GITHUB

GitHub is a developer platform that allows developers to create, store, manage and share their code. It uses Git software, providing the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project. GitHub provides a centralized location for developers to work on projects, share their code, and collaborate with others.

Basic Git Commands

Repository: A storage location where your project lives, containing all the files and revision history.

Branch: A version of the codebase that diverges from the main branch to isolate changes for specific features, fixes, or experiments.

```
$ git branch <branchname> (create new branch)
```

Commit: A snapshot of your changes, saved to your local repository. Each commit is uniquely identified by a checksum.

```
$ git commit m "Add file to feature branch"
```

Merge: The process of integrating changes from one branch into another, typically the main branch.

Pull Request: A proposal to merge changes from one branch into another, often used in collaborative environments to review and discuss changes before they are merged.

Fork: A personal copy of someone else's project that lives on your GitHub account.

Clone: The act of downloading a repository from a remote source to your local machine.

```
$ git clone https://github.com/YOURUSERNAME/YOURREPOSITORY
```

Push: The action of sending your commits to a remote repository.

```
$ git push origin featurebranch
```

Pull: The action of fetching changes from a remote repository and merging them into your current branch.

DATABASE

A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner. It allows users to create, modify, and query a database, as well as manage the security and access controls for that database.

Key features of DBMS are data modeling, data storage and retrieval, concurrency control, data integrity and security, backup and recovery, etc.

The different types of database languages are:

Data definition language which deals with database schemas and descriptions, of how the data should reside in the database. (CREATE, ALTER, DROP, RENAME)

Data manipulation language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

Data control language which acts as an access specifier to the database.basically to grant and revoke permissions to users in the database (GRANT, REVOKE)

Transaction control language which acts as a manager for all types of transactional data and all transactions. (ROLL BACK, COMMIT)

Queries

SQL is the standard language used to communicate with a relational database. It is used to perform various operations such as querying, updating, and managing data.

SELECT: retrieves data from database

```
SELECT column1, column2 FROM table_name;
```

UPDATE: Modifies existing data in database

```
UPDATE table_name SET column1 = value1 WHERE condition;
```

DELETE: removes data from database

```
DELETE FROM table_name WHERE condition;
```

ALTER TABLE: Modifies the structure of an existing table

```
ALTER TABLE table_name
ADD column_name datatype;
```

KEYS IN DBMS

Keys in a DBMS ensure that each record within a table can be uniquely identified.

Primary key: A column or a combination of column that uniquely identifies each row in a table. A table can only have one primary key

Foreign key: Key that uniquely identifies rows in another table. It establishes relation between the tables.

Unique key: Ensures that all values in a column are distinct

Candidate key: A column, set of columns that can qualify as a primary key.

ER DIAGRAMS

ER Diagrams are a visual representation of the database structure. They illustrate the entities in the system and the relationships between them.

Entities: Objects or things in a system, represented by rectangles

Attributes: Properties of entities, represented by ovals

Relationships: Association between entities, represented by diamonds

HTML

HTML (HyperText Markup Language) is the standard language for creating web pages. It describes the structure of a webpage using a system of nested elements. It consists of a series of elements (tags) that describe the structure and content of a webpage. Forms are an essential part of HTML, allowing users to submit data to a server.

Tags

<html>: The root element of an html document

<body>: Contains content of the document

<h1> to <h6>: Headings where <h1> is the highest level and <h6> is the lowest

<p>: paragraph

<b>: bold text

<u>: underline

<br>: Line break

<ul>: unordered list

<ol>: ordered list

<li>: List item

<a>: anchor used to create hyperlinks

<img>: Image

<table>: Defines a table

<tr>: table row

<th>: table header

Form Elements

<form>: The container for all form elements

<label>: Defines a label for an input element

<input>: Defines an input field

<button>: Defines a clickable button

CSS

CSS is a styling language used to control the presentation and layout of HTML documents.

Basic Structure:

```
        Selector {
                Property: value;
        }
```

Inline CSS: Applied directly within the HTML element using the ‘style’ attribute

<p style="color: red; fontsize: 16px;">This is a paragraph with inline CSS.</p>

Internal CSS: Defined within the <style> element in the <head> section of an HTML document

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
            fontfamily: Arial, sansserif;
        }
        h1 {
            color: blue;
        }
    </style>
</head>
<body>
    <h1>This is a heading with internal CSS.</h1>
    <p>This is a paragraph styled with internal CSS.</p>
</body>
</html>
```

External CSS: Defined in a separate CSS file and linked to an HTML document using the <link> element

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <h1>This is a heading styled with external CSS.</h1>
    <p>This is a paragraph styled with external CSS.</p>
</body>
</html>
```

Styles.css

```
body {
    fontfamily: Arial, sansserif;
}


h1 {
  color: green;
}
```

Basic CSS Selectors

Element Selector: Targets all elements of a specific type.

```
p { color: blue; }
```

Class Selector: Targets elements with a specific class attribute, using a dot (.) before the class name.

```
.example { color: red; }
```

ID Selector: Targets a single element with a specific ID attribute, using a hash (#) before the ID name.

```
#unique { color:green; }
```

Styling Text

Color: Sets the text color.

```
p { color: red; }
```

Font Size: Sets the size of the text.

```
p { fontsize: 18px; }
```

Font Family: Sets the font of the text.

```
p { fontfamily: Arial, sansserif; }
```

Styling Background

Background Color: Sets the background color of an element.

```
body { backgroundcolor: lightblue; }
```

Background Image: Sets a background image for an element.

```
body { backgroundimage: url('background.jpg'); }
```

JAVASCRIPT

JavaScript is a highlevel, interpreted programming language primarily used for clientside web development.

Syntax:

Variables: Declared using `var`, `let`, or `const`. Example: `let name = 'John';`

Data Types: Includes `number`, `string`, `boolean`, `null`, `undefined`, `object`, `symbol`.

Operators: Arithmetic (`+`, ``, `*`, `/`), assignment (`=`, `+=`, `=`), comparison (`==`, `===`, `!=`, `!==`), logical (`&&`, `||`, `!`), etc.

Control Flow:

Conditional Statements: `if`, `else if`, `else`, `switch`.

```
if (condition) {
    // code block
} else if (anotherCondition) {
    // code block
} else {
    // code block
}
```

Loops: `for`, `while`, `dowhile`, `for...in`, `for...of`.

```
for (let i = 0; i < 5; i++) {
    // code block
}
```

Functions: Declared using `function` keyword or arrow functions (`=>`).

```
function greet(name) {
    return `Hello, ${name}!`;
}
```

Objects and Arrays:

Objects: Keyvalue pairs enclosed in `{}`.

```
javascript
let person = {
    name: 'John',
    age: 30,
    isAdmin: true
};
```

Arrays: Ordered collections of values enclosed in `[]`.

```
javascript
let numbers = [1, 2, 3, 4, 5];
```

Events and Event Handling:

Events: User actions or occurrences detected by the browser.

Event Handlers: Functions that respond to events.

```
javascript
document.getElementById('myButton').addEventListener('click', function() {
    alert('Button clicked!');
});
```

REACT

React is a JavaScript library for building user interfaces.

```
import React from 'react';

function App() {
  return (
    <div>
      <h1>Hello, React!</h1>
      <p>This is a paragraph in JSX.</p>
    </div>
  );
}

export default App;
```

Components:

Building Blocks: Encapsulate UI functionality.

Functional Components: Simple functions returning JSX.

```
function Welcome(props) {
  return <h1>Hello, {props.name}!</h1>;
}
```

Class Components: ES6 classes extending `React.Component`.

```
class Welcome extends React.Component {
        render() {
            return <h1>Hello, {this.props.name}!</h1>;
        }
    }
```

Props and State:

Props (Properties): Passed from parent to child components.

State: Managed within a component, mutable and used for dynamic UI.

```
    class Counter extends React.Component {
        constructor(props) {
            super(props);
            this.state = { count: 0 };
        }

        render() {
            return (
                <div>
                    <p>Count: {this.state.count}</p>
                    <button onClick={() => this.setState({ count: this.state.count + 1
})}>Increment</button>
                </div>
            );
        }
    }
```

Creating Forms in React

Controlled Components in React:

Definition: Inputs where form data is controlled by React state.

Example:

```
 import React, { useState } from 'react';


 function MyForm() {
    const [name, setName] = useState('');


    const handleChange = (event) => {
       setName(event.target.value);
    };


    const handleSubmit = (event) => {
       event.preventDefault();
       alert(`You typed: ${name}`);
    };


    return (
       <form onSubmit={handleSubmit}>
          <label>
             Name:
             <input type="text" value={name} onChange={handleChange} />
          </label>
          <button type="submit">Submit</button>
       </form>
    );
 }
 export default MyForm;
```

Form Handling

Event Handlers: `onChange`, `onSubmit`.

State Management: Use `useState` hook to manage input values.

Form Validation:

ClientSide Validation: Validate input fields before submission.

Example:

```
 function MyForm() {
    const [email, setEmail] = useState('');
    const [isValid, setIsValid] = useState(false);

    const handleChange = (event) => {
       setEmail(event.target.value);
       // Perform validation
       setIsValid(event.target.value.includes('@'));
    };

    const handleSubmit = (event) => {
       event.preventDefault();
       alert(`Email submitted: ${email}`);
    };

    return (
        <form onSubmit={handleSubmit}>
          <label>
            Email:
            <input type="text" value={email} onChange={handleChange} />
          </label>
          <button type="submit" disabled={!isValid}>Submit</button>
        </form>
    );
}
```