HTML & CSS cheat sheet · Building web pages

Basics · Text · Links · Semantics · Images · Help

Holger Sebastião · www.sebastiao.org
Version 1.1 · 2026-10-25

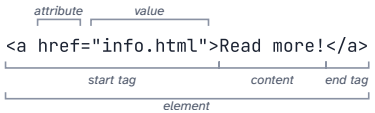
Page skeleton — index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content=
    "width=device-width, initial-scale=1">
  <title>My page</title>
  <link rel="stylesheet"
        href="css/style.css">
</head>
<body>
  <h1>Hello world</h1>
</body>
</html>
```

`<!DOCTYPE html>`	This is HTML5. Always the first line.
`<head>`	Info for the browser, invisible: character set, tab title, CSS.
`<body>`	Everything you see on the page.

The homepage is called `index.html`.

Anatomy of an element



The end tag has a `/`. Attributes go in the start tag: `name="value"`, value in quotation marks.

Empty elements have no content and no end tag: `<br>` `<hr>` `<img>` `<meta>` `<link>`

Nest correctly, close the inner one first: `<p><strong>…</strong></p>`, not `<p><strong>…</p></strong>`

Write tags in lower case and use indentation to show what sits inside what.

Special characters

`<`	<	`>`	>
`&`	&	` `	non-breaking space
`©`	©		

Write characters like é, ü or ç directly, thanks to `charset="UTF-8"`.

Text

```
<h1>Main heading</h1>
<h2>Subheading</h2>   <!-- up to h6 -->

<p>A paragraph. The text here simply
keeps flowing in the browser.</p>

<p>First line<br>
second line in the same paragraph</p>

<strong>important</strong>  <em>stressed</em>
<hr>   <!-- dividing line -->
```

`h1` … `h6`	headings. `h1` only once per page, don't skip a level.
`p`	paragraph
`br`	line break inside a paragraph
`strong`, `em`	important (bold), stressed (italic)

The browser ignores spaces and line breaks in the code. Don't use `<br>` for spacing, that's what CSS is for.

Comment: `<!-- not displayed -->`

Lists

```
<ul>
  <li>Milk</li>
  <li>Bread</li>
</ul>
```

```
<ol>
  <li>Alarm</li>
  <li>Shower</li>
</ol>
```

`ul`	bulleted list (unordered list)
`ol`	numbered list (ordered list)
`li`	a list item, only inside `ul` or `ol`

Nesting: the inner list goes **inside** the `<li>` of the outer one.

Table

```
<table>
  <tr><th>Name</th><th>Grade</th></tr>
  <tr><td>Ada</td><td>1</td></tr>
  <tr><td>Max</td><td>2</td></tr>
</table>
```

`tr` row (table row), `th` header cell, `td` data cell. Tables only for data, never for the page layout.

Links — a href

```
<a href="contact.html">Contact</a>
```

TARGET	HREF
page in the same folder	`contact.html`
page in a subfolder	`pages/team.html`
one level up	`../index.html`
another website	`https://visucode.de`
place on the page	`#contact`
email	`mailto:info@school.org`

```
<!-- open in a new tab -->
<a href="https://en.wikipedia.org"
   target="_blank">Wikipedia</a>

<!-- jump target: id without # -->
<h2 id="contact">Contact</h2>

<!-- image as a link -->
<a href="index.html"><img src="logo.png"
   alt="To the homepage"></a>
```

Relative path (`pages/team.html`): from the folder of the current file, also works on the server. An **absolute path** starts with `https://`.

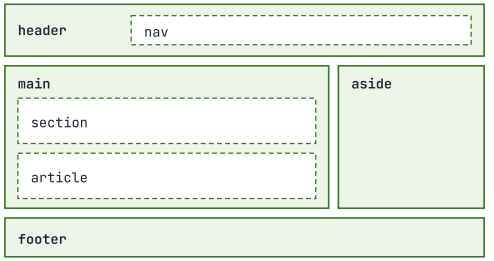
Never use `C:\Users\…` or `/Users/…` as a target: that only exists on your computer.

Navigation — nav

```
<header>
  <nav>
    <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="team.html">Team</a></li>
      <li><a href="info.html">Info</a></li>
    </ul>
  </nav>
</header>
```

A menu is a **list of links** in `<nav>`. The same menu appears on every page. CSS turns it into a bar (page 2).

Semantic HTML5 — page structure



`header`	top: logo, title, menu
`nav`	main navigation
`main`	main content, only once per page
`section`	section with its own heading
`article`	stands on its own, e.g. a post
`aside`	side info, e.g. a box at the edge
`footer`	bottom: legal notice, contact

Semantic means: the tag says **what** the content is, not how it looks. That helps screen readers and search engines. `div` and `span` only when no tag fits (e.g. just for CSS).

Images — img

```
<figure>
  <img src="images/dog.jpg"
       alt="Brown dog in the grass"
       width="300">
  <figcaption>Our school dog</figcaption>
</figure>
```

`alt` is required: it describes the image for blind people and when it doesn't load. `figure` + `figcaption` = image with a caption.

Typical mistakes

- End tag forgotten or nested wrongly
- File names with spaces, special characters or capitals: better `about-us.html`. The server treats `Photo.JPG` and `photo.jpg` as different.
- Wrong path: is the file really in that folder?
- Check: `validator.w3.org`, in the browser F12 or right-click → Inspect.

HTML & CSS cheat sheet · Building web pages

Including Selectors Styling Box Layout Help

Holger Sebastião · www.sebastiao.org
Version 1.1 · 2026-10-25

Including CSS — 3 ways

1 · EXTERNAL: THIS IS HOW WE DO IT

```
<!-- in the head of every HTML page -->
<link rel="stylesheet" href="css/style.css">
```

```
/* file css/style.css: only CSS, no HTML */
body {
  font-family: Arial, sans-serif;
}
```

One file for all pages: change it, and every page changes.

2 · INTERNAL IN THE HEAD

```
<head>
  <style>
    h1 { color: navy; }
  </style>
</head>
```

Only applies to this one page.

3 · DIRECTLY IN THE TAG (INLINE)

```
<p style="color: red;">Warning</p>
```

Only applies to this one element. Only for quick tests.

> **Who wins?** Inline beats internal and external. Otherwise whatever comes later wins.

Anatomy of a rule



The **selector** chooses **what** is styled. Inside `{ }` are declarations, each ending with `;`. Comment: `/* … */`

Inheritance in CSS

```
body {
  font-family: Arial, sans-serif;
  color: #1d2433;
}
```

The font and text colour of `body` apply to everything inside it. Spacing, borders and backgrounds are **not** inherited.

Selectors

CSS	MATCHES	IN THE HTML
`p`	all paragraphs	`<p>`
`.notice`	class, any number of times	`class="notice"`
`#top`	id, only once per page	`id="top"`
`nav a`	links **inside** nav	
`h1, h2`	h1 and h2	
`a:hover`	link under the mouse	
`*`	all elements	

```
<p class="notice">No lessons tomorrow</p>
```

```
.notice {
  background-color: #fff3c4;
}
```

If several rules match: id beats class, class beats element. If they are equal, the later rule wins.

Colours & text

```
h1 {
  color: #1f5fae;       /* text colour */
  background-color: white;
  font-family: Arial, sans-serif;
  font-size: 2rem;
  font-weight: bold;
  text-align: center;   /* or left */
  line-height: 1.5;     /* line spacing */
}
a {
  text-decoration: none; /* no underline */
}
```

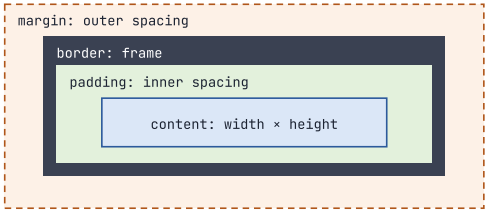
colour	`red` · `#ff0000` · `rgb(255, 0, 0)`
transparent	`rgba(0, 0, 0, 0.5)`
units	`px` fixed · `%` of the parent · `rem` of the base text (16px)

Styling links

```
a:link    { color: #1f5fae; } /* new */
a:visited { color: #6b4aa6; } /* visited */
a:hover   { color: #b0561a; } /* mouse */
a:active  { color: #b3261e; } /* click */
```

Keep this order, otherwise they override each other. Memory aid: **LoVe HAte**.

Box model




```
.card {
  width: 300px;
  padding: 1rem;
  border: 2px solid #3a4152;
  border-radius: 8px;  /* round corners */
  margin: 20px auto;   /* auto = centred */
  box-sizing: border-box;
}
```

`box-sizing: border-box`: `width` includes padding and border. Shorthand `margin: 10px 20px;` = top/bottom 10, left/right 20. Four values: top, right, bottom, left (clockwise).

Navigation as a bar — Flexbox

```
nav ul {
  list-style: none;   /* no bullets */
  margin: 0;
  padding: 0;
  display: flex;      /* side by side */
  gap: 1rem;
}
nav a {
  color: white;
  text-decoration: none;
  padding: 0.5rem 1rem;
}
nav a:hover {
  background-color: #1f5fae;
}
```

`display: flex` puts the children side by side. Also: `justify-content: center` (horizontal), `align-items: center` (vertical), `flex-direction: column` (stacked).

Phone & tablet — @media

```
/* only applies up to 600px wide */
@media (max-width: 600px) {
  nav ul {
    flex-direction: column;
  }
}
```

Needs the viewport line from the page skeleton in the HTML head.

Typical mistakes

- Forgot `;` or `}`: nothing works from there on.
- In CSS `.notice` with a dot, in HTML `class="notice"` without.
- Wrong path in `<link>`: is `style.css` in the `css` folder?
- Old version in the browser: reload with Ctrl+F5 or Cmd+Shift+R.
- See which rule applies: F12 or right-click → Inspect.

Folders of a website

```
website/
├── index.html
├── team.html
├── contact.html
├── css/
│   └── style.css
└── images/
    └── logo.png
```

From `index.html`: `css/style.css`, `images/logo.png`. All lower case, no spaces.

Page layout — Flexbox

```
<div class="content">
  <main>…</main>
  <aside>…</aside>
</div>
```

```
.content {
  display: flex;
  gap: 2rem;
}
main  { flex: 3; }   /* 3/4 of the width */
aside { flex: 1; }   /* 1/4 of the width */
```

HTML & CSS cheat sheet · Building web pages

Structure · Fields · Attributes · Choices · CSS · Help

Holger Sebastião · www.sebastiao.org
Version 1.1 · 2026-10-25

Structure of a form `form`

```
<form action="send.php" method="post">
  <label for="firstname">First name</label>
  <input type="text" id="firstname"
         name="firstname" required>

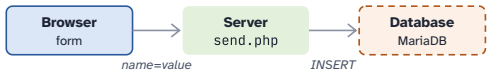
  <button type="submit">Send</button>
</form>
```

`action`	where the data goes (file on the server)
`method`	`get`: data appears in the address bar · `post`: invisible, for passwords and lots of data
`name`	the value arrives at the server under this name
`id`	connects the field with its `label`

label for = id: clicking the text puts the cursor in the field. Screen readers read the text aloud.

Without `name` the field is **not** sent.

What happens when sending?



HTML only shows the form. Only a program on the server can store the data, e.g. in PHP with SQL.

Buttons `button`

```
<button type="submit">Send</button>
<button type="reset">Clear</button>
<button type="button">Just a button</button>
```

`submit` sends, `reset` clears all fields, `button` does nothing (for JavaScript). Always give a `type`.

Testing without a server

```
<form action="thanks.html" method="get">
```

Create an empty page `thanks.html`. After sending, the address bar shows:
`thanks.html?firstname=Ali`
That way you see what the form sends.

Input fields `input`

TYPE	FOR
`text`	short text, one line
`email`	email, checks for @
`password`	shows only dots
`number`	number, with `min` `max` `step`
`date`	date with a calendar
`tel`	phone number
`url`	web address, checks for https://
`color` · `range`	colour picker · slider
`file`	upload a file
`checkbox`	tick box, several possible
`radio`	exactly one from a group
`submit`	send button (better `<button>`)

```
<label for="age">Age</label>
<input type="number" id="age"
       name="age" min="14" max="99">

<label for="mail">Email</label>
<input type="email" id="mail" name="mail"
       placeholder="name@school.org">
```

Important attributes

`required`	required field, otherwise no sending
`placeholder`	grey sample text in the empty field
`value`	value that is sent (or a default)
`maxlength="20"`	at most 20 characters
`min` · `max`	smallest and largest number or date
`checked`	tick or dot already set
`disabled`	greyed out, not sent
`readonly`	read only, is sent
`autofocus`	cursor is in the field on load

placeholder doesn't replace a label: the text disappears when you type.

Choosing `radio` · `checkbox`

```
<fieldset>
  <legend>Year</legend>
  <input type="radio" id="y11"
         name="year" value="11" checked>
  <label for="y11">11</label>
  <input type="radio" id="y12"
         name="year" value="12">
  <label for="y12">12</label>
</fieldset>

<input type="checkbox" id="terms"
       name="terms" required>
<label for="terms">I agree</label>
```

radio: same `name` = one group, only one can be chosen. Each needs its own `value`.
`fieldset` frames fields that belong together, `legend` is the heading.

Drop-down and long text

```
<label for="subject">Subject</label>
<select id="subject" name="subject">
  <option value="">Please choose</option>
  <option value="cs">Computing</option>
  <option value="ma">Maths</option>
</select>

<label for="text">Message</label>
<textarea id="text" name="text"
          rows="4"></textarea>
```

`textarea` has an end tag and no `value`: default text goes between the tags.

What reaches the server?

FIELD	WHAT IS SENT
text field	`firstname=Ali`
radio	`year=12` (the chosen `value`)
checkbox on	`terms=on` (without `value`)
checkbox off	nothing at all
select	`subject=cs` (`value`, not the text)

With `get`, the pairs appear in the address bar joined by `&`: `?firstname=Ali&year=12`

Styling forms `CSS`

```
label {
  display: block;      /* own line */
  margin-top: 0.8rem;
}
input, select, textarea {
  width: 100%;
  padding: 0.4rem;
  border: 1px solid #8a92a3;
  border-radius: 4px;
  box-sizing: border-box;
}
input:focus, textarea:focus {
  outline: 2px solid #1f5fae; /* active */
}
input:invalid {
  border-color: #b3261e; /* error */
}
button {
  margin-top: 1rem;
  padding: 0.5rem 1.5rem;
}
```

For radio and checkbox keep `width: auto` and put the `label` next to it with `display: inline`.

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

label without for

```
<label>
  <input type="checkbox" name="news">
  Subscribe to the newsletter
</label>
```

If the field is **inside** the `label`, it needs no `for` and no `id`. Handy for checkbox and radio.

Typical mistakes

- `name` missing: the value doesn't arrive.
- `for` and `id` spelled differently.
- Radio buttons with different `name`s: all can be chosen at once.
- `<input>` has no end tag, `<textarea>` and `<select>` do.
- Button outside the `<form>`: sends nothing.
- `required` doesn't replace a check on the server.