



**Gurukul Technicals Institute**

Computer education · Robotics · Science projects · Delhi

**FREE EBOOK**

PROGRAMMING AND WEB

# HTML and CSS Study Book

The complete study notes, practice questions and glossary for HTML and CSS

---

## INSIDE THIS BOOK

- The structure of an HTML5 document
- Headings, paragraphs, lists, links and images
- Tables and forms
- Semantic HTML5 elements
- CSS basics: selectors, colours and fonts
- ...and 2 more chapters, practice questions and a glossary

---

**Beginner**

Anyone who wants to build web pages

**engineergurukul.com**

+91 97116 99511

# HTML and CSS Study Book

Build web pages with HTML5 and style them with CSS3, from document structure, text, tables and forms to selectors, the box model, Flexbox, Grid and responsive design.

|                |                                                             |
|----------------|-------------------------------------------------------------|
| WHO IT IS FOR  | Anyone who wants to build web pages                         |
| LEVEL          | Beginner                                                    |
| SUBJECT AREA   | Programming and web                                         |
| WHAT IS INSIDE | 7 chapters, practice questions with answers, and a glossary |

CONTENTS

|   |                                               |    |
|---|-----------------------------------------------|----|
| 1 | The structure of an HTML5 document            | 3  |
| 2 | Headings, paragraphs, lists, links and images | 5  |
| 3 | Tables and forms                              | 8  |
| 4 | Semantic HTML5 elements                       | 10 |
| 5 | CSS basics: selectors, colours and fonts      | 13 |
| 6 | The box model and Flexbox                     | 15 |
| 7 | CSS Grid and responsive design                | 18 |
|   | Practice questions                            | 21 |
|   | Answers                                       | 24 |
|   | Glossary                                      | 26 |

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.

# The structure of an HTML5 document

---

## What HTML is

HTML stands for **HyperText Markup Language**. It describes the content and structure of a web page: which text is a heading, which is a paragraph, where an image goes and what a link points to. HTML is a markup language, not a programming language. It has no variables, loops or calculations; it only labels content so that a web browser such as Chrome, Edge or Firefox knows how to display it.

The current version is **HTML5**. It is maintained as a "living standard", which means browsers keep adding small improvements instead of waiting for a big new version.

## Tags, elements and attributes

HTML is written with **tags** inside angle brackets. Most tags come in pairs: an opening tag such as `<p>` and a closing tag such as `</p>`. The opening tag, the content and the closing tag together form an **element**.

```
HTML
```

```
<p class="intro">Namaste! Welcome to my first web page.</p>
```

Here `p` is the element name, `class="intro"` is an **attribute** and the text between the tags is the content. Attributes give extra information and are always written in the opening tag as `name="value"`.

Some elements have no content and therefore no closing tag. These are called **void** (or empty) elements, for example `<br>` (line break), `<hr>`, `<img>`, `<input>`, `<meta>` and `<link>`.

## The basic skeleton

Every HTML5 page follows the same structure:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Ravi's Mobile Shop</title>
  </head>
  <body>
    <h1>Ravi's Mobile Shop</h1>
    <p>New and refurbished phones in Karol Bagh.</p>
  </body>
</html>

```

- `<!DOCTYPE html>` tells the browser this is an HTML5 document, so it uses standards mode.
- `<html>` is the root element that wraps everything. Its `lang` attribute states the language of the page, which helps screen readers and search engines.
- `<head>` holds information about the page: the character set, the viewport setting for phones, the title shown on the browser tab and links to stylesheets. It is not shown inside the page.
- `<body>` holds everything the visitor actually sees.

`<meta charset="utf-8">` lets the page show text in any language correctly, including Hindi and the rupee sign ! .

## Writing good HTML

1. Nest elements properly: close the inner element before the outer one.  
`<p><strong>Sale</strong></p>` is correct; `<p><strong>Sale</p></strong>` is wrong.
2. Write tag and attribute names in lowercase. HTML does not require it, but it is the common convention.
3. Put attribute values in double quotes.
4. Leave notes for yourself with comments such as `<!-- price list starts here -->`. Browsers do not display them.
5. Save the file with the `.html` extension, for example `index.html`, and open it in a browser to see the result.

Right-click any web page and choose View page source to see its HTML. Reading other pages is a good way to learn.

## KEY POINTS

- HTML is a markup language that labels the structure of content; it is not a programming language.
- An element is usually an opening tag, content and a closing tag; attributes go in the opening tag as `name="value"`.
- Every page starts with `<!DOCTYPE html>` and has one html element containing a head and a body.
- The head holds information about the page, such as charset, viewport and title; the body holds the visible content.
- Void elements such as `br`, `img`, `input` and `meta` have no closing tag.

## TRY THESE EXERCISES

1. Open Notepad, type the HTML5 skeleton with the doctype, html (lang set to en), head and body elements, set the title to My First Page, save it as index.html in a new folder named website with All files as the save type and UTF-8 encoding, and open it in Chrome or Edge to check that the title shows on the browser tab.
2. In the same page, add an h1 heading with your name, two paragraphs about your favourite subject, an hr line between them, a br line break inside one paragraph and a comment that marks where the second paragraph starts; reload the page and confirm that the comment is not displayed.
3. Add a sentence containing the rupee sign ! and a few words in Hindi and check that they display correctly, then right-click any web page, choose View page source and write down five element names and two attributes you recognise.

## CHAPTER 2

# Headings, paragraphs, lists, links and images

## Headings and paragraphs

HTML has six levels of headings, from `<h1>` (most important) to `<h6>` (least important). Use one `<h1>` for the main title of the page, `<h2>` for its sections and `<h3>` for subsections inside them. Do not skip levels just to get a smaller font; size can always be changed with CSS. Screen readers and search engines use headings to understand how a page is organised.

Text goes in paragraphs with `<p>`. The browser joins extra spaces and line breaks in your code into a single space, so use `<br>` only when a line break is part of the content, such as in a postal address.

Two pairs of elements are often confused:

- `<strong>` marks text of strong importance, while `<b>` only draws attention, for example to a product name. Both look bold by default.
- `<em>` adds emphasis, the stress you would put on a word when speaking, while `<i>` marks a different voice, such as a technical term. Both look italic by default.

## Lists

- An **unordered list** `<ul>` shows bullet points, for items where order does not matter.
- An **ordered list** `<ol>` shows numbers, for steps where order matters. Its `type` attribute can switch to letters or Roman numerals, and `start` sets the first number.
- Every item in either kind of list is an `<li>`. A complete list can be placed inside an `<li>` to make a nested list.

### HTML

```
<h2>Documents to bring</h2>
<ul>
  <li>Identity proof</li>
  <li>Two passport-size photos</li>
</ul>
```

## Links

Links use the anchor element `<a>` and its `href` attribute:

### HTML

```
<a href="https://www.example.com">Example website</a>
<a href="contact.html">Contact us</a>
<a href="#timings">See shop timings</a>
<a href="mailto:info@example.com">Email us</a>
```

An **absolute URL** contains the full address, including `https://`. A **relative URL** such as `contact.html` points to a file on the same website, relative to the current page. An `href` that starts with `#` jumps to the element on the same page with that `id`. A `mailto:` link opens an email app and a `tel:` link offers to call the number on a mobile phone. Adding `target="_blank"` opens the link in a new tab. Write link text that says where the link goes: "Download the price list" is far more useful than "click here".

## Images

Images use the void element `<img>`:

```

```

- `src` is the path or URL of the image file.
- `alt` is a text alternative, read aloud by screen readers and shown if the image fails to load. For a purely decorative image, use an empty `alt=""`.
- `width` and `height` let the browser reserve space before the image loads, so the page does not jump.

Use JPEG for photographs, PNG for graphics that need sharp edges or transparency, SVG for logos and icons that must scale to any size, and WebP for smaller files.

#### KEY POINTS

- Use one h1 for the page title and go down the heading levels in order, without skipping.
- `ul` makes a bulleted list and `ol` a numbered list; every item is an `li`.
- The `a` element creates links; `href` can be an absolute URL, a relative path, a `#id` on the same page, `mailto:` or `tel:`.
- Every `img` needs `alt` text that describes it, or `alt=""` if it is purely decorative.

#### TRY THESE EXERCISES

1. Build a page about your town with one h1, three h2 sections and an h3 under one of them, write a paragraph in each section, use `strong` for one important warning and `em` for one stressed word, and put a postal address on three lines using `br`.
2. Add an unordered list of five things to carry to an exam, an ordered list of the steps to make tea with a nested list under one step, and change the ordered list to Roman numerals with its `type` attribute; then create `contact.html` and link the two pages to each other with relative links.
3. Save a photo in an `images` folder, show it with an `img` element that has descriptive `alt` text, `width` and `height`, then misspell the file name to see the `alt` text appear in place of the image; also add a link that jumps to a section by its `id`, an email link using `mailto:` and a phone link using `tel:`.

# Tables and forms

## Tables for tabular data

A table shows data in rows and columns, such as a timetable, a marksheet or the items on a GST invoice.

| Element                   | Meaning                                     |
|---------------------------|---------------------------------------------|
| <table>                   | The whole table                             |
| <caption>                 | Title of the table, written first inside it |
| <tr>                      | One table row                               |
| <th>                      | A header cell                               |
| <td>                      | A data cell                                 |
| <thead>, <tbody>, <tfoot> | Groups of header, body and footer rows      |

HTML

```
<table>
  <caption>Invoice items</caption>
  <thead>
    <tr><th>Item</th><th>Qty</th><th>Amount (₹)</th></tr>
  </thead>
  <tbody>
    <tr><td>USB keyboard</td><td>2</td><td>1,200</td></tr>
    <tr><td>Mouse</td><td>1</td><td>350</td></tr>
  </tbody>
  <tfoot>
    <tr><th colspan="2">Total</th><td>1,550</td></tr>
  </tfoot>
</table>
```

The `colspan` attribute stretches a cell across several columns, and `rowspan` stretches it down several rows. Use tables only for data that really is tabular, never to arrange the layout of a page; CSS does that job.

## Forms for collecting input

A form collects information from the visitor, such as an enquiry or a registration.

```

<form action="/enquiry" method="post">
  <label for="name">Full name</label>
  <input type="text" id="name" name="name" required>

  <label for="email">Email</label>
  <input type="email" id="email" name="email">

  <fieldset>
    <legend>Contact me by</legend>
    <input type="radio" id="by-phone" name="contact" value="phone">
    <label for="by-phone">Phone</label>
    <input type="radio" id="by-email" name="contact" value="email">
    <label for="by-email">Email</label>
  </fieldset>

  <label for="city">City</label>
  <select id="city" name="city">
    <option>Delhi</option>
    <option>Noida</option>
  </select>

  <label for="msg">Message</label>
  <textarea id="msg" name="message" rows="4"></textarea>

  <button type="submit">Send</button>
</form>

```

- `action` is the address that receives the data and `method` is how it is sent. With `get` (the default) the data is added to the URL, which suits searches. With `post` it travels in the body of the request, which suits passwords and personal details.
- Every control whose value must be sent needs a `name`, because that is the key sent to the server with its value. A control without a name is not sent.
- A `<label>` whose `for` matches an input's `id` belongs to that input: clicking the label focuses the input, and screen readers read it out. A `placeholder` is not a replacement for a label.
- Useful `type` values include `text`, `email`, `password`, `number`, `tel`, `date`, `checkbox`, `radio` and `file`.
- Radio buttons that share the same `name` form a group in which only one can be selected. Checkboxes allow several choices.
- `<fieldset>` groups related controls and `<legend>` gives the group a caption.
- Attributes such as `required`, `min` and `maxlength` let the browser check input before sending it. The server must still check the data again, because browser checks are easy to bypass.

## KEY POINTS

- Use tables only for tabular data: tr makes a row, th a header cell and td a data cell.
- colspan joins cells across columns and rowspan joins cells down rows.
- Link every input to a label by matching the label's for attribute to the input's id.
- method="get" puts form data in the URL; method="post" sends it in the request body, which suits passwords.
- Radio buttons with the same name allow one choice; checkboxes allow several.

## TRY THESE EXERCISES

1. Make a marksheet table for five students with a caption, a thead row of Roll No, Name, Hindi, English, Maths and Total headings, a tbody with their marks, and a tfoot row in which the words Class average span the first two columns using colspan.
2. Create your weekly class timetable as a table with the periods down the first column and the days across the top, use rowspan to show a two-period practical as one tall cell and colspan to show the lunch break as one cell across all the days.
3. Build an admission enquiry form with a label for every control: text inputs for name and email, a tel input for phone, a date input for date of birth, radio buttons for morning or evening batch inside a fieldset with a legend, checkboxes for three subjects, a select for the city, a textarea and a submit button; make the name required, try submitting it empty, then leave out the action attribute so that the form sends its data back to the same page, set method to get, submit it filled in and read the name=value pairs added to the address bar.

## CHAPTER 4

# Semantic HTML5 elements

## What semantic means

A **semantic** element tells the browser what its content is, not just how it should look. Before HTML5, page layouts were built almost entirely from `<div>` elements with class names such as `class="menu"`. A `<div>` has no meaning of its own, so browsers and screen readers could not tell the menu from the main story. HTML5 added elements that name the common parts of a page.

| Element                     | Use it for                                                 |
|-----------------------------|------------------------------------------------------------|
| <code>&lt;header&gt;</code> | Introductory content: logo, site name, often the main menu |
| <code>&lt;nav&gt;</code>    | A block of major navigation links                          |

| Element                                                         | Use it for                                                                                          |
|-----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>&lt;main&gt;</code>                                       | The main content of the page, unique to that page                                                   |
| <code>&lt;article&gt;</code>                                    | Self-contained content that makes sense on its own, such as a blog post, news story or product card |
| <code>&lt;section&gt;</code>                                    | A group of related content, usually with its own heading                                            |
| <code>&lt;aside&gt;</code>                                      | Content related to the page but separate from it, such as a sidebar or a tip box                    |
| <code>&lt;footer&gt;</code>                                     | Closing content: copyright, contact details, related links                                          |
| <code>&lt;figure&gt;</code> and <code>&lt;figcaption&gt;</code> | An image, chart or code sample with its caption                                                     |

## A semantic page layout

### HTML

```

<body>
  <header>
    <p>Sharma Electronics</p>
    <nav>
      <a href="index.html">Home</a>
      <a href="products.html">Products</a>
      <a href="contact.html">Contact</a>
    </nav>
  </header>
  <main>
    <h1>LED Televisions</h1>
    <article>
      <h2>32-inch Smart TV</h2>
      <p>Built-in Wi-Fi and two HDMI ports.</p>
    </article>
    <aside>
      <h2>Need help choosing?</h2>
      <p>Visit our showroom for advice.</p>
    </aside>
  </main>
  <footer>
    <p>Sharma Electronics, Lajpat Nagar, New Delhi</p>
  </footer>
</body>

```

## Why it matters

- **Accessibility.** Screen readers announce these elements as landmarks, so a blind or low-vision user can jump straight to the navigation or the main content instead of listening to the whole page.
- **Search engines** can better tell which part of the page is the real content.

- **Readable code.** `<nav>` is clearer than `<div class="nav">` for anyone who edits the page later.

## Rules of thumb

1. Use only one visible `<main>` per page, and do not put it inside `<header>`, `<footer>`, `<nav>`, `<article>` or `<aside>`.
2. `<header>` and `<footer>` are not only for the top and bottom of the page. An `<article>` can have its own header and footer.
3. Choose `<article>` when the content could be shared or reused on its own; choose `<section>` to group related content inside a page.
4. Not every group of links needs `<nav>`; keep it for major navigation blocks.
5. When no semantic element fits, `<div>` (a block-level box) and `<span>` (an inline piece of text) are still the right choice, usually as hooks for styling.

Ask yourself: if the styles failed to load, would the HTML alone still make the structure of the page clear? If yes, your markup is semantic.

### KEY POINTS

- Semantic elements describe what content is, which helps screen readers, search engines and other developers.
- header, nav, main, article, section, aside and footer name the common parts of a page.
- Use one visible main per page for the content unique to that page.
- article is for self-contained content; section groups related content under a heading.
- div and span carry no meaning and are used when no semantic element fits.

### TRY THESE EXERCISES

1. Write a page for an imaginary electronics shop using header, nav with three links, main, two article elements for two products, an aside with a buying tip and a footer with the address, without using a single div.
2. Write a short page built only from div elements with classes such as menu, content, sidebar and bottom, then replace each div with the matching semantic element and check that there is exactly one main and that it is not inside header, footer, nav, article or aside.
3. Give one article its own header with the product name and a footer with the date it was added, place a figure with an image and a figcaption inside it, add a section with its own h2 that groups both products, and write a comment beside each article, section and aside explaining why you chose that element.

# CSS basics: selectors, colours and fonts

## What CSS does

**CSS (Cascading Style Sheets)** controls how HTML looks: colours, fonts, spacing and layout. Keeping style in CSS and content in HTML means one stylesheet can change the look of a whole website. CSS can be added in three ways:

1. **External stylesheet**, the best choice for real websites: `<link rel="stylesheet" href="style.css">` in the `<head>`.
2. **Internal stylesheet**: CSS inside a `<style>` element in the `<head>`.
3. **Inline style**: a `style` attribute on one element. Avoid it except for quick tests.

A CSS **rule** has a selector and a block of declarations:

CSS

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

`h1` is the selector, `color` is a property and `navy` its value. Each declaration ends with a semicolon.

## Selectors

| Selector     | Example                           | Selects                                       |
|--------------|-----------------------------------|-----------------------------------------------|
| Element      | <code>p</code>                    | every paragraph                               |
| Class        | <code>.price</code>               | every element with <code>class="price"</code> |
| ID           | <code>#cart</code>                | the one element with <code>id="cart"</code>   |
| Descendant   | <code>nav a</code>                | links anywhere inside a nav                   |
| Child        | <code>ul &gt; li</code>           | list items directly inside a ul               |
| Group        | <code>h1, h2</code>               | all h1 and h2 elements                        |
| Pseudo-class | <code>a:hover, input:focus</code> | an element in a particular state              |

An element can have several classes, as in `class="price sale"`. An ID must be unique on the page, while a class can be reused on many elements.

## The cascade and specificity

When two rules set the same property on the same element, the browser decides which one wins:

1. A declaration marked **!important** beats normal declarations.
2. Otherwise the more **specific** selector wins: an inline style beats an ID, an ID beats a class or pseudo-class, and a class beats an element selector.
3. If specificity is equal, the rule written **later** wins.

Some properties, such as **color** and **font-family**, are **inherited**: set them on **body** and the text inside picks them up.

## Colours

### CSS

```
.brand { color: #e65100; }           /* hex: red, green, blue */
.box   { background-color: rgb(255, 243, 224); }
.shade  { background-color: rgba(0, 0, 0, 0.5); } /* 50% transparent black */
.badge  { color: hsl(120, 60%, 30%); }          /* hue, saturation, lightness */
.error  { color: crimson; }                  /* named colour */
```

Hex colours use two hexadecimal digits (00 to ff) for each of red, green and blue, so **#ffffff** is white and **#000000** is black. Always keep enough contrast between text and its background.

## Fonts and text

### CSS

```
body {
  font-family: "Noto Sans", Arial, sans-serif;
  font-size: 1rem;
  line-height: 1.6;
}
```

List several fonts and end with a generic family such as **serif**, **sans-serif** or **monospace**; the browser uses the first one available. **px** is a fixed unit, while **rem** is relative to the root font size (16px by default in most browsers), so it respects the reader's own text-size setting. Other useful properties are **font-weight** (400 is normal, 700 is bold), **text-align** and **text-decoration**.

## KEY POINTS

- A CSS rule is a selector followed by declarations of the form property: value;
- Use an external stylesheet linked from the head for real websites.
- Class selectors start with a dot and can be reused; ID selectors start with # and must be unique.
- When rules conflict, !important wins, then higher specificity, then the rule written later.
- Colours can be named, hex, rgb() or hsl(); font-family lists fallbacks ending in a generic family.

## TRY THESE EXERCISES

1. Create style.css, link it from the head of index.html with a link element, set the body font-family to "Noto Sans", Arial, sans-serif with a font-size of 1rem and a line-height of 1.6, then colour the h1 with a named colour, a hex code and an rgb() value in turn.
2. Give two paragraphs the class price and one heading the id offer, style them with .price and #offer selectors, add a nav a rule and an a:hover rule that changes the link colour, and style h1 and h2 together with one grouped selector.
3. Write two rules that give the same paragraph different colours, one using the element selector p and one using a class selector, predict which colour wins and check in the browser; then swap the order of the rules, and finally add !important to the element rule and note the result each time.

## CHAPTER 6

# The box model and Flexbox

---

## Every element is a box

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

1. **Content:** the text or image itself.
2. **Padding:** space between the content and the border, filled by the background colour.
3. **Border:** a line around the padding.
4. **Margin:** transparent space outside the border that pushes other elements away.

```
.card {  
  width: 300px;  
  padding: 20px;  
  border: 5px solid #ccc;  
  margin: 16px;  
}
```

By default (`box-sizing: content-box`), `width` sets only the content area. The card above therefore takes  $300 + 20 + 20 + 5 + 5 = \mathbf{350px}$  of width, plus its margins. Many developers switch every element to `box-sizing: border-box`, where `width` includes the padding and border, so the same card is exactly 300px wide.

## Shorthand values

`padding` and `margin` accept one to four values, which go clockwise from the top:

- `margin: 10px;` sets all four sides.
- `margin: 10px 20px;` sets top and bottom to 10px, left and right to 20px.
- `margin: 5px 10px 15px 20px;` sets top, right, bottom and left.
- `margin: 0 auto;` centres a block element that has a width.

In normal page flow, the vertical margins of two block elements next to each other **collapse**: a 20px bottom margin above a 30px top margin gives a 30px gap, not 50px.

## Block and inline

Block elements such as `<p>`, `<div>` and `<h1>` start on a new line and stretch to the full available width. Inline elements such as `<a>`, `<span>` and `<strong>` sit inside a line of text, and `width` and `height` have no effect on them. The `display` property changes this: `block`, `inline`, `inline-block` or `none`, which hides the element completely.

## Flexbox: layout in one direction

Flexbox arranges items in a row or a column. Set `display: flex` on the **container**, and its direct children become flex items.

```
.menu {  
  display: flex;  
  flex-direction: row;          /* or column */  
  justify-content: space-between;  
  align-items: center;  
  flex-wrap: wrap;  
  gap: 12px;  
}  
.menu .search { flex: 1; }      /* take the leftover space */
```

- `flex-direction` sets the **main axis**: `row` (the default) or `column`.
- `justify-content` places items along the main axis: `flex-start`, `center`, `flex-end`, `space-between`, `space-around` or `space-evenly`.
- `align-items` places items along the **cross axis**. By default items stretch to fill the cross axis, so items in a row become equal in height; `center` centres them instead.
- `flex-wrap: wrap` lets items move to a new line instead of squeezing together.
- `gap` adds space between items without using margins.
- `flex: 1` on an item lets it grow to fill the free space.

To centre something inside a box, give the box `display: flex; justify-content: center; align-items: center;`.

#### KEY POINTS

- Every element is a box of content, padding, border and margin.
- With the default content-box, width covers only the content; border-box includes padding and border.
- Margin and padding shorthand runs clockwise: top, right, bottom, left.
- `display: flex` on a container lines up its children along one axis.
- `justify-content` works on the main axis and `align-items` on the cross axis.

### TRY THESE EXERCISES

1. Style a .card with width 300px, padding 20px, a 5px solid border and margin 16px, check in the browser developer tools (F12) that it takes 350px of width, then add box-sizing: border-box and confirm that it becomes exactly 300px wide.
2. Give four boxes margins written with one, two and four values and label each box with the values you used, centre a box that has a fixed width with margin: 0 auto, and place a paragraph with a 20px bottom margin above one with a 30px top margin to see that the gap is 30px, not 50px.
3. Build a navigation bar with display: flex containing a logo, four links and a search box, use justify-content: space-between, align-items: center and gap: 12px, give the search box flex: 1, add flex-wrap: wrap and narrow the window to watch the items wrap onto a new line.

## CHAPTER 7

# CSS Grid and responsive design

## Grid: layout in two directions

Flexbox handles one direction at a time. **CSS Grid** controls rows and columns together, which suits page layouts, product listings and photo galleries.

### CSS

```
.products {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  gap: 16px;  
}
```

- `grid-template-columns` defines the columns. `repeat(3, 1fr)` creates three equal columns.
- The **fr** unit means "one fraction of the free space". `2fr 1fr` makes the first column twice as wide as the second.
- `gap` sets the space between rows and columns.
- Items fill the cells in order and start a new row when one is full.

An item can cover more than one cell:

## CSS

```
.featured { grid-column: span 2; } /* two columns wide */
.sidebar {
  grid-column: 3 / 4;           /* from line 3 to line 4 */
  grid-row: 1 / 3;             /* two rows tall */
}
```

Grid lines are numbered from 1, so a three-column grid has lines 1 to 4.

This pattern fits as many columns as there is room for, with no media queries at all:

## CSS

```
.gallery {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 16px;
}
```

## Responsive design

A responsive page adapts to every screen, from a small phone to a wide desktop monitor.

Three things make it work:

1. **The viewport meta tag** in the head: `<meta name="viewport" content="width=device-width, initial-scale=1">`. Without it, phones lay the page out at desktop width and shrink it.
2. **Flexible sizes**: percentages, `fr`, `rem` and `max-width` instead of fixed pixel widths. The rule `img { max-width: 100%; height: auto; }` stops images from overflowing a narrow screen.
3. **Media queries**, which apply CSS only when a condition is true.

## Media queries and mobile-first

### CSS

```
/* Base styles: phones, one column */
.products { display: grid; grid-template-columns: 1fr; gap: 16px; }

/* Tablets and wider */
@media (min-width: 600px) {
  .products { grid-template-columns: repeat(2, 1fr); }
}

/* Laptops and desktops */
@media (min-width: 1024px) {
  .products { grid-template-columns: repeat(4, 1fr); }
}
```

This is the **mobile-first** approach: write the simple one-column layout for small screens first, then use `min-width` queries to add columns as the screen gets wider. A `max-width` query works the other way round, applying styles only up to a given width.

The widths 600px and 1024px are **breakpoints**. There is no official list of breakpoints; add one wherever your own design starts to look cramped or stretched.

## Test on a phone size

Open the browser's developer tools (F12 in Chrome or Edge on Windows) and press Ctrl+Shift+M to switch on device emulation, which previews the page at phone widths such as 390px. Check that nothing scrolls sideways, text is readable without zooming, and buttons are big enough to tap with a thumb.

### KEY POINTS

- Grid lays out rows and columns together; Flexbox handles one direction at a time.
- `repeat(3, 1fr)` makes three equal columns; `fr` is a share of the free space.
- `grid-column: span 2` makes an item two columns wide.
- Responsive pages need the viewport meta tag, flexible sizes and media queries.
- Mobile-first means base styles for phones, then `min-width` media queries for larger screens.

### TRY THESE EXERCISES

1. Lay out six product cards with `display: grid, grid-template-columns: repeat(3, 1fr)` and a 16px gap, make the first card span two columns with `grid-column: span 2`, then change the columns to `2fr 1fr 1fr` and see how the space is shared.
2. Build a gallery of nine photos with `grid-template-columns: repeat(auto-fit, minmax(200px, 1fr))`, add the rule `img { max-width: 100%; height: auto; }`, and drag the browser window from wide to narrow, counting how many columns fit at each width.
3. Write a mobile-first product page that shows one column by default, two columns from a `min-width` of 600px and four from 1024px, check that the viewport meta tag is in the head, then open device emulation with F12 and Ctrl+Shift+M and test it at 390px for sideways scrolling, readable text and buttons large enough to tap.

# Practice questions

---

Choose the best answer for each question. The answers, with a short explanation, are in the next chapter.

**1. What does HTML stand for?**

- a) HyperText Markup Language
- b) High Transfer Markup Language
- c) HyperText Machine Language
- d) Home Tool Markup Language

**2. Which line should come first in an HTML5 document?**

- a) `<!DOCTYPE html>`
- b) `<html5>`
- c) `<head>`
- d) `<body>`

**3. Which of these belongs inside the head element rather than the body?**

- a) The main heading of the page
- b) A paragraph of welcome text
- c) The page title shown on the browser tab
- d) A photo of the shop

**4. Which of these is a void element that has no closing tag?**

- a) `<p>`
- b) `<ul>`
- c) `<img>`
- d) `<title>`

**5. In `<a href="contact.html">Contact</a>`, what is `href="contact.html"`?**

- a) An element
- b) A comment
- c) A doctype
- d) An attribute

**6. Which heading element is normally used once, for the main title of a page?**

- a) `<h1>`
- b) `<h6>`
- c) `<head>`
- d) `<title>`

**7. Which element creates a numbered list?**

- a) `<ul>`
- b) `<li>`
- c) `<ol>`
- d) `<nl>`

**8. What happens when a visitor clicks a link written as `<a href="#timings">`?**

- a) The page jumps to the element with `id="timings"` on the same page
- b) The browser opens a file named `timings.html`
- c) The browser starts an email to an address called `timings`
- d) The browser downloads a file called `timings`

**9. Why should every informative image have an alt attribute?**

- a) It makes the image download faster
- b) It sets the width of the image
- c) It adds a visible caption below the image
- d) It gives a text alternative for screen readers and for when the image fails to load

**10. Which image format is best for a logo that must stay sharp at any size?**

- a) JPEG
- b) SVG
- c) GIF
- d) BMP

**11. In an HTML table, which element creates a header cell?**

- a) <th>
- b) <td>
- c) <tr>
- d) <thead>

**12. Which attribute makes a table cell stretch across three columns?**

- a) rowspan="3"
- b) colspan="3"
- c) cols="3"
- d) width="3"

**13. Which method should a login form use when it sends a password?**

- a) method="get"
- b) method="send"
- c) method="post"
- d) method="secure"

**14. How do you connect a label to the input <input id="phone">?**

- a) <label name="phone">
- b) <label id="phone">
- c) <label input="phone">
- d) <label for="phone">

**15. A form asks the user to choose exactly one payment mode: Cash, UPI or Card. Which controls should it use?**

- a) Three checkboxes with different names
- b) Three radio buttons, each with a different name
- c) Three radio buttons that share the same name
- d) Three text boxes with placeholders

**16. Which element should contain the main content that is unique to a page?**

- a) <section>
- b) <aside>
- c) <header>
- d) <main>

**17. Which element is meant for a block of major navigation links?**

- a) <navigation>
- b) <links>
- c) <aside>
- d) <nav>

**18. A blog post that would make sense if shared on its own should be wrapped in which element?**

- a) <section>
- b) <article>
- c) <aside>
- d) <span>

**19. How do semantic elements help blind users?**

- a) Screen readers announce them as landmarks, so users can jump to parts such as the navigation or main content
- b) They make the text larger on screen
- c) They play audio descriptions of images automatically
- d) They print the page in Braille without any extra software

- 20. Which element has no meaning of its own and is used for a small inline piece of text, usually as a hook for styling?**
- a) <div>
  - b) <strong>
  - c) <em>
  - d) <span>
- 21. Which CSS selector selects every element with class="price"?**
- a) .price
  - b) #price
  - c) price
  - d) @price
- 22. What is the recommended way to add CSS to a website with many pages?**
- a) A style attribute on every element
  - b) An external stylesheet linked with <link rel="stylesheet" href="style.css">
  - c) A separate <style> block pasted into each page's body
  - d) A <css> element inside the head
- 23. An element is written <p id="notice" class="alert">. The stylesheet has #notice { color: red; } and, later, .alert { color: blue; }. What colour is the text?**
- a) Blue, because the rule written later always wins
  - b) Red, because an ID selector is more specific than a class selector
  - c) Purple, because the two colours are mixed
  - d) Black, because conflicting rules cancel each other
- 24. Which colour is #ffffff?**
- a) Black
  - b) Red
  - c) White
  - d) Blue
- 25. Why does font-family usually list several fonts, ending with a word such as sans-serif?**
- a) So the browser blends the fonts together
  - b) So the browser can fall back to the next font if the first is not available
  - c) So each paragraph uses a different font in turn
  - d) Because CSS requires exactly three fonts
- 26. In the CSS box model, which layer lies between the content and the border?**
- a) Margin
  - b) Outline
  - c) Padding
  - d) Gap
- 27. With the default box-sizing: content-box, an element has width: 200px, padding: 10px and border: 2px solid. How much horizontal space does it take, not counting margins?**
- a) 200px
  - b) 212px
  - c) 220px
  - d) 224px
- 28. What does margin: 10px 20px; set?**
- a) Top and left 10px, bottom and right 20px
  - b) Left and right 10px, top and bottom 20px
  - c) Top and bottom 10px, left and right 20px
  - d) Top 10px and bottom 20px only

- 29. In a flex container with flex-direction: row, which property spaces the items along the horizontal (main) axis?**
- a) justify-content
  - b) align-items
  - c) flex-wrap
  - d) vertical-align
- 30. Which declaration makes an element's width include its padding and border?**
- a) box-sizing: content-box;
  - b) display: block;
  - c) box-sizing: border-box;
  - d) overflow: hidden;
- 31. What does grid-template-columns: repeat(3, 1fr); create?**
- a) Three rows of equal height
  - b) One column three times wider than the others
  - c) Three columns that are 1 pixel wide each
  - d) Three columns of equal width
- 32. In CSS Grid, what does the fr unit stand for?**
- a) A fraction of the free space in the grid container
  - b) A fixed width of 16 pixels
  - c) The font size of the root element
  - d) A percentage of the screen height
- 33. Which tag makes phones display a page at their real screen width instead of shrinking a desktop-width layout?**
- a) <meta charset="utf-8">
  - b) <meta name="viewport" content="width=device-width, initial-scale=1">
  - c) <meta name="description" content="mobile">
  - d) <meta name="author" content="phone">
- 34. In a mobile-first stylesheet, which media query adds styles for screens 768px wide and wider?**
- a) @media (max-width: 768px)
  - b) @media (min-width: 768px)
  - c) @media (width: 768px)
  - d) @media (min-height: 768px)
- 35. Which CSS rule stops large images from overflowing a narrow phone screen?**
- a) img { width: 1200px; }
  - b) img { max-width: 100%; height: auto; }
  - c) img { position: fixed; }
  - d) img { float: right; }

## ANSWER KEY

# Answers

---

**1. (a) HyperText Markup Language.** HTML is HyperText Markup Language: hypertext is text with links, and markup means tags that label the content.

**2. (a) <!DOCTYPE html>.** The doctype declaration tells the browser the page is HTML5 and must come before the html element.

3. (c) **The page title shown on the browser tab.** The head holds information about the page, such as the title, character set and stylesheet links; visible content goes in the body.
4. (c) **<img>.** img has no content and no closing tag; p, ul and title all wrap content and need closing tags.
5. (d) **An attribute.** href is an attribute: extra information written inside the opening tag as name="value".
6. (a) **<h1>.** h1 is the most important heading level and is normally used once for the main visible title; title sets the browser tab text, not a heading on the page.
7. (c) **<ol>.** ol is an ordered list, numbered by default; ul is a bulleted list and li is an item inside either.
8. (a) **The page jumps to the element with id="timings" on the same page.** An href that starts with # points to the element on the current page whose id matches.
9. (d) **It gives a text alternative for screen readers and for when the image fails to load.** Alt text describes the image to people who cannot see it and is shown if the image cannot be loaded.
10. (b) **SVG.** SVG is a vector format, so it scales to any size without becoming blurry.
11. (a) **<th>.** th is a header cell; td is a data cell, tr is a row and thead groups the header rows.
12. (b) **colspan="3".** colspan sets how many columns a cell covers; rowspan does the same for rows.
13. (c) **method="post".** With post the data travels in the request body instead of being added to the URL, where it could be seen and saved in the browser history.
14. (d) **<label for="phone">.** The label's for attribute must match the input's id; clicking the label then focuses the input.
15. (c) **Three radio buttons that share the same name.** Radio buttons that share one name form a group in which only one option can be selected.
16. (d) **<main>.** main wraps the central content of the page; the header, navigation and footer that repeat on every page stay outside it.
17. (d) **<nav>.** nav is the HTML5 element for major navigation blocks; navigation and links are not HTML elements.
18. (b) **<article>.** article is for self-contained content that can stand alone; section groups related content inside a page.
19. (a) **Screen readers announce them as landmarks, so users can jump to parts such as the navigation or main content.** Screen readers expose semantic elements as landmarks for quick navigation around the page.
20. (d) **<span>.** span is a generic inline container; div is the generic block-level container, and strong and em add meaning.
21. (a) **.price.** A dot selects by class and # selects by id; a bare name such as price would select elements of that type.
22. (b) **An external stylesheet linked with <link rel="stylesheet" href="style.css">.** One external stylesheet can style every page, and changing it updates the whole site.
23. (b) **Red, because an ID selector is more specific than a class selector.** Specificity is compared before the order of rules, and an ID selector is more specific than a class selector.
24. (c) **White.** Each pair of hex digits is red, green and blue; ff is the maximum, so all three at full strength give white.

- 25. (b) So the browser can fall back to the next font if the first is not available.** The browser uses the first available font in the list, and the generic family at the end guarantees a fallback.
- 26. (c) Padding.** Padding is the space inside the border around the content; margin is outside the border.
- 27. (d) 224px.** Content-box adds padding and border on both sides:  $200 + 10 + 10 + 2 + 2 = 224\text{px}$ .
- 28. (c) Top and bottom 10px, left and right 20px.** With two values, the first applies to top and bottom and the second to left and right.
- 29. (a) justify-content.** justify-content works along the main axis, which is horizontal for a row; align-items works on the cross axis.
- 30. (c) box-sizing: border-box;.** With border-box, the width you set covers content, padding and border together; content-box, the default, covers only the content.
- 31. (d) Three columns of equal width.** repeat(3, 1fr) creates three columns, each taking one equal share of the free space.
- 32. (a) A fraction of the free space in the grid container.** fr shares out the free space; a 2fr column gets twice as much as a 1fr column.
- 33. (b) <meta name="viewport" content="width=device-width, initial-scale=1">.** Without the viewport tag, mobile browsers lay the page out at desktop width and scale it down.
- 34. (b) @media (min-width: 768px).** A min-width query applies when the viewport is at least that wide, which is how mobile-first CSS adds layouts for larger screens.
- 35. (b) img { max-width: 100%; height: auto; }.** max-width: 100% keeps an image no wider than its container, and height: auto keeps its proportions.

## WORD LIST

# Glossary

### Alt text

A short description of an image, written in the img element's alt attribute, that screen readers read aloud and browsers show if the image fails to load.

### Attribute

Extra information written in an opening tag as name="value", such as the href of a link or the src of an image.

### Box model

The way a browser draws every element as a rectangular box made of content, padding, border and margin, from the inside out.

### Cascade

The rules CSS uses to decide which declaration wins when several set the same property on an element: importance first, then specificity, then the order the rules are written in.

---

## Class selector

A CSS selector that starts with a dot, such as .price, and styles every element whose class attribute contains that name.

---

## CSS

Cascading Style Sheets, the language that controls how HTML looks, including colours, fonts, spacing and layout.

---

## CSS Grid

A CSS layout method, switched on with display: grid, that places items in rows and columns at the same time.

---

## Doctype

The declaration on the first line of an HTML file that tells the browser the page is HTML5, so that it displays it in standards mode.

---

## Element

One part of an HTML page, usually made of an opening tag, its content and a closing tag, such as a paragraph or a heading.

---

## Flexbox

A CSS layout method, switched on with display: flex on a container, that lines up its direct children along one direction, as a row or a column, letting them wrap onto new lines when flex-wrap is set.

---

## fr unit

A CSS Grid unit meaning one fraction of the free space, so 2fr 1fr makes the first column twice as wide as the second.

---

## HTML

HyperText Markup Language, the markup language that labels the structure of a web page, such as its headings, paragraphs, links and images.

---

## ID selector

A CSS selector that starts with #, such as #cart, and styles the one element whose id attribute has that name, which must be unique on the page.

---

## Label

A form element tied to an input by matching its for attribute to the input's id, so that clicking it focuses the input and screen readers announce it.

---

## Margin

The transparent space outside an element's border that pushes neighbouring elements away.

---

## Media query

A block of CSS written with @media whose rules apply only when a condition is true, such as the browser window (the viewport) being at least 600px wide.

---

## Mobile-first

A way of writing CSS in which the base styles suit phones and min-width media queries add layout for wider screens.

---

## Padding

The space inside an element's box between its content and its border, which is filled by the element's background colour.

---

## Pseudo-class

A selector part written after a colon that targets an element in a particular state, such as a:hover or input:focus.

---

## Responsive design

Designing a page so that its layout adapts to every screen size, using the viewport meta tag, flexible sizes and media queries.

---

### **Selector**

The part of a CSS rule that chooses which HTML elements the rule's declarations apply to.

---

### **Semantic element**

An HTML element whose name describes the meaning of its content, such as header, nav, main, article or footer.

---

### **Specificity**

The weight of a CSS selector, used to decide which of two conflicting declarations wins when neither is marked !important: an inline style beats an ID, an ID beats a class or pseudo-class, and a class beats an element selector.

---

### **Tag**

A name written inside angle brackets that marks where an element starts or ends, such as the opening and closing tags of a paragraph.

---

### **Void element**

An element such as br, img, input or meta that has no content and therefore no closing tag.

---



# 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](mailto:engineergurukul@yahoo.co.in)

**Website:** [engineergurukul.com](http://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/](http://engineergurukul.com/resources/ebooks/)

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