FULL STACK DEVELOPMENT WORKSHOP

BY LIMAT SOFT SOULUTIONS PVT LIMITED

Curriculum (30 Hours)

Introduction to Full Stack Development (2 hours)

Frontend Development (8 hours)

Backend Development (8 hours)

Full Stack Application Development (6 hours)

Advanced Topics (4 hours)

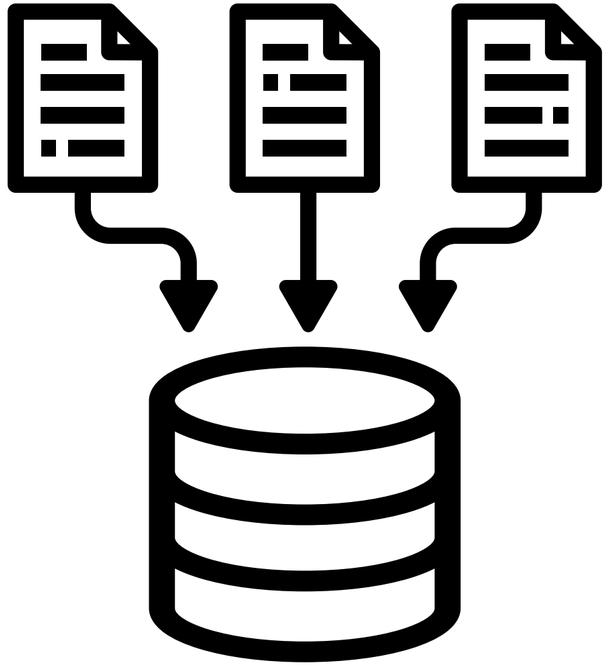
Final Project and Review (2 hours)

1. Introduction to Full Stack Development (2 hours)

- Overview of Full Stack Development
- Understanding Frontend, Backend, and Databases
- Tools and Technologies (HTML, CSS, JavaScript, Node.js, Express, MongoDB, etc.)

Full Stack Development

Overview and Fundamentals



What is Full Stack Development?

Full Stack Development- refers to the process of developing both the frontend (client-side) and backend (server-side) parts of a web application. A Full Stack Developer is proficient in all layers of a web application's architecture, which includes:

Frontend Development: Involves designing and developing the user interface (UI) that users interact with, using technologies like HTML, CSS, JavaScript, and frontend frameworks (e.g., React, Angular).

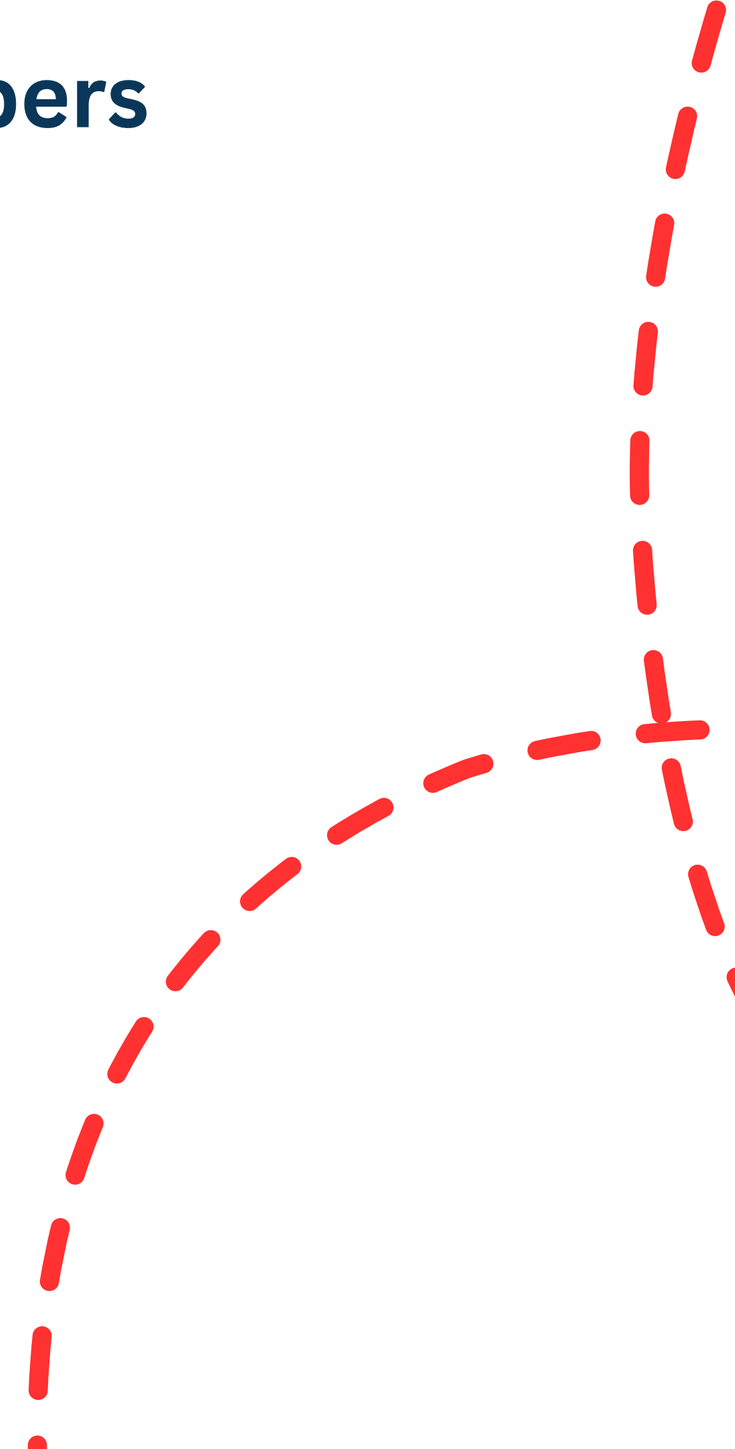
Backend Development: Focuses on server-side logic, databases, APIs, and business processes that power the frontend, using technologies like Node.js, Express, Python, or Java, and databases like MongoDB, MySQL, or PostgreSQL.

Importance of mastering both frontend and backend

1. Versatility and Flexibility
2. Better Collaboration and Communication
3. Improved Problem Solving
4. Higher Demand and Salary Potential
5. Complete Ownership of Projects
6. Streamlined Development Process

Career opportunities for full stack developers

1. Full Stack Developer
2. Web Developer
3. Software Engineer
4. DevOps Engineer
5. Freelancer / Consultant

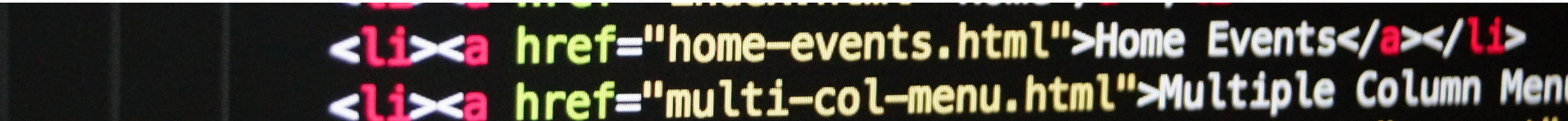


OVERVIEW OF FRONTEND DEVELOPMENT

Technologies: HTML, CSS, JavaScript

What is client-side development?

Importance of UI/UX in frontend development



Overview of Backend Development

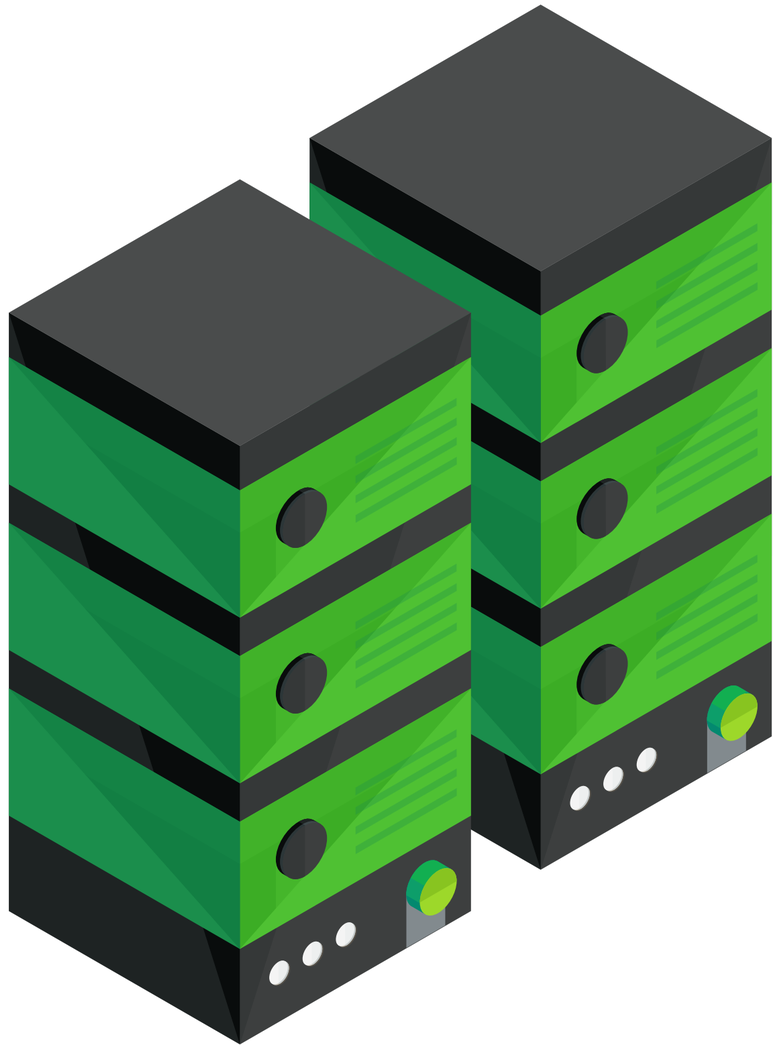
Technologies: Node.js, Express

Server-side programming

Handling business logic and database operations

Overview of Databases

- Types of databases: SQL vs NoSQL
- Introduction to MongoDB (NoSQL database)
- Why databases are crucial for dynamic web apps



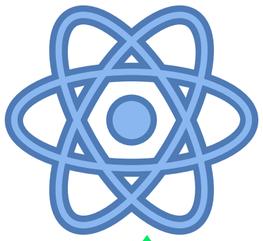
Tools & Technologies for Full Stack Development

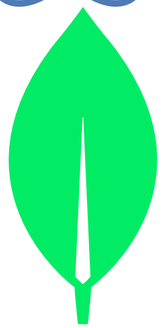
Frontend: HTML, CSS, JavaScript, React

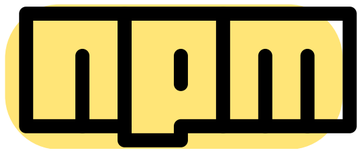
Backend: Node.js, Express

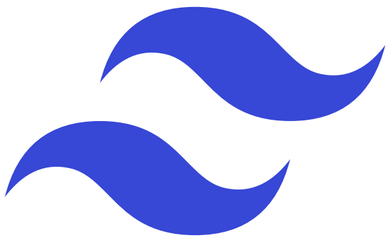
Database: MongoDB

Other tools: Git, VSCode, Postman

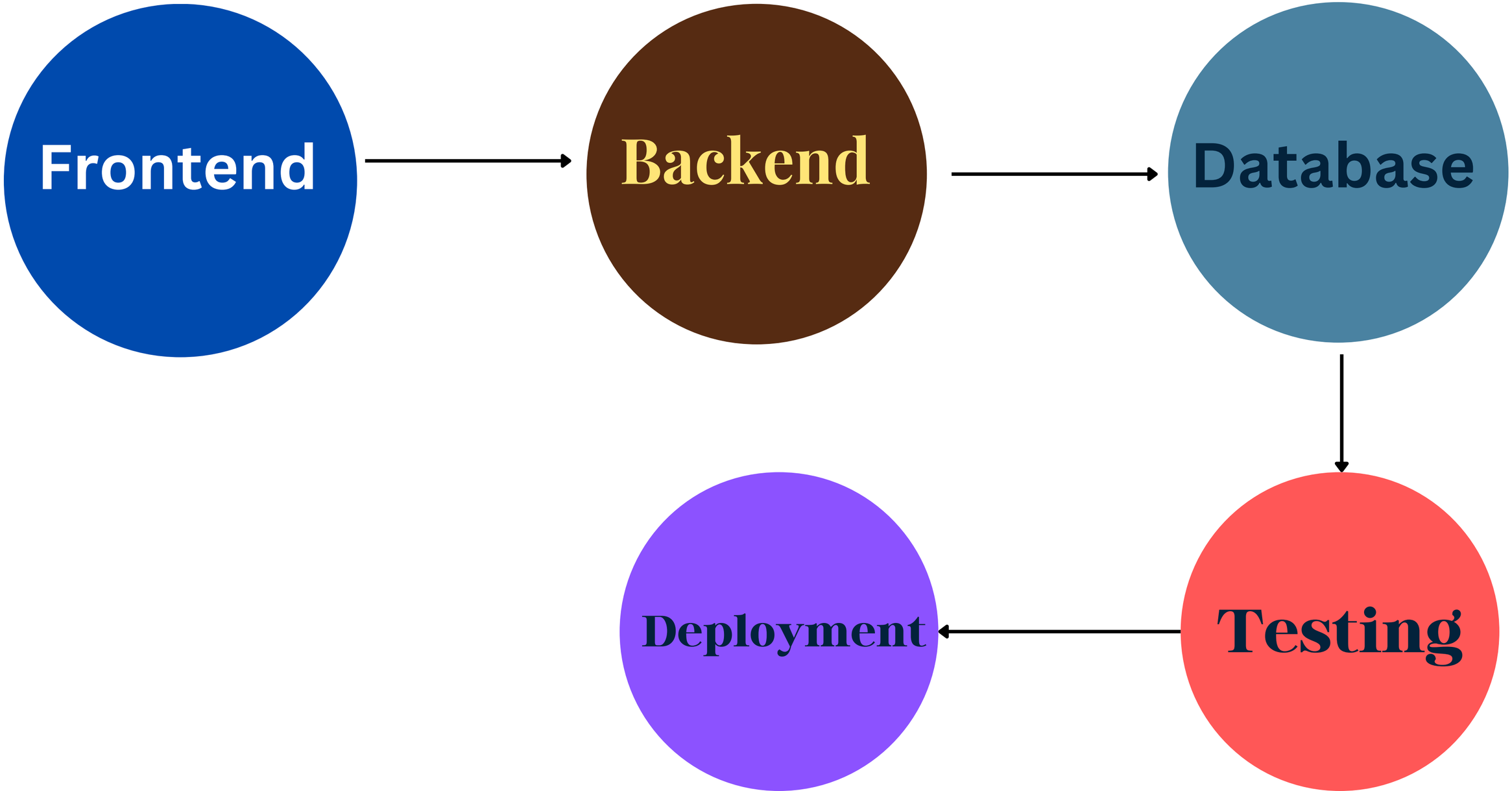








The Full Stack Development Workflow



HTML, CSS, and JavaScript

Role of HTML: Structure of web pages

Role of CSS: Styling and layout

Role of JavaScript: Interactivity and functionality

Introduction to Node.js

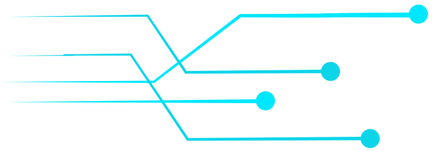
- What is Node.js and why it’s important for full stack development?
- Advantages of using JavaScript for both frontend and backend
- Node.js as a runtime environment for server-side code

Introduction to Express.js

What is Express.js?

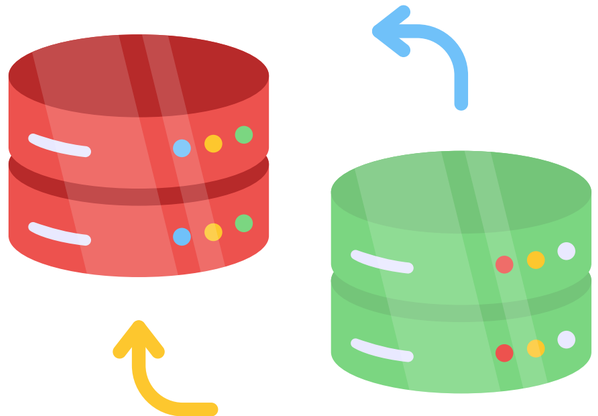
How it simplifies building web servers and APIs

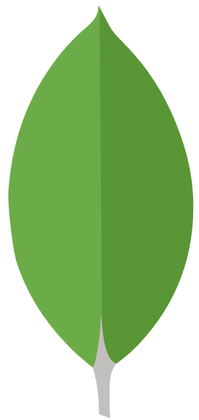
Routing and middleware in Express



Introduction to MongoDB

- What is MongoDB?
- Difference between NoSQL and relational databases
- Why MongoDB is used in modern web apps





let's start.....

HTML AND CSS

BASICS

Agenda:

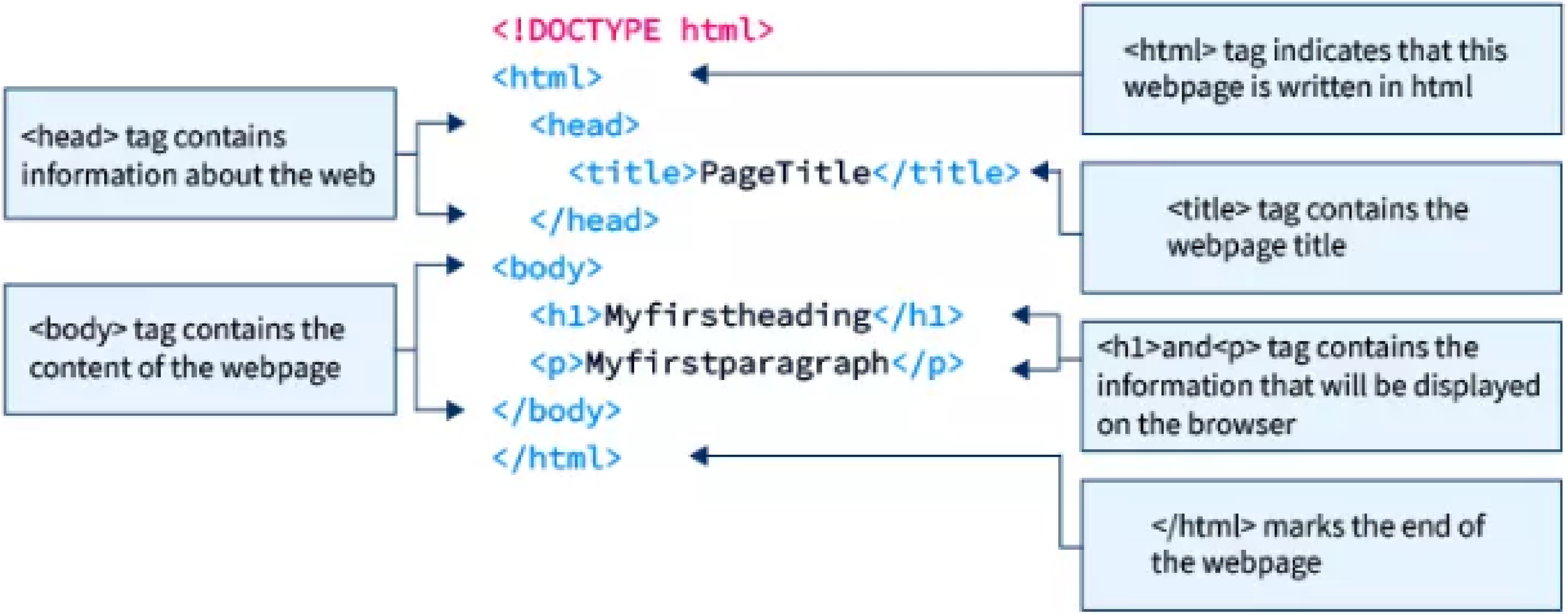
- Structure of HTML (Tags, Elements, Attributes)
- Styling with CSS (Selectors, Flexbox, Grid)
- Responsive Design (Media Queries, Mobile-first Design)
- CSS Frameworks: Tailwind CSS

What is HTML?

Hypertext Markup Language (HTML) is the standard language for creating web pages.
HTML describes the structure of web pages using tags.

```
<html>
  <head><title>My Web Page</title></head>
  <body><h1>Hello World!</h1></body>
</html>
```

HTML Document Structure



HTML Tags

Tags define the structure and content of elements on the page.

Example: <h1>, <p>, <img>, .

Self-Closing Tags

List of HTML Tags

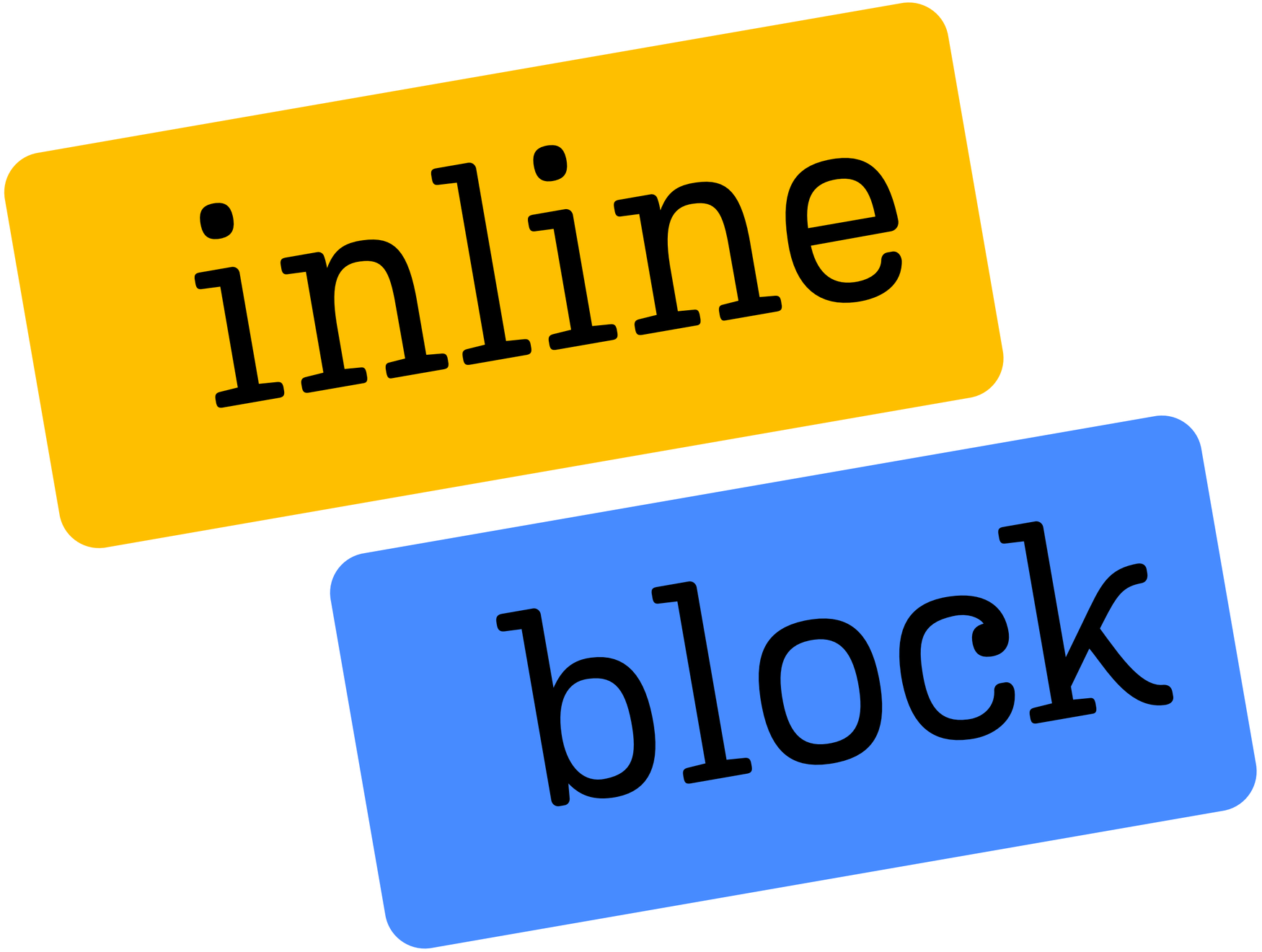
- <!-...->
- <IDOCTYPE>
- <a>
- <abbr>
- <acronym>
- <address>
- <applet>
- <area>
- <article>
- <aside>
- <audio>
- <b>
- <base>
- <basefont>
- <bdi>
- <bdo>
- <big>
- <blockquote>
- <body>
-
- <button>
- <canvas>
- <caption>
-
- <cite>
- <code>
- <col>
- <colgroup>
- <data>
- <datalist>
- <dd>
- <del>
- <details>
- <dfn>
- <dialog>
- <dir>
-
- <dl>
- <dt>
-
- <embed>
- <fieldset>
- <figcaption>
- <figure>
- <font>
- <footer>
- <form>
- <frame>
- <frameset>
- <h1>to<h6>
- <head>
- <header>
- <hr>
- <html>
- <i>
- <iframe>
- <img>
- <input>
- <ins>
- <kbd>
- <label>
- <legend>
-
- <link>
- <main>
- <map>
- <mark>
- <meta>
- <meter>
- <nav>
- <noframes>
- <noscript>
- <object>
-
- <optgroup>
- <option>
- <output>
- <p>
- <param>
- <picture>
- <pre>
- <progress>
- <q>
- <rp>
- <rt>
- <ruby>
- <s>
- <samp>
- <script>
- <section>
- <select>
- <small>
- <source>
-
- <strike>
-
- <style>
- <sub>
- <summary>
- <sup>
- <svg>
- <table>
- <tbody>
- <td>
- <template>
- <textarea>
- <tfoot>
- <th>
- <thead>
- <time>
- <title>
- <tr>
- <track>
- <tt>
- <u>
-
- <var>
- <video>
- <wbr>

HTML Attributes

attributes provide additional information about elements (e.g., src, alt, href).
Attributes are written within the opening tag.

```
<a href="https://example.com">Visit Example</a>

<img src="image.jpg" alt="Image description" />
```


inline
block

Semantic HTML

Semantic tags clearly describe their purpose: <header>, <footer>, <article>, <section>.
Improves SEO and accessibility.

```
<header><h1>Welcome to My Website</h1></header>
<section><p>This is an article about web development.</p></section>
```

Project: Personal Profile Page

LETS STYLE!

What is CSS?

- **Cascading Style Sheets (CSS) is used to control the appearance of HTML elements.**
- **CSS applies styles such as colors, fonts, layout, and spacing.**

Inline, Internal, and External CSS

Inline CSS: style attribute in HTML tags.

Internal CSS: <style> inside the <head> section.

External CSS: Separate .css file linked with <link>.

Inline CSS

Internal CSS

External CSS

CSS Selectors

- TAG SELECTOR
- ID SELECTOR
- CLASS SELECTOR

Styling Text in CSS

Font-family

font-size

font-weight

color

text-align

Styling Backgrounds in CSS

- background-color
- background-image
- background-repeat

Margins and Padding

Margins: Space outside an element.

Padding: Space inside an element.

Display Property

```
.block-element {
    display: block;
}

.inline-element {
    display: inline;
}
```

INTRODUCTION TO FLEXBOX

- Flexbox is a layout model that makes it easy to align items.
- Properties: display: flex;, justify-content, align-items.

Flexbox: Flex Direction

The flex-direction property defines the direction of the flex items: row, column.

MEDIA QUERIES IN CSS

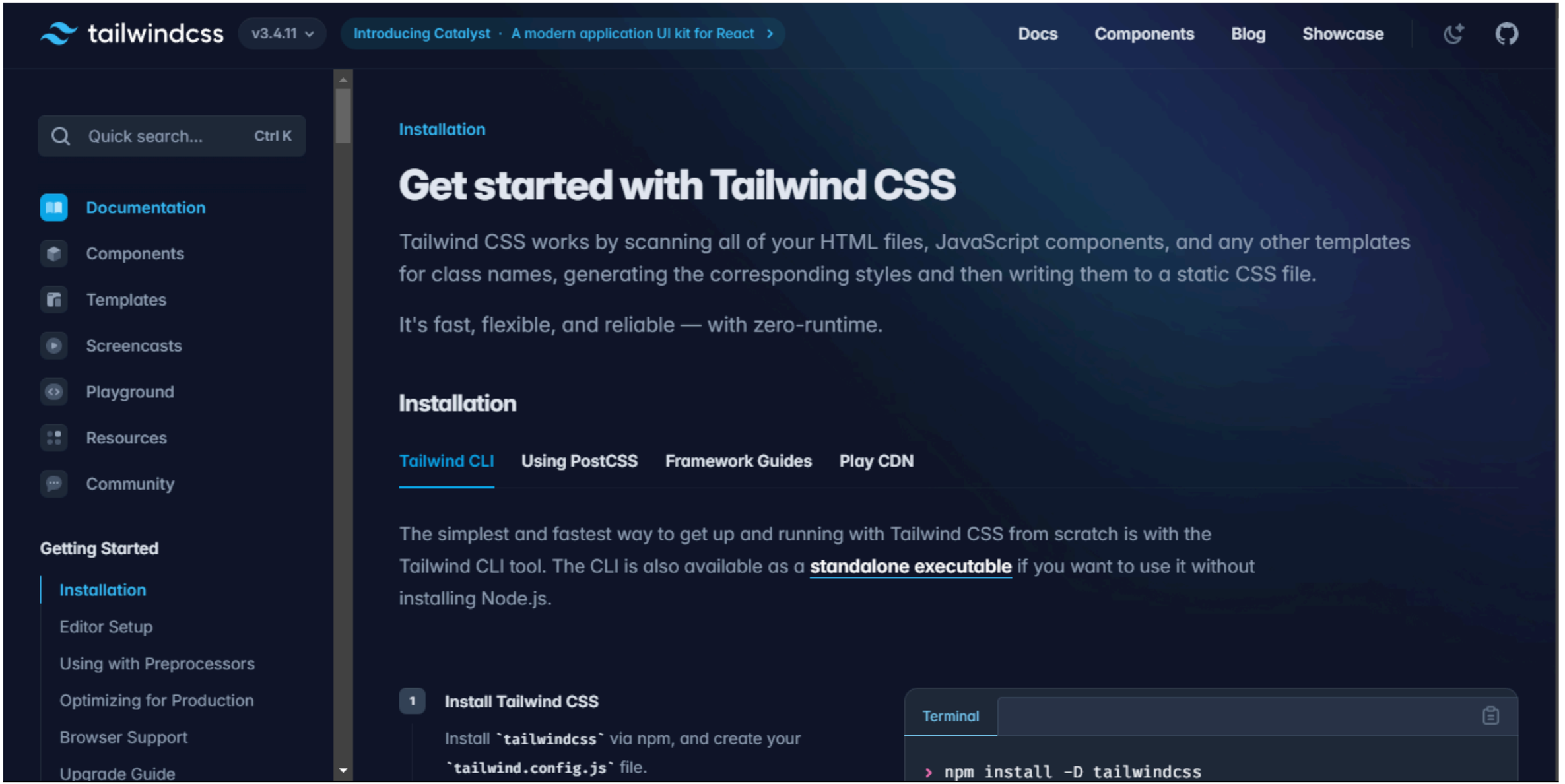
MEDIA QUERIES ALLOW YOU TO APPLY STYLES BASED ON DEVICE SIZE.

```css
@media (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

Introduction to Tailwind CSS

Tailwind CSS

TALWIND CSS


tailwindcss
v3.4.11
Introducing Catalyst · A modern application UI kit for React
Docs
Components
Blog
Showcase
Quick search...
Ctrl K
Documentation
Components
Templates
Screencasts
Playground
Resources
Community
Getting Started
Installation
Editor Setup
Using with Preprocessors
Optimizing for Production
Browser Support
Upgrade Guide
Installation
Get started with Tailwind CSS
Tailwind CSS works by scanning all of your HTML files, JavaScript components, and any other templates for class names, generating the corresponding styles and then writing them to a static CSS file.
It's fast, flexible, and reliable — with zero-runtime.
Installation
Tailwind CLI
Using PostCSS
Framework Guides
Play CDN
The simplest and fastest way to get up and running with Tailwind CSS from scratch is with the Tailwind CLI tool. The CLI is also available as a standalone executable if you want to use it without installing Node.js.
1
Install Tailwind CSS
Install `tailwindcss` via npm, and create your `tailwind.config.js` file.
Terminal
> npm install -D tailwindcss

Steps to include Tailwind CSS in a project via CDN or NPM.

Steps to include Tailwind CSS in a project via CDN or NPM.

Examples of Tailwind CSS utility classes:

text-color, bg-color, padding.

```
<div class="bg-blue-500 text-white p-4">Tailwind Box</div>
```

Tailwind CSS: Flexbox Classes

```
<div class="flex justify-center items-center">Centered Content</div>
```

TAILWIND CSS: GRID CLASSES

```
<div class="grid grid-cols-3 gap-4">
  <div class="bg-red-500">Item 1</div>
  <div class="bg-green-500">Item 2</div>
  <div class="bg-blue-500">Item 3</div>
</div>
```

TAILWIND CSS: RESPONSIVE CLASSES

```
<div class="text-base md:text-xl lg:text-2xl">Responsive Text</div>
```

LETS CREATE SOMETHING

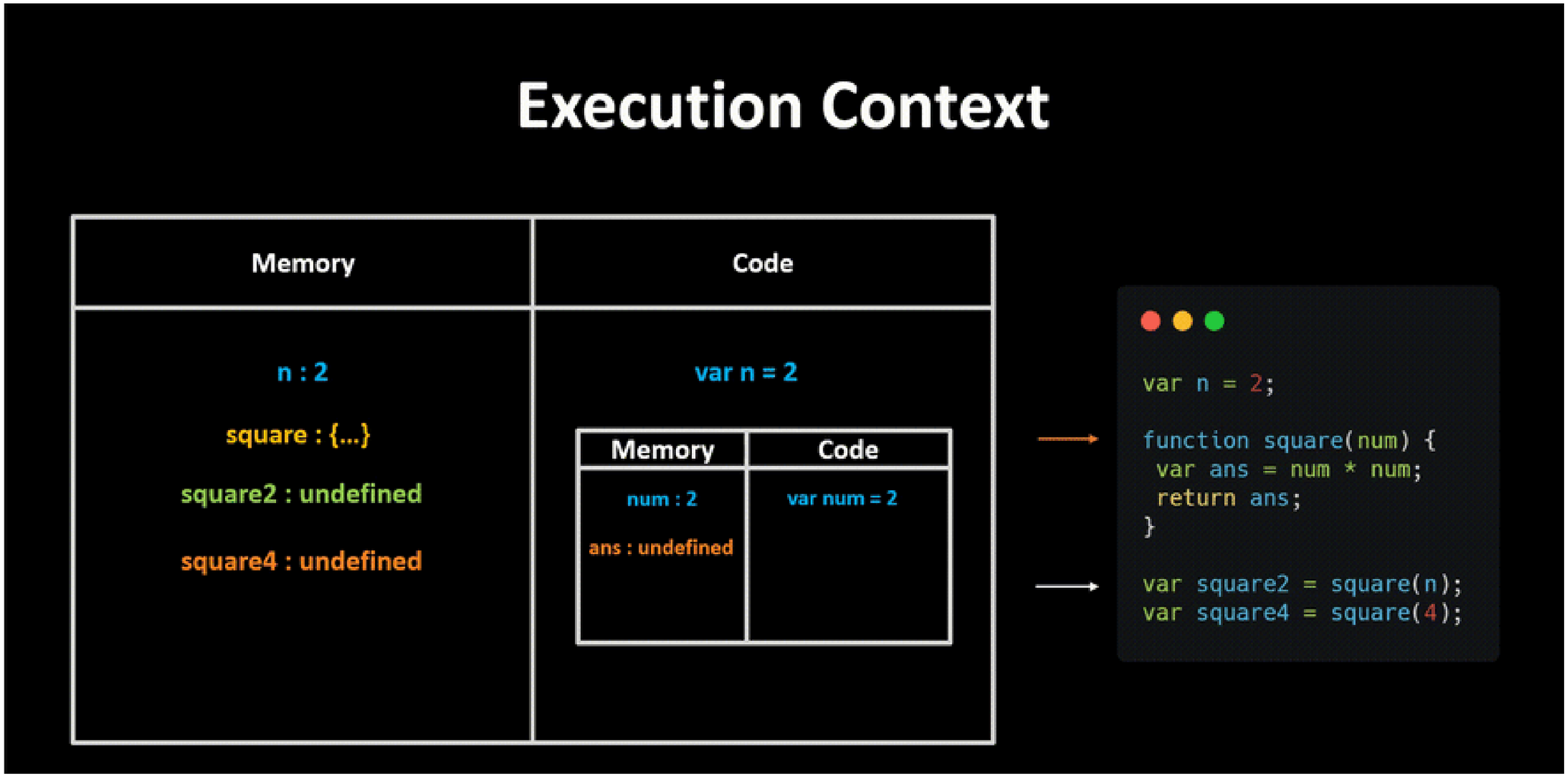

JS

INTRODUCTION TO JAVASCRIPT

- **High-level, Dynamic Language:** JavaScript is a high-level programming language primarily used for enhancing the interactivity and functionality of web pages.
- **Client-Side Scripting:** It is primarily used as a client-side scripting language, meaning it runs directly in the user's web browser to enable dynamic content and user interaction without needing to reload the page.
- **Supports Multiple Paradigms:** JavaScript supports multiple programming paradigms, including object-oriented, functional, and imperative styles, making it versatile and adaptable.
- **Interpreted, Not Compiled:** Unlike some languages that require a compilation step, JavaScript code is interpreted by the browser in real-time, allowing for faster development and easier debugging.
- **Event-Driven Programming:** JavaScript is built around event-driven programming, which enables developers to create responsive, interactive web applications by reacting to user events like clicks, keystrokes, or page loads.

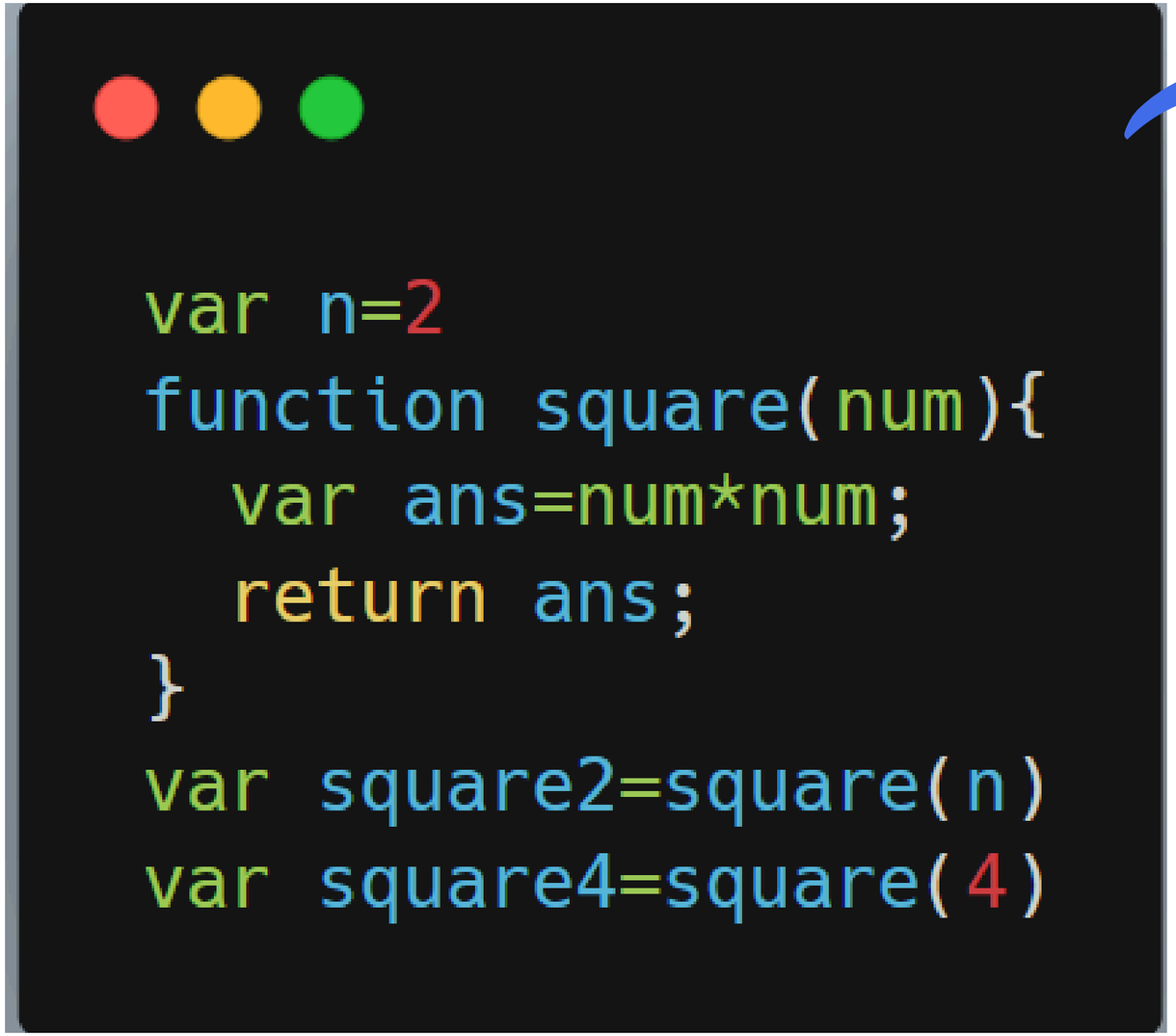
- **ECMAScript Standard:** JavaScript is based on the ECMAScript specification, which standardizes its core features and ensures compatibility across different browsers.
- **Widely Used for Web Development:** It is the foundation of modern web development, alongside HTML and CSS, for creating responsive, interactive websites and applications.
- **Backend Development with Node.js:** While traditionally used for frontend development, JavaScript can also be used on the server-side with frameworks like Node.js, allowing developers to use a single language for both client-side and server-side development.
- **Asynchronous Programming with Promises and Async/Await:** JavaScript provides mechanisms like Promises and async/await to handle asynchronous operations, which is essential for tasks like making API calls or managing file I/O without blocking the main thread.
- **Extensive Libraries and Frameworks:** JavaScript has a vast ecosystem of libraries and frameworks such as React, Angular, and Vue.js, which simplify and streamline the process of building complex web applications.

- Everything in javascript happens inside an execution context



-Javascript is a synchronous single threaded language

let's understand this code through execution context...



```
var n=2
function square(num){
   var ans=num*num;
   return ans;
}
var square2=square(n)
var square4=square(4)
```

What if we run empty Javascript file ????

Concept of undefined Keyword

UNDEFINED VS NOT DEFINED

Javascript is a loosely type programming language means it does not attach its variables to any specific datatype.

HOISTING IN JAVASCRIPT

Hoisting in JavaScript refers to the behavior in which variable and function declarations are moved ("hoisted") to the top of their containing scope (either global or function scope) before the code is executed. This means that you can use variables and functions in your code even before they are declared, though the behavior can differ depending on whether you're using var, let, const, or function declarations.

Variable Hoisting with var

No Initialization with let or const

Function Hoisting: Function declarations are fully hoisted, meaning you can invoke a function even before it is declared in the code.

```
greet(); // Output: "Hello"

function greet() {
  console.log("Hello");
}
```

Function Expressions are Not Hoisted

If you declare a function using a function expression, it behaves like a variable. The function will only be available after the line where it's defined.

```
sayHi(); // TypeError: sayHi is not a function


var sayHi = function() {
  console.log("Hi");
};
```

TYPES OF ERRORS

Syntax Error

A syntax error in JavaScript occurs when the code violates the rules of the language's syntax.

Type Error

If you are declaring a const , you can not declare its value later on so it's a type error.

```
const b;
b=100;
```

Reference Error

Reference error is when a javascript engine tries out to find a specific variable inside the memory space & you can not access it.

```
console.log(a);
let a=100;
```

JavaScript Operators

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- String Operators
- Logical Operators
- Bitwise Operators
- Ternary Operators
- Type Operators

JavaScript Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

Assignment Operators

Operator	Example
=	x = y
+=	x += y
-=	x -= y
*=	x *= y
/=	x /= y
%=	x %= y
**=	x **= y

Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

function
are the
heart of js

- What is Anonymous function?
- What are first class functions?
- What is difference between Function Statement V/s Function Expression V/s Function Declaration ?

Anonymous Function

-Anonymous function does not have their own identity

-You can't directally use anonymous function you can only use it using assign its value to a variable same like function expression.