# 2 - HTML & CSS

**React education, 2024.**

MITHRIL

## Overview

- About HTML
- Element categories
- Common tags & attributes
- Structuring content
- About CSS
- Cascade and specificity
- Responsive design
- Box model
- Hands-on

MITHRIL

## About HTML

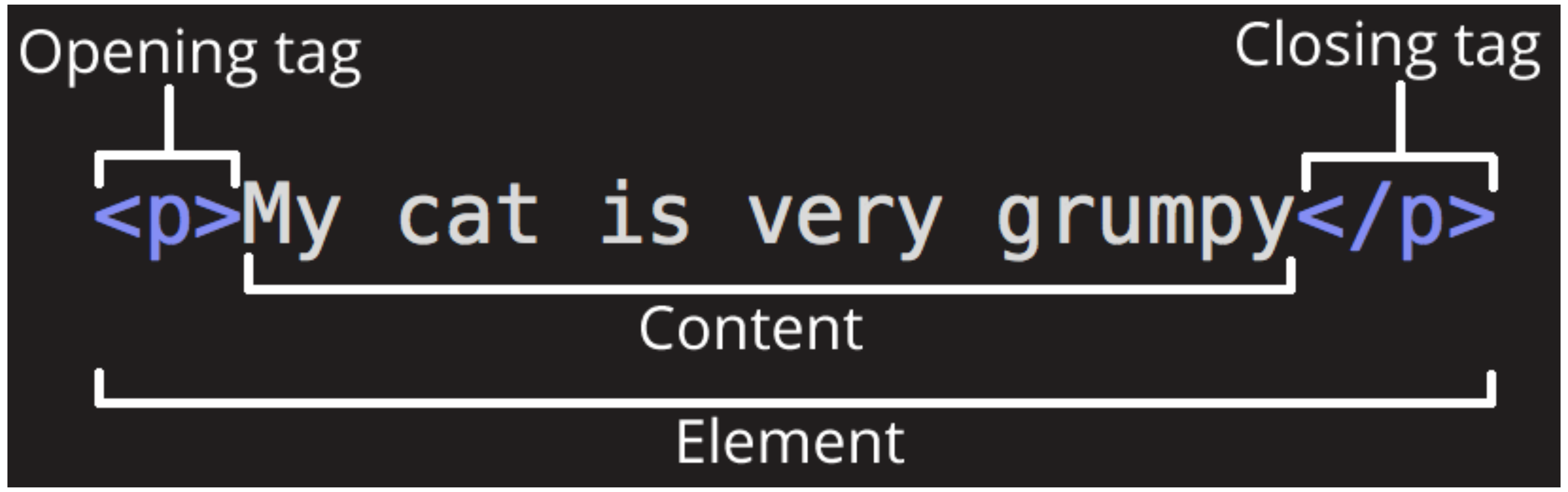
## HTML

- *Hypertext Markup Language* (not programming language)
- Created in 1989/1991 by Tim Berners Lee
- Markup language used to tell your browser how to structure web pages
- Current version:
  - HTML5 - better semantics, performance, device access etc.
  - HTML elements
  - HTML & CSS style guide

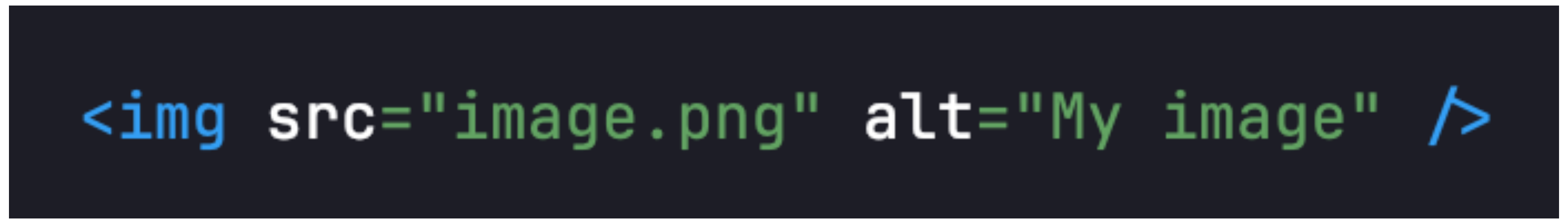
MITHRIL

# Anatomy of HTML element

- Most of HTML elements consists of opening tag, content and closing tag.
- Opening tag consists of name of the element.
- Closing tag is the same as opening, except it includes a forward slash before the element name.
- Content can be text or another HTML element.
- Elements can also be empty (without the closing tag) - *void elements*



```
<p>My cat is very grumpy</p>
```

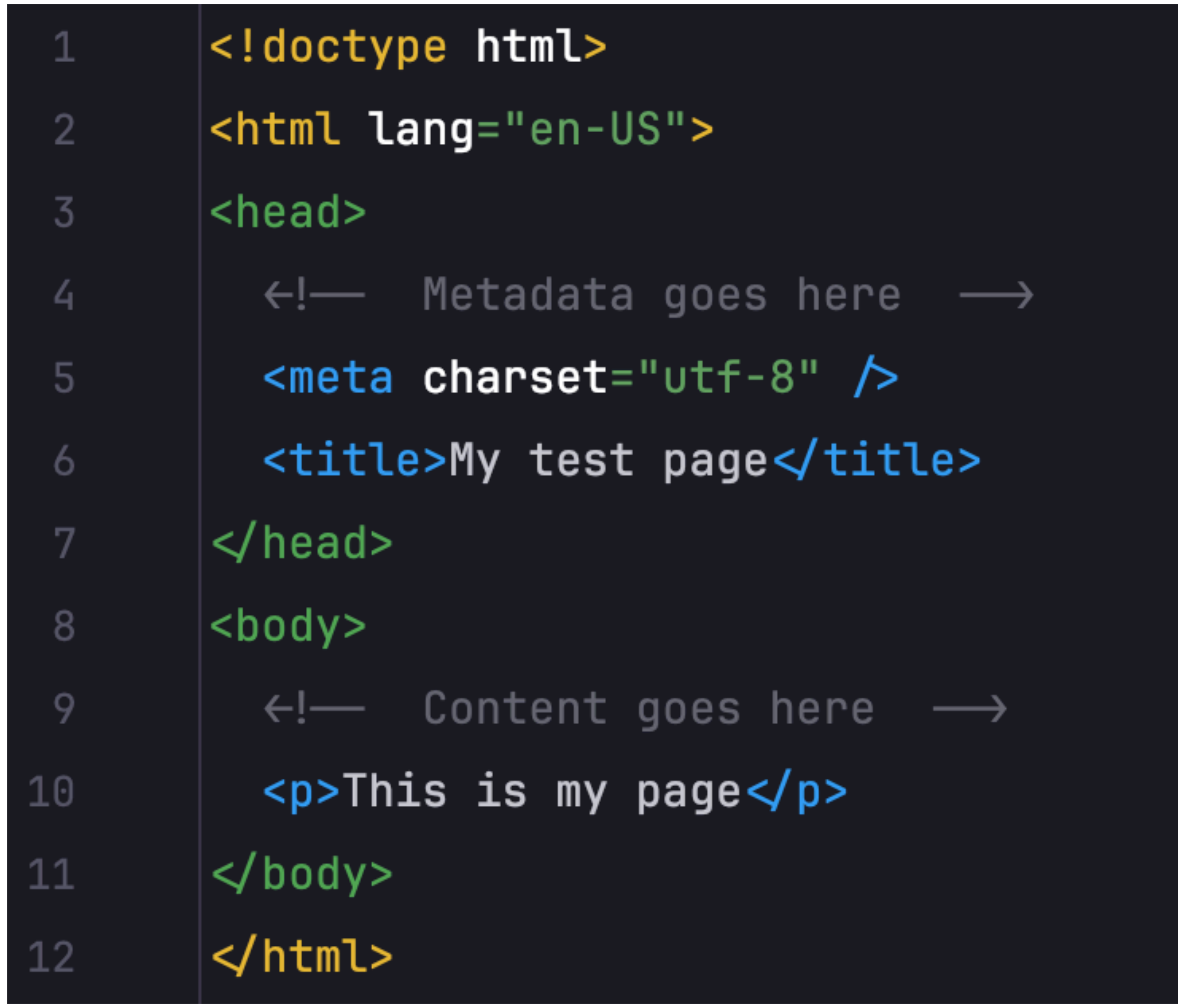


```
<img src="image.png" alt="My image" />
```

MITHRIL

# Boilerplate

- Anatomy of an HTML document
- Begins with declaration of artifact
- <html> - wraps all the content of the page
- <head> - container for metadata
- <body> - container for content
- <!— HTML comment —>

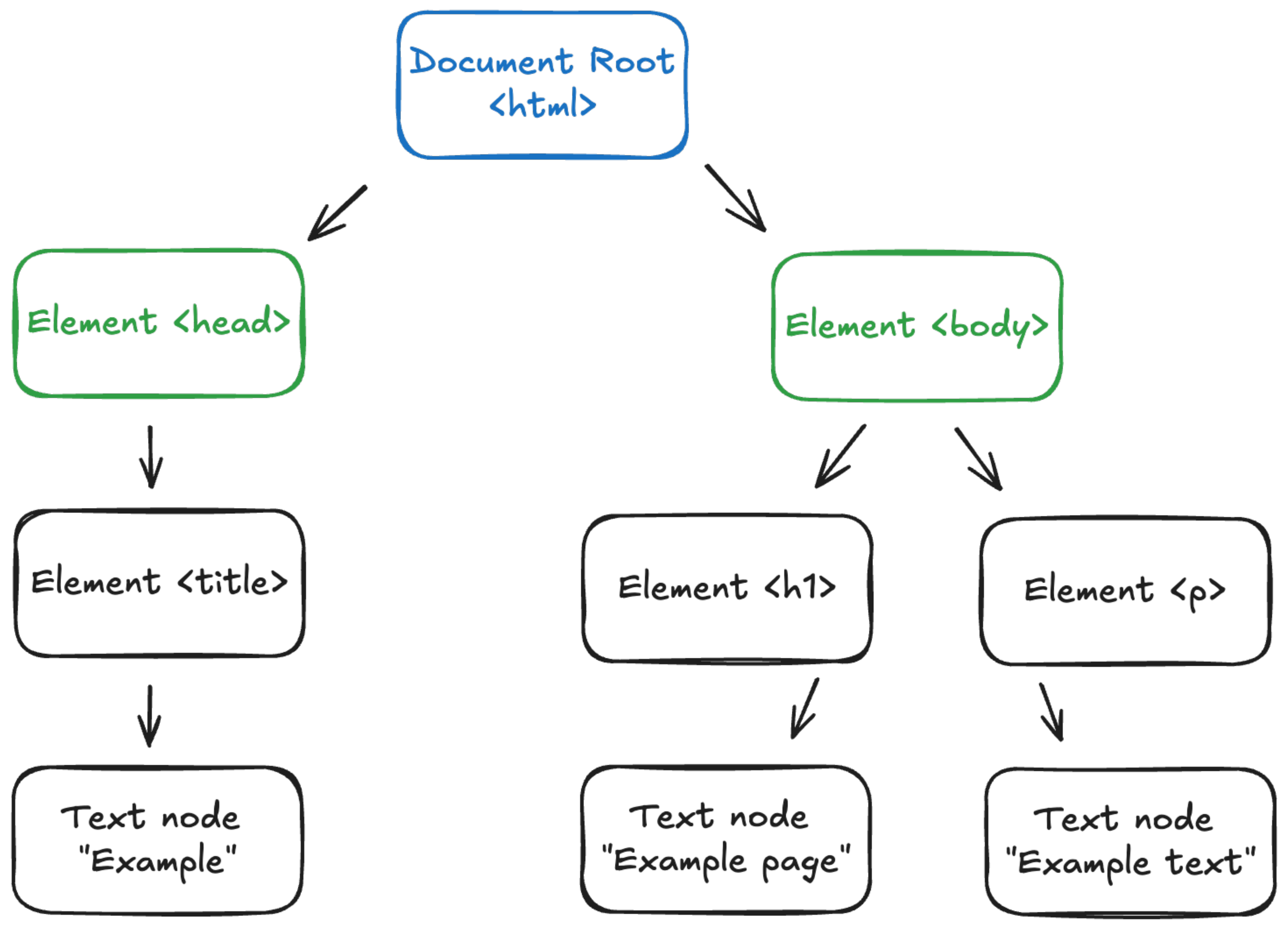


```html
<!doctype html>
<html lang="en-US">
<head>
  <!-- Metadata goes here -->
  <meta charset="utf-8" />
  <title>My test page</title>
</head>
<body>
  <!-- Content goes here -->
  <p>This is my page</p>
</body>
</html>
```

# DOM

## Document Object Model

- When the web page is loaded, browser creates a DOM.
- It represents a web page so that programs can change the document structure, style and content.
- Logical tree - each branch of the tree ends in a node, and each node contains objects.



MITHRIL

## Element categories

# Element categories

- Block-level elements - always starts on a new line and takes up the full width available

  <div>, <p>, <li>, <nav>, <header>, <section>, <footer>

- Inline elements - do not start on a new line and only take up as much width as necessary

  <a>, <button>, <input>, <span>, <label>, <small>

- Element content categories
- By using CSS we can change the display property of each element, but changing the CSS display type doesn't change the category of the element.

MITHRIL

## Common tags and attributes

## Common tags

- Example of good semantics
- Other common tags:

  <div>, <span>, <p>

  <figure>, <picture>, <img>

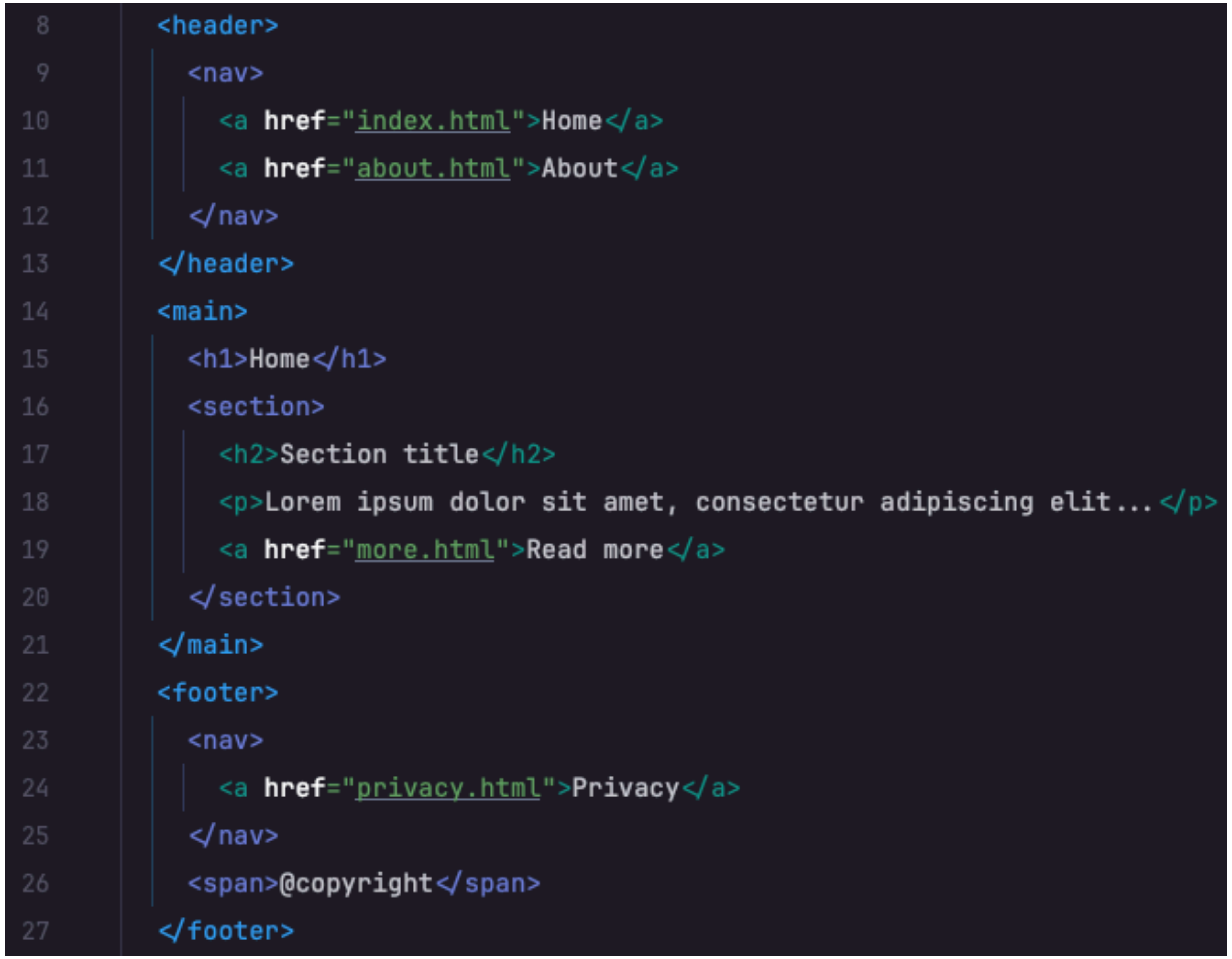
  <ul>, <ol>, <li>

  <form>, <label>, <input>

  <h1>, <h2>…<h6>

  <select>, <option>

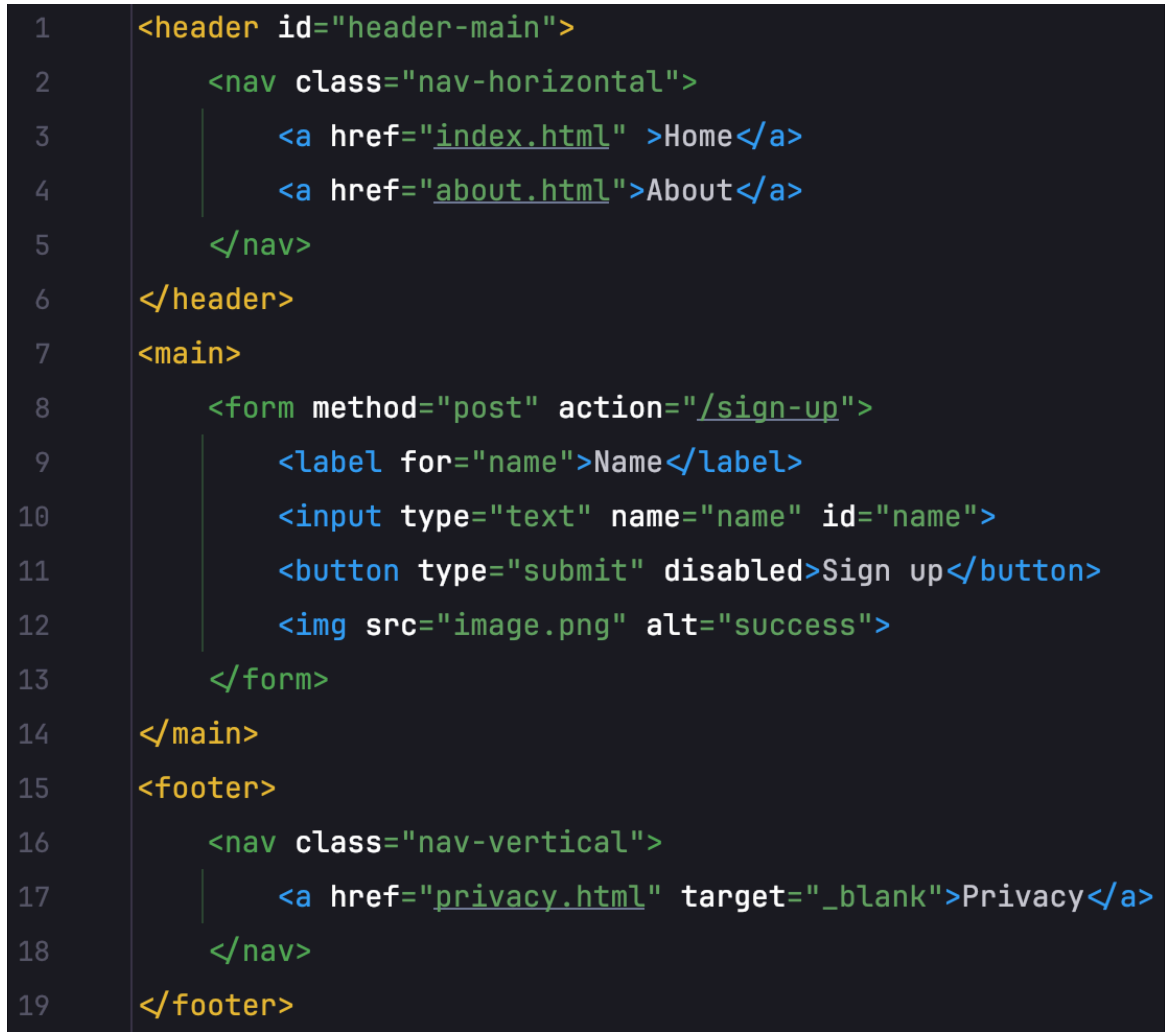
  <article>



```html
<header>
  <nav>
    <a href="index.html">Home</a>
    <a href="about.html">About</a>
  </nav>
</header>
<main>
  <h1>Home</h1>
  <section>
    <h2>Section title</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
    <a href="more.html">Read more</a>
  </section>
</main>
<footer>
  <nav>
    <a href="privacy.html">Privacy</a>
  </nav>
  <span>@copyright</span>
</footer>
```

# Common attributes

- List of all HTML attributes
- Attributes contain extra information about the element that you don't want to appear in the actual content



```html
<header id="header-main">
    <nav class="nav-horizontal">
        <a href="index.html" >Home</a>
        <a href="about.html">About</a>
    </nav>
</header>
<main>
    <form method="post" action="/sign-up">
        <label for="name">Name</label>
        <input type="text" name="name" id="name">
        <button type="submit" disabled>Sign up</button>
        <img src="image.png" alt="success">
    </form>
</main>
<footer>
    <nav class="nav-vertical">
        <a href="privacy.html" target="_blank">Privacy</a>
    </nav>
</footer>
```

MITHRIL

## Structuring content

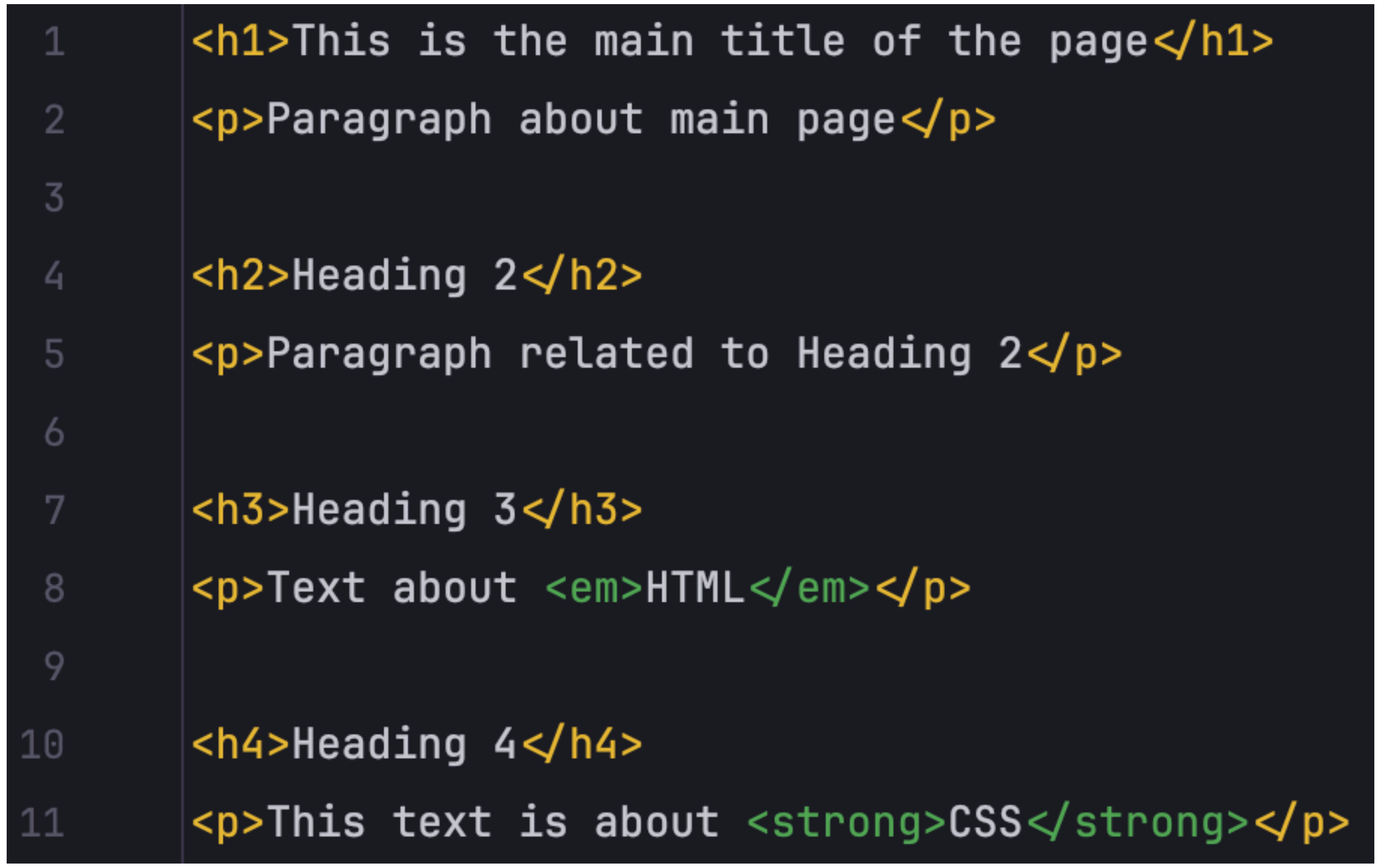
# Structuring content

- Why we need structure ?
  - Users looking at the web page tend to scan quickly to find relevant content
  - Search engine indexing
  - Screen readers
  - Styling and targeting content with JS
- Most common structures:
  - Headings and paragraphs
  - Lists
  - Forms

MITHRIL

## Headings & paragraphs

- Single <h1> should be used per page.
- Emphasize certain words to alter the meaning of a sentence (<em>, <strong>), useful for screen readers.

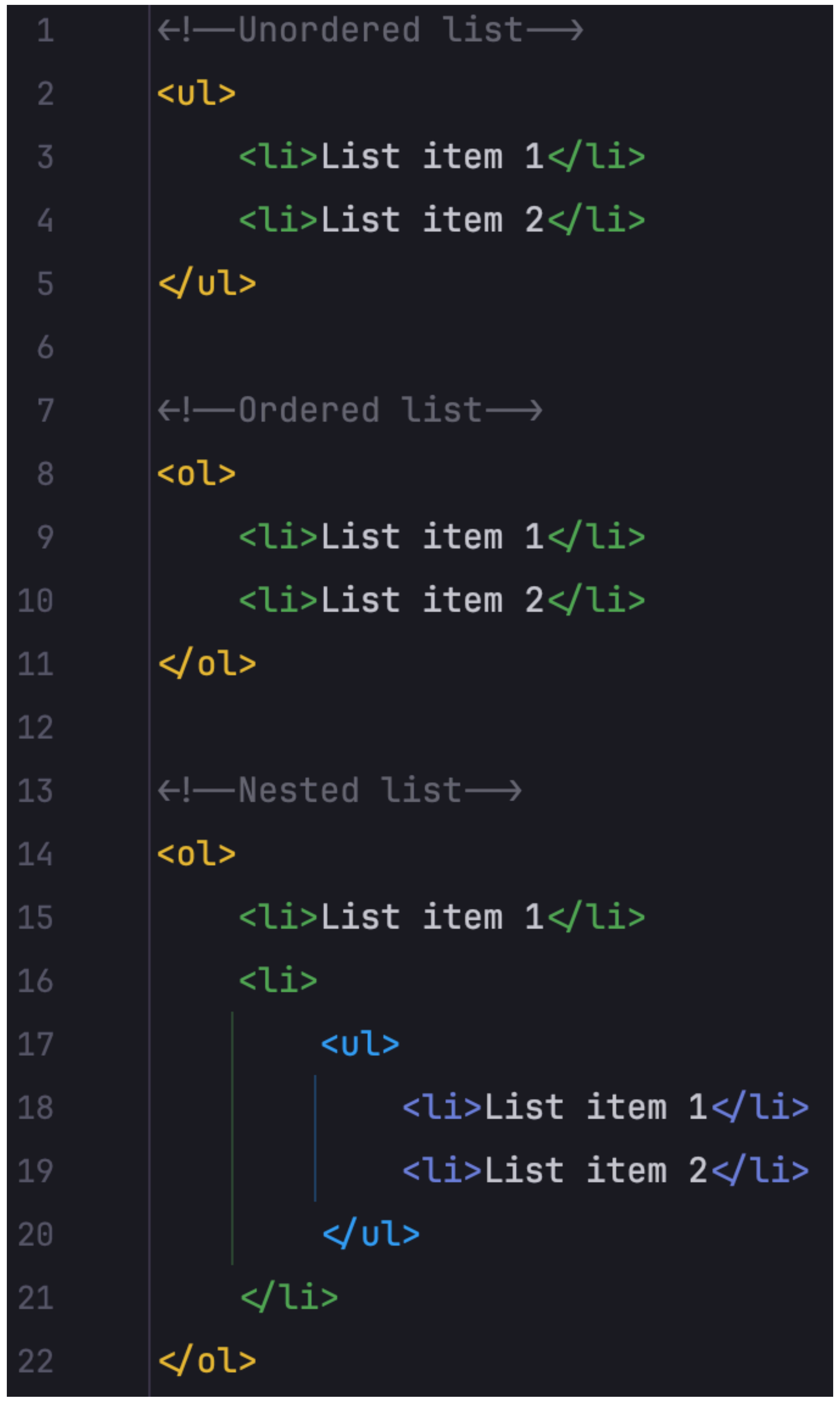


```html
<h1>This is the main title of the page</h1>
<p>Paragraph about main page</p>

<h2>Heading 2</h2>
<p>Paragraph related to Heading 2</p>

<h3>Heading 3</h3>
<p>Text about <em>HTML</em></p>

<h4>Heading 4</h4>
<p>This text is about <strong>CSS</strong></p>
```

MITHRIL

# Lists

- Unordered lists are used to mark up lists of items for which order of items *doesn't* matter.
- Ordered lists are lists in which order of the items *does* matter.
- One list can be nested inside another list, often used in navigation menus.

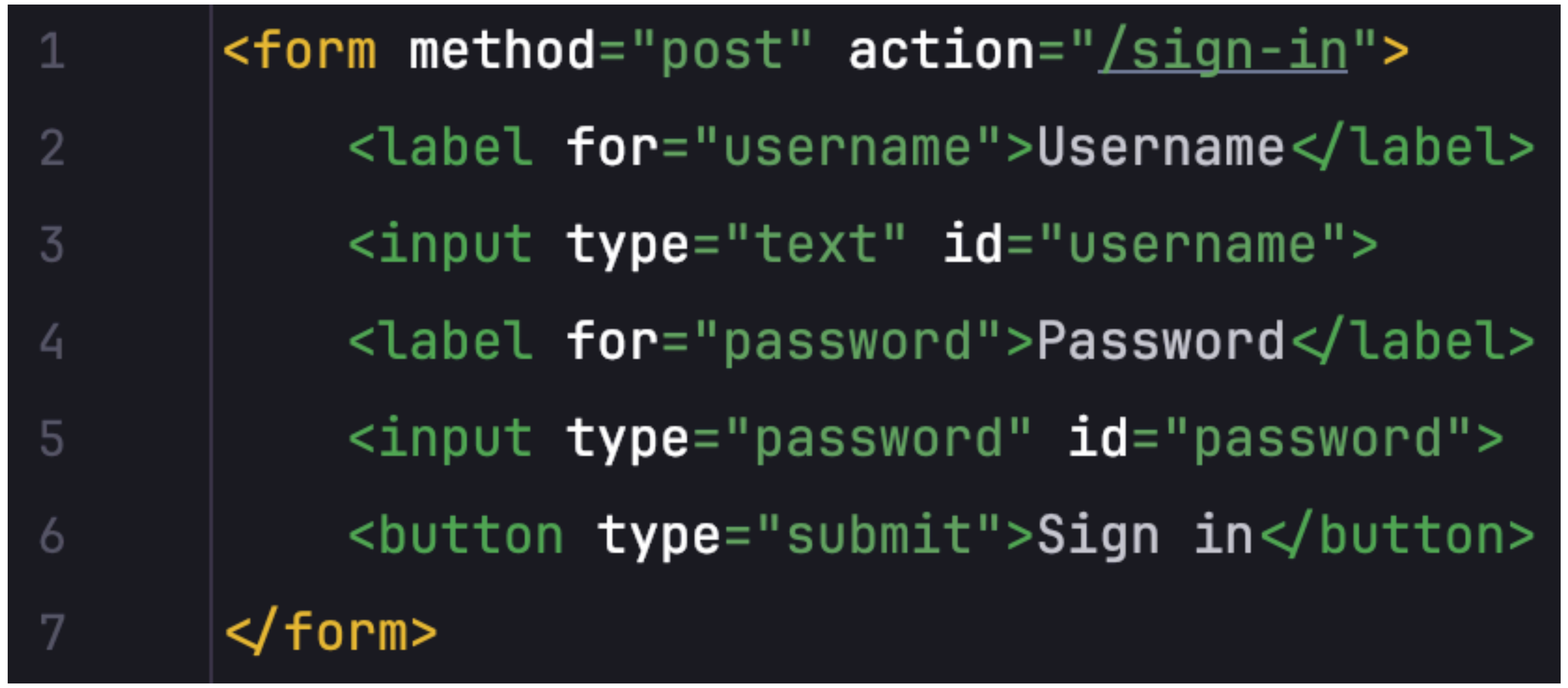


```html
<!--Unordered list-->
<ul>
    <li>List item 1</li>
    <li>List item 2</li>
</ul>

<!--Ordered list-->
<ol>
    <li>List item 1</li>
    <li>List item 2</li>
</ol>

<!--Nested list-->
<ol>
    <li>List item 1</li>
    <li>
        <ul>
            <li>List item 1</li>
            <li>List item 2</li>
        </ul>
    </li>
</ol>
```

MITHRIL

# Forms

- Method - the type of HTTP request
- Action - the URL to send form data to
- Attribute "for" references to element id with the same value (when user clicks on label, then input will be focused)
- HTML input types



```html
<form method="post" action="/sign-in">
    <label for="username">Username</label>
    <input type="text" id="username">
    <label for="password">Password</label>
    <input type="password" id="password">
    <button type="submit">Sign in</button>
</form>
```

MITHRIL

## About CSS

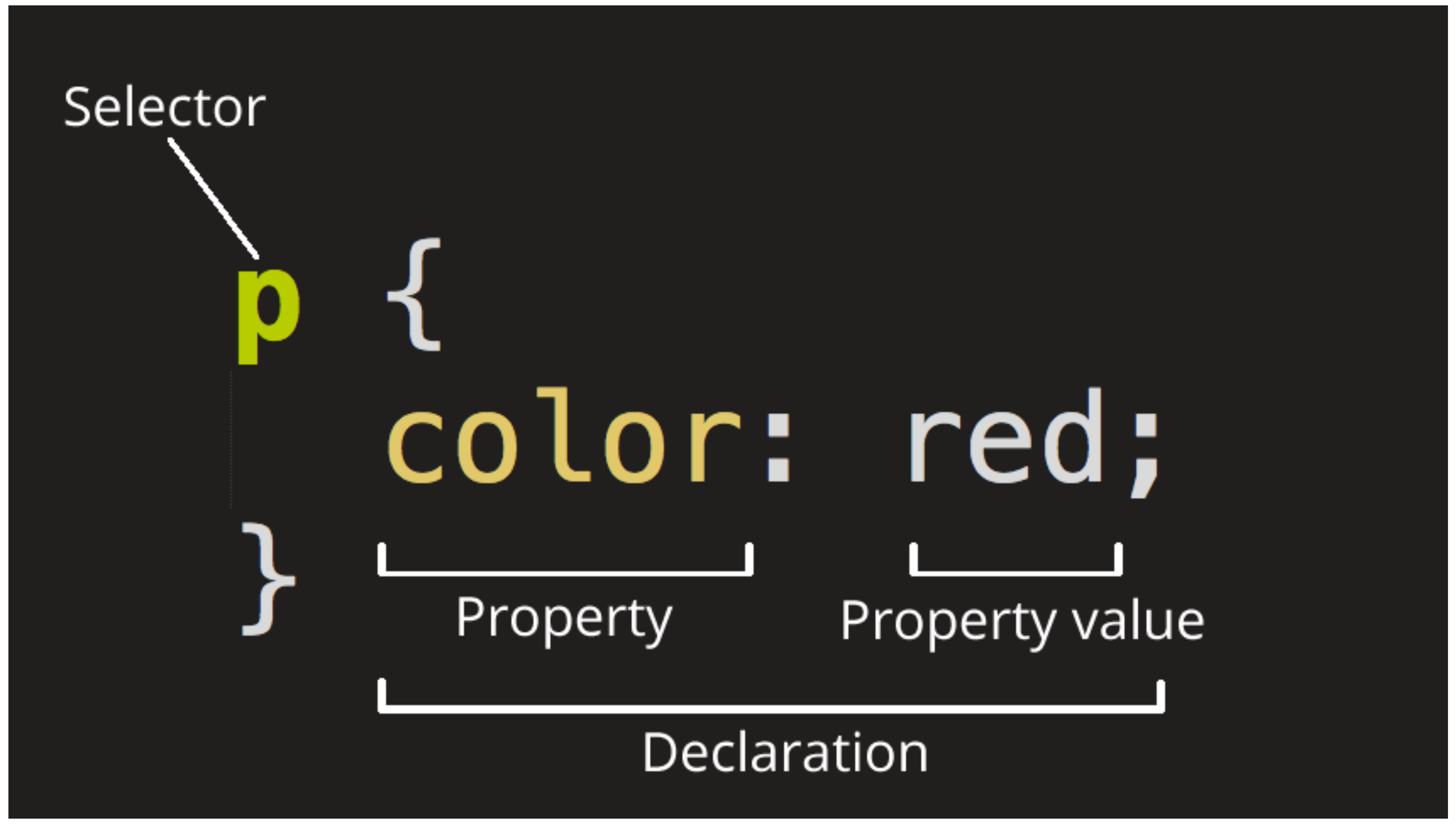
## About

- Cascading Style Sheets
- Set of rules defining how an HTML element will be presented in the browser
- Allows you to separate your web sites HTML content from it’s style
- CSS properties
- How to use CSS?
  - External stylesheet
  - Internal stylesheet
  - Inline styles

MITHRIL

## Syntax

- Image shows CSS ruleset or rule.
- Selector can be HTML element or attribute (class, id, name…).
- Declaration block consists of property and value.

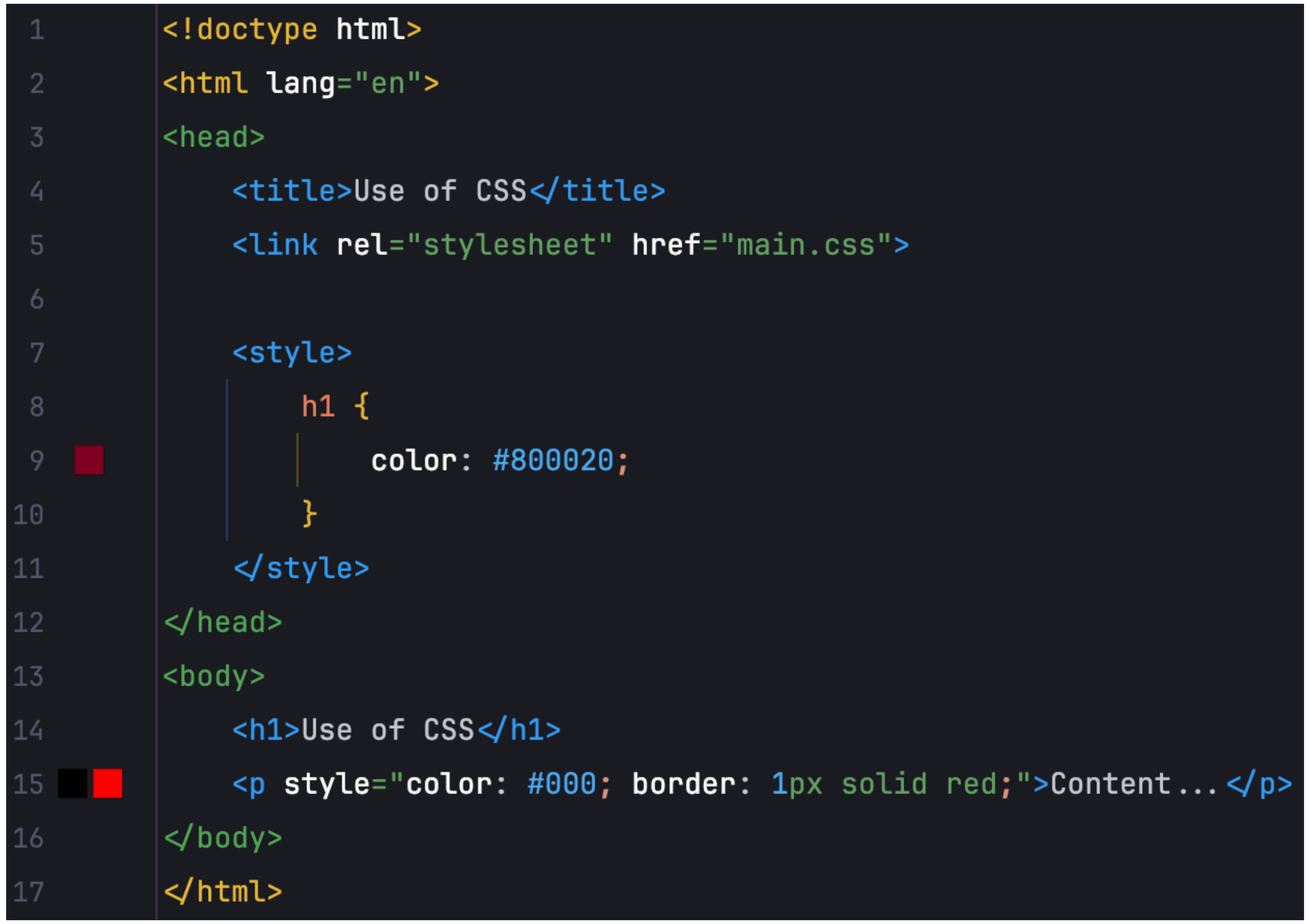


```
p {
  color: red;
}
```

MITHRIL

# Use of CSS

- How to use CSS?
- We can write all of our CSS in separate file, for example “main.css” and then link that file inside <head>.
- Style can also be written inside head <style> tags or inline within element.
- Inline styles always overwrite external styles.
- Inline CSS > Internal CSS > External CSS



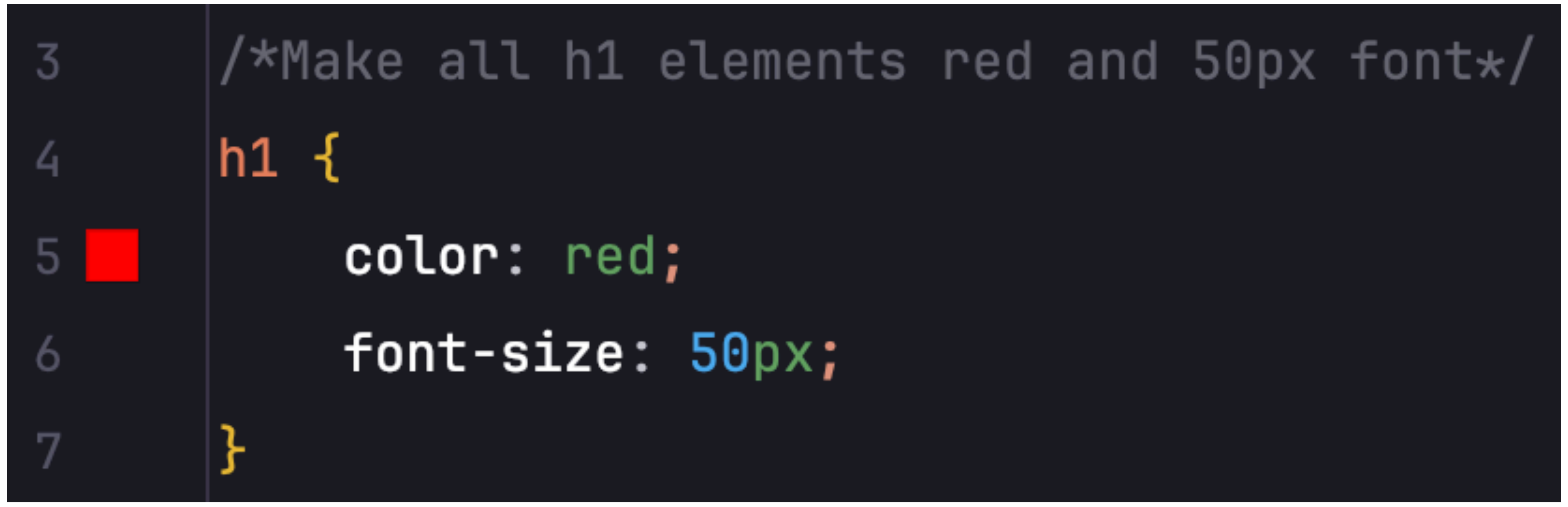
```html
<!doctype html>
<html lang="en">
<head>
    <title>Use of CSS</title>
    <link rel="stylesheet" href="main.css">

    <style>
        h1 {
            color: #800020;
        }
    </style>
</head>
<body>
    <h1>Use of CSS</h1>
    <p style="color: #000; border: 1px solid red;">Content...</p>
</body>
</html>
```

## Selectors

- Element selector
- ID selector
- Class selector
- Attribute selector
- Pseudo-class selector

## Element selector

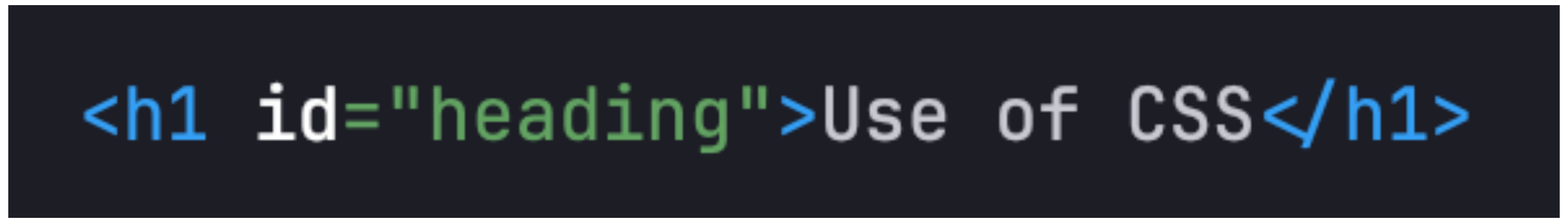
- Any HTML element can be CSS element selector.



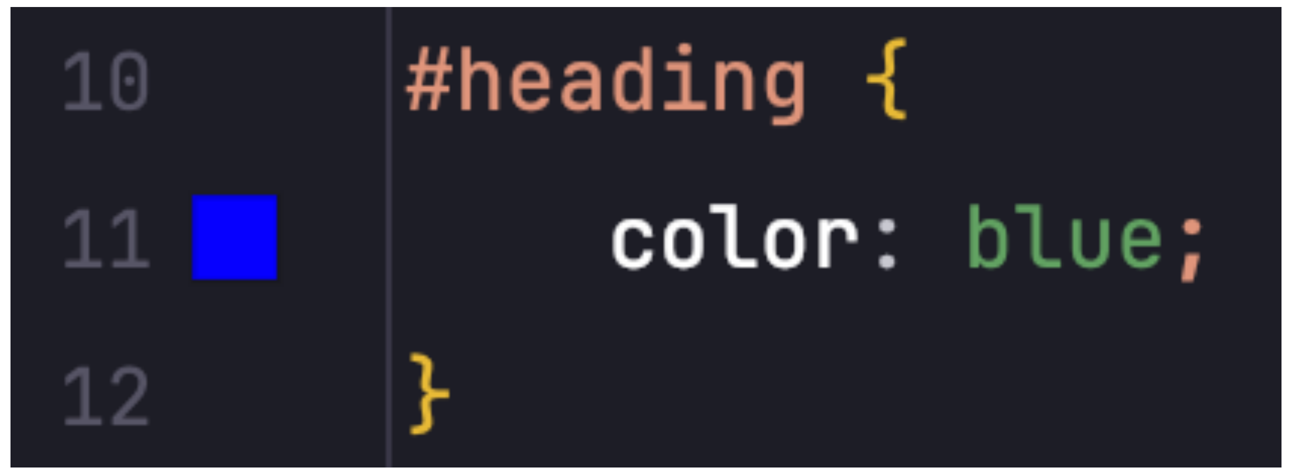
```
/*Make all h1 elements red and 50px font*/
h1 {
    color: red;
    font-size: 50px;
}
```

## ID selector

- Only one element with same ID is allowed on one page.



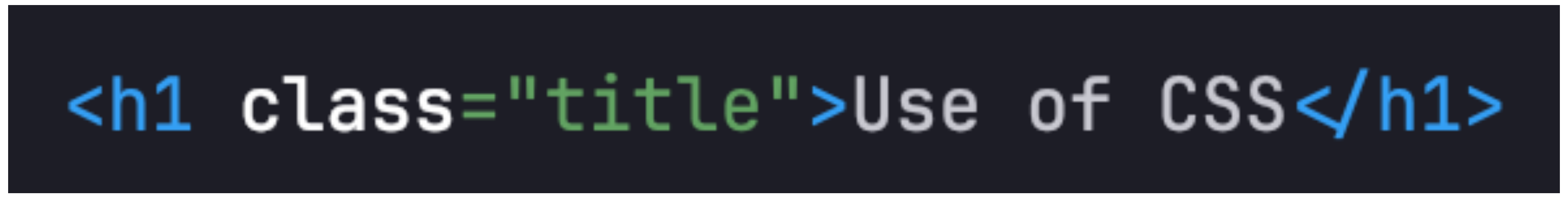
```html
<h1 id="heading">Use of CSS</h1>
```



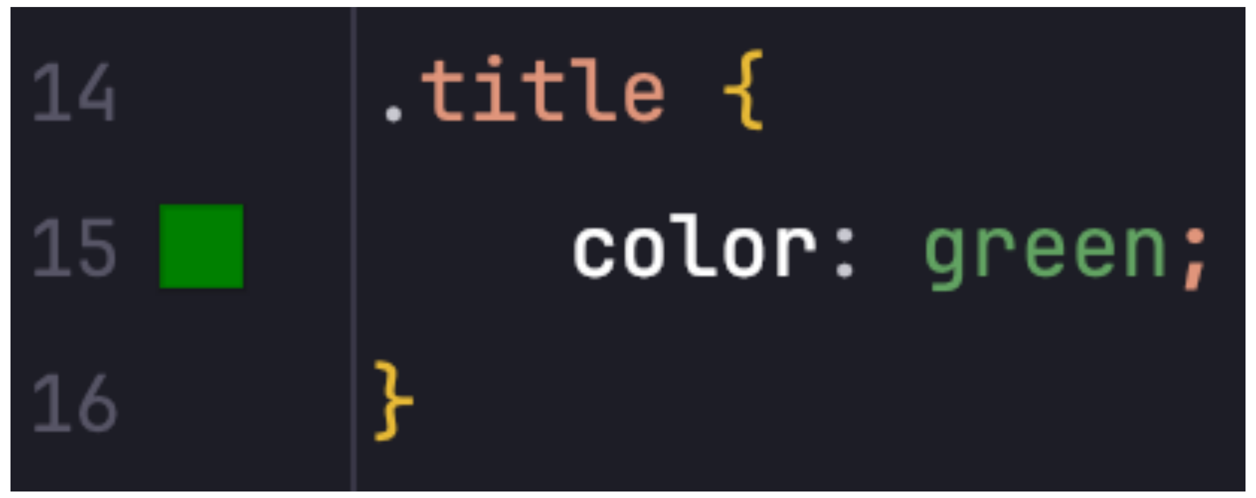
```css
#heading {
    color: blue;
}
```

## Class selector

- Compared to ID selectors, any number of class selectors with the same name can be written on the same page.



```html
<h1 class="title">Use of CSS</h1>
```

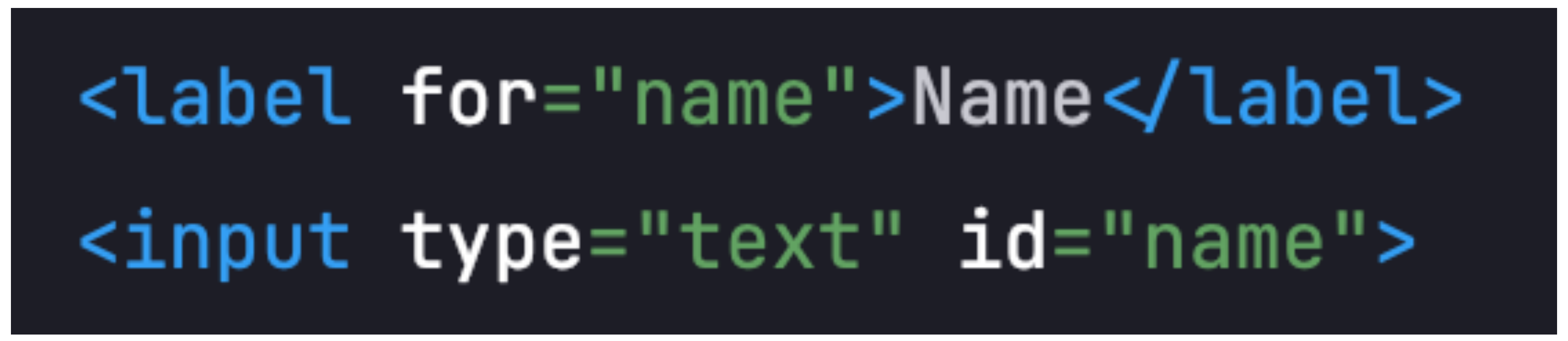


```css
.title {
    color: green;
}
```

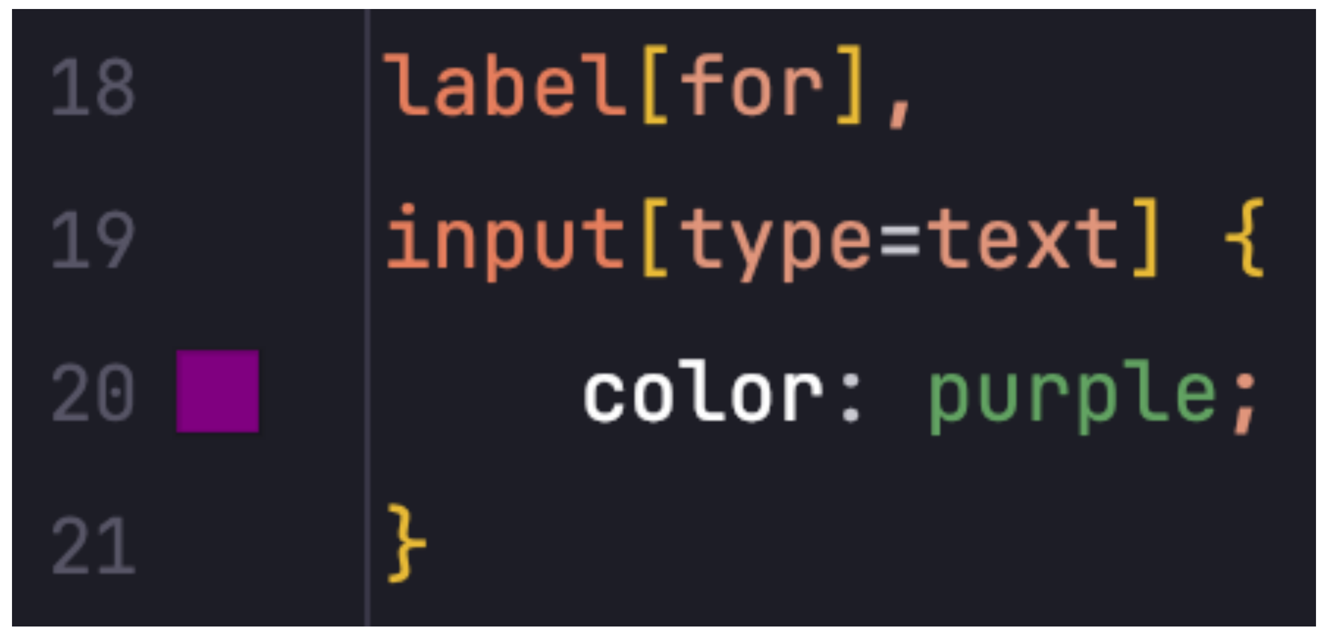
MITHRIL

# Attribute selector

- Any HTML attribute can be a selector.
- We can also target specific attribute values.



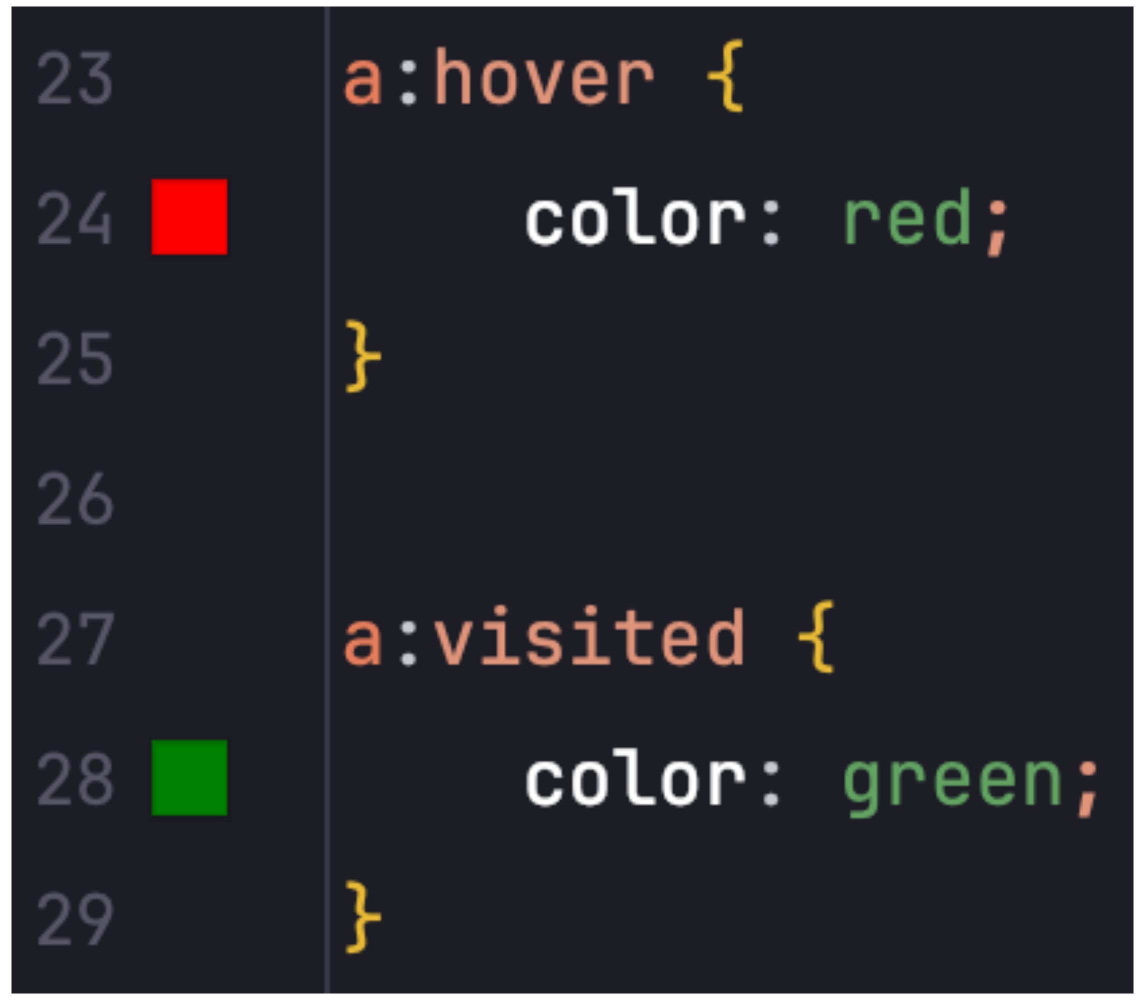
```html
<label for="name">Name</label>
<input type="text" id="name">
```



```css
label[for],
input[type=text] {
    color: purple;
}
```

# Pseudo-class selector

- A pseudo-class is used to define a special state of an element.
- Pseudo-classes can be combined with CSS classes.



```css
a:hover {
    color: red;
}

a:visited {
    color: green;
}
```

MITHRIL

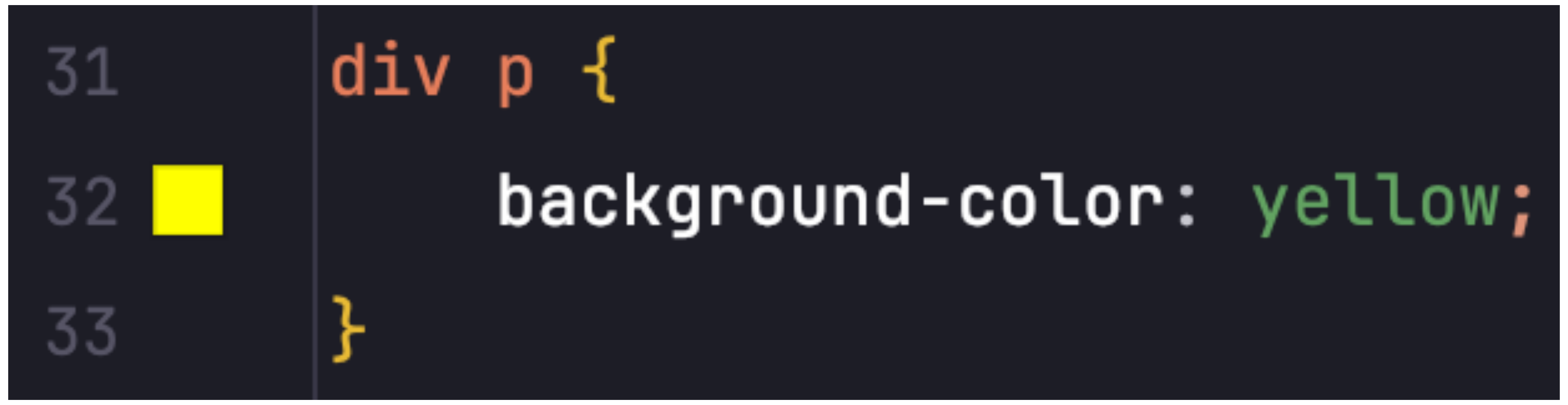
## CSS combinators

- A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.
  - Descendant combinator (space)
  - Child combinator (>)
  - Next sibling combinator (+)
  - Subsequent-sibling combinator (~)

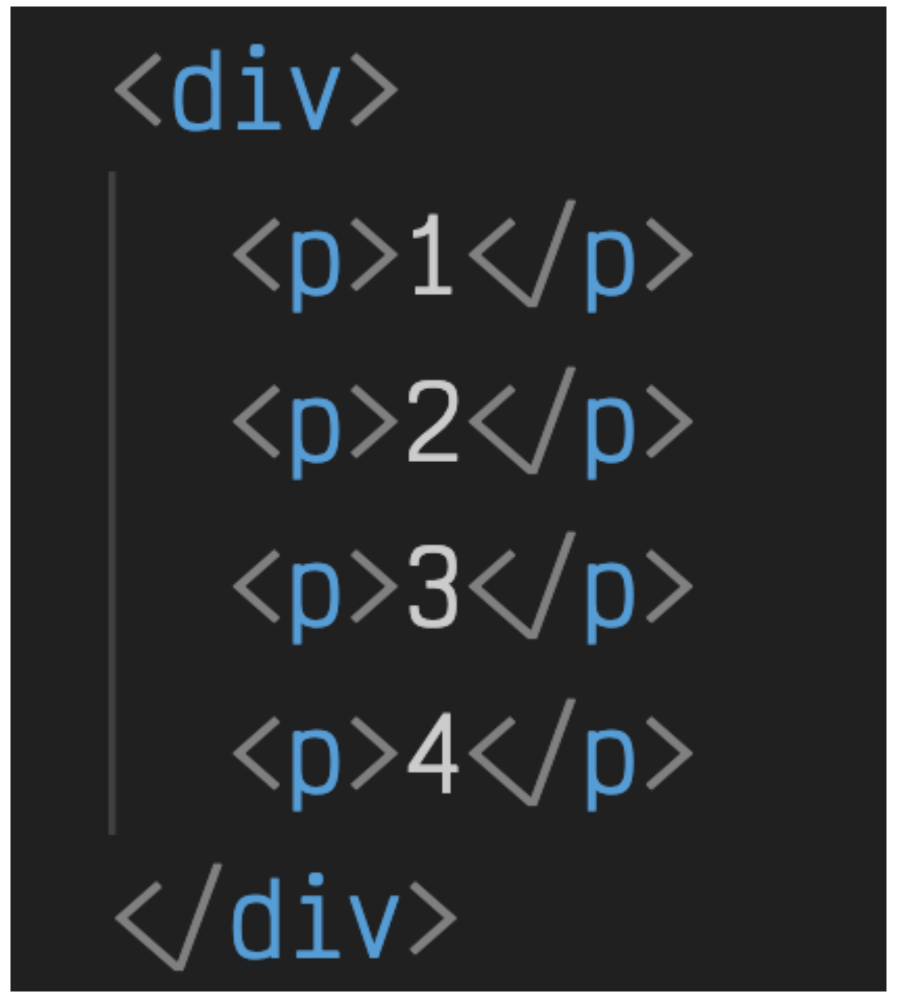
MITHRIL

# Descendant combinator (space)

- The descendant combinator matches all elements that are descendants of a specified element.
- The following example selects all <p> elements inside <div> elements:



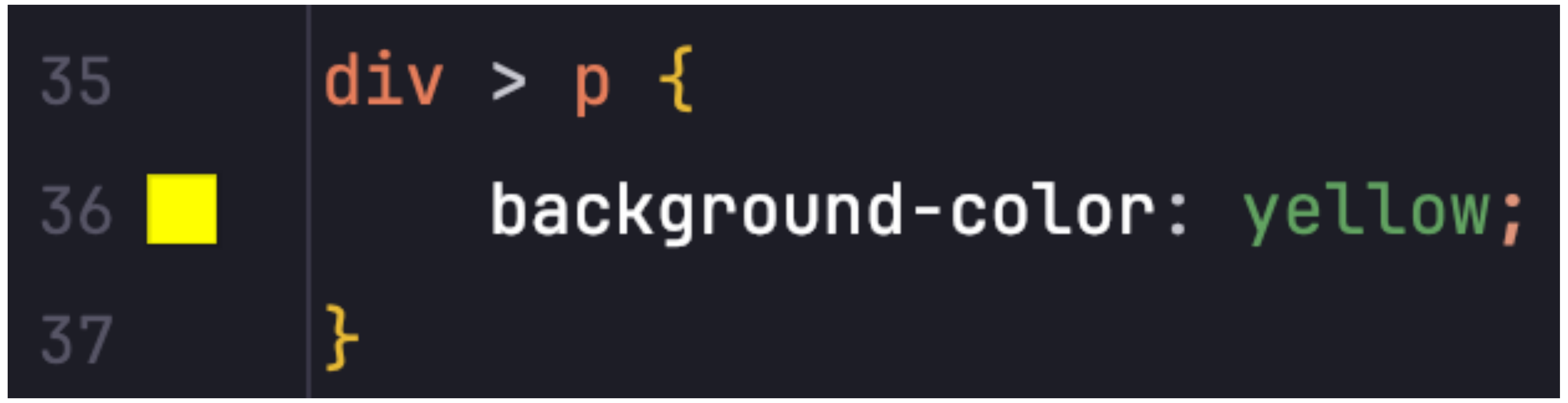
```css
div p {
    background-color: yellow;
}
```



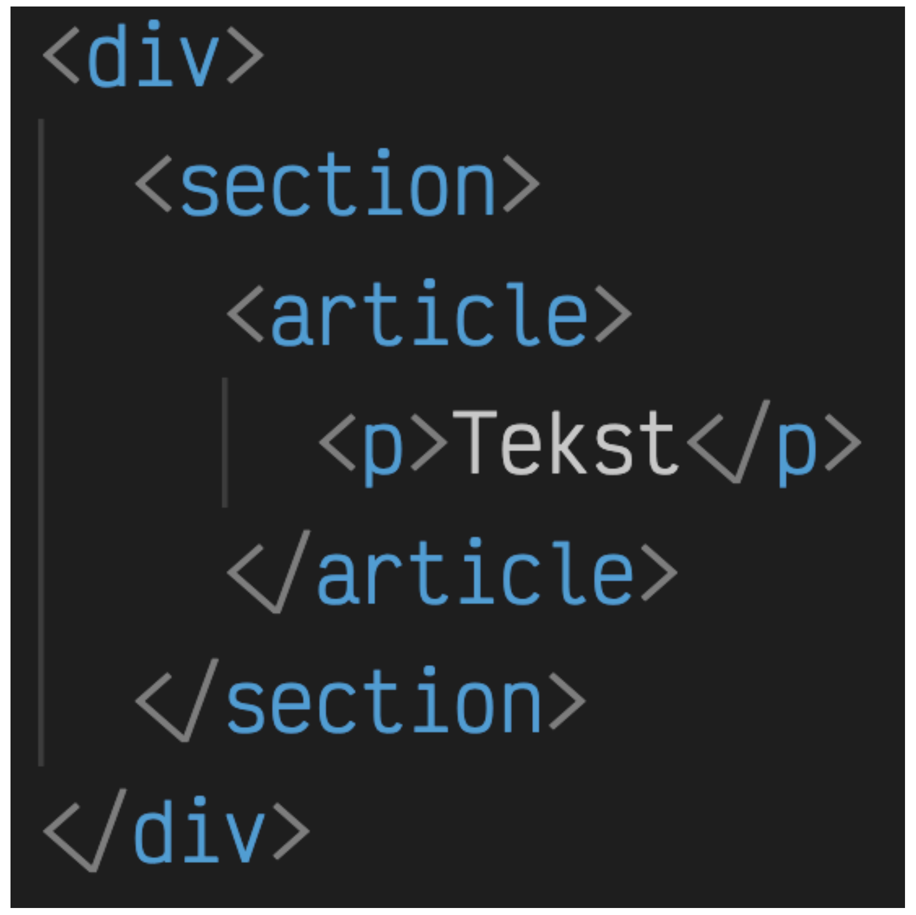
```html
<div>
  <p>1</p>
  <p>2</p>
  <p>3</p>
  <p>4</p>
</div>
```

## Child combinator (>)

- The child combinator selects all elements that are the children of a specified element.
- The following example selects all <p> elements that are children of a <div> element:



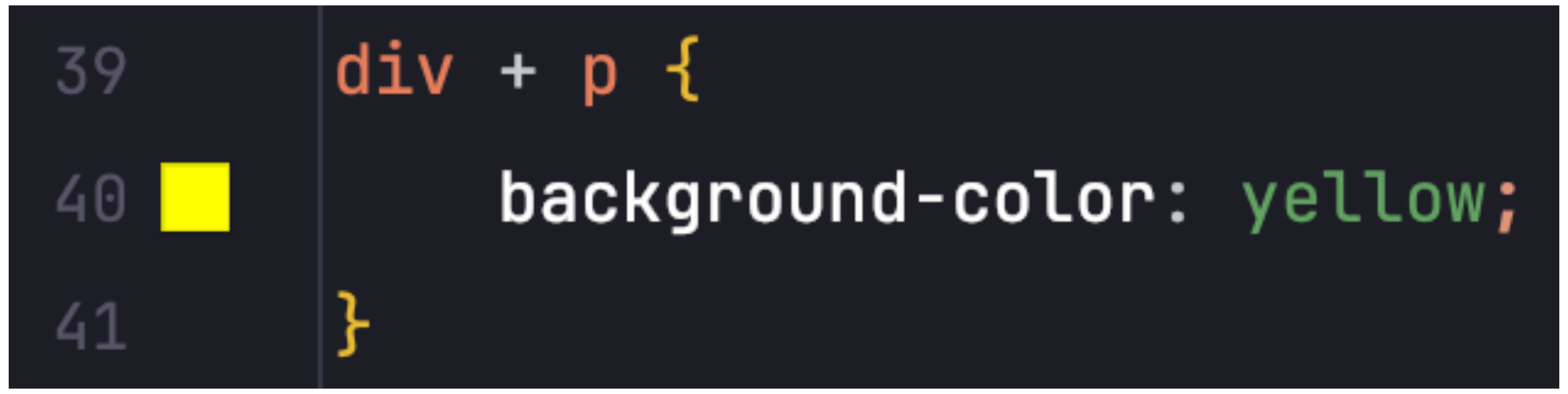
```css
div > p {
    background-color: yellow;
}
```



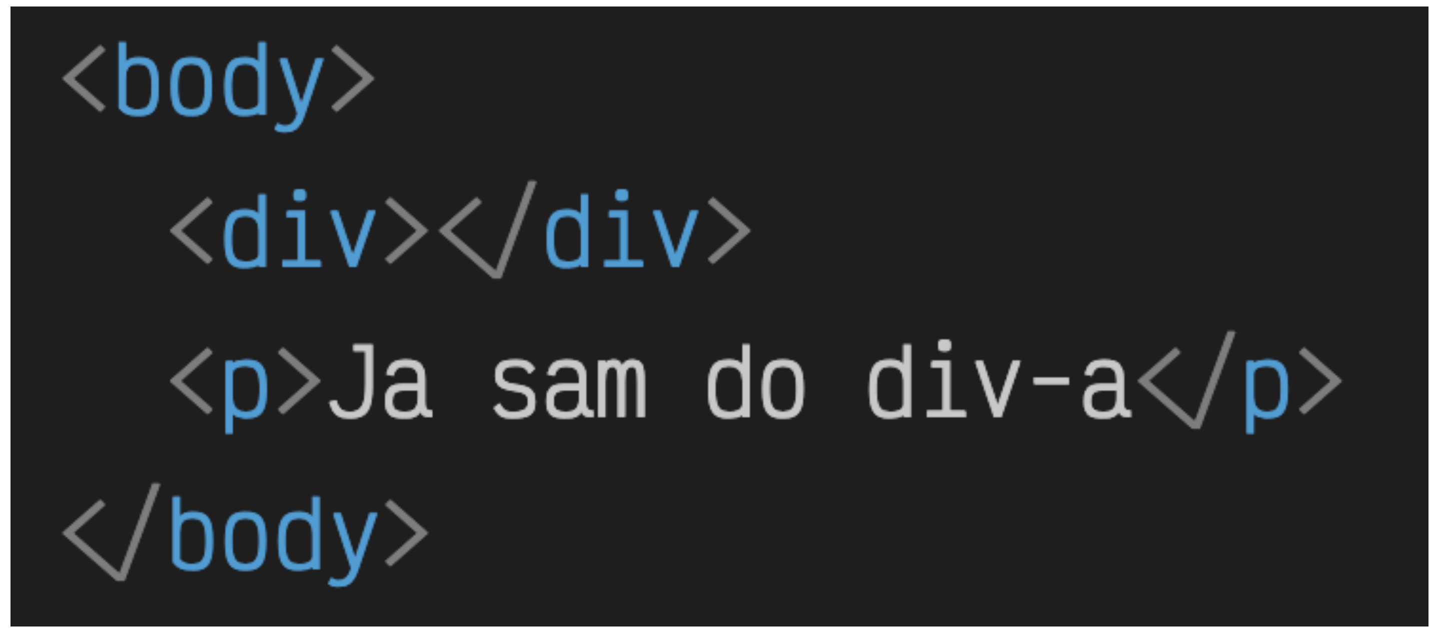
```html
<div>
  <section>
    <article>
      <p>Tekst</p>
    </article>
  </section>
</div>
```

## Next Sibling combinator (+)

- The next sibling combinator is used to select an element that is directly after another specific element.
- The following example selects the first <p> element that are placed immediately after <div> elements:



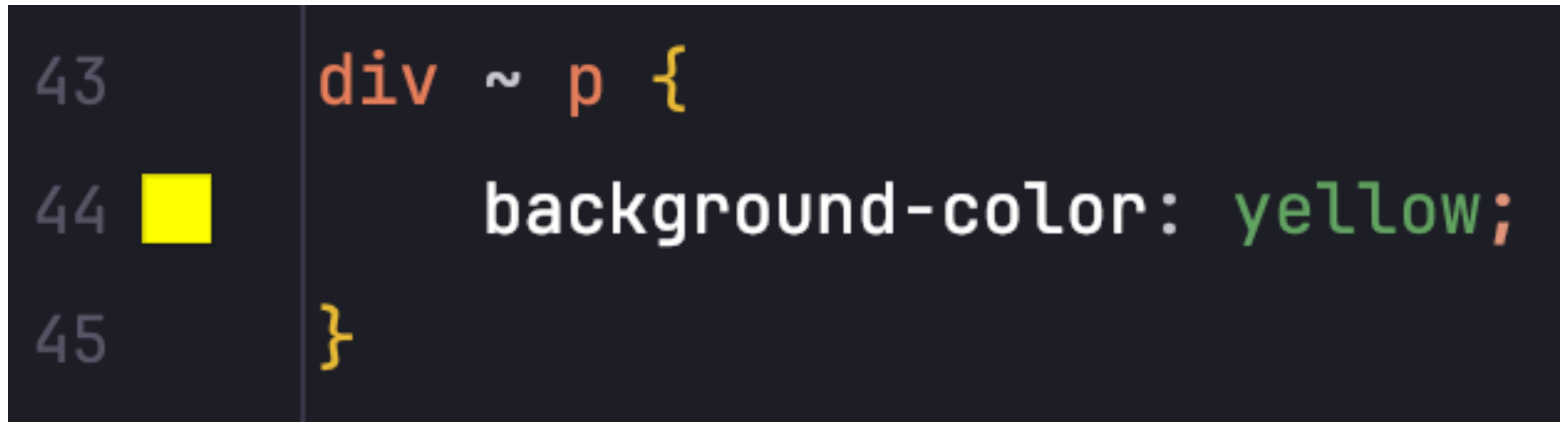
```css
div + p {
    background-color: yellow;
}
```



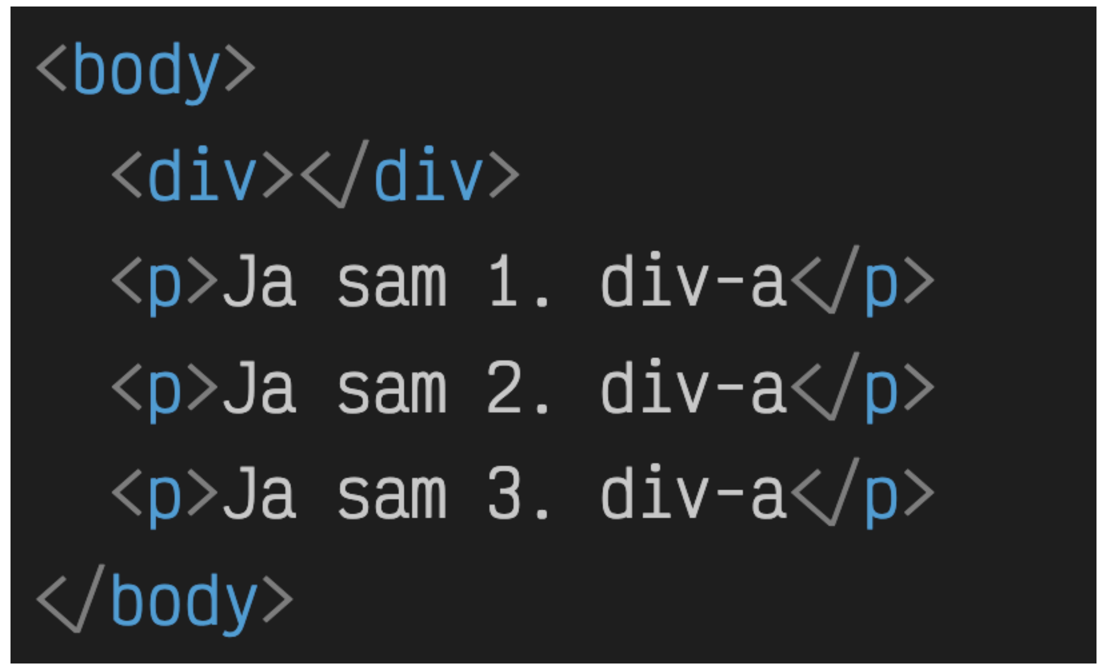
```html
<body>
  <div></div>
  <p>Ja sam do div-a</p>
</body>
```

# Subsequent-sibling combinator (~)

- The subsequent-sibling combinator selects all elements that are next siblings of a specified element.
- The following example selects all <p> elements that are next siblings of <div> elements:



```css
div ~ p {
    background-color: yellow;
}
```



```html
<body>
  <div></div>
  <p>Ja sam 1. div-a</p>
  <p>Ja sam 2. div-a</p>
  <p>Ja sam 3. div-a</p>
</body>
```

## Cascade and specificity

## Cascade & specificity

- The CSS language has rules to control which rule will win in the event of collision - these are called *cascade* and *specificity (and inheritance)*.
- Cascade is an algorithm that defines how to combine property values originating from different sources - read more.
- Specificity is a weight that is applied to a given CSS declaration, determined by the number of each selector type in matching selector - read more.
- Read more details here.

MITHRIL

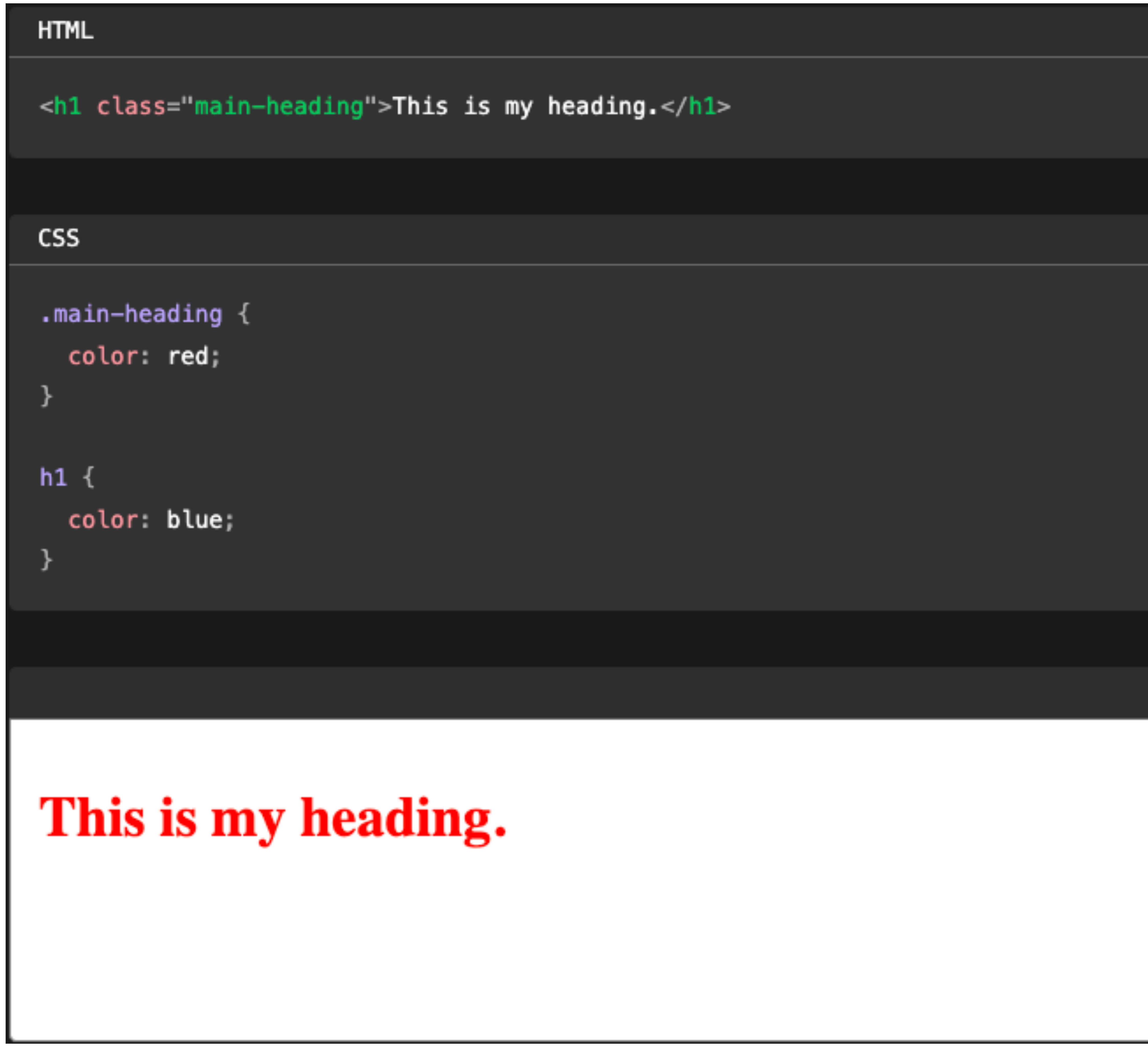
# Specificity

- The following selector types increases by specificity:

1. Type selectors (i.e. h1) and pseudo-elements (i.e. ::before)

2. Class selectors (i.e. .example), attributes selector (i.e. [type=input]) and pseudo-classes (i.e. :hover)

3. ID selectors (i.e. #example)



HTML

```html
<h1 class="main-heading">This is my heading.</h1>
```

CSS

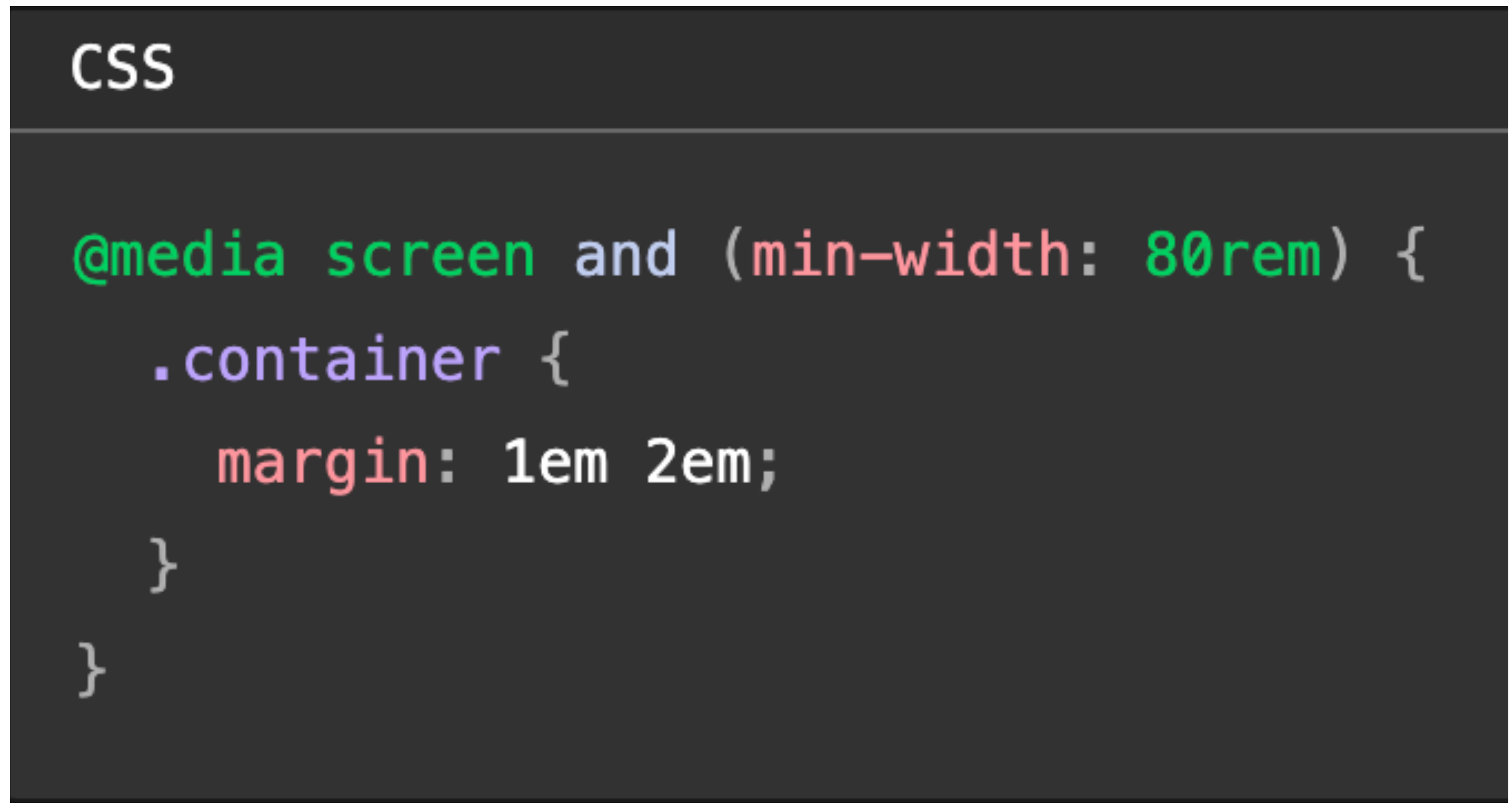
```css
.main-heading {
  color: red;
}

h1 {
  color: blue;
}
```

This is my heading.

## Responsive design

# Responsive design

- *Responsive web design* (RWD) is a web design approach to make web pages render well on all screen sizes and resolutions while ensuring good usability.
- Mobile-first design.
- Some of the web platform features you might want to use when creating a responsive site:
  - Media queries allow us to run a series of tests (e.g. whether the user's screen is greater than a certain width, or a certain resolution) and apply CSS selectively to style the page appropriately for the user's needs.
  - Several layout methods, including Multiple-column layout, Flexbox, and Grid are responsive by default. They all assume that you are trying to create a flexible grid and give you easier ways to do so.


CSS

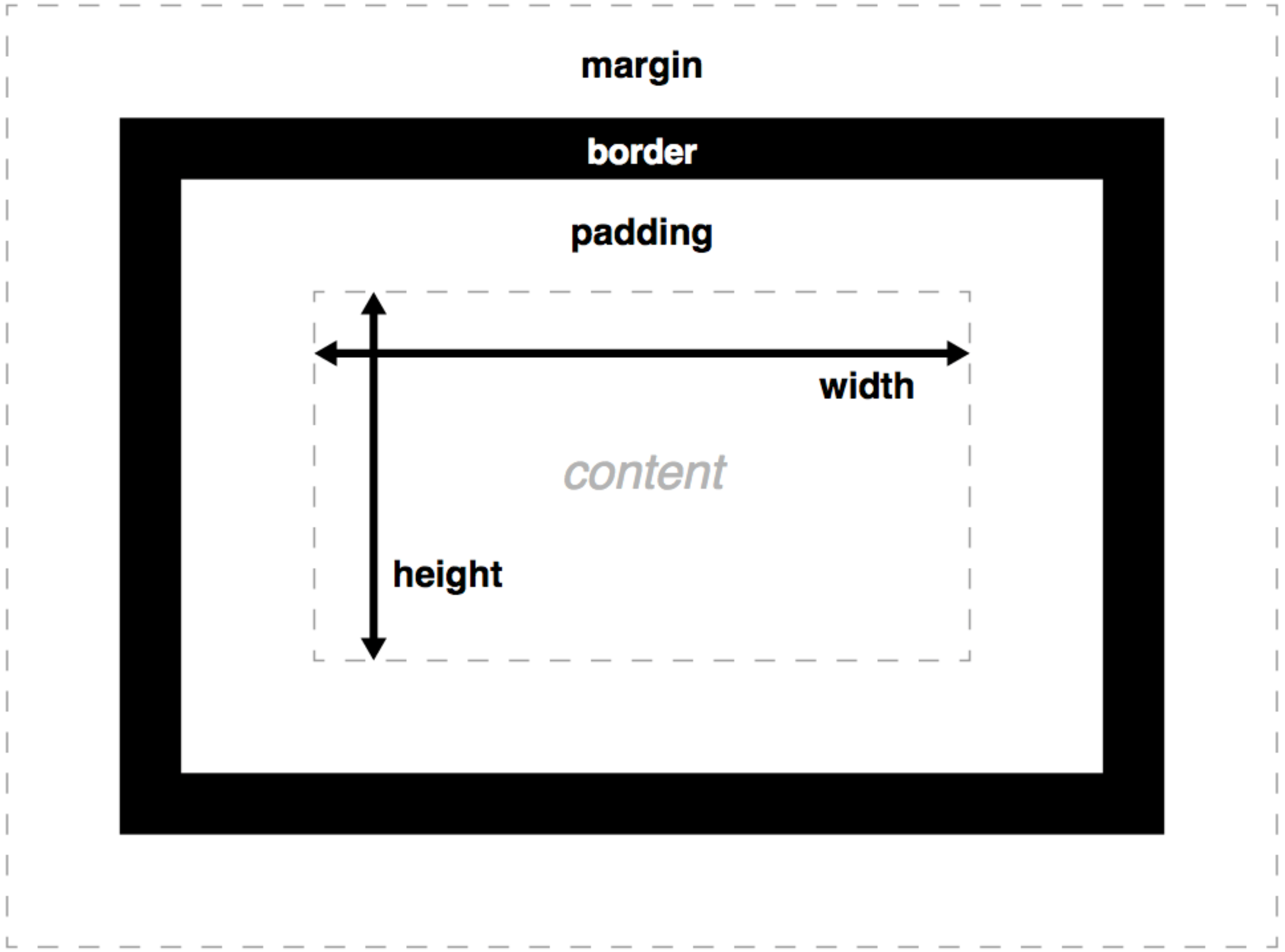
```css
@media screen and (min-width: 80rem) {
  .container {
    margin: 1em 2em;
  }
}
```

MITHRIL

## Box model

## Box model

- CSS box model
- For easier development setup the *box-sizing* property to value *border-box*. This property sets how the total width and height if an element is calculated.
  - More about box-sizing property



MITHRIL

## Hands-on

## Hands-on

- Create a simple HTML page
- Page has to concur to HTML standards
- Page consists of header, footer, and a main section
- Add a title to the HTML document
- Add a title of the page
- In the main section, add a form with input (type text) and a button, which we will use to input our TODOs.
- Add a “mock” list of initial TODOs - unordered list with several list items
- Using CSS style the page and it’s elements by your liking

MITHRIL