WEB DEV FOR BEGINNERS

June 2025 30 13:23

1 Introduction to Programming Languages and Tools of the Trade -

- **Programming** (or **coding**) is the process of **writing instructions** that a computer can understand and follow to perform tasks.
- Programming is just like a recipe and out is dish.
- Example-print("Hello, world!") this line shows the message hello world on the screen.
- What is program??

  Ans- A **program** is something made with **code**.

  It can be things like:

- A **website** (like Google),
- A **game** (like Minecraft),
- Or a **phone app** (like WhatsApp).
- When a program is **running**, it means the computer or phone is **doing what the code tells it to do**, step by step.
- The world's **first computer programmer** is widely considered to be **Ada Lovelace.**
- **Famous for**: Writing the **first algorithm** meant to be carried out by a machine.

- What are programming languages?
- Ans- Computers only understand **binary** (just 1s and 0s).

  But writing in 1s and 0s is hard for humans. So we use **programming languages** to write instructions in a way that's easier for people to understand.
- Think of programming languages like a **translator** between humans and computers.
- Fibonacci sequence- The **Fibonacci sequence** is a list of numbers where:
- The first number is **0**
- The second number is **1**
- After that, **each number is made by adding the two numbers before it**
- Start with: 0
- Next: 1

- Then: 0 + 1 = 1
- Then: 1 + 1 = 2
- Then: 1 + 2 = 3
- Then: 2 + 3 = 5
- Then: 3 + 5 = 8
- Then: 5 + 8 = 13
- Then: 8 + 13 = 21
- Then: 13 + 21 = 34
- First 10 numbers are 1,2,3,5,8,13,21,34.

- **Tools used by developers are -**
- **Editors-** where developers write code
- **Browser-** Web developers rely on the browser to see how their code runs on the web. It's also used to display the visual elements of a web page that are written in the editor, like HTML..
- **DevTools** (Developer Tools) are built-in tools in web browsers like **Chrome**, **Firefox**, or **Edge** that help developers **see what’s happening inside a website**
- **Version Control System**
- **Programing Language like** javascript,python etc

2 Introduction to GitHub

- What is GitHub?

GitHub is a website and platform where developers can store, share, and collaborate on code.

It works with Git, a tool that keeps track of code changes. GitHub adds extra features like backups, collaboration, and a nice user interface.

- **Git** = A tool to track changes in your code (like saving versions)
- **GitHub** = A place online to store your code and work with others

Some basic git terms-

Term	Simple Meaning
Repository (Repo)	A folder on GitHub where your project lives. It stores code, files, and the full history of changes.

◯ **Commit**	A saved snapshot of your code changes. Like a save point in your project.
◯ **Push**	Sends your commits (saved changes) from your computer to GitHub.
◯ **Pull**	Brings the latest changes from GitHub to your local machine.
◯ **Clone**	Copies a GitHub repo to your computer so you can work on it.
◯ **Branch**	A separate version of your code where you can work on new features without affecting the main code.
◯ **Main (or Master)**	The default branch of your project. Usually holds the final or production-ready code.
◯ **Merge**	Combines changes from one branch into another (often used to bring updates into the main branch).
◯ **Pull Request (PR)**	A request to merge code changes from one branch to another. Used for code review and collaboration.
◯ **Fork**	Makes a copy of someone else's GitHub repo under your own account so you can make changes.
◯ **README**	A file that describes your project—what it is, how to use it, etc. (written in Markdown: .md)
◯ **Git**	A tool used to track changes in your code. Git runs on your computer.
◯ **Version Control System**	A system that tracks changes in code over time and lets you go back to earlier versions. Git is one example.
◯ **git init**	Initializes a new Git repository in your project folder (starts version tracking).
◯ **git add**	Tells Git which files you want to include in your next commit (save).
◯ **git commit -m "message"**	Saves the added changes with a short message explaining what you did.
◯ **git config**	Used to set your Git user information, like your name and email (needed for commits). Example: git config --global user.name "Your Name"

3 Accessibility Fundamentals-

🌐 What is Web Accessibility?

Web accessibility means making websites usable for everyone, including people with disabilities.

Imagine someone:

- who can’t see (blind),
- who can’t hear (deaf),

- who can’t use a mouse (uses a keyboard or voice),
- or has trouble reading (cognitive disabilities).

Web accessibility ensures they can still use and enjoy websites.

Tools To Use-

1-👁️‍🗨️ What is a Screen Reader?

A screen reader is a tool that reads out loud what’s on the screen.

It helps people who are blind or have low vision use websites, apps, and computers.

🔊 How does it work?

- It reads the web page from top to bottom — out loud.
- If your page has text, it reads the text.

2- what is zoom??

- Another tool commonly used by people with vision impairments is zooming. The most basic type of zooming is static zoom, controlled through Control + plus sign (+) or by decreasing screen resolution. This type of zoom causes the entire page to resize, so using responsive design is important to provide a good user experience at increased zoom levels.

- Another type of zoom relies on specialized software to magnify one area of the screen and pan, much like using a real magnifying glass.

3-🎨 What is Contrast?

- Contrast means the difference between text color and background color.
- ✅ High contrast = Easy to read (like black text on white)
  ❌ Low contrast = Hard to read (like light gray text on white

Contrast checkers

Colors on web sites need to be carefully chosen to answer the needs of color-blind users or people who have difficulty seeing low-contrast colors.
✅ Test a web site you enjoy using for color usage with a browser extension such as WCAG's color checker. What do you learn?

3🔦 What is Lighthouse?

Lighthouse is a tool built into your browser (like Chrome).
It checks how good a website is in different areas, like:

- ✅ Accessibility (can people with disabilities use it?)
- 🚀 Performance (is it fast?)
- 📱 Mobile friendly?
- 🔐 Safe to use?

- A 100% score is great, but real testing (like with screen readers) is still important.

IT HELPS YOU TO IMPROVE YOUR WEB SITE

Good display principles

Color safe palettes
People see the world in different ways, and this includes colors. When selecting a color scheme for your site, you should ensure it's accessible to all.
One great..

What is Semantic HTML?

Semantic HTML means using the right HTML tags for the right purpose.

Just like we use the right tools in real life (e.g., you wouldn’t eat soup with a fork 🍴), we should also use the correct HTML elements when building web pages.

🎯 Why It Matters

Even though you can style anything to look like a button or link, screen readers (used by visually impaired users) and search engines rely on correct tags to understand your page.

For example:

- ✅ Use <a> for links
- ✅ Use <button> for buttons

- ❌ Don't use <span> or <div> just because they're easy to style

Why Headings Matter

Headings (like <h1>, <h2>, <h3>) are like chapter titles in a book.

They help organize content and make it easy to scan and navigate, especially for:

- Screen reader users
- Keyboard users
- Search engines

📋 What Is a Heading Hierarchy?

A heading hierarchy is using headings in the right order:

ARIA

ARIA stands for **Accessible Rich Internet Applications**.

It's a set of **HTML attributes** that help make websites more understandable for people using **screen readers** (especially those who are blind or visually impaired).

Imagine a page like this:

Product	Description	Order
Widget	Description	Order
Super Widget	Description	Order

To a sighted user, this is clear:

- "Description" next to "Widget" means it's the description of Widget
- "Order" next to "Super Widget" means order that product

But a screen reader would just say:

"Description, Order, Description, Order..."

Without context — the user has no idea what product each one refers to.

✅ **The Solution: ARIA (like aria-label)**

You can add extra context using an ARIA attribute called aria-label.

🧾 Example:

Instead of:

html
CopyEdit
```
<a href="description.html">Description</a>
```

Use:

html

```
<a href="description.html" aria-label="Description of Widget">Description</a>
```

Now a screen reader would say:

“Description of Widget”

JavaScript Basics: Data Types

What is a Variable?

A **variable** is like a **box** where you can store some information
(like a number or a word), and then use or change that information
later in your code.

How to Create a Variable?

You write it like this:

```
let name;
```

This has two parts:

1. Keyword:
- You write let or var.
- We recommend using let because it's safer and newer (var is old-fashioned).
2. Variable Name:
- You choose this. It's like naming your box.
- Example: name, age, score

What is a Constant?

A **constant** is like a variable — it stores a value — **but you can’t change it once it’s set**

How to Declare a Constant

Use the const keyword:

```
const MY_VARIABLE = 123;
```

What is a Data Type?

A data type tells the computer what kind of value a variable holds — like text, number, or true/false.

Think of it like different kinds of boxes:
One for text, 📦 one for numbers, 📦 one for true/false answers, and so on.

JavaScript has 7 primitive (basic) data types:

Data Type	**What it holds**	**Example**
string	Text	"hello", 'zebra'
number	Numbers (whole or decimal)	5, -2, 3.14

bigint	Very big numbers	12345678901234567890n
boolean	True or false	true, false
undefined	A variable with no value yet	let x; → undefined
null	Empty or nothing on purpose	let x = null;
symbol	Unique value (for advanced use)	Symbol("id")

Example:

```
let name = "Zebra";      // string
let age = 5;             // number
let isWild = true;       // boolean
let bigNum = 12345678901234567890n; // bigint
let nothing = null;      // null
let notSet;              // undefined
```

🔢 Numbers in JavaScript

You can store any number in a variable:

```
let score = 100;
let price = 9.99;
l[illegible]
```

And you can do math with them too! Like:

```
let total = 10 + 5; // 15
```

Arithmetic Operators

There are several types of operators to use when performing arithmetic functions, and some are listed here:

Symbol	**Description**	**Example**
+	Addition: Calculates the sum of two numbers	1 + 2 //expected answer is 3
-	Subtraction: Calculates the difference of two numbers	1 - 2 //expected answer is -1
*	Multiplication: Calculates the product of two numbers	1 * 2 //expected answer is 2
/	Division: Calculates the quotient of two numbers	1 / 2 //expected answer is 0.5
%	Remainder: Calculates the remainder from the division of two numbers	1 % 2 //expected answer is 1

- **Strings**
- A string is just text wrapped in quotes.

- Examples:
- 'Hello'
  "World"
  '123'

- You can also store a string in a variable:
- let message = "Hello, world!";
-

- **Joining Strings (Concatenation)**
- You can combine strings using the + sign:

-
  ```
  let first = "Hello";
  let second = "World";

  let combined = first + " " + second + "!"; // Hello World!
  ```

- Why does '1' + '1' give 11?
- Because:
- '1' is a string, not a number.
- When you + two strings, it just sticks them together → '1' + '1' = '11'
- But:

-
  ```
  1 + 1        // 2 (number + number)
  '1' + 1      // '11' (string + number = string!)
  ```

- JavaScript treats everything like a string if at least one part is a string!
-

- **Template Literals**
- Instead of using +, you can use **backticks** ` like this:

-
  ```
  let name = "Alice";
  let message = `Hello, ${name}!`; // Hello, Alice!
  ```

- It's cleaner and easier when combining many variables or using line breaks.
- When to use what?

Use this	When
"quotes"	Just writing a simple string
+	Combining small strings
\`template ${literals}\`	Combining **variables** or making **multi-line** strings

- Booleans
- A **Boolean** is either:
- true (yes)
- false (no)
- Examples:

-
  ```
  let isSunny = true;
  let isRaining = false;
  ```
- Booleans are used to help **make decisions** in code. For example

-
  ```
  if (isSunny) {
    console.log("Go outside!");
  }
  ```

JavaScript Basics: Methods and Functions

- **Function Kya Hota Hai?**

Function ek block of code hota hai jo hum baar-baar reuse kar sakte hain.

Socho agar tumhe ek hi kaam 5 jagah karna hai — toh har jagah wohi code likhne ki jagah, tum ek function bana lo, aur jab chaho usko call kar lo.

Jaise:

- Ek function banaya jo "Hello" bole

- Ab chahe 1 baar bolo ya 100 baar — sirf functionName() likhke use use karo

- **Function Ka Naam-**

  Function ka naam ek label ki tarah hota hai — jaise button pe likha ho "Cancel Timer" toh samajh aata hai kya hoga.

  Waise hi agar function ka naam ho startGame() toh samajh jaate ho game shuru hoga.

Function Kaise Banate Hain?

Syntax:

```
function nameOfFunction() { // yahan woh code likho jo baar-baar chalana hai }
```

Ex-

```
function displayGreeting() { console.log('Hello, world!'); }
```

Yeh function ban gaya, ab jab bhi tum displayGreeting() likhoge, yeh line chalegi:

```
Hello, world!
```

Function Ko Kaise Call Karte Hain?

Bas uska naam likho aur () laga do:

```
displayGreeting(); // output: Hello, world!
```

Jaise hum kisi button ko press karte hain, waise hi function ko call/invoke karte hain.

Function vs Method

Function: Free hota hai, kisi object se link nahi hota.

Method: Kisi object ke andar hota hai.

Example:

```
console.log('Hi'); // yeh method hai
```

Yahan console ek object hai, aur log() uska method hai.

- **Passing Information to a Function — Matlab kya?**
- Agar tum function ko thoda zyada smart banana chahte ho, toh tum usko extra information de sakte ho. Yeh extra info ko hum parameter (ya argument) kehte hain.

Without Parameter Example:

```
function displayGreeting() { console.log("Hello, world!"); } displayGreeting(); // Output: Hello, world!
```

Yeh sirf "Hello, world!" hi print karega — har baar wahi.

- **With Parameter (Smart Function)**

Ab socho tum chahte ho ki function kisi naam wale insaan ko greet kare.
Toh hum function ke andar parameter pass karenge:

```
function displayGreeting(name) { const message = `Hello, ${name}!`; console.log(message); }
```

Yahan:

- name = parameter (function ko milne wali input value)
- ${name} = template literal (yeh value string ke andar use hoti hai)

- **Call Karna (Function Chalana with Argument)**

```
displayGreeting("Christopher");
```

Output: Hello, Christopher!

Agar tum:\

```
displayGreeting("Peaky");
```

➡️ Output: Hello, Peaky!

What is a Default Value?

Sometimes when you create a function, you want to make some parameters optional — so that if the user doesn't provide a value, the function will use a default instead.
This makes the function more flexible and easier to use.

🔧 Example:

```
function displayGreeting(name, salutation = 'Hello') { console.log(`${salutation}, ${name}`); }
```

In this function:
name is required (you must give it)
salutation = 'Hello' → this means: if you don’t pass a greeting, it will use "Hello" as the default

- **Calling the Function:**

Case 1: Only name is given

displayGreeting('Christopher');

🟢 Output: Hello, Christopher

Since no greeting (salutation) is passed, it uses the default "Hello".

Case 2: Name and custom greeting given

displayGreeting('Christopher', 'Hi');

🟢 Output: Hi, Christopher

Now, we passed a custom greeting "Hi", so the function uses that instead of the default.

Return Value kya hota hai?

Kabhi-kabhi hum chahte hain ki function koi result calculate kare aur woh value wapas (return) kare, jise hum baad mein store ya use kar sakein.

Uske liye hum use karte hain:

javascript

CopyEdit

return someValue;

Iska matlab: function ka result ye hai, aur jahan se function call hua tha, wahan wapas bhej diya.

Example:

function createGreetingMessage(name) { const message = `Hello, ${name}`; return message; }

What's happening?

- message banaya inside the function
- return message; → function ka result wapas bhej diya

📦 Using the Return Value

const greetingMessage = createGreetingMessage("Christopher"); console.log(greetingMessage); // Output: Hello, Christopher

Yahan:

- createGreetingMessage() ne value return ki
- Humne us value ko greetingMessage naam ke variable mein store kar liya

- Fir console pe print kiya

Functions as parameters for functions

In JavaScript, **functions can be passed into other functions** — just like you pass numbers or strings as values.

That’s right — **a function can be used as an input (parameter) to another function.**

Anonymous Function kya hoti hai?

"Anonymous" ka matlab hota hai **"without a name"**.

Normally, jab hum function banate hain to uska naam dete hain:

```
function displayDone() { console.log("3 seconds has elapsed"); }
```

Yeh function ka naam hai: displayDone

❓Par agar ye function sirf **ek hi jagah** use hona hai?

Toh naam dene ki zarurat nahi hai!

Bas **direct function likh do** jahan use karna hai — bina naam ke. Yehi hota hai **anonymous function**.

✅ Example: Anonymous Function with setTimeout

```
setTimeout(function() { console.log("3 seconds has elapsed"); }, 3000);
```

Yahan:

- Humne function banaya
- Lekin **koi naam nahi diya**
- Directly setTimeout ke andar likh diya

➡️ Output same hai: **"3 seconds has elapsed"**

🏹 Fat Arrow Function (Arrow Function) kya hoti hai?

JavaScript mein ek shortcut hai function likhne ka:

Hum function keyword hata ke => (arrow) use karte hain.

Isko kehte hain **arrow function** ya **fat arrow function**

✅ Example: Using Arrow Function

```
setTimeout(() => { console.log("3 seconds has elapsed"); }, 3000);
```

Yeh aur bhi short aur clean hai.

- function() ki jagah humne likha () =>
- Baaki sab same

() { console.log('3 seconds has elapsed'); } // timer value is in milliseconds setTimeout(displayDone, 3000);