previous versions.
  - Track *who changed what, when*.

Why Git?

- **Distributed** Version Control System (DVCS).
- Most popular in industry → transferable skills.

Key Terminology

- **Repository (repo):** Project directory tracked by Git.
- **Commit:** Snapshot of changes (like a video game save point).
- **SHA:** Unique commit ID (long hash string).
- **Branch:** Independent line of development.

- **Checkout:** Switching to a branch/commit/file.
- **Staging Area:** Prepares files for a commit.

Common Git Commands

- `git init` → start repository
- `git status` → check working directory state
- `git add <file>` or `git add .` → stage changes
- `git commit -m "msg"` → save snapshot
- `git log` → view history
- `git clone <url>` → copy repo
- `git push` / `git pull` → sync with remote
- `git branch <name>` / `git checkout <name>` → branch mgmt
- `git merge <branch>` → merge changes
- `git revert <SHA>` → undo commit

GitHub

- Remote hosting platform for Git repositories.
- Enables collaboration, pull requests, project visibility.

Extra Exam Notes

- **Merge conflicts** happen when changes overlap → must resolve manually.
- **Best Practice:** Commit often with meaningful messages.
- **Pull Requests:** GitHub feature to propose changes before merging.

Unix Commands Notes

Navigation

- `pwd` → print working directory.

- `cd` → change directory (`cd -` returns to last dir, `cd` alone goes home).
- `ls`, `ls -l`, `ls -lrt` → list files (long format, sorted).
- `mkdir <name>` → create directory.

File Management

- `mv old new` → move/rename file.
- `cp old new` → copy file.
- `rm file` / `rm -f` → delete permanently (no recycle bin).

Viewing Files

- `more file` → view text page by page.
- `less file` → better than `more`, allows search.

Permissions

- `chmod a+r file` → everyone can read.
- `chmod u+rwx file` → user gets full access.
- `chmod -R u+rw dir` → recursive permission change.

Processes

- `top` → live CPU/memory usage.
- `ps` → list running processes.
- `jobs` → background jobs.
- `kill <PID>` / `kill -9 <PID>` → terminate process.

Utilities

- `man <cmd>` → manual/help.
- `gzip file` → compress.
- `gunzip file.gz` → decompress.
- `find ./ -name "*.txt"` → search files.

- `df` → disk space usage.
- `du -sk . | sort -g` → folder size usage.

Extra Exam Notes

- **Wildcards:** = any string, `?` = single char.
- **Tilde (~):** shorthand for home directory.
- **No undo in Unix:** Be cautious with `rm`.

HTML Notes

Basics

- **HTML = Hypertext Markup Language** → structure of all websites.
- **Markup Language:** Uses tags (`<tag>`) to define structure.
- **Boilerplate:** Standard template including `<!DOCTYPE html>`, `<html>`, `<head>`, `<body>`.

Tags

- **Headings:** `<h1>` ... `<h6>`
- **Paragraphs:** `<p>`
- **Lists:**
  - Ordered → `<ol><li></li></ol>`
  - Unordered → `<ul><li></li></ul>`
- **Images:** `<img src="url" alt="desc">` (self-closing).
- **Links:** `<a href="url">text</a>`
- **Tables:** `<table><tr><th><td></td></th></tr></table>`
- **Forms:** `<form> <input> <label> </form>`

Attributes

- Provide extra info about an element. Example:

```
<img src="cat.jpg" alt="A cat">
```

Hosting Websites

- Local files are only visible to you.
- To publish, use services like **GitHub Pages** (free).

Extra Exam Notes

- **HTML5:** Current version, supports multimedia (`<video>`, `<audio>`).
- **Semantic Tags:** `<header>`, `<footer>`, `<article>` → improve readability & SEO.
- **UTF-8 Charset:** Ensures text displays correctly worldwide.

CSS Notes

Why Was CSS Created?

- Originally, HTML was used to handle both structure and design (e.g., `<center>`, `<h1 bgcolor="#990000">`).
- Problem: mixing layout with content (e.g., using tables for positioning).
- **CSS separates content from design**, making styling more consistent and reusable.

CSS Types

Inline CSS

- Applies style to **one element at a time**.
- Example:

```
<body style="background-color:aquamarine;">
```

Internal CSS

- Applies styles **to a single HTML page**.
- Defined inside `<head>` → `<style>` tag.
- Syntax: `selector { property: value; }`.

External CSS

- CSS written in a **separate `.css` file**.
- Best practice: separates design from content.
- Linked in HTML using `<link rel="stylesheet" href="style.css">`.

HTML Elements in CSS

Block Elements

- Start on a **new line**.
- Occupy the full width (e.g., `<p>`, `<div>`, `<h1>`).

Inline Elements

- Only take as much space as needed (e.g., `<span>`, `<a>`, `<img>`).

Inline-Block

- Behaves like inline but **accepts width/height** changes.

None

- Hides the element completely.

CSS Syntax (The Anatomy)

- **Selector**: “Who” to style (e.g., `h1`).
- **Property**: “What” to change (e.g., `color`).
- **Value**: “How” to change (e.g., `blue`).

Example:

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

Selectors

- **Tag Selector**: applies to all instances of a tag (`h1 {}`).
- **Class Selector**: targets elements with a `class` (`.classname {}`).
- **ID Selector**: targets element with a unique `id` (`#idname {}`).
- **Specificity Rules**:
  - IDs > Classes > Tags.
- **Pseudo-classes**: define different element states (e.g., `:hover`, `:first-child`).

Divs & Structure

- `<div>`: **generic container** with no meaning, used to structure layouts.
- Helps split content into **separate boxes** for styling.

CSS Box Model

Every element is a box made of:

1. **Content** – text, images.
2. **Padding –** space between content & border.
3. **Border** – surrounds padding/content.
4. **Margin** – space between element and others.

Key Notes:

- Units: `px`, `%`, `em`, `rem`.

- Box total size = content + padding + border + margin.
- Shorthand order: **top → right → bottom → left**.

CSS Display Property

- **Block**: takes full width, forces line break. (e.g., `<div>`, `<p>`)
- **Inline**: fits within text flow (e.g., `<span>`, `<a>`).
- **Inline-block**: inline positioning + block sizing.
- **None**: hides element.

CSS Positioning

Static

- Default flow of HTML.

Relative

- Positioned relative to **its normal place**.
- Uses: `top`, `bottom`, `left`, `right`.
- Other elements **not affected**.

Absolute

- Positioned relative to the **nearest parent with positioning**.
- Removes element from normal flow.

Fixed

- Stays **in same position on screen** (e.g., navbars).

Font Styling

- Font families:
  - **Serif** (traditional, decorative strokes).

    - **Sans-serif** (clean, modern).
- Use **Google Fonts** for custom typefaces.
- Example:

```
font-family: Verdana, sans-serif;
```

CSS Sizing

- **Pixels (px)**: fixed size.
- **em**: relative to parent (`1em = 16px`).
- **rem**: relative to root (`html`) element, ignores parents.
- **%**: relative to parent's size.

Why use em/rem?

- **Accessibility**: scales with browser/user settings.
- **px** is fixed → not responsive.

Favicons

- Small icon representing the website in browser tabs.
- Added via:

```
<link rel="icon" href="favicon.ico" type="image/x-icon">
```

---

1. Bootstrap

Overview

- **Created by:** Mark Otto and Jacob Thornton (Twitter, 2010)
- **Purpose:** Front-end CSS framework for building responsive websites efficiently.

- **Open-source:** Available on GitHub → github.com/twbs/bootstrap Bootstrap_1

Key Concepts

- **Front End:** User-visible part of a website/app.
- **Back End:** Server logic, data, and processing layer.
- **Responsiveness:** Automatically adapts to different viewports (desktop, tablet, mobile).

Benefits

- Pre-built components (buttons, navbars, grids).
- Consistency and rapid UI development.
- Supports responsive design by default.

Core Components

- **Grid System:**
  - 12-column layout built on Flexbox.
  - Uses `.container`, `.row`, and `.col-*` classes.
- **Breakpoints:** Adjust layout based on screen width (`<576px`, etc.).
- **Buttons:** Use predefined classes like `.btn`, `.btn-primary`.
- **Navs/Jumbotron:** Quick creation of headers, navigation bars, and hero sections.

Installation

- Add via **CDN** (copy CSS/JS links from getbootstrap.com) or download starter template.

Wireframing

- Sketch layout and structure before coding.
- Common workflow: **Wireframe → Mockup → Implementation**.

---

2. CSS Flexbox

Purpose

- Simplifies layout alignment compared to floats or positioning.
- Enables responsive, dynamic resizing of containers and itemsCSS Flexbox.

Float (Old Approach)

- Use only for text wrapping around images.
- Avoid for page layout.

Flexbox Basics

- **Parent container:** `display: flex` or `display: inline-flex`.
- **Main Axis:** Direction of item placement (`row` by default).

Core Properties

Property	Applied To	Description
`flex-direction`	parent	Sets main axis (`row` or `column`)
`flex-wrap`	parent	Controls wrapping (`nowrap` or `wrap`)
`justify-content`	parent	Aligns items on main axis (`flex-start`, `center`, `space-between`, etc.)
`align-items`	parent	Aligns items on cross axis
`order`	child	Controls order of items
`flex-basis`	child	Defines base size of an element

Key Notes

- Default layout = `row` direction.
- Wrapping helps prevent overflow.
- Understanding parent vs child properties is essential.

---

3. Document Object Model (DOM)

Concept

- **DOM:** Tree-structured representation of an HTML document.
- **Purpose:** Allows JavaScript to dynamically read and modify webpage contentDOM.

Types of JS Integration

Type	Description	Recommended?
Inline JS	Code in element attributes	❌ Not modular
Internal JS	Code inside `<script>` in HTML	✅ OK for small scripts
External JS	Linked JS file via `<script src="">`	✅ Best practice

Structure

- Browser converts HTML → **DOM Tree**.
- Each HTML tag becomes a **node/object**.
- Elements relate as **parent**, **child**, or **siblings**.

Selecting Elements

Method	Description	Returns
`getElementById("id")`	Selects by ID	Single element
`getElementsByClassName("class")`	Selects by class	HTMLCollection
`getElementsByTagName("tag")`	Selects all tags	HTMLCollection
`querySelector("selector")`	Returns first match	Single element
`querySelectorAll("selector")`	Returns all matches	NodeList

Manipulation

- **Change content:** `element.innerHTML` or `element.textContent`
- **Change style:** `element.style.property = "value"`
- **Add/remove classes:** `element.classList.add("className")`
- **Change attributes:** `element.setAttribute("attr", "value")`

Separation of Concerns

- **Structure:** HTML
- **Style:** CSS
- **Behavior:** JavaScript

---

4. JavaScript

Background

- **Created by:** Brendan Eich (1995) in 10 days at Netscape.
- Initially called *LiveScript*.
- Adds dynamic behavior to static HTML/CSS websitesJavaScript.

Data Types

- **String:** `"text"`
- **Number:** `1, 2, 3`
- **Boolean:** `true / false`

Variables

- Declare using `var`, `let`, or `const`.
- Follow naming conventions:
  - No spaces or leading numbers.
  - Use camelCase (`userName`).
  - Must be descriptive.

Strings

- **Concatenation:** `"a" + "b" = "ab"`
- **Length:** `"text".length`
- **Slice:** `"hello".slice(0, 2)` → `"he"`

Arithmetic

- Basic operators: `+`, , , `/`, `%`
- `%` gives remainder → used to check even/odd.

Functions

- Define using `function name() {}`
- **Parameters:** Inputs to the function.
- **Return:** Output value.
- **Arrow functions (ES6):** `const add = (a, b) => a + b`

Conditionals

- `if / else` statements control flow.
- Comparators: `==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, `<=`
- Logical operators: `&&`, `||`, `!`

Arrays

- Store multiple items in one variable.
  - Example: `let cars = ["BMW", "Volvo", "Saab"]`
- **Methods:**
  - `push()` → Add item
  - `pop()` → Remove last item

Loops

- **While loop:** Repeats while condition is true.
- **For loop:** Runs fixed number of times.
- Example:

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

Randomization Example

```
Math.floor(Math.random() * 10);
```

Generates a random integer between 0–9.