Learn Web Design as a Beginner: A 7-Day Starter Guide

✅ Table of Contents:

📘 Introduction: Why Learn Web Design?

Welcome to “Learn Web Design as a Beginner”, your 7-day crash course to becoming confident in building websites from scratch—even if you’ve never written a single line of code!

Whether you're looking to:

- Start a freelance career 💼
- Build your personal blog 📝
- Launch a business website 🛍️
- Or just learn a valuable modern skill 💡

Web design is the perfect place to begin. Why? Because the web is everywhere. It's how we read, shop, learn, and communicate.

💡 What is Web Design?

Web Design is the process of planning, creating, and styling the content that people see on websites. It includes:

- Layout (structure of the page)
- Visuals (colors, fonts, images)
- User Experience (how easy it is to use).

In this eBook, you’ll learn the core tools of web design:

- HTML: The structure of your web page
- CSS: The style and design
- Responsive Design: Making websites look great on all devices
- JavaScript (basic): Adding interactivity
- Tools like Bootstrap to speed up your workflow

✅ What You Need to Follow Along

- A laptop or desktop
- A code editor like VS Code
- A modern web browser like Chrome
- No prior experience—just curiosity!

🚀 Day 1: Understanding the Web (How Websites Work)

Before you build a website, it helps to know how the web actually works.

🌐 What happens when you type a URL?

When you go to a site like www.google.com, here's what's happening behind the scenes:

1. Your browser sends a request to a server.
2. The server finds the files for that website (like HTML, CSS, images).
3. The files are sent back to your browser.
4. The browser displays the page using those files.

Websites are made up of files, and your browser knows how to read them. These files are usually:

- HTML (structure)
- CSS (style)
- JS (interactivity)
- Images, fonts, videos, etc.

🛠️ Let's Build Your First Web Page (HTML only)

We'll create a simple page that says Hello World with your name.

📁 Step 1: Create Your Files

1. Create a folder on your desktop named web-design-day1.
2. Inside it, create a file named: index.html

✨ Paste this code into index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>My First Web Page</title>
</head>
<body>

  <h1>Hello, World!</h1>
  <p>My name is Sarah and I'm learning web design 🚀</p>

</body>
</html>
```

✅ Save and Open It

- Double-click the index.html file.
- Your browser will open the page, and you'll see your message!

🎉 You've just written your first webpage!

💡 HTML Structure Breakdown

```
<!DOCTYPE html>   <!-- Tells browser this is HTML5 -->
<html>            <!-- Root of the document -->
<head>            <!-- Metadata, title, settings -->
  <title>My First Web Page</title>
</head>
<body>            <!-- Visible content of the page -->
  <h1>Hello, World!</h1>     <!-- Heading -->
  <p>My name is Josh...</p> <!-- Paragraph -->
</body>
</html>
```

🧠 *Bonus Exercise*

Try changing the content:

- Replace the heading with your own name
- Add more <p> tags with fun facts about yourself
- Add another heading using <h2>

✅ *What You Learned Today*

- How the web works.
- What a basic website is made of.
- How to create and open an HTML file.
- The structure of a simple HTML page.

NOTE- In other to perform these tasks that you have just learnt, you need a code editor like Notepad, Sublime Text, VsCode.
Preferably, VsCode is better.

You're to download and install VsCode on your computer. After installation, launch your VsCode. Click on Extensions on the left hand side of the VsCode, search for

- Live Server and install.
- Prettier and install.
- ESLint and install.
- Code Spell Checker.

Extensions	Description
1. Live Server	Live server Launches a local dev server with live reload.
2. Prettier	Code formatter to keep your code clean and consistent.
3. ESLint	Identifies and fixes JavaScript/ TypeScript problems.
4. Code Spell Checker	Catches common spelling errors in code.

After pasting the code in your folder that you've created, on your VsCode, click on the 3 bars on the top left hand corner of your VsCode, locate File, click on open folder and locate the folder where you have your code, click on the folder and finally click on open folder.

📘 Day 2: HTML Basics – Structure Your Web Page

Welcome to Day 2! Now that you've created your first simple HTML page, it's time to explore more HTML building blocks that make up every web page you see online.

HTML stands for HyperText Markup Language. It's not a programming language—it's a markup language that tells the browser how to structure the content on your page.

🧰 Common HTML Elements You'll Use Often

Tag	Purpose
<h1> to <h6>	Headings (titles & subtitles)
<p>	Paragraphs (text)
<a>	Links (to other pages or websites)
<img>	Images
, ,	Lists (unordered/ordered)
	Generic container
	Inline container
 	Line break
,	Bold, Italic

🛠️ Let's Build a Simple Profile Page

📁 Step 1: Create a New File
Create a new file in your folder and name it:
profile.html

💻 Paste this code:

<!DOCTYPE html>

```
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>My Profile</title>
</head>
<body>

  <h1>👋 Hello, I'm Josh</h1>
  <p>Welcome to my profile page. I’m learning web design and loving it!</p>

  <h2>🌟 My Skills</h2>
  <ul>
    <li>HTML</li>
    <li>CSS</li>
    <li>Creative Design</li>
  </ul>

  <h2>📸 My Photo</h2>
  <img src="profile.jpg" alt="My Profile Photo" width="200" />

  <h2>📫 Contact Me</h2>
  <p>Email: <a href="mailto:Josh@example.com">Josh@example.com</a></p>

  <h2>📍 Where I'm From</h2>
  <p>Lagos, Nigeria</p>

</body>
</html>
```

📝 Note: Replace "profile.jpg" with an actual image in the same folder for it to display.

✅ **Let’s Break it Down**

🔤 **Headings**:
<h1>Title</h1> to <h6>Smallest Heading</h6>
Headings help structure your content. Use only one <h1> per page, and then organize with <h2>, <h3>, etc.

📄 **Paragraphs**:
This is a paragraph of text.
Used for plain text content like introductions, descriptions, and more.

📸 Images:

src: Path to the image
alt: Text shown if image can’t load (important for accessibility)
width: Size control
🔗 Links:
Visit my website

- href is the destination,
- Can link to external sites, other pages, or email addresses

📝 **Lists-**
Unordered list:

html

  HTML
  CSS

Ordered list:

html

  First
  Second

💡 **Pro Tips for Clean HTML**

- Always use semantic tags (like <header>, <section>, <footer>) when possible (we’ll cover this in detail later).
- Keep your code indented and organized.
- Use alt text on all images for SEO and accessibility.

🧠 Practice Challenge

Create a new HTML file called about-me.html with:

- A heading (your name)
- Two paragraphs about your interests
- One image of your choice
- A list of your favorite hobbies
- A link to your favorite website

✅ What You Learned Today

- How to use HTML elements to structure a page
- How to add text, images, lists, and links
- How to write semantic, organized HTML.

🎨 Day 3: CSS Basics – Styling Your Page

Welcome to Day 3! Previously, you learned how to structure your page with HTML. Now, bring it to life using CSS—the design language of the web.

💡 *What is CSS?*

CSS stands for Cascading Style Sheets. It controls how HTML elements look on the screen, such as:

- Fonts 🅰️
- Colors 🎨
- Spacing 📏
- Layouts 📐

You can think of HTML as the skeleton, and CSS as the clothing and makeup that makes your site visually appealing.

🛠️ CSS in Action: Your First Styled Page

📁 Step 1: Create a New HTML + CSS File

- Create a file named: styled-profile.html
- Create a file named: style.css Put both in the same folder.

🧾 **styled-profile.html**

html
👇
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Styled Profile</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>

  <div class="container">
    <h1>Hello, I'm Josh 🌸</h1>
    <p>Welcome to my profile page. I'm learning web design and loving it!</p>

    <h2>My Skills</h2>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>Creative Design</li>
    </ul>

    <img src="profile.jpg" alt="My Profile" class="profile-img"/>

    <p>📫 <a href="mailto:sarah@example.com">Contact Me</a></p>
  </div>

</body>
</html>
```

🎨 **style.css**
```
/* General Page Styles */ - This is a comment in CSS.

body {
  font-family: Arial, sans-serif;
```

```
  background-color: #fdf6f0;
  color: #333;
  margin: 0;
  padding: 0;
}

/* Container Styling */ - CSS comments

.container {
  max-width: 700px;
  margin: 40px auto;
  padding: 20px;
  background-color: #ffffff;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
  border-radius: 10px;
}

/* Headings */ - CSS comments

h1 {
  color: #e91e63;
  font-size: 2.5rem;
}

h2 {
  color: #333;
  margin-top: 30px;
}

/* Lists */

ul {
  padding-left: 20px;
  list-style-type: square;
}

/* Images */
```

```
.profile-img {
  width: 200px;
  border-radius: 8px;
  margin-top: 20px;
}

/* Links */

a {
  color: #0077cc;
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}
```

🔍 *CSS Breakdown*

🧾 **How CSS Works**

- The browser loads your HTML
- It sees <link rel="stylesheet" href="style.css">
- It applies styles to match the HTML tags and classes

🔤 **Selectors**

CSS targets HTML using selectors:

Selector	Targets
body	The entire page
h1, p	Specific tags
.container	Class named “container”
#id-name	An element with a specific id

💅 **Styling Elements**

```
h1 {
  color: blue;
  font-size: 3rem;
```

}
Changes the color and size of all <h1> tags.

🎨 *Design Tip: Use a Color Palette*

Try a free tool like https://coolors.co to generate aesthetic color schemes for your site.

🧠 Practice Challenge

1. Add a background color to your page
2. Change the font to 'Courier New'
3. Add some padding around your text
4. Change the link color on hover

Bonus: Try adding a border to your image!

✅ *What You Learned Today*

- How to link an external CSS file to HTML
- How to style fonts, colors, images, and spacing
- What selectors and properties are
- How to make your first custom-styled web page

📱 Day 4: Responsive Design with Flexbox & Media Queries

Welcome to Day 4! You've built and styled your first web page—now it's time to make sure it looks great on any screen, whether it's a phone, tablet, or desktop. *This is called Responsive Design.*

📦 What is Responsive Design?

Responsive design means your website adapts to different screen sizes. It ensures that:

1. Text is readable on mobile
2. Images resize correctly
3. Layouts shift to fit the screen.

To do this, we use:

- Flexbox (for flexible layouts)
- Media Queries (for custom rules on different screen sizes)

🧰 *Meet Flexbox: Flexible Layout Made Simple ~*

Flexbox is a CSS layout tool that helps you align items in rows or columns and control spacing—even when screen size changes.

💻 Let's Create a Responsive Card Layout

We'll build a section that shows your profile, with text on one side and an image on the other.

📁 Files you'll need:

- responsive-profile.html
- style.css (add new code here or create a new one)

🧾 responsive-profile.html

html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Responsive Profile</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>

  <div class="container flex">
    <div class="text-box">
      <h1>Hello, I'm Sarah 👋</h1>
      <p>I'm learning web design. I love creating responsive, user-friendly websites that work on all devices!</p>
    </div>
    <div class="img-box">
      <img src="profile.jpg" alt="Sarah" class="profile-img"/>
    </div>
  </div>
```

```
</body>
</html>
```

🎨 Add to style.css

```
/* Flex container */ - CSS comments
.flex {
  display: flex;
  align-items: center;
  justify-content: space-between;
  flex-wrap: wrap;
}

/* Text and Image boxes */
.text-box {
  flex: 1;
  padding: 20px;
}

.img-box {
  flex: 1;
  padding: 20px;
  text-align: center;
}

/* Image styling */
.profile-img {
  max-width: 100%;
  height: auto;
  border-radius: 10px;
}
```

🔁 How Flexbox Works

```
css
👇
display: flex;
```

Turns the container into a flexible box.
Other properties:

- flex-wrap: wrap; → stacks items on smaller screens
- justify-content → controls horizontal alignment
- align-items → controls vertical alignment

🎯 Media Queries: Custom Styles for Mobile

Media queries let you change styles at different screen widths.

css
👇

```
@media (max-width: 768px) {
  .flex {
    flex-direction: column;
    text-align: center;
  }

  .text-box, .img-box {
    padding: 10px;
  }

  h1 {
    font-size: 1.8rem;
  }
}
```

✅ What it does:

On screens 768px or less (like tablets/phones), the layout becomes vertical.
Headings and padding adjust for smaller screens.

📱 Try It Out

Open your browser, and shrink the window. The layout will change thanks to Flexbox and the media query. Your site is now responsive! 🎉

🧠 Practice Challenge

Create a responsive "About Me" section:

- One side: a short paragraph about you
- Other side: an image
- Use Flexbox to arrange them
- Add a media query to stack them vertically on small screens

✅ What You Learned Today

- What responsive design means
- How to use Flexbox for layout control
- How to use media queries to adjust styles for different screen sizes

✍️ Day 5: Modern Typography and Google Fonts

Typography is more than just picking fonts — it's about creating a beautiful reading experience. Well-chosen fonts and spacing can make your content easier to read, more visually appealing, and more professional.

🎯 Why Typography Matters

- 👁️ Improves readability
- 🎨 Establishes visual hierarchy
- ✨ Builds brand personality
- 📱 Boosts user experience on all devices

📚 Let's Talk Terminology

Term	Meaning
Font family	The name of the typeface (e.g., Arial, Roboto)
Font size	The size of the text, typically in px, em, or rem
Line height	Vertical space between lines of text
Letter spacing	Space between characters

🌐 Using Google Fonts

Google Fonts is a free, easy way to get beautiful fonts on your website.

🔗 Step 1: Choose a Font

Let’s choose “Poppins” and “Roboto”, a popular combo.

- Go to https://fonts.google.com
- Search for Poppins and Roboto
- Click "Select Font" → Click the “@import” tab
- Copy the @import code snippet

Example:

css
👇
```
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&family=Roboto&display=swap');
```

Paste it at the top of your style.css file.

🧾 Updated style.css

```
/* Import Google Fonts */ - Css come

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&family=Roboto&display=swap');

/* Base Styles */
body {
  font-family: 'Roboto', sans-serif;
  background-color: #fdf6f0;
  color: #333;
  line-height: 1.6;
  margin: 0;
  padding: 0;
```

```
}

/* Headings */
h1, h2, h3 {
  font-family: 'Poppins', sans-serif;
  font-weight: 600;
}

/* Paragraphs */
p {
  font-size: 1.1rem;
  margin-bottom: 20px;
}
```

📏 **Styling Tips for Better Typography**

✅ **Font Sizes**

Use rem or em instead of px so text can scale across devices.

Example:

```
p {
  font-size: 1.1rem;
}
```

✅ **Line Height**

Makes text more breathable.

Example:

```
line-height: 1.6;
```

✅ **Text Alignment & Spacing**

```
.text-box {
  text-align: left;
  letter-spacing: 0.5px;
  padding: 10px;
}
```

🖼️ **Example Typography Section**

```
html
```

👇

```
<div class="text-box">
  <h1>Hi, I'm Sarah</h1>
  <p>I specialize in creating clean, responsive websites using modern web design techniques.</p>
</div>
```

css
👇

```
.text-box {
  font-family: 'Roboto', sans-serif;
  line-height: 1.6;
  max-width: 600px;
  margin: 0 auto;
  padding: 20px;
}
```

🧠 Practice Challenge

1. Visit Google Fonts and pick your own heading and body font pair
2. Use them to update your website typography
3. Experiment with font sizes, spacing, and line height
4. Preview on mobile and desktop

✅ What You Learned Today

- How to use Google Fonts in your project
- Difference between heading and body fonts
- How to improve typography with spacing and sizing
- Best practices for web readability

⚙️ Day 6: Add Interactivity with JavaScript

So far, you've built a website that looks great. Now, it’s time to make it come alive with interactivity using JavaScript (JS)—the language that powers the behavior of web pages.

🧠 What is JavaScript?

JavaScript is a programming language used to create:

- 🎯 Buttons that respond to clicks
- 📩 Forms that validate inputs
- 🕹️ Slideshows and animations
- 🔄 Dynamic content updates

Together with HTML (structure) and CSS (style), JavaScript is the third core piece of web design.

🧪 Let’s Write Our First Script

📁 Create Files

1. interactive.html
2. script.js

🧾 interactive.html

html
👇

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Interactive Page</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>

  <div class="container">
    <h1>Welcome to My Interactive Page!</h1>
    <button id="greetBtn">Click Me</button>
    <p id="message"></p>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

🎯 script.js

JavaScript
👇
```
// Wait until the page loads - JS comments

document.addEventListener('DOMContentLoaded', function() {
  const greetBtn = document.getElementById('greetBtn');
  const message = document.getElementById('message');

  greetBtn.addEventListener('click', function() {
    message.textContent = "👋 Hello, thanks for clicking!";
    message.style.color = "#e91e63";
  });
});
```

💡 **JavaScript Concepts You Just Used**

Concept	Explanation
document.getElementById()	Gets the element by ID from HTML
.addEventListener()	Waits for an event (like a click)
.textContent	Changes the text inside an element
.style.property	Dynamically changes the element's style

🧠 **Practice: Build a Simple Form**

Let's add a small form that gives a confirmation message when submitted.

✍️**Add to interactive.html**

html
👇
```
<form id="contactForm">
  <input type="text" placeholder="Your Name" id="nameInput" required />
  <button type="submit">Submit</button>
</form>
<p id="formMessage"></p>
```

➕ Update script.js

javascript
👇
```
const form = document.getElementById('contactForm');
const nameInput = document.getElementById('nameInput');
const formMessage = document.getElementById('formMessage');

form.addEventListener('submit', function(e) {
  e.preventDefault(); // Prevent actual form submission
  formMessage.textContent = `✅ Thank you, ${nameInput.value}! We'll be in touch.`;
});
```

✅ What You Learned Today

- How to link a JavaScript file to your HTML
- How to respond to button clicks
- How to manipulate text and styles using JS
- How to interact with form inputs

🧠 Bonus: Make a Light/Dark Mode Toggle

Challenge: Add a button that toggles between light and dark themes using JavaScript.

🌍 Day 7: Publishing Your Website Online

Congratulations! 🎉 You've built, styled, and made your first website interactive. Now it's time to publish it so anyone in the world can access it via a URL.
This is the final step — and a huge milestone! ✅

🧠 What Does "Hosting" Mean?

Hosting means putting your website's files (HTML, CSS, JS, images) on a web server so others can visit it using a browser.

🆓 Best Free Hosting Options for Beginners

Platform	Features	Ideal For
GitHub Pages	Free hosting for static sites	Portfolio sites, eBooks, resumes
Netlify	Fast deploys + custom domains	Landing pages, blogs
Vercel	Modern frontend deployment	React, Next.js apps

We'll focus on GitHub Pages – simple and great for beginners.

🧰 What You'll Need

- A GitHub account → https://github.com
- Your project folder (HTML, CSS, JS)

🔧 Step-by-Step: Deploy with GitHub Pages

1. **Create a GitHub Repository**
- Go to GitHub.com
- Click + > New repository
- Name it something like my-website
- Choose Public and click Create

2. **Upload Your Project**

Option 1: Use GitHub Desktop (easier)
Option 2: Upload files manually:

- Open your repo
- Click Add file > Upload files
- Upload index.html, style.css, and any other files
- Click Commit changes

3. **Enable GitHub Pages**
- Go to your repo's Settings
- Scroll to Pages
- Under "Branch," select main and root (/)
- Click Save

GitHub will give you a URL like:
https://yourusername.github.io/my-website/

🎉 That's your live site!

✅ Optional: Use a Custom Domain

If you own a domain (e.g., yoursite.com), you can connect it:

1. Buy a domain via Namecheap, GoDaddy, etc.
2. Point it to GitHub Pages using DNS settings
3. Add your domain to GitHub → Settings > Pages > Custom domain

🌈 Your Site is Now Live!

Share it on:

- LinkedIn, Twitter, Instagram
- Developer communities
- Your resume or portfolio
- Email newsletters

💡 Bonus Tip: Use Netlify for Drag & Drop Hosting

1. Go to https://netlify.com
2. Create account
3. Drag your folder (with index.html) into the dashboard
4. Done! You'll get a URL instantly

🎉 What You've Achieved in 7 Days

✔️ Built HTML structure
✔️ Styled it beautifully with CSS
✔️ Made it responsive with Flexbox
✔️ Added custom fonts
✔️ Added JavaScript interactivity
✔️ Connected a form

✔️ Published it online

👏 You've taken a complete beginner journey from zero to a published web designer.

✅ **Bonus: How to Keep Learning & Monetize Your Skills**
Learning web design is just the beginning. To truly grow, here's how you can level up and start earning:

📚 **Keep Learning**

- Practice daily: Build mini projects like a portfolio site, a to-do app, or a blog.
- Follow tutorials: Sites like freeCodeCamp, MDN Web Docs, and YouTube channels like "Traversy Media" and "Kevin Powell" are goldmines.
- Take real feedback: Share your work on forums (Reddit, Discord, etc.) and ask for constructive critique.

💼 **Monetize Your Skills**

- Freelancing: Start with platforms like Fiverr or Upwork.
- Sell digital products: Turn your designs into templates or ebooks and sell them on Gumroad or Etsy.
- Offer Web Services: Help local businesses get online by offering basic site packages.
- Start a blog or YouTube channel: Share what you're learning and earn through ads, sponsorships, or products.

Remember: You don't have to be an expert to earn — you just need to be one step ahead of someone else.

✅ **Resources & Tools**

Here's a handpicked list of tools and resources to help you grow:

🛠️ **Design Tools**

- Figma (UI Design)
- Canva (Graphics & Branding)
- Adobe XD (Web Design Prototyping)

💻 Code Editors

- VS Code (Highly recommended)
- Sublime Text
- Brackets

📚 Learning Platforms

- freeCodeCamp.org
- MDN Web Docs
- W3Schools.com

🌐 Hosting & Deployment

- GitHub Pages (Free)
- Netlify (Fast and simple)
- Vercel (Great for frontend)

📦 Starter Kits

- HTML5 Boilerplate
- CSS Reset or Normalize.css
- Tailwind CSS CDN

Good luck 👍