



Gurukul Technicals Institute

Computer education · Robotics · Science projects · Delhi

FREE EBOOK

PROGRAMMING AND WEB

Build Five Web Pages with HTML and CSS

A project book: a profile page, a school page, a shop page, a form and a photo gallery

INSIDE THIS BOOK

- Getting ready: tools and the basics
- Project 1: a personal profile page
- Project 2: a school information page
- Colours and fonts
- ...and 4 more chapters, practice questions and a glossary

Beginner

Students who want to practise web design with real projects

engineergurukul.com

+91 97116 99511

Build Five Web Pages with HTML and CSS

Build a small website of five real pages, one project at a time: a profile, a school page, a shop, an enquiry form and a photo gallery. Along the way you learn HTML structure, the box model, flexbox, grid, media queries, colours, fonts, accessibility and publishing.

WHO IT IS FOR	Students who want to practise web design with real projects
LEVEL	Beginner
SUBJECT AREA	Programming and web
WHAT IS INSIDE	8 chapters, practice questions with answers, and a glossary

CONTENTS

1	Getting ready: tools and the basics	3
2	Project 1: a personal profile page	6
3	Project 2: a school information page	9
4	Colours and fonts	11
5	Project 3: a shop page and its layout	13
6	Project 4: an admission enquiry form	17
7	Project 5: a responsive photo gallery	21
8	Accessibility, testing and publishing	23
	Practice questions	25
	Answers	26
	Glossary	27

Free ebook: you may read, print and share it for learning; it may not be sold or changed. © 2026 Gurukul Technicals Institute. First edition, October 2026. Contact details are on the back cover.

Getting ready: tools and the basics

A web page is just a text file that a browser knows how to read. You do not need expensive software or even an internet connection to build one. You need two free tools.

Tool	What it does	Examples
Text editor	where you type the code	Notepad (comes with Windows), VS Code
Browser	shows the finished page	Chrome, Firefox, Edge

Notepad is enough to start. VS Code, a free code editor, colours your code and closes tags for you, so install it when you can.

HTML (HyperText Markup Language) describes what each part of a page is: a heading, a paragraph, an image, a link. **CSS** (Cascading Style Sheets) describes how it looks: colours, fonts, spacing and layout. Keep the two in separate files. Then one CSS file can style every page of your site.

Make the project folder

All five projects live in one folder, so the pages can link to each other and share one stylesheet. Create this structure before you start:

```
my-website/
  index.html      Project 1: profile page
  school.html     Project 2: school page
  shop.html       Project 3: shop page
  enquiry.html    Project 4: enquiry form
  gallery.html    Project 5: photo gallery
  css/
    style.css     one stylesheet for all pages
  images/
    (your photos go here)
```

Follow these naming rules for every file and folder; they save hours of confusion once a site is online.

- Use small letters only: `school.html`, not `School.HTML`. Many web servers treat capital and small letters as different.
- No spaces. Use a hyphen instead: `hawa-mahal.jpg`.
- Keep the right extension: `.html` for pages, `.css` for styles, `.jpg` or `.png` for images.

TIP

Windows hides file extensions by default, so "index.html" can secretly become "index.html.txt". In File Explorer, open the View options and turn on "File name extensions".

REMEMBER

The home page of any website folder is called `index.html`. Servers open it automatically when someone visits the folder.

Elements and tags

HTML is made of **elements**. Most elements have an opening tag, some content and a closing tag with a slash:

HTML

```
<p>Delhi is the capital of India.</p>
```

Here `<p>` is the opening tag, `</p>` the closing tag, and together they make a paragraph element. Some elements, such as `<img>` and `<br>`, have no content and no closing tag. They are called **void elements**.

An **attribute** gives extra information inside the opening tag, written as `name="value"`. In `<a href="school.html">`, the attribute `href` tells the link where to go.

Elements sit inside each other like boxes inside boxes. This is called **nesting**. Always close the inner element before the outer one: `<p><strong>Hi</strong></p>` is right, while `<p><strong>Hi</p></strong>` is wrong.

The skeleton of every page

Every page in this book starts with the same skeleton. Projects 2 to 5 show only what goes inside `<body>`: start each one from this skeleton, or from a copy of `index.html`, and change the title.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Page title</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <p>Everything the visitor sees goes here.</p>
</body>
</html>
```

Line	Why it is there
<code><!DOCTYPE html></code>	tells the browser this is modern HTML5
<code><html lang="en"></code>	the whole page; <code>lang</code> names its language for screen readers and search engines
<code><head></code>	information about the page, mostly not shown
<code><meta charset="utf-8"></code>	lets the page show any character, including ! and Hindi letters
<code><meta name="viewport" ...></code>	makes the page fit phone screens properly
<code><title></code>	the text on the browser tab and in search results
<code><link rel="stylesheet" ...></code>	connects the CSS file
<code><body></code>	the visible page

CSS rules

A CSS **rule** has a **selector**, which picks elements, and **declarations**, which say how they look. Each declaration is a **property** and a **value**, ending with a semicolon:

```
CSS

h1 {
  color: navy;
  font-size: 2rem;
}
```

This rule makes every `h1` navy blue and twice the normal text size. Three kinds of selector cover most needs:

Selector	Picks	Example
element name	every element of that kind	<code>p</code>
<code>.name</code>	every element with <code>class="name"</code>	<code>.price</code>
<code>#name</code>	the one element with <code>id="name"</code>	<code>#products</code>

"Cascading" means that when two rules disagree, the more specific one wins, and between equal ones, the later one wins.

Seeing your page

Save the file, then double-click it to open it in your browser. After every change, save and press F5 to reload. To look inside a page, press F12 to open the browser's **developer tools** (DevTools). The Elements panel shows the HTML, and clicking an element shows every CSS rule that applies to it.

TRY THIS

Type the skeleton above into Notepad, change the paragraph to your own name, save it as `test.html` and open it in a browser.

CHAPTER 2

Project 1: a personal profile page

Our first page introduces a student named Riya, from Bhopal. Use your own name, photo and details instead. You will use headings, paragraphs, an image and three kinds of link. Put a square photo of yourself in the `images` folder and name it `profile.jpg`.

The HTML

Save this as `index.html`:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Riya Sharma | My Profile</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <header>
    <h1>Riya Sharma</h1>
    <p>Class 10 student from Bhopal who loves maths and sketching</p>
  </header>
  <main>
    
    <h2>About me</h2>
    <p>I am learning to build web pages. My dream is to design
      websites for small shops in my city.</p>
    <h2>My hobbies</h2>
    <p>I enjoy <strong>badminton</strong>, sketching old buildings and
      reading <em>Panchatantra</em> stories to my little brother.</p>
    <h2>Links</h2>
    <p>Visit my <a href="school.html">school page</a>.</p>
    <p>I learn from <a href="https://developer.mozilla.org/">MDN Web Docs</a>.</p>
    <p>Email me: <a href="mailto:riya@example.com">riya@example.com</a></p>
  </main>
  <footer>
    <p>Made by Riya Sharma</p>
  </footer>
</body>
</html>

```

What each part does

- **Headings** run from `<h1>` (most important) to `<h6>`. Use one `<h1>` for the page title, `<h2>` for sections and `<h3>` for parts of a section. Never pick a heading level just because of its size; CSS controls size.
- `<header>`, `<main>` and `<footer>` are **semantic elements**: their names describe their role, which helps search engines and screen readers. `<main>` holds the page's main content and appears once per page.
- `<p>` is a paragraph. The browser ignores extra spaces and line breaks in your code, so the lines inside a paragraph join up on screen.
- `<strong>` marks important words (shown in bold) and `<em>` marks emphasis (shown in italics).
- `<img>` shows a picture. `src` is the file's path, and `alt` is a description read aloud to blind users and shown if the image fails to load. `width` and `height` let the browser keep the right space while the image loads, so the text does not jump.

There are three kinds of link here. `school.html` is a **relative link** to a file in the same folder. The MDN link is an **absolute link**: a full web address starting with `https://`. `mailto:` opens the visitor's email app.

The CSS

Create `css/style.css` and type the first rules:

CSS

```
body {
  margin: 0;
  font-family: Arial, Helvetica, sans-serif;
  line-height: 1.6;
  color: #222222;
  background: #fafafa;
}

header,
footer {
  background: #1a5fb4;
  color: #ffffff;
  text-align: center;
  padding: 1rem;
}

footer a { color: #ffffff; }

main {
  max-width: 800px;
  margin: 0 auto;
  padding: 1rem;
}

img {
  max-width: 100%;
  height: auto;
}

.profile-photo {
  display: block;
  margin: 1rem auto;
  border-radius: 50%;
}
```

`max-width: 800px` with `margin: 0 auto` keeps lines short enough to read and centres the content. `img { max-width: 100%; height: auto; }` stops a large photo from spilling off a phone screen. `border-radius: 50%` turns the square photo into a circle.

REMEMBER

The path in `src` and `href` is counted from the HTML file. Because `index.html` sits next to the `images` folder, the path is `images/profile.jpg`. A broken image almost always means a wrong path or a misspelt file name.

CHAPTER 3

Project 2: a school information page

This page describes a school using **lists** and a **table**. Our example is a made-up school called Kiran Public School; fill in your own school's real details if you like.

Three kinds of list

List	Tag	Use it for
Unordered	<code></code>	items in no special order, shown with bullets
Ordered	<code></code>	steps or rankings, shown with numbers
Description	<code><dl></code>	terms with their meanings

Each item of `<ul>` or `<ol>` goes in `<li>` (list item). In a `<dl>`, each term is a `<dt>` and its meaning a `<dd>`.

Tables

A table is built row by row. `<tr>` is a table row, `<th>` a heading cell and `<td>` a data cell. `<thead>` holds the heading row and `<tbody>` the data rows. `<caption>` gives the table a title. Use tables only for data in rows and columns, such as a timetable, never to lay out a page.

The HTML

Make `school.html` from the skeleton, with the title `Kiran Public School | About Our School`, and put this inside `<body>`:

```

<header>
  <h1>Kiran Public School</h1>
  <p>A made-up school for this practice page</p>
</header>
<main>
  <h2>Facilities</h2>
  <ul>
    <li>Computer lab</li>
    <li>Science laboratories</li>
    <li>Library with a reading corner</li>
    <li>Playground and indoor games room</li>
  </ul>
  <h2>How to reach the school</h2>
  <ol>
    <li>Get off at the main bus stand.</li>
    <li>Walk straight past the post office.</li>
    <li>Turn left at the park gate.</li>
  </ol>
  <h2>Class 7 timetable</h2>
  <table>
    <caption>First three periods, Monday and Tuesday</caption>
    <thead>
      <tr><th scope="col">Time</th><th scope="col">Monday</th><th
scope="col">Tuesday</th></tr>
    </thead>
    <tbody>
      <tr><th scope="row">8:00</th><td>Maths</td><td>English</td></tr>
      <tr><th scope="row">8:40</th><td>Science</td><td>Hindi</td></tr>
      <tr><th scope="row">9:20</th><td colspan="2">Assembly in the hall</td></tr>
    </tbody>
  </table>
  <h2>School words</h2>
  <dl>
    <dt>PTM</dt>
    <dd>Parent-teacher meeting, where parents discuss a child's progress.</dd>
    <dt>House</dt>
    <dd>A team of students from all classes that competes in events.</dd>
  </dl>
</main>
<footer>
  <p><a href="index.html">Back to my profile</a></p>
</footer>

```

`scope="col"` and `scope="row"` tell screen readers which cells each heading belongs to.

`colspan="2"` makes one cell stretch across two columns, which suits the assembly that fills both days. (`rowspan` does the same downwards.)

The CSS

Add these rules to the end of `style.css`:

CSS

```
table {
  width: 100%;
  border-collapse: collapse;
  margin: 1rem 0;
}

caption {
  font-weight: bold;
  text-align: left;
  padding-bottom: 0.5rem;
}

th,
td {
  border: 1px solid #999999;
  padding: 0.5rem;
  text-align: left;
}

thead th { background: #e3ecfa; }

tbody tr:nth-child(even) { background: #f0f0f0; }

dt { font-weight: bold; }
```

`border-collapse: collapse` merges the double lines between cells into single ones. `tbody tr:nth-child(even)` picks every second row, giving the soft stripes that help the eye follow a row across.

EXAMPLE

Add a third column for Wednesday. You need one more `<th>` in the heading row and one more `<td>` in each data row, and the assembly cell becomes `colspan="3"`.

CHAPTER 4

Colours and fonts

Before the next project, let us choose colours and fonts properly, so that every page looks like part of one site.

Writing colours

Way	Example	Notes
Name	navy, white	easy, but only about 140 names exist
Hex code	#1a5fb4	# then red, green and blue as two hex digits each

Way	Example	Notes
rgb()	rgb(26, 95, 180)	red, green and blue from 0 to 255
rgba()	rgba(0, 0, 0, 0.5)	the fourth value is transparency, 0 to 1

#1a5fb4 and rgb(26, 95, 180) are the same blue. Most code editors show a little colour box next to a colour code and let you pick a shade with the mouse.

CSS variables

A **custom property**, often called a CSS variable, stores a value once so you can reuse it everywhere. Its name starts with two hyphens, and `var()` reads it. Add these lines to the very top of `style.css`:

```

CSS
:root {
  --brand: #1a5fb4;
  --accent: #c64600;
}

```

`:root` means the whole page. The later projects use `var(--brand)`, so changing one line here recolours all of them.

Contrast

Text must stand out clearly from its background. Dark grey on white, or white on a deep blue, is easy to read. Light grey on white or red on green is not, especially in sunlight on a phone or for people with weak eyesight or colour blindness. DevTools in Chrome and Edge shows a contrast ratio when you inspect a text colour. Aim for at least 4.5 to 1 for normal text, the level that the Web Content Accessibility Guidelines (WCAG) recommend.

Fonts

`font-family` takes a list called a **font stack**. The browser uses the first font the visitor's device has, and falls back to the next. Always end with a generic family: `sans-serif` (plain letters, good on screens), `serif` (letters with small feet) or `monospace` (every letter the same width, used for code).

Size text with `rem`, not `px`. One `rem` equals the root font size, which is usually 16 pixels but follows the visitor's own setting. A person who sets a bigger default text size then gets a bigger page. Add this rule to give headings a different font from the body:

CSS

```
h1,
h2,
h3 {
  font-family: Georgia, "Times New Roman", serif;
  line-height: 1.2;
}
```

TIP

Use at most two fonts on a site, one for headings and one for body text, and three or four colours. Restraint looks professional; ten colours look messy.

CHAPTER 5

Project 3: a shop page and its layout

Project 3 is a page for a made-up stationery shop, Meera Stationery. It has a header with a menu and a grid of product cards with prices in rupees. First you need the most important idea in CSS layout: the box model.

The box model

The browser draws every element as a rectangular box made of four layers, from the inside out:

1. **Content:** the text or image itself.
2. **Padding:** space inside the box, between the content and the border. It takes the box's background colour.
3. **Border:** a line around the padding.
4. **Margin:** space outside the border that pushes other boxes away. It is always transparent.

Think of a framed photo on a wall: the photo is the content, the white mount around it is the padding, the frame is the border and the wall space between frames is the margin.

By default, `width` sets only the content. So a card with `width: 300px`, `padding: 16px` and a 1-pixel border actually takes $300 + 16 + 16 + 1 + 1 = 334$ pixels, which surprises most beginners. The fix is one rule that makes `width` include the padding and border. Add it near the top of `style.css`, just under the `:root` block:

CSS

```
*, *::before, *::after { box-sizing: border-box; }
```

Setting

`width: 300px` with 16px padding and a 1px border

`content-box` (default)

334px wide on screen

border-box

300px wide on screen; the content shrinks to 266px

* is the **universal selector**: it picks every element. You can watch the box model live: in DevTools, select an element and look for the coloured box diagram in the Styles or Computed panel.

The HTML

Make `shop.html` from the skeleton, with the title `Meera Stationery | School Supplies`, and put this inside `<body>`. Add three product photos to `images`.

HTML

```
<header class="shop-header">
  <h1>Meera Stationery</h1>
  <nav>
    <a href="#products">Products</a>
    <a href="enquiry.html">Enquiry</a>
    <a href="gallery.html">Gallery</a>
  </nav>
</header>
<main>
  <h2 id="products">Back-to-school products</h2>
  <div class="product-grid">
    <article class="card">
      
      <h3>Ruled notebook</h3>
      <p class="price">₹60</p>
      <p>Long size, 172 pages.</p>
    </article>
    <article class="card">
      
      <h3>Geometry box</h3>
      <p class="price">₹120</p>
      <p>Compass, divider, protractor and scales.</p>
    </article>
    <article class="card">
      
      <h3>Steel water bottle</h3>
      <p class="price">₹250</p>
      <p>Holds 750 ml.</p>
    </article>
  </div>
</main>
<footer>
  <p><a href="index.html">Back to my profile</a></p>
</footer>
```

`<nav>` holds the main menu. `href="#products"` jumps to the element whose `id` is `products` on the same page. Each product is an `<article>`, an element for a self-contained piece of content. To add more products, copy a whole `<article>` block and change its details.

Styling the cards

Add the card rules to `style.css`:

```
CSS

.card {
  background: #ffffff;
  border: 1px solid #dddddd;
  border-radius: 8px;
  padding: 1rem;
}

.card h3 { margin: 0.5rem 0; }

.price {
  margin: 0;
  font-size: 1.25rem;
  font-weight: bold;
  color: var(--accent);
}
```

Each card now has 1rem of padding inside its border, so the text never touches the edge. The next section places the cards side by side.

DID YOU KNOW

The gap between two paragraphs comes from their margins. When the bottom margin of one box meets the top margin of the next, the browser keeps only the larger of the two, instead of adding them. This is called **margin collapsing**.

Flexbox and grid

So far the boxes stack one under another. Two CSS layout systems arrange them side by side.

	Flexbox	Grid
Works in	one direction: a row or a column	two directions: rows and columns together
Best for	menus, toolbars, lining up a few items	page sections, cards, galleries
Switched on with	<code>display: flex;</code>	<code>display: grid;</code>

The element you apply it to is the **container**, and its direct children are the **items**.

Flexbox for the header

We want the shop name on the left and the menu on the right. Add:

CSS

```
.shop-header {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  align-items: center;
  gap: 0.5rem 1.5rem;
  text-align: left;
}

.shop-header h1 { margin: 0; }

.shop-header nav {
  display: flex;
  gap: 1rem;
}

.shop-header a {
  color: #ffffff;
  text-decoration: none;
  font-weight: bold;
}
```

Property	What it does here
<code>justify-content: space-between</code>	pushes the items to the two ends of the main direction (left and right)
<code>align-items: center</code>	centres them in the other direction (up and down)
<code>flex-wrap: wrap</code>	lets the menu drop to a new line on a narrow phone instead of squashing
<code>gap</code>	space between items, with no need for margins

The `nav` is a flex container too, so its three links sit in a row with 1rem between them. `text-align: left` overrides the centring the header got from Project 1's rule, because this selector is more specific.

Grid for the products

Add:

CSS

```
.product-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
  gap: 1rem;
}
```

This one line is worth understanding piece by piece:

- `minmax(220px, 1fr)`: each column is at least 220 pixels wide and at most one **fraction** (`fr`) of the free space.
- `repeat(auto-fit, ...)`: make as many such columns as fit in the row.

So the grid shows three cards in a row on a wide screen, two on a narrower one and one on a phone, with no media queries at all. Make the browser window narrower and wider to watch it happen.

TRY THIS

Change 220px to 300px and watch how the number of columns changes. Then try `grid-template-columns: 1fr 2fr`; to see two columns where the second is twice as wide.

CHAPTER 6

Project 4: an admission enquiry form

Forms let visitors send information: a name, a phone number, a choice. Project 4 is an enquiry form that a coaching centre or school might put on its site.

Form controls

Control	Tag	Use
Text box	<code><input type="text"></code>	short text such as a name
Email	<code><input type="email"></code>	checks for an @ and a domain
Phone	<code><input type="tel"></code>	brings up the number keypad on phones
Date	<code><input type="date"></code>	shows a date picker
Drop-down	<code><select></code> with <code><option></code>	choose one from a list
Radio buttons	<code><input type="radio"></code>	choose exactly one of a few
Checkbox	<code><input type="checkbox"></code>	yes or no, or several choices
Text area	<code><textarea></code>	longer, multi-line text
Button	<code><button></code>	send or clear the form

Every control needs a **label**, the visible text that says what to type. Connect them by giving the control an `id` and the label a matching `for`. Then clicking the label puts the cursor in the box, and a screen reader announces the label. Placeholder text inside a box is not a label, because it disappears as soon as you type.

The HTML

Make `enquiry.html` from the skeleton, with the title `Admission Enquiry Form`, and put this inside `<body>`:

HTML

```
<header>
  <h1>Admission enquiry</h1>
</header>
<main>
  <form class="enquiry">
    <label for="name">Full name</label>
    <input type="text" id="name" name="name" required minlength="3"
autocomplete="name">
    <label for="email">Email</label>
    <input type="email" id="email" name="email" required autocomplete="email">
    <label for="phone">Mobile number (10 digits)</label>
    <input type="tel" id="phone" name="phone" required pattern="[0-9]{10}"
autocomplete="tel">
    <label for="dob">Date of birth</label>
    <input type="date" id="dob" name="dob">
    <label for="course">Course</label>
    <select id="course" name="course" required>
      <option value="">Choose a course</option>
      <option value="basics">Computer basics</option>
      <option value="web">Web design</option>
      <option value="coding">Programming</option>
    </select>
    <fieldset>
      <legend>Preferred batch</legend>
      <label><input type="radio" name="batch" value="morning" checked> Morning</
label>
      <label><input type="radio" name="batch" value="evening"> Evening</label>
    </fieldset>
    <label for="question">Your question</label>
    <textarea id="question" name="question" rows="4" maxlength="300"></textarea>
    <label><input type="checkbox" name="agree" required> I agree to be contacted
about this enquiry</label>
    <div class="buttons">
      <button type="submit">Send enquiry</button>
      <button type="reset">Clear</button>
    </div>
  </form>
</main>
<footer>
  <p><a href="index.html">Back to my profile</a></p>
</footer>
```

- `name` is the label the data travels under when the form is sent. Radio buttons share one `name`, which is what makes them a group where only one can be picked.
- `<fieldset>` groups related controls and `<legend>` is the group's caption.

- A label can also wrap its control, as with the radio buttons and the checkbox; then no `for` is needed.
- `autocomplete` lets the browser fill in details the visitor has saved, which helps a lot on phones.

Validation attributes

The browser checks the form before sending it, with no programming needed:

Attribute	Rule it adds
<code>required</code>	the field cannot be left empty
<code>type="email"</code>	the text must look like an email address
<code>minlength="3" / maxlength="300"</code>	the shortest and longest text allowed
<code>pattern="[0-9]{10}"</code>	the text must match the pattern: here, exactly ten digits
<code>min / max</code>	the smallest and largest number or date allowed

Press "Send enquiry" with an empty name and the browser shows a message and refuses to send.

SAFETY

Browser checks only help honest visitors fix mistakes. Anyone can switch them off, so a real website must check the data again on its server before saving it. Never ask for passwords, Aadhaar numbers or bank details on a practice form.

This practice form has nowhere to send its data. Pressing the button reloads the page, and the address bar shows what you typed after a question mark. A real site adds `method="post"` and an `action` that points to a server program, which a web developer or the hosting company sets up.

The CSS

CSS

```
.enquiry {
  display: grid;
  gap: 0.5rem;
  max-width: 480px;
}

.enquiry input,
.enquiry select,
.enquiry textarea {
  width: 100%;
  padding: 0.5rem;
  font: inherit;
  border: 1px solid #888888;
  border-radius: 4px;
}

.enquiry input[type="radio"],
.enquiry input[type="checkbox"] {
  width: auto;
}

.enquiry :focus {
  outline: 3px solid var(--accent);
  outline-offset: 1px;
}

.enquiry fieldset {
  border: 1px solid #888888;
  border-radius: 4px;
}

.buttons {
  display: flex;
  gap: 1rem;
}

button {
  padding: 0.6rem 1.2rem;
  font: inherit;
  color: #ffffff;
  background: var(--brand);
  border: none;
  border-radius: 4px;
  cursor: pointer;
}
```

`font: inherit` makes form controls use the page's font, since browsers otherwise give them their own smaller one. `input[type="radio"]` is an **attribute selector**: it picks only inputs whose `type` is `radio`. The `:focus` outline shows clearly which box is active when someone moves through the form with the Tab key.

Project 5: a responsive photo gallery

A **responsive** page rearranges itself to suit the screen: one column on a phone, more on a tablet and a laptop. Project 5 is a gallery of photos from a school trip to Jaipur, and it uses **media queries** to change the layout at different widths.

The viewport meta tag

Phones pretend to be wide screens unless told otherwise, and then they shrink the whole page until the text is too small to read. The line `<meta name="viewport" content="width=device-width, initial-scale=1">`, already in every page of this book, tells the phone to use its real width. Without it, media queries do not work as expected on phones.

The HTML

Make `gallery.html` from the skeleton, with the title `Jaipur Trip | Photo Gallery`, and put this inside `<body>`. Put three photos in `images`.

HTML

```
<header>
  <h1>Our class trip to Jaipur</h1>
</header>
<main>
  <div class="gallery">
    <figure>
      
      <figcaption>Hawa Mahal</figcaption>
    </figure>
    <figure>
      
      <figcaption>Amer Fort</figcaption>
    </figure>
    <figure>
      
      <figcaption>All of us, tired but happy</figcaption>
    </figure>
  </div>
</main>
<footer>
  <p><a href="index.html">Back to my profile</a></p>
</footer>
```

`<figure>` holds an image with its caption, and `<figcaption>` is the caption. `loading="lazy"` tells the browser to fetch a photo only when the visitor scrolls near it, which saves mobile data. The first photo loads normally because it is visible straight away.

Mobile-first CSS

We write the phone layout first, then add media queries for wider screens. This is called **mobile-first** design.

CSS

```
.gallery {
  display: grid;
  grid-template-columns: 1fr;
  gap: 1rem;
}

.gallery figure {
  margin: 0;
  background: #ffffff;
  border-radius: 8px;
  overflow: hidden;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.15);
}

.gallery img {
  display: block;
  width: 100%;
  height: 220px;
  object-fit: cover;
}

.gallery figcaption { padding: 0.5rem 0.75rem; }

@media (min-width: 600px) {
  .gallery { grid-template-columns: repeat(2, 1fr); }
}

@media (min-width: 900px) {
  .gallery { grid-template-columns: repeat(3, 1fr); }
}
```

A **media query** applies its rules only when its condition is true. `@media (min-width: 600px)` means "when the screen is at least 600 pixels wide". So:

Screen width	Columns
under 600px (most phones)	1
600px to 899px (many tablets)	2
900px and over (laptops, desktops)	3

The widths where the layout changes are called **breakpoints**. Choose them where your design starts to look cramped, not for particular phone models.

`object-fit: cover` fills the 220-pixel-tall box with the photo, trimming the edges instead of stretching it, so photos of different shapes still line up neatly. `overflow: hidden` clips the photo to the card's rounded corners.

DID YOU KNOW

Photos straight from a phone camera are often several megabytes each. Before using them on a website, resize them to around the size they are shown (600 pixels wide here) and save them as compressed JPG or WebP files. The page then loads far faster on mobile data.

CHAPTER 8

Accessibility, testing and publishing

Accessibility means making pages that everyone can use, including people who are blind and use a **screen reader** (software that reads the page aloud), people who cannot use a mouse, and people with weak eyesight or colour blindness. They are also easier for everyone else, and search engines understand them better.

An accessibility checklist

Check	How these projects do it
Every image has useful <code>alt</code> text	<code>alt</code> describes what the photo shows; a purely decorative image gets <code>alt=""</code>
Headings are in order	one <code><h1></code> , then <code><h2></code> , then <code><h3></code> , with no levels skipped
Every form control has a label	<code><label for="..."></code> or a wrapping label
Link text makes sense alone	"school page", not "click here"
Text has strong contrast	dark text on light backgrounds, white on deep blue
The page works with a keyboard	try it: Tab moves forward, Shift+Tab back, Enter follows a link
Focus is visible	the <code>:focus</code> outline from Project 4
The language is set	<code><html lang="en"></code>

Good `alt` text is short and says what matters in that context. For Riya's photo, "Riya smiling in her school uniform" is useful. Words like "image of" or "photo" are not needed, because the screen reader already says it is an image.

TRY THIS

Put your mouse away and use only the keyboard on `enquiry.html`. Can you reach every box, pick a batch, tick the checkbox and press Send? If anything is skipped or invisible, fix it.

Checking in mobile view

You do not need a phone to test a phone layout, although testing on a real phone at the end is still a good idea.

1. Open the page in Chrome or Edge and press F12 to open DevTools.
2. Press Ctrl+Shift+M to turn on device mode. The page now shows at phone size.
3. Pick a phone from the list at the top, or drag the edge of the page to any width.
4. Check the gallery at different widths: 1, then 2, then 3 columns.
5. Check that no page scrolls sideways, all text is readable without zooming, and buttons are big enough to tap with a thumb.

Firefox has the same feature, called Responsive Design Mode, from its own developer tools.

REMEMBER

A page that scrolls sideways on a phone usually has one element wider than the screen. In device mode, scroll to the right edge and inspect what sticks out. The usual culprits are a fixed width in pixels, a large image without `max-width: 100%`, or a long word or web address that cannot wrap.

Publishing basics

Your site works on your computer. To let others see it, its files must be copied to a **web server**, a computer that is always connected to the internet and sends pages to browsers when asked.

Three things you need

Thing	What it is	Everyday comparison
Hosting	space on a web server for your files	renting a shop
Domain name	the name people type, ending in <code>.in</code> , <code>.com</code> and so on	the shop's signboard and address
Upload method	a way to copy your files to the server	the van that carries your stock

A domain name is rented from a company called a **registrar** for one or more years at a time, and then renewed. Hosting companies sell space, often with a domain included. Some services also offer free plans for small static sites like this one; GitHub Pages is a well-known example. Read the current terms and prices of any service on its own website before you sign up.

Before you upload

1. Open every page and click every link. A **broken link** leads to a "404 Not Found" error page.
2. Check that every image appears. Remember that online servers usually treat `Photo.JPG` and `photo.jpg` as different files.
3. Test each page in device mode at phone width.
4. Check your HTML with the W3C's free online validator, which lists mistakes such as unclosed tags. Fix them one by one, starting with the first.
5. Make sure the home page is called `index.html` and sits at the top of the folder.

Uploading

Most hosting companies give you a control panel with a **file manager**, where you upload files from your browser. Keep the same folder structure: `index.html` at the top, with the `css` and `images` folders beside it. Another common method is an **FTP** client (File Transfer Protocol), a program that shows your computer's files on one side and the server's on the other, so you can drag files across. After uploading, open your domain in a browser, press Ctrl+F5 to skip any old saved copy, and test everything again on a real phone.

SAFETY

Never put private details on a public website, such as your home address, phone number or school timetable. Ask a parent or teacher before putting your own photo online, and never post photos of other people without their permission. Riya's page uses an example email address for exactly this reason.

What to learn next

You now know the core of HTML and CSS. Natural next steps are JavaScript, which makes pages interactive, and a code editor's extensions that preview your page live as you type. Keep building small real pages: a menu card for a local dhaba, a family event invitation or a page for your school club. Each one will teach you something new.

CHECK YOURSELF

Practice questions

1. What is the difference between HTML and CSS?
2. Why should file names use small letters and no spaces?
3. What is the name of the home page file in a website folder, and why does it matter?
4. What is an attribute? Give an example from Project 1.
5. What does `<meta charset="utf-8">` do?
6. Why must every image have `alt` text?

7. What is the difference between a relative link and an absolute link?
8. When would you use `<ol>` instead of `<ul>`?
9. What do `<thead>`, `<tbody>`, `<th>` and `<td>` hold?
10. What does `colspan="2"` do?
11. What is the difference between a class selector and an id selector?
12. Why is `rem` better than `px` for text size?
13. Name the four layers of the box model, from the inside out.
14. A box has `width: 200px`, `padding: 10px` and a 2px border. How wide is it with `content-box` and with `border-box`?
15. When would you choose flexbox, and when grid?
16. What does `repeat(auto-fit, minmax(220px, 1fr))` do?
17. How do you connect a label to an input, and why does it matter?
18. Why do the radio buttons in Project 4 share the same `name`?
19. Name four validation attributes.
20. Why must a real website check form data on its server as well?
21. What does `@media (min-width: 600px)` mean?
22. What is mobile-first design?
23. How do you turn on device mode in Chrome's developer tools?
24. What are hosting and a domain name?

ANSWER KEY

Answers

1. HTML describes what each part of a page is (headings, paragraphs, images); CSS describes how it looks (colours, fonts, layout).
2. Many web servers treat capital and small letters as different, and spaces cause problems in web addresses, so links can break once the site is online.
3. `index.html`. Servers open it automatically when someone visits the folder or the site's address.
4. Extra information inside an opening tag, written as `name="value"`, such as `src="images/profile.jpg"` on the `<img>`.
5. It tells the browser the file uses UTF-8, so characters such as ! and Hindi letters display correctly.
6. Screen readers read it aloud to blind users, and it appears if the image fails to load.
7. A relative link points to a file by its path from the current page, such as `school.html`; an absolute link is a full address starting with `https://`.

8. When the order matters, such as steps of directions or a ranking.
9. `<thead>` holds the heading rows and `<tbody>` the data rows; `<th>` is a heading cell and `<td>` a data cell.
10. It makes one cell stretch across two columns.
11. A class selector (`.price`) can style many elements with that class; an id selector (`#products`) styles the one element with that id.
12. `rem` follows the visitor's chosen text size, so people who need bigger text get it; `px` ignores their setting.
13. Content, padding, border, margin.
14. With `content-box: 200 + 10 + 10 + 2 + 2 = 224px`. With `border-box: 200px`.
15. Flexbox for items in one direction, such as a menu; grid for layouts in rows and columns together, such as product cards.
16. It creates as many columns as fit, each at least 220px and sharing the extra space equally, so the number of columns changes with the screen width.
17. Give the input an `id` and the label the same value in `for`, or wrap the input in the label. Clicking the label then focuses the input, and screen readers announce it.
18. A shared `name` makes them one group, so only one batch can be selected.
19. Any four of `required`, `minlength`, `maxlength`, `pattern`, `min`, `max` and `type="email"`.
20. Browser checks can be switched off, so only a check on the server can be trusted before data is saved.
21. Apply the rules inside it only when the screen is at least 600 pixels wide.
22. Writing the CSS for small screens first, then adding media queries for larger screens.
23. Press F12 to open DevTools, then Ctrl+Shift+M.
24. Hosting is space on a web server that stores your site's files; a domain name is the address people type to reach it.

WORD LIST

Glossary

Absolute link

a full web address, starting with `https://`.

Accessibility

making pages usable by everyone, including people with disabilities.

Attribute

extra information in an opening tag, written as `name="value"`.

Box model

the content, padding, border and margin that make up every element's box.

Breakpoint

a screen width at which a media query changes the layout.

CSS

Cascading Style Sheets, the language that controls how a page looks.

Custom property

a reusable CSS value, such as `--brand`, read with `var()`.

DevTools

the developer tools built into a browser, opened with F12.

Domain name

the address people type to reach a website.

Element

a part of a page made by a tag, such as a paragraph or an image.

Flexbox

a CSS layout system that arranges items in one row or one column.

Font stack

a list of fonts in `font-family`, tried in order.

Grid

a CSS layout system that arranges items in rows and columns.

Hosting

space on a web server where a site's files are kept.

HTML

HyperText Markup Language, which describes the parts of a web page.

Media query

a CSS rule that applies only when a condition, such as a screen width, is true.

Relative link

a link to a file given by its path from the current page.

Responsive design

pages that adapt their layout to the screen size.

Screen reader

software that reads a page aloud for blind and low-vision users.

Selector

the part of a CSS rule that picks which elements to style.

Semantic element

an element whose name describes its role, such as `header` or `nav`.

Viewport

the visible area of the page in the browser window or phone screen.



Gurukul Technicals Institute

Computer education institute in Delhi, with authorised member centres, robotics classes and science projects, working models and DIY kits.

COURSES

Computer & Office Skills · Accounting & Business Software · Programming & Web Development · Design & Digital Skills · Hardware & Networking · Robotics & Coding · Other Programmes

PROJECTS AND KITS

Ready-made school projects · DIY and STEM kits · Robotics and coding · Components and spares · Engineering projects

CONTACT

Head office: A-10, Chief Commissioner Colony, opposite Rana Pratap Bagh, Delhi 110007

Phone: 011 2721 1512 · **Mobile and WhatsApp:** +91 97116 99511

Email: engineergurukul@yahoo.co.in

Website: engineergurukul.com · **Instagram:** [@science_projects_club](https://www.instagram.com/science_projects_club)

More free ebooks for computer courses, robotics and science projects: engineergurukul.com/resources/ebooks/

Free to read, print and share. Not for sale.