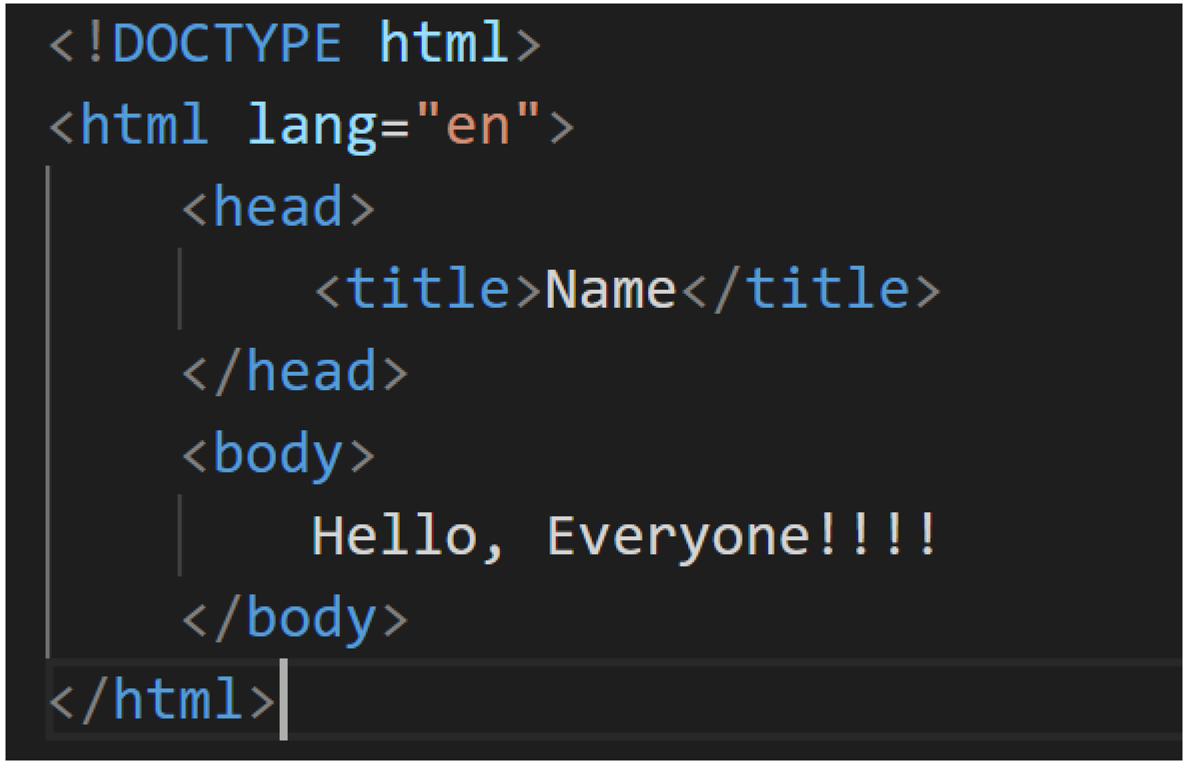
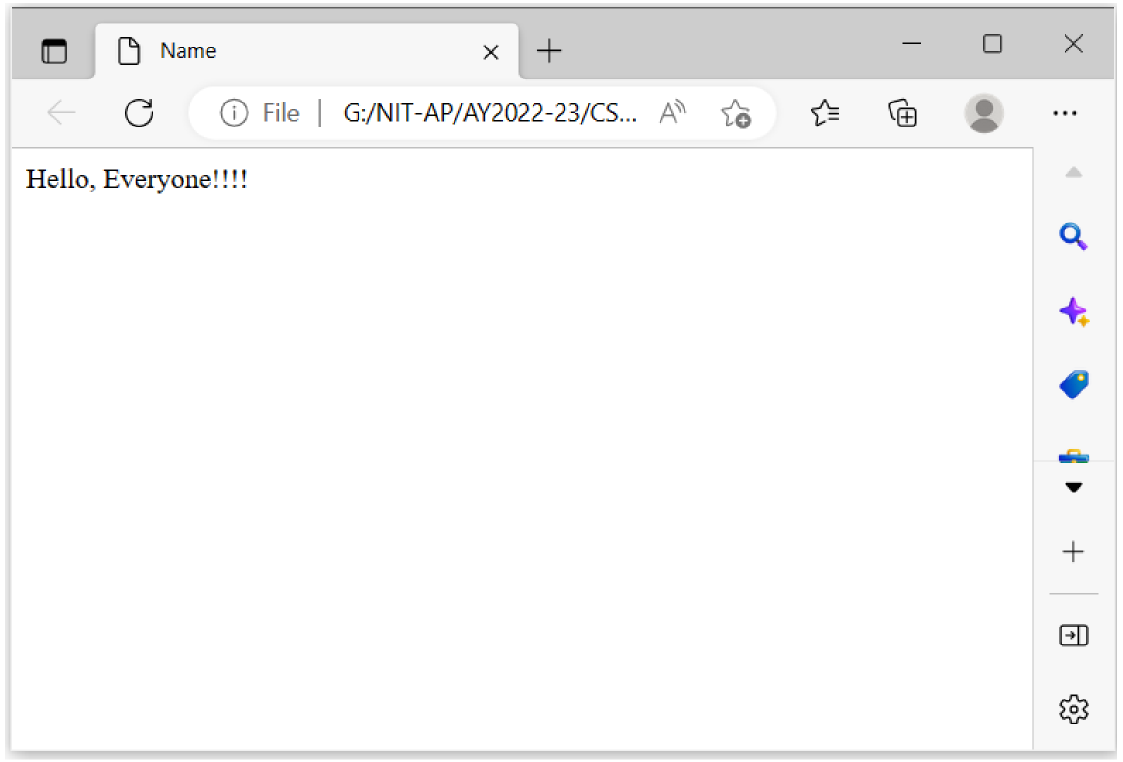
HTML & CSS

First HTML Code

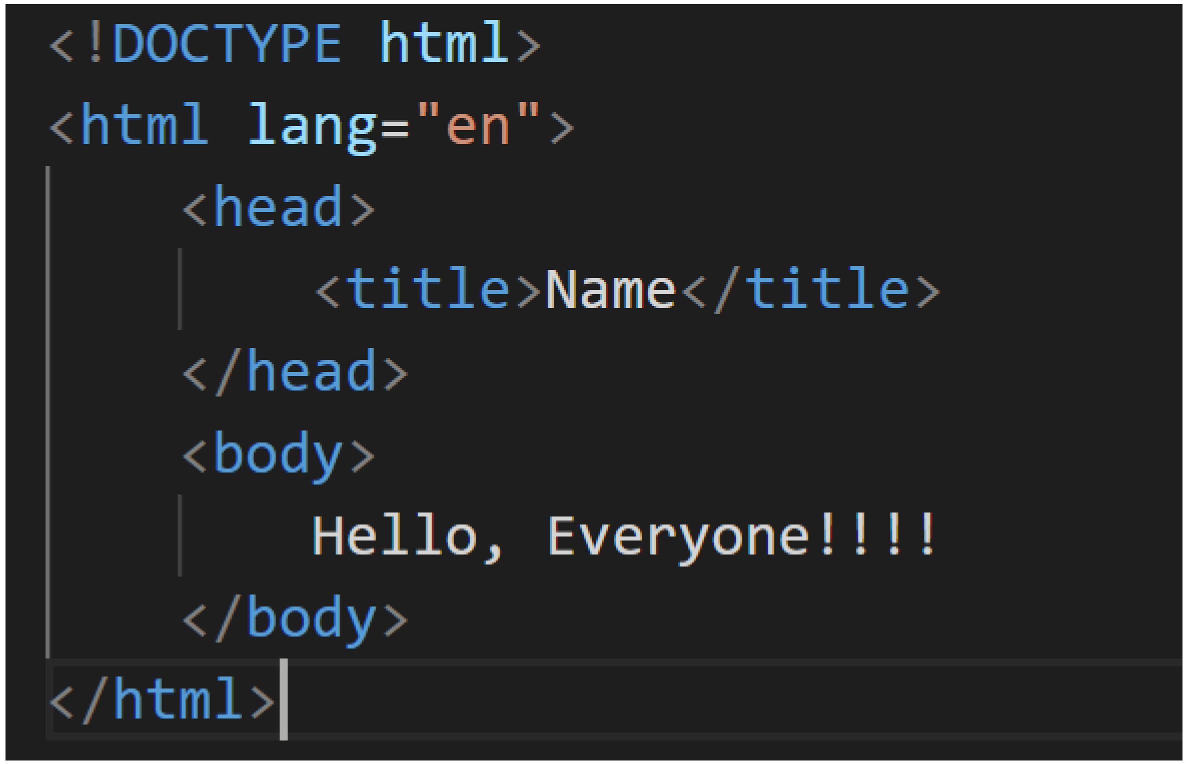


```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Name</title>
    </head>
    <body>
        Hello, Everyone!!!!
    </body>
</html>
```

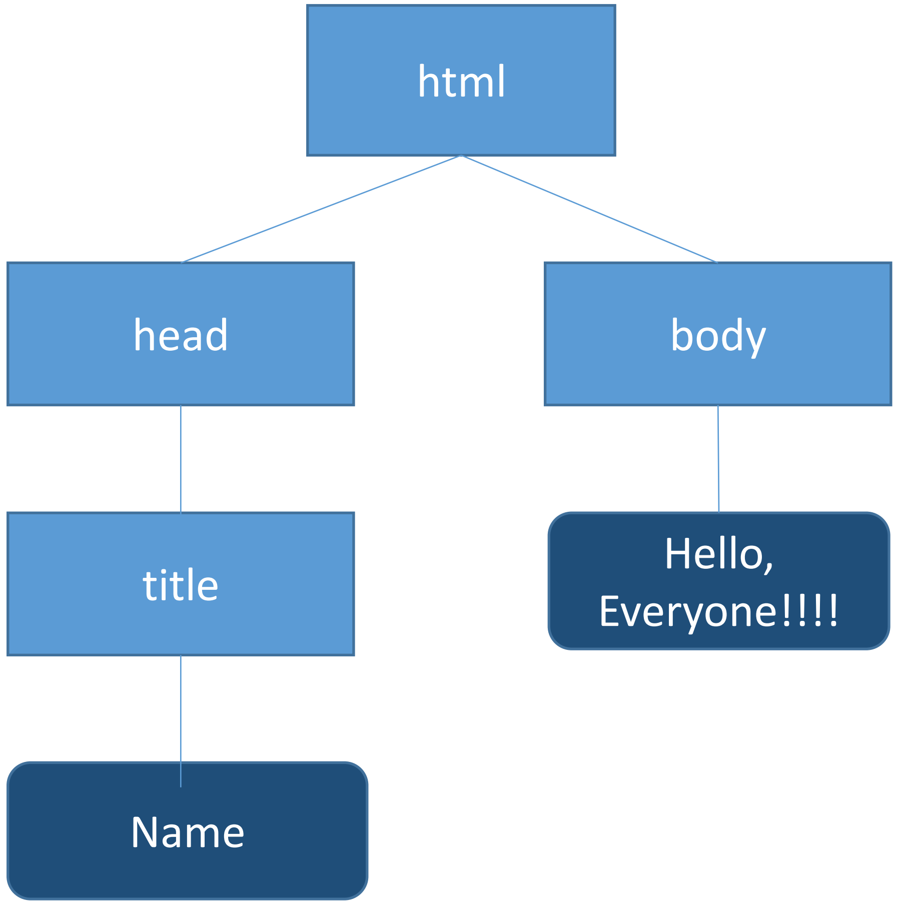


Hello, Everyone!!!!

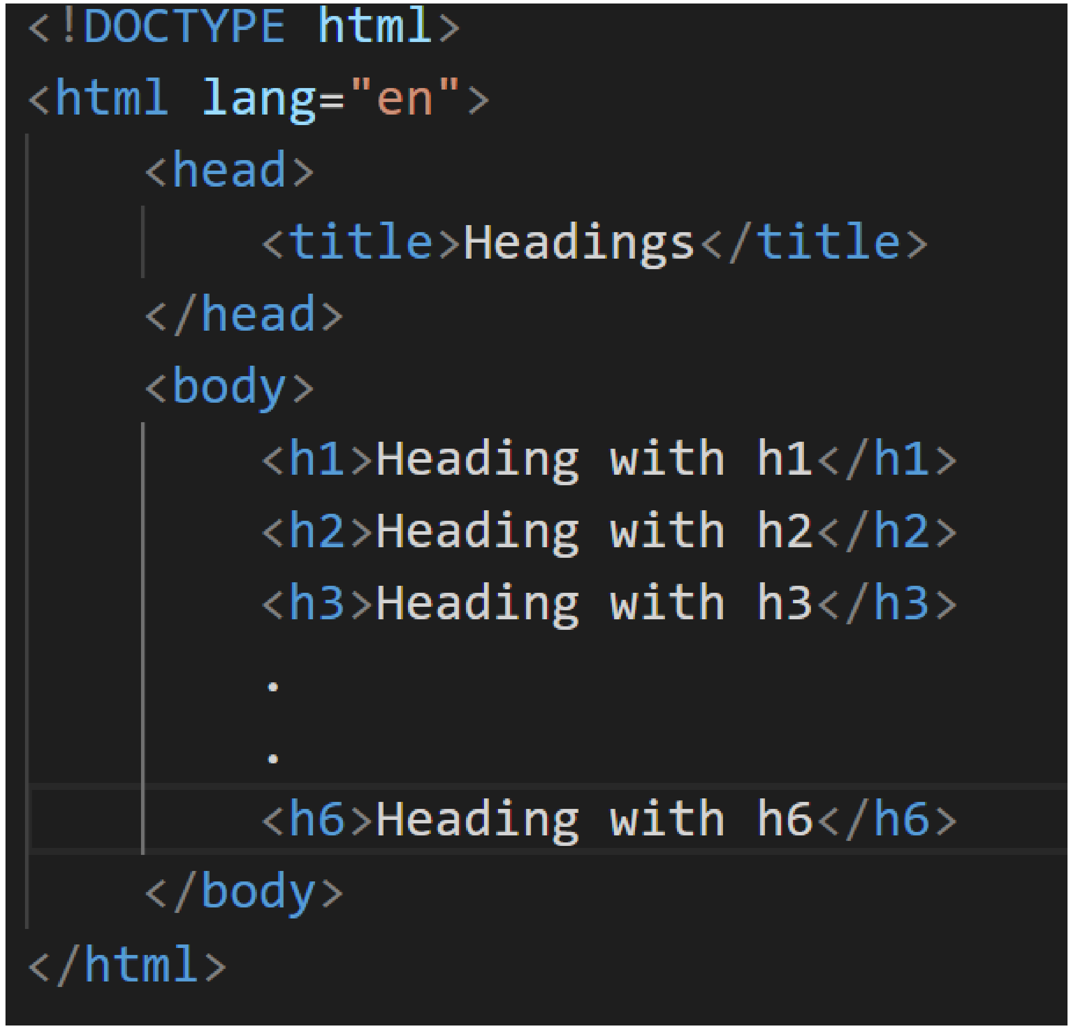
Document Object Model



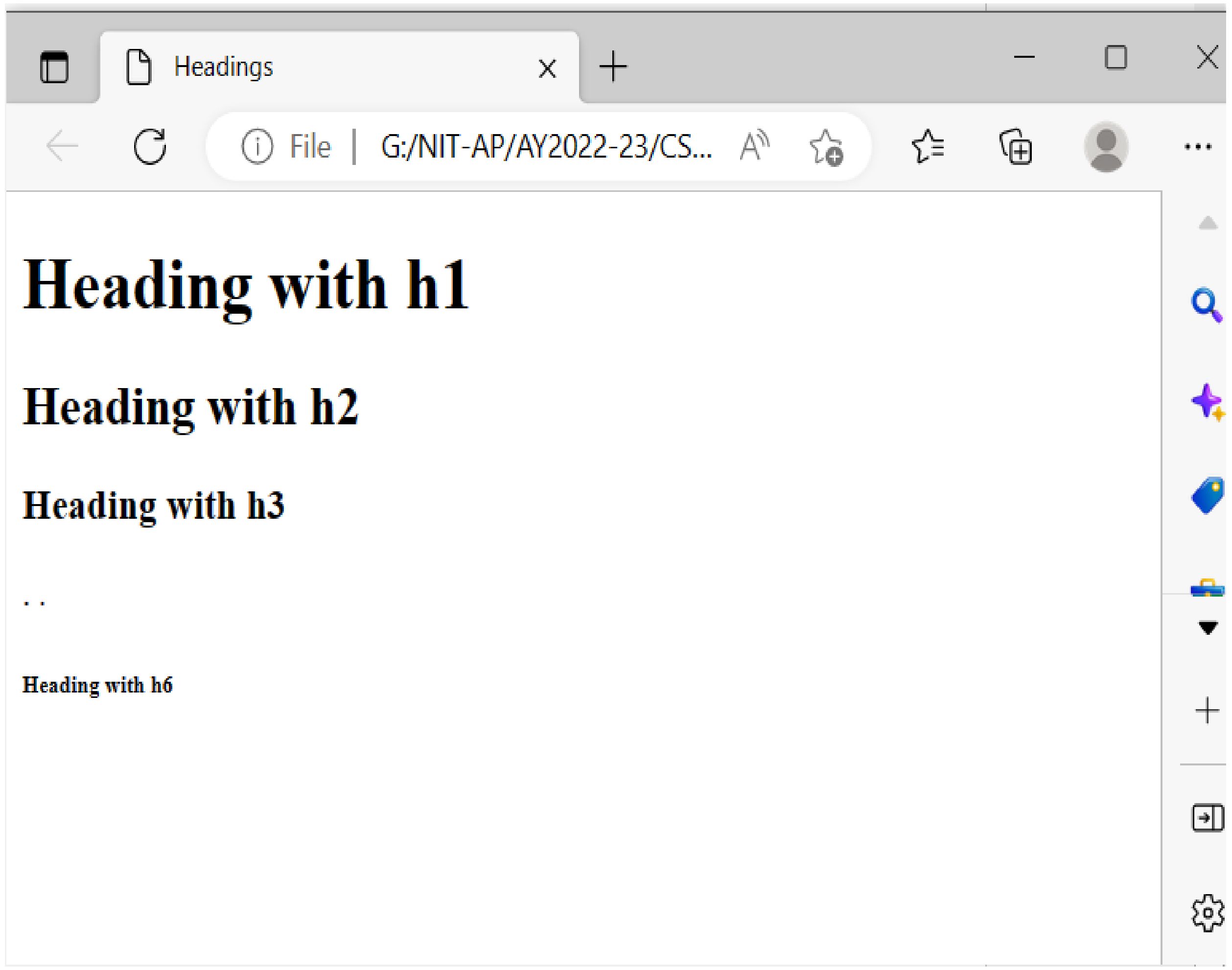
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Name</title>
    </head>
    <body>
        Hello, Everyone!!!!
    </body>
</html>
```



Headings

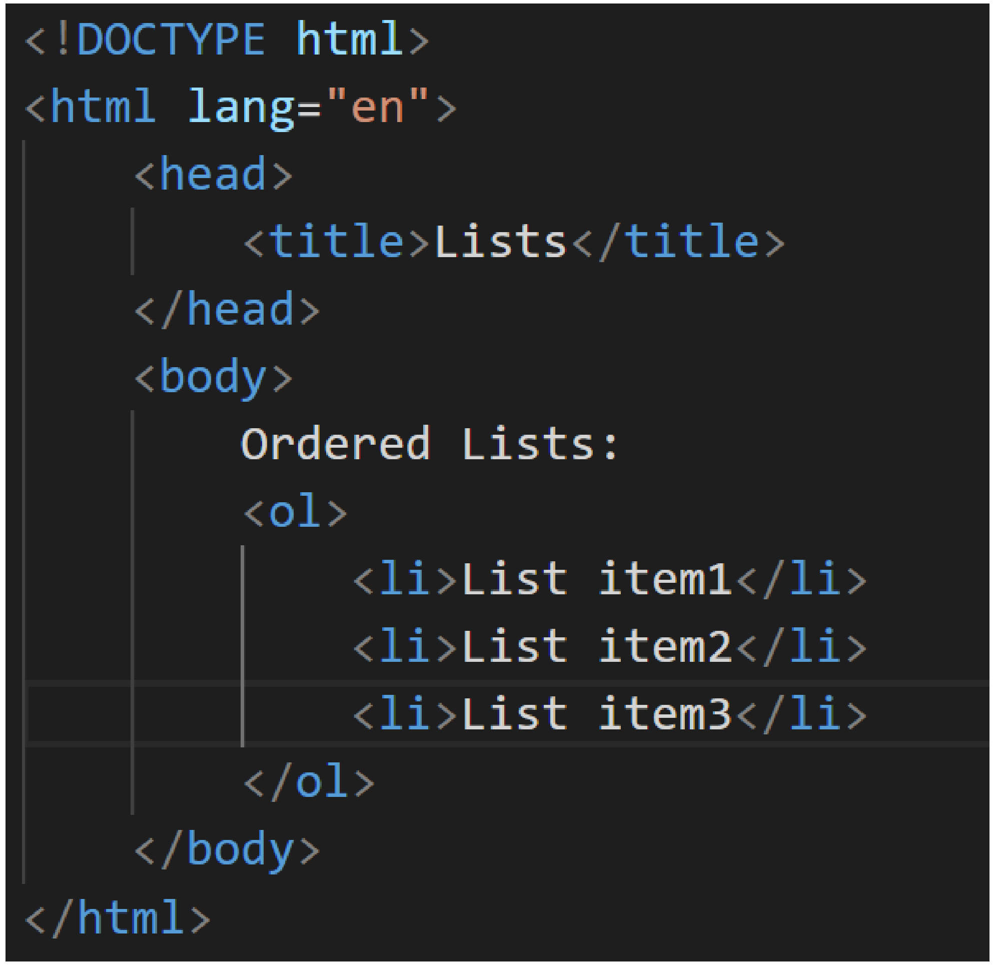


```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Headings</title>
    </head>
    <body>
        <h1>Heading with h1</h1>
        <h2>Heading with h2</h2>
        <h3>Heading with h3</h3>
        .
        .
        <h6>Heading with h6</h6>
    </body>
</html>
```

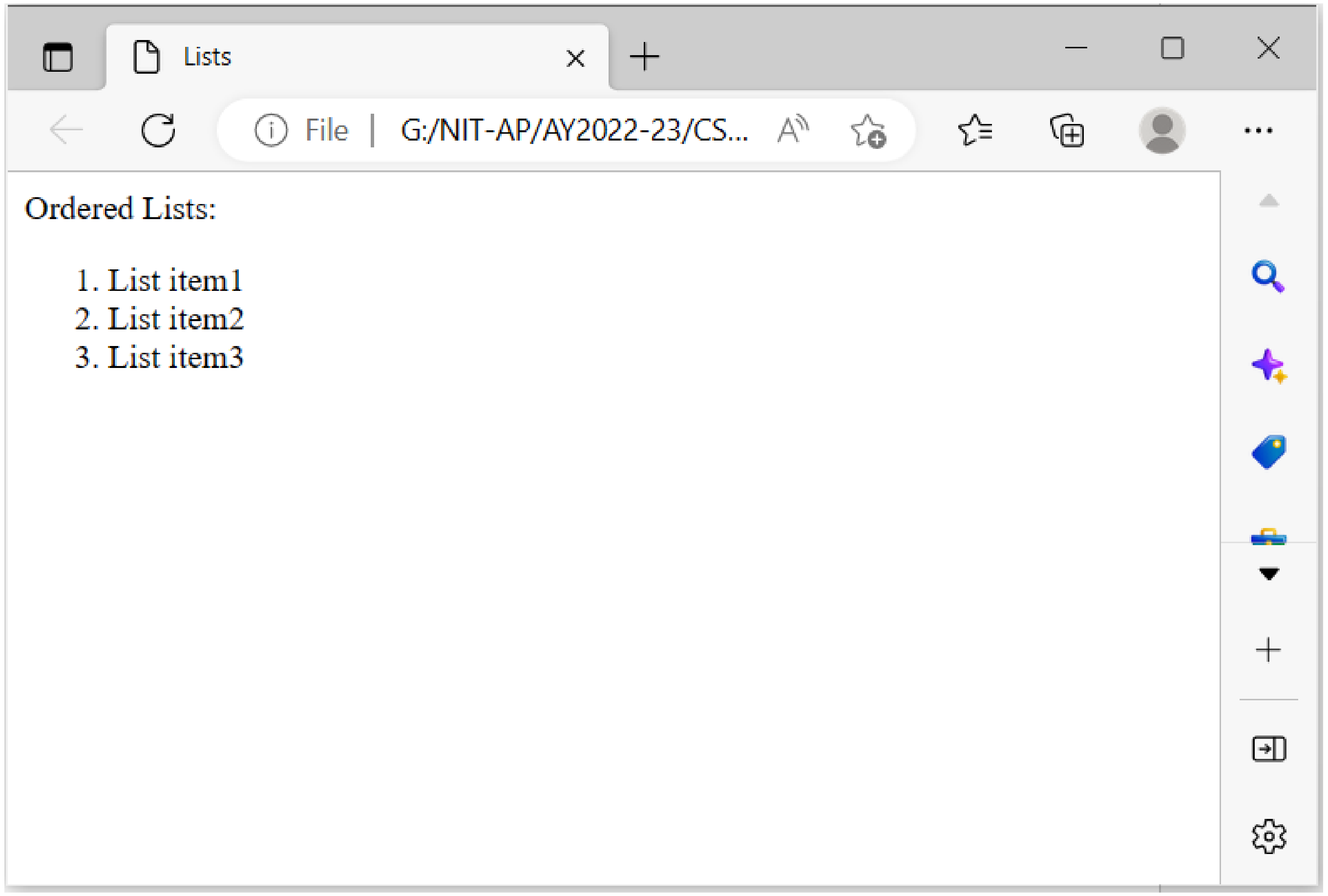


```
Heading with h1

Heading with h2

Heading with h3

. .

Heading with h6
```

Lists: Ordered List

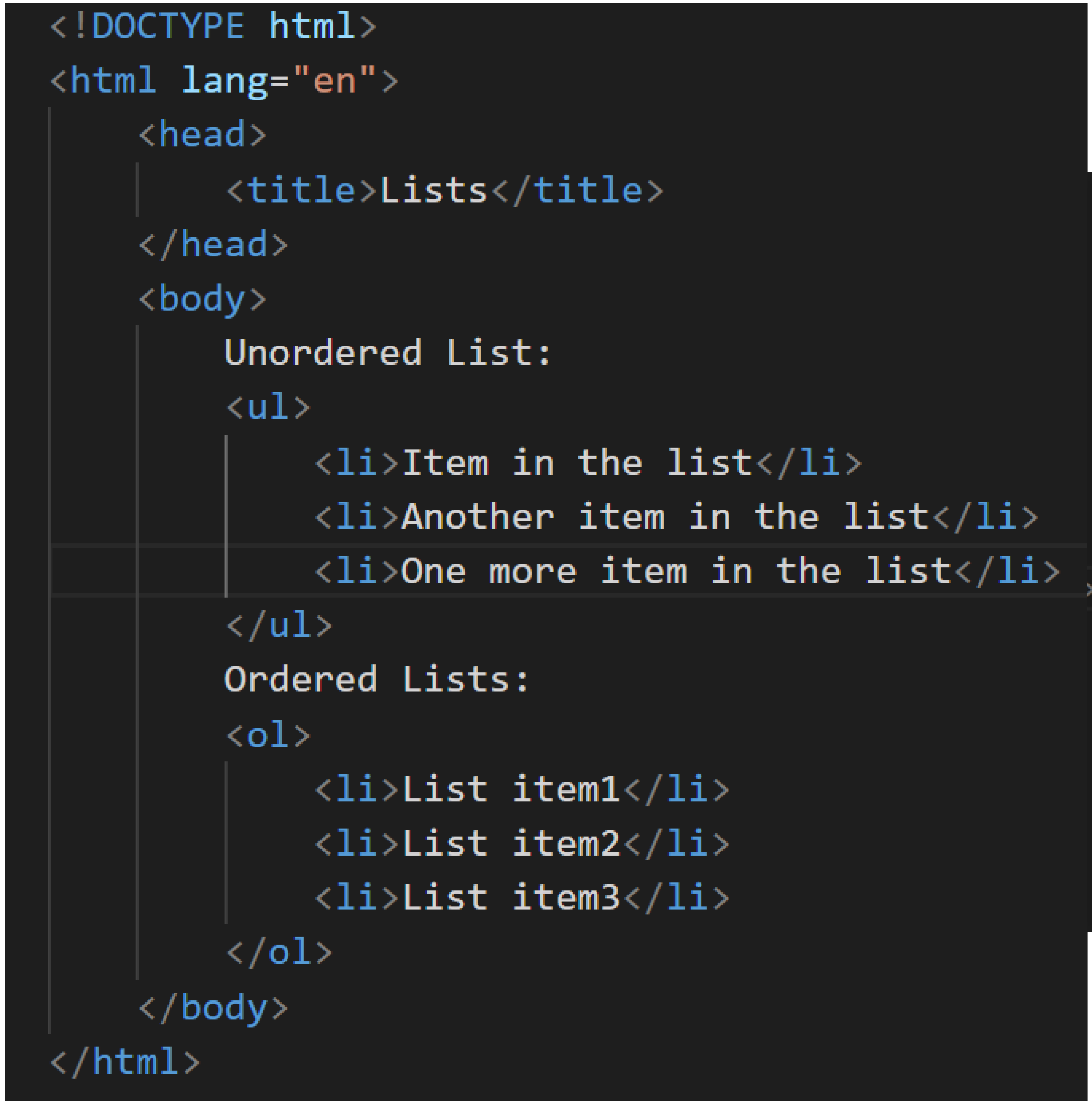


```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Lists</title>
    </head>
    <body>
        Ordered Lists:
        <ol>
            <li>List item1</li>
            <li>List item2</li>
            <li>List item3</li>
        </ol>
    </body>
</html>
```

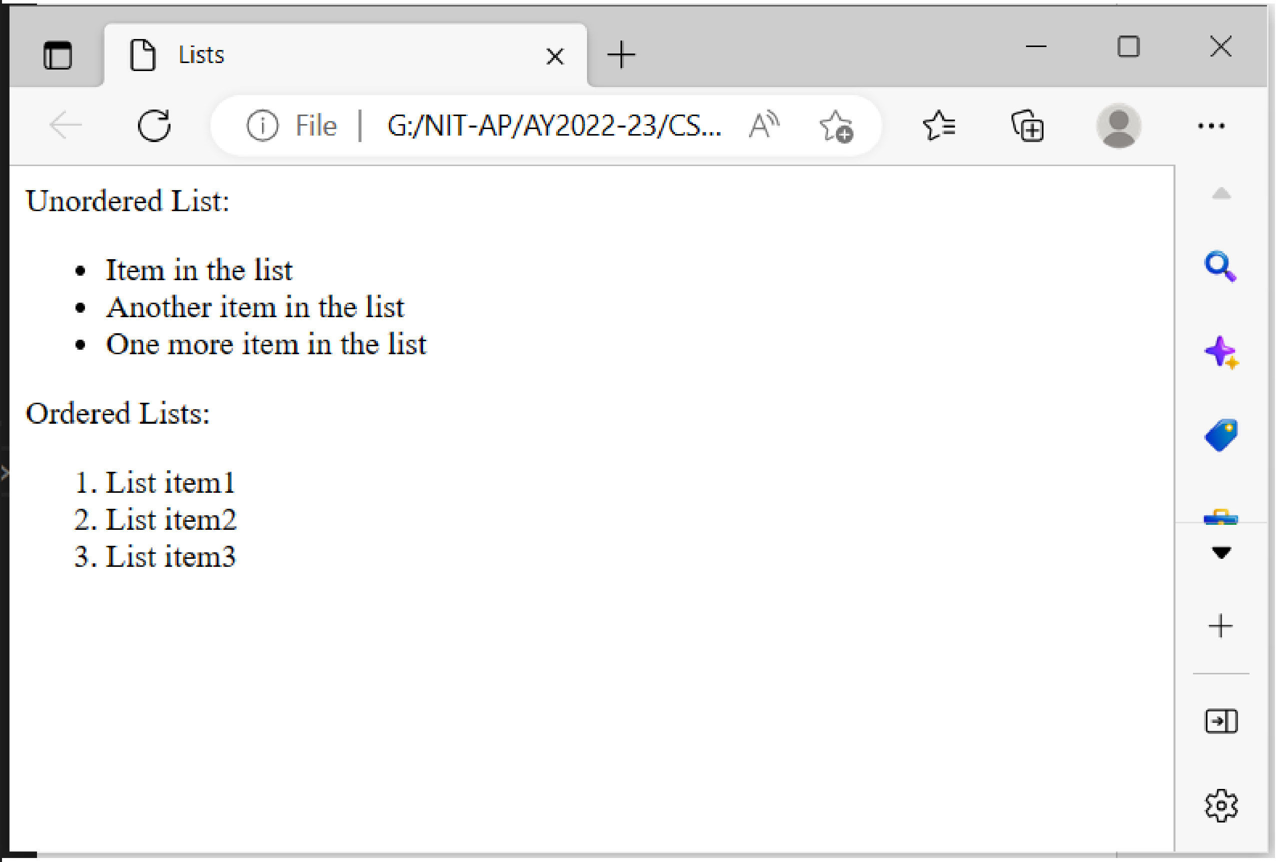


```
Ordered Lists:

    1. List item1
    2. List item2
    3. List item3
```

Lists: UnOrdered List

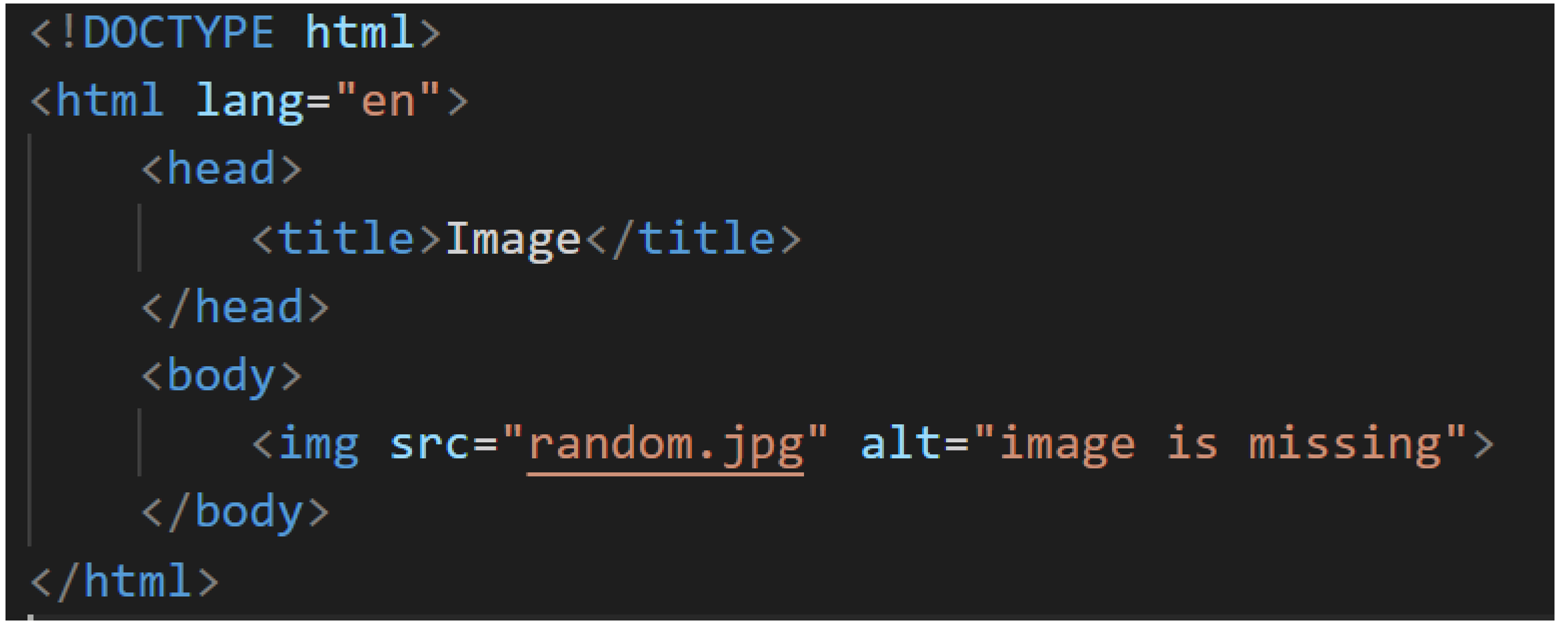


```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Lists</title>
    </head>
    <body>
        Unordered List:
        <ul>
            <li>Item in the list</li>
            <li>Another item in the list</li>
            <li>One more item in the list</li>
        </ul>
        Ordered Lists:
        <ol>
            <li>List item1</li>
            <li>List item2</li>
            <li>List item3</li>
        </ol>
    </body>
</html>
```



```
Unordered List:

  • Item in the list
  • Another item in the list
  • One more item in the list

Ordered Lists:

  1. List item1
  2. List item2
  3. List item3
```

Images



```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Image</title>
    </head>
    <body>
        <img src="random.jpg" alt="image is missing">
    </body>
</html>
```

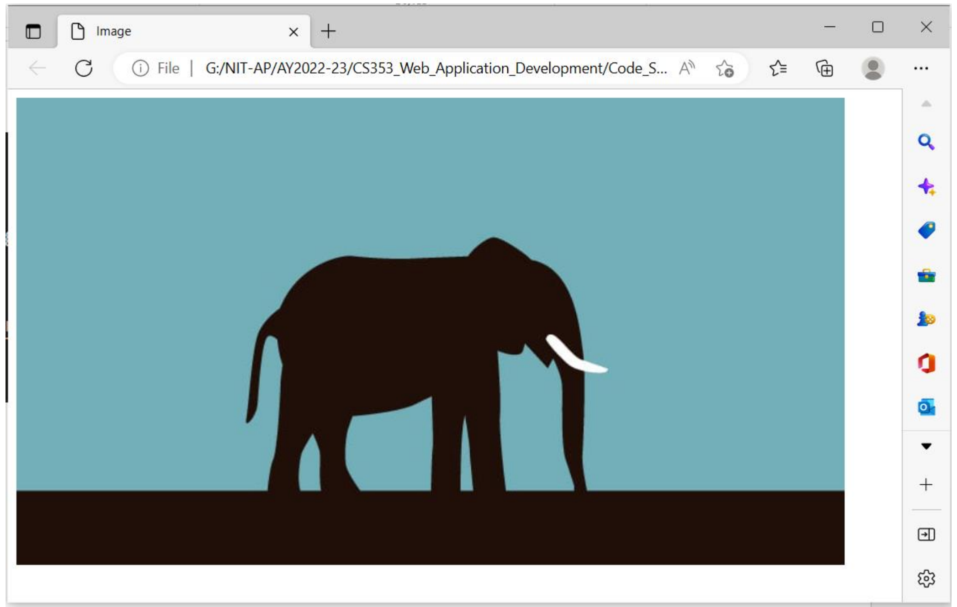
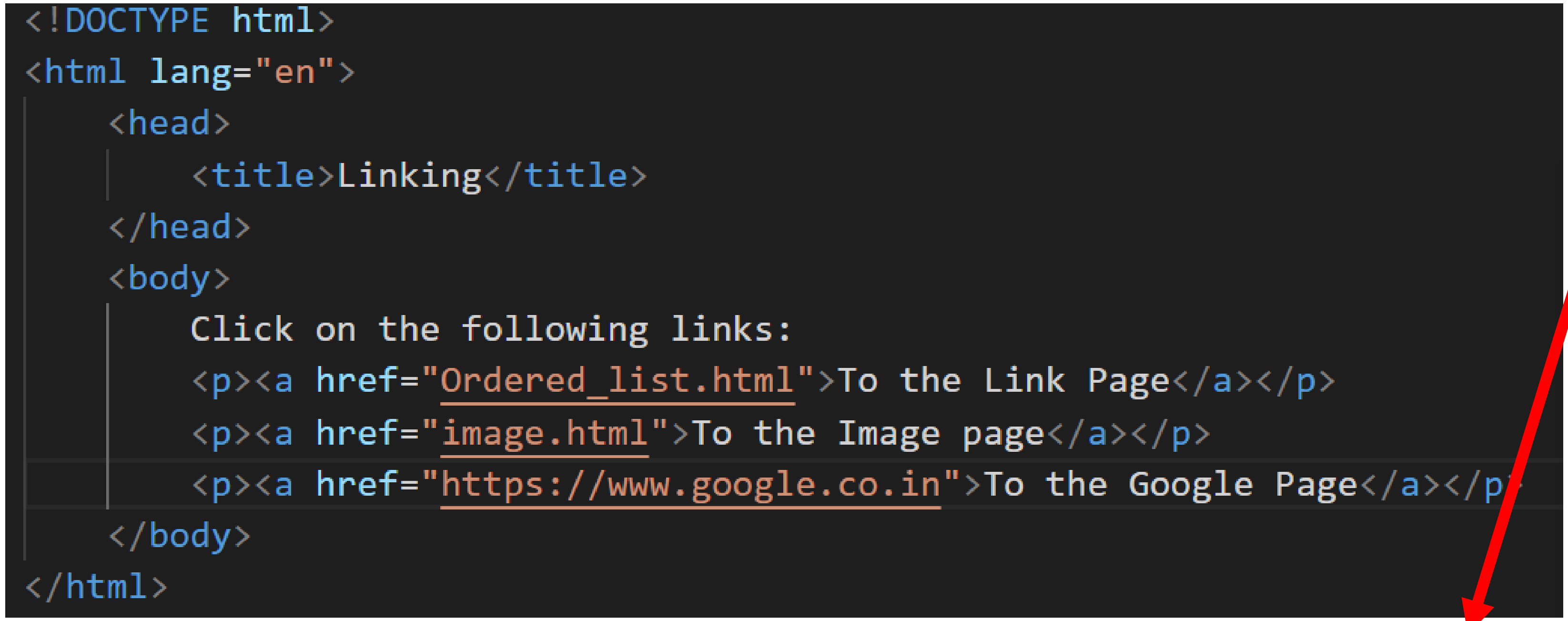


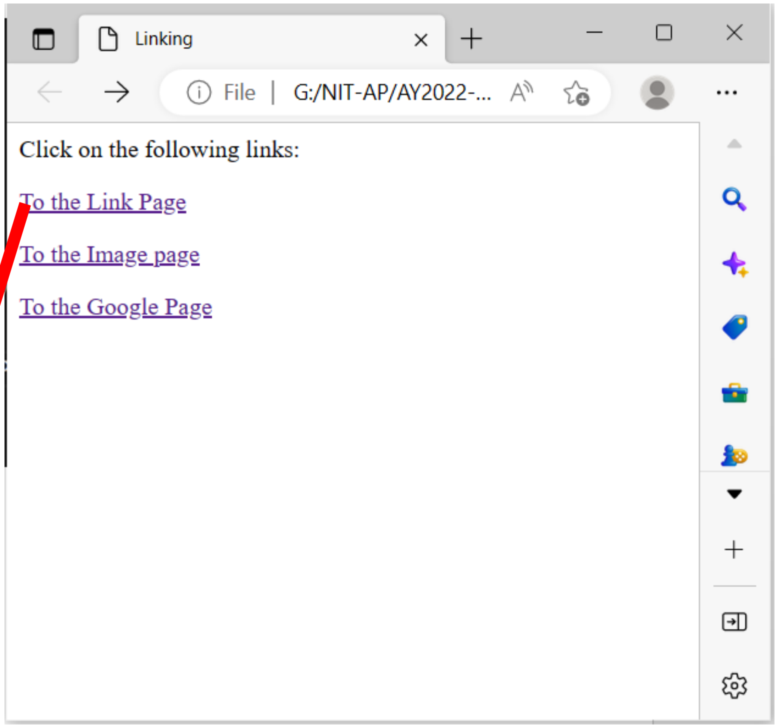
Image
File | G:/NIT-AP/AY2022-23/CS353_Web_Application_Development/Code_S...

Other attributes to <img>:
Width = “120”
Height = “200”

Linking



```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Linking</title>
    </head>
    <body>
        Click on the following links:
        <p><a href="Ordered_list.html">To the Link Page</a></p>
        <p><a href="image.html">To the Image page</a></p>
        <p><a href="https://www.google.co.in">To the Google Page</a></p
    </body>
</html>
```

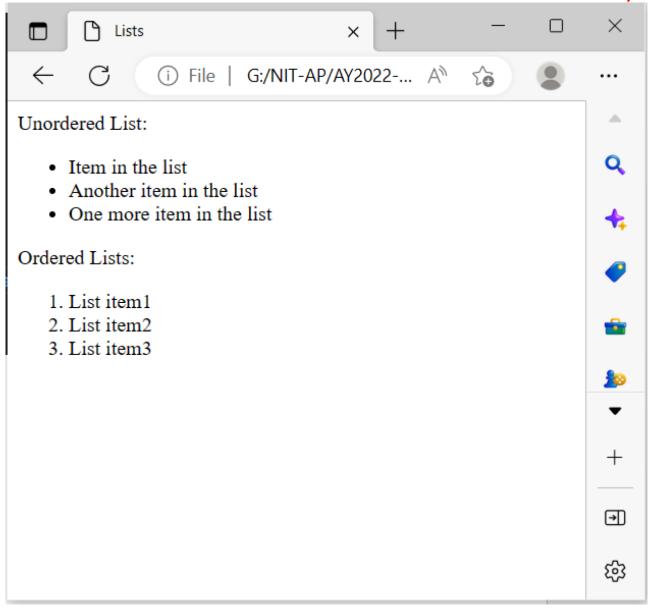


Click on the following links:

To the Link Page

To the Image page

To the Google Page



Unordered List:

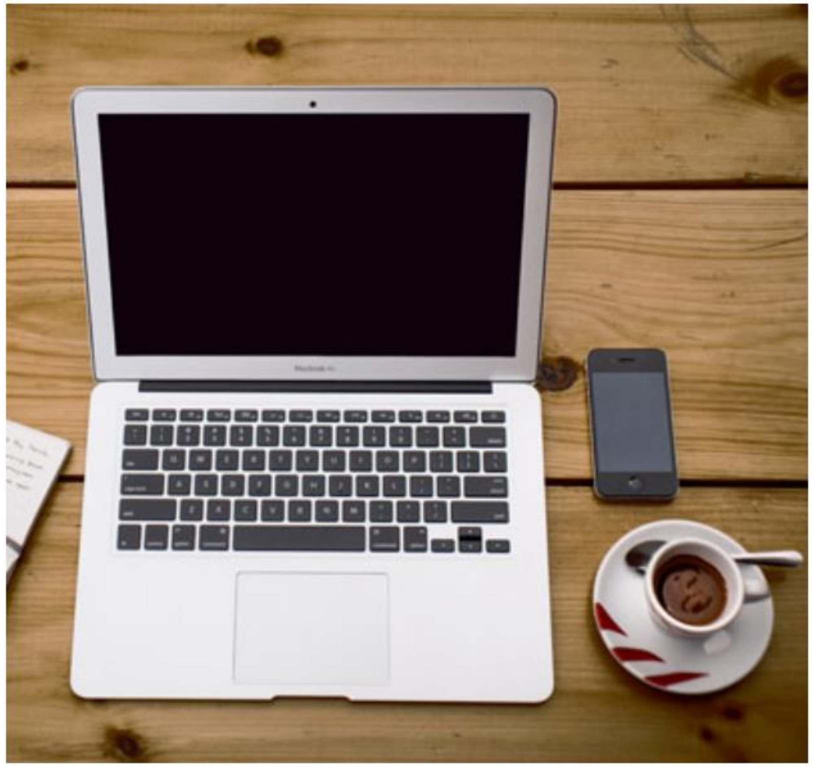
- Item in the list
- Another item in the list
- One more item in the list

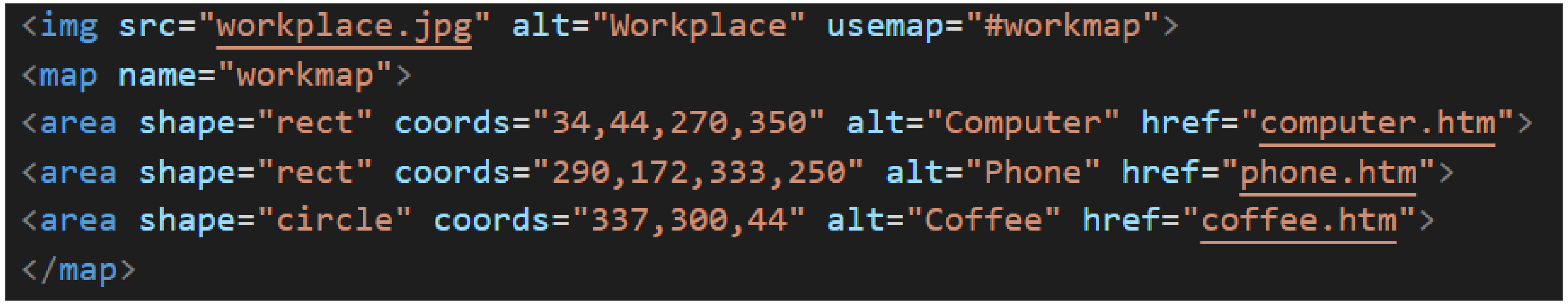
Ordered Lists:

1. List item1
2. List item2
3. List item3

Image Maps

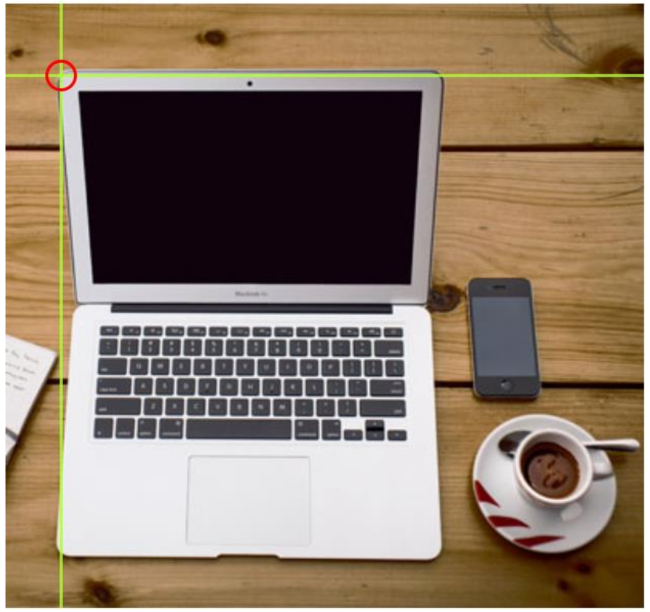
- Image map is an image with *clickable area.*
- *<map>*
- *<area>*

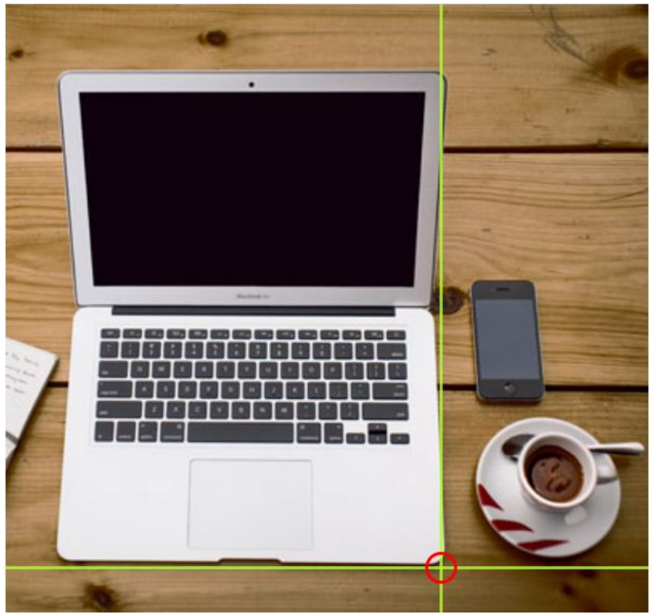


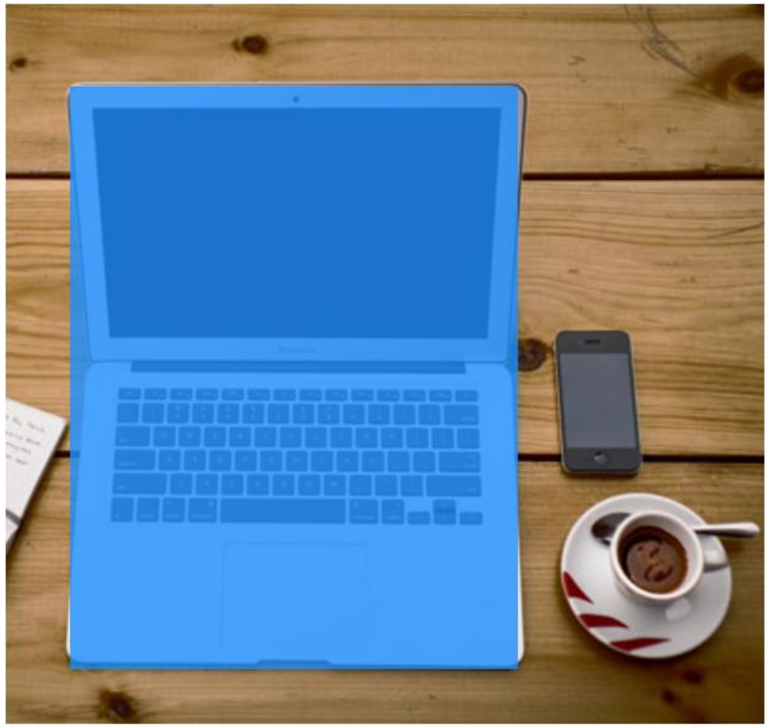


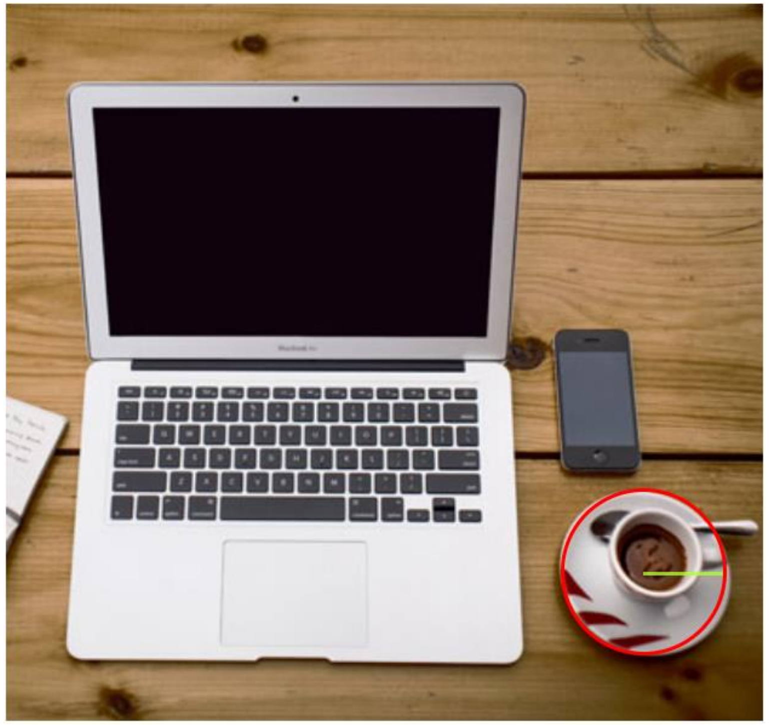
```
<img src="workplace.jpg" alt="Workplace" usemap="#workmap">
<map name="workmap">
<area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">
<area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">
<area shape="circle" coords="337,300,44" alt="Coffee" href="coffee.htm">
</map>
```

Create Image Map









Areas:

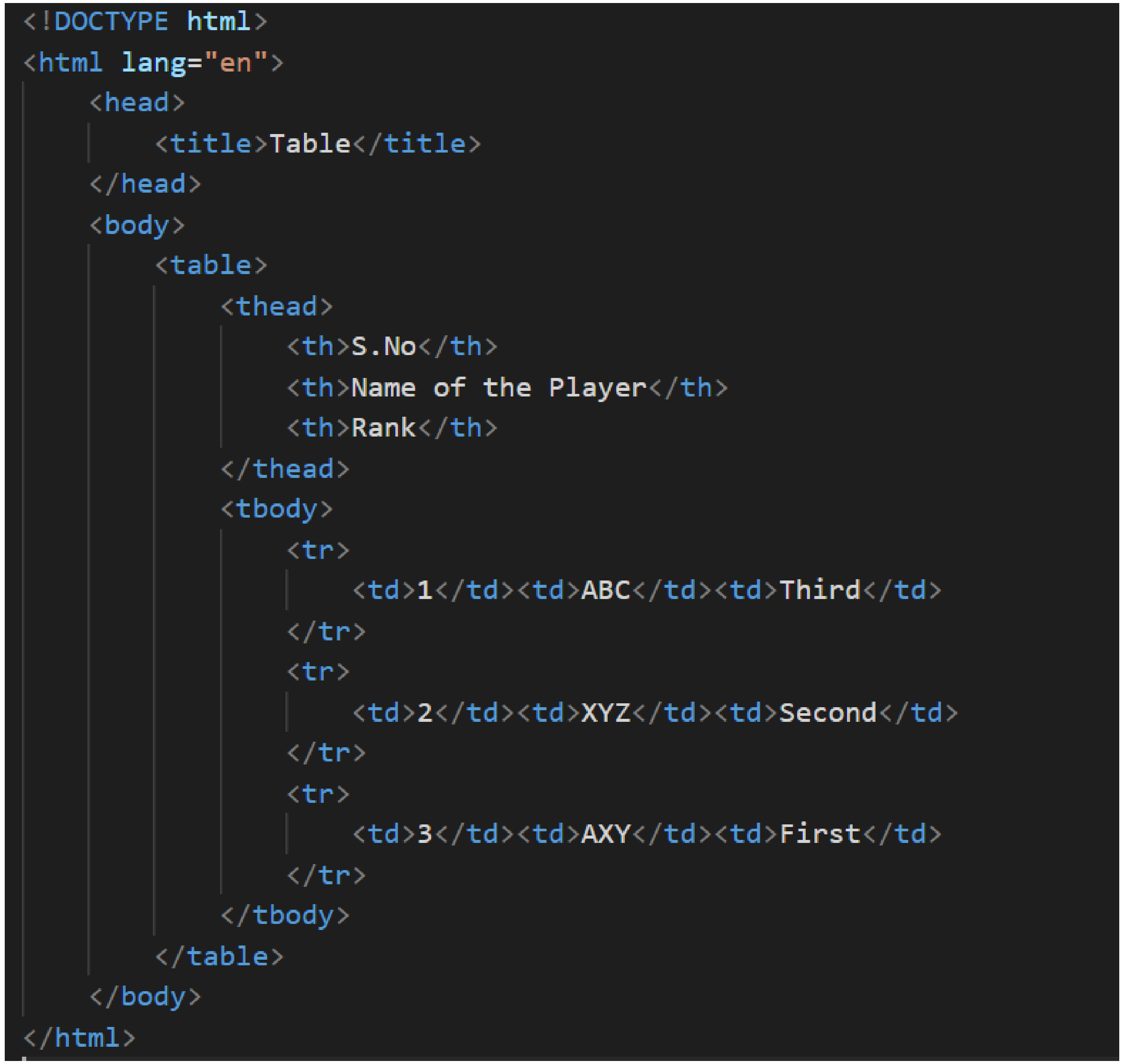
rect

circle

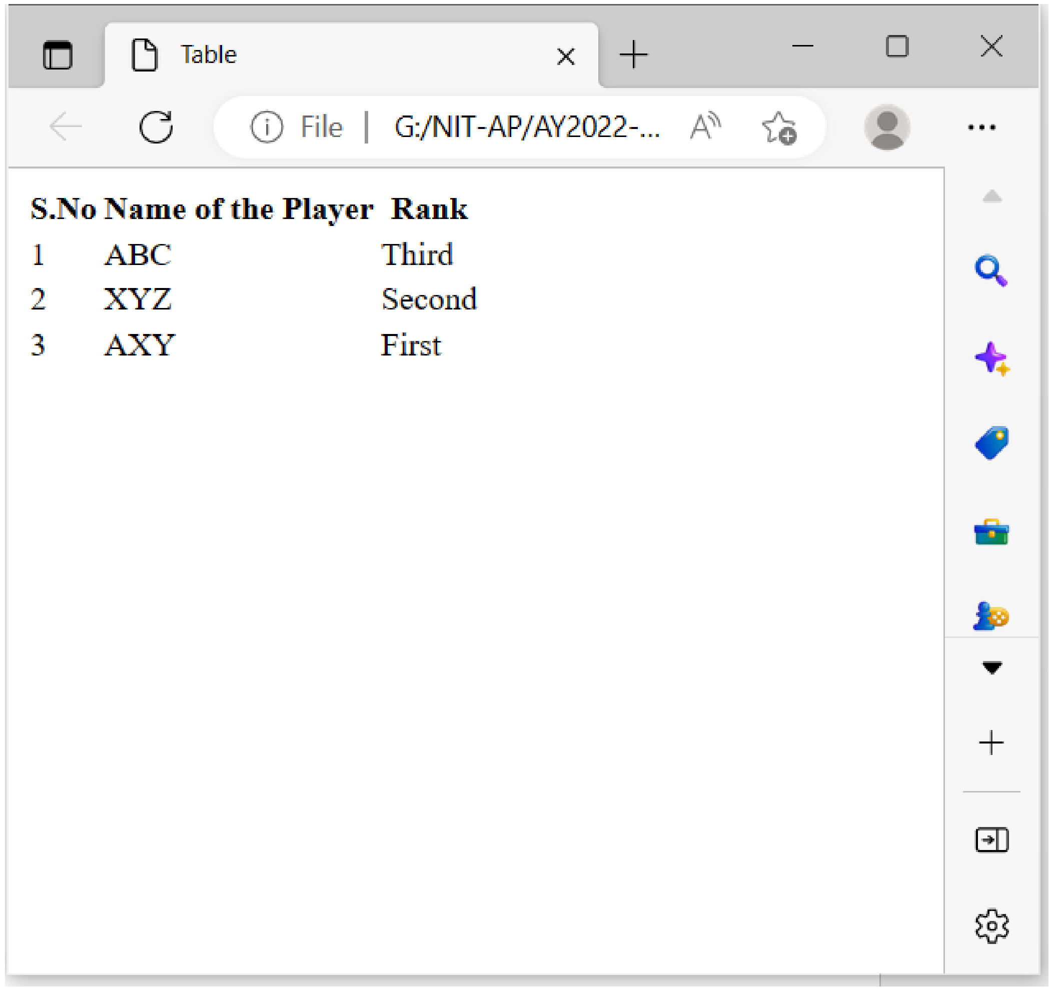
poly

default

Table

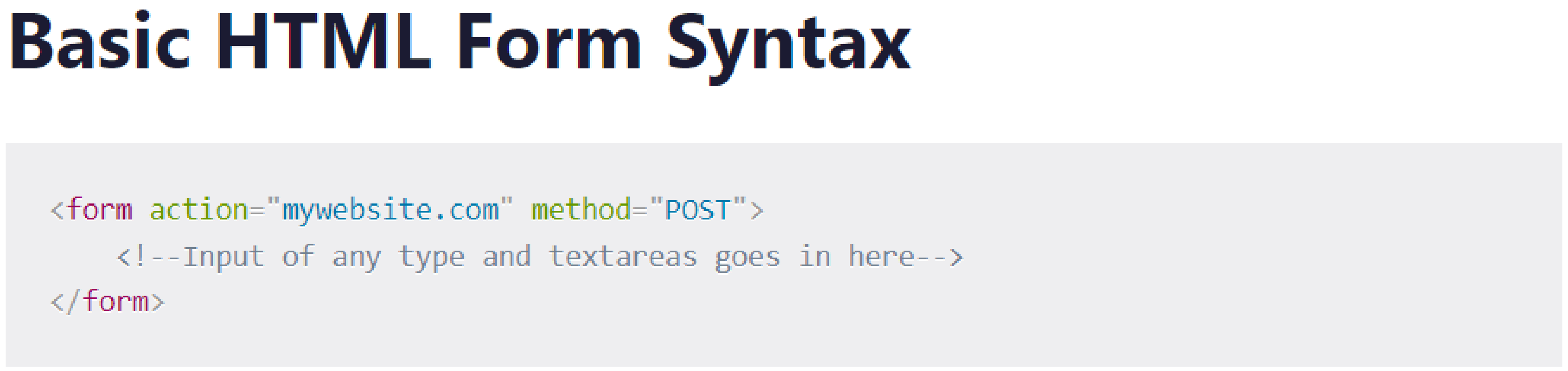


```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Table</title>
    </head>
    <body>
        <table>
            <thead>
                <th>S.No</th>
                <th>Name of the Player</th>
                <th>Rank</th>
            </thead>
            <tbody>
                <tr>
                    <td>1</td><td>ABC</td><td>Third</td>
                </tr>
                <tr>
                    <td>2</td><td>XYZ</td><td>Second</td>
                </tr>
                <tr>
                    <td>3</td><td>AXY</td><td>First</td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```



S.No	Name of the Player	Rank
1	ABC	Third
2	XYZ	Second
3	AXY	First

Form



Basic HTML Form Syntax

```
<form action="mywebsite.com" method="POST">
    <!--Input of any type and textareas goes in here-->
</form>
```

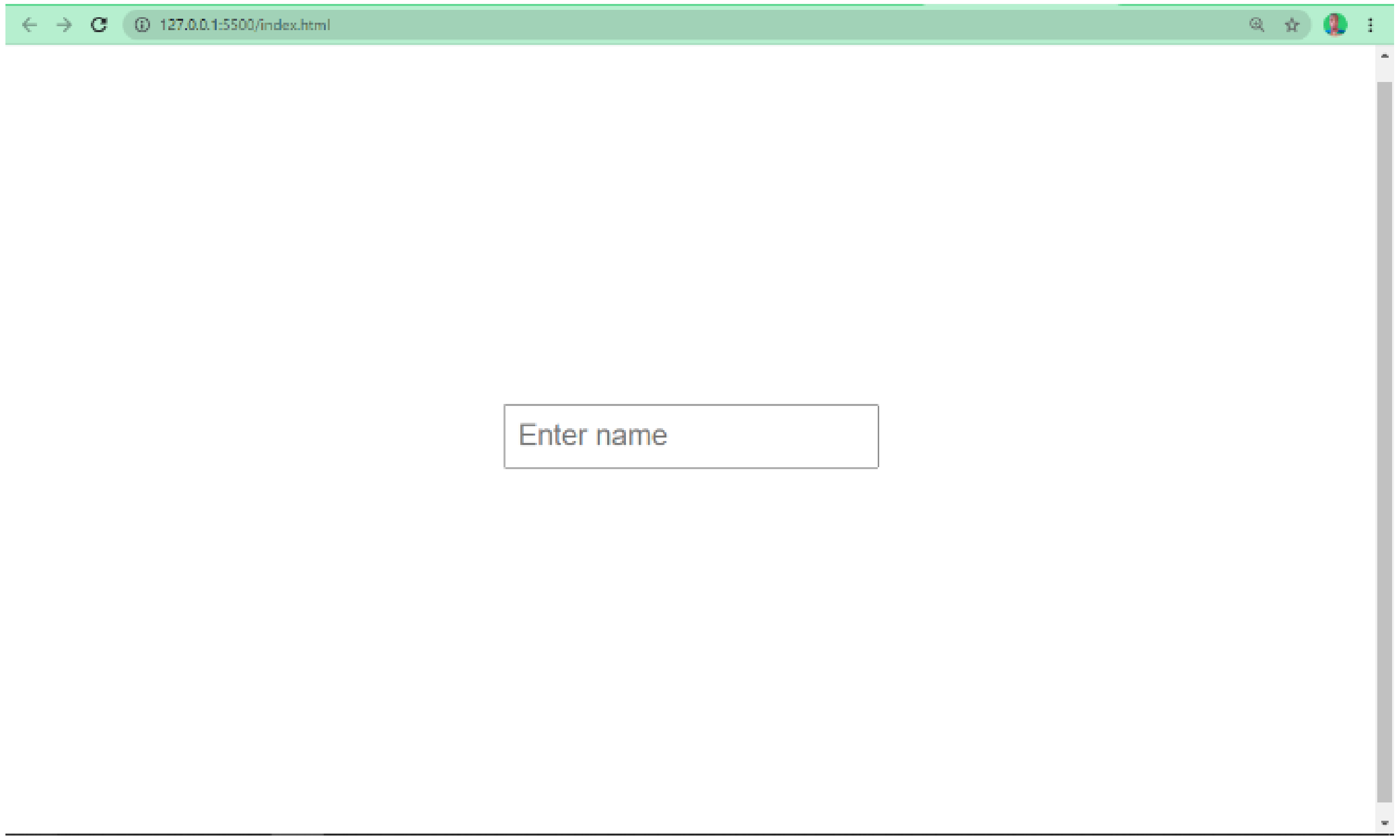
You may Use <input> tag to create various form controls in HTML

Input is an inline tag and takes attributes such as:

- type
- name
- minlength
- maxlength
- placeholder

HTML Form Input Types

```
<input type="text" placeholder="Enter name" />
```

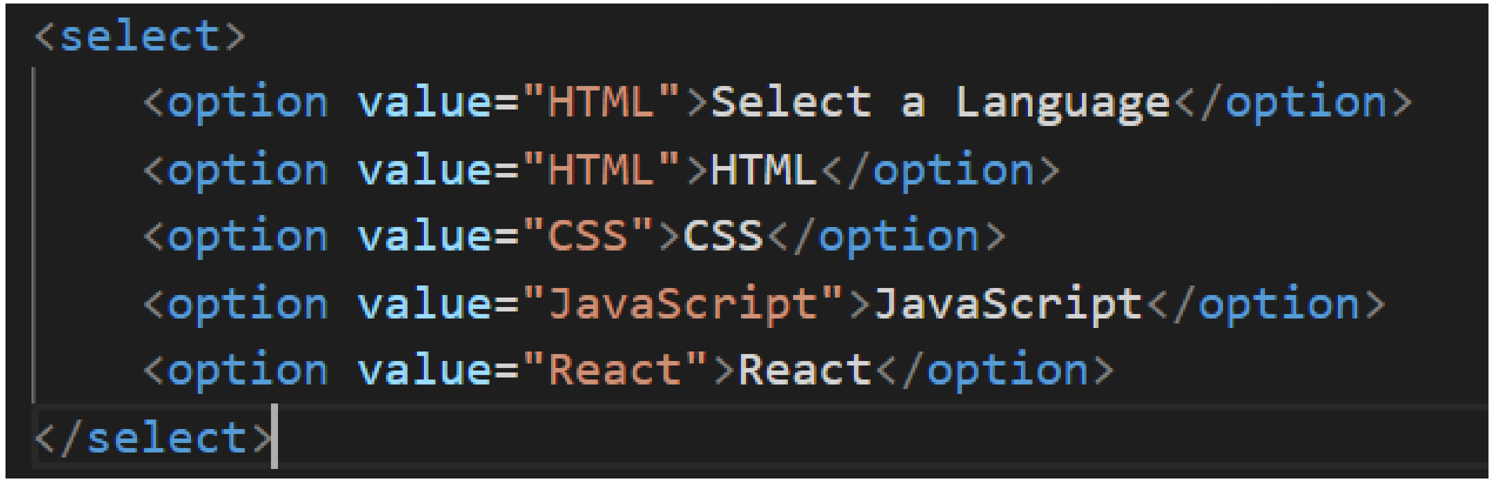

127.0.0.1:5500/index.html

Enter name

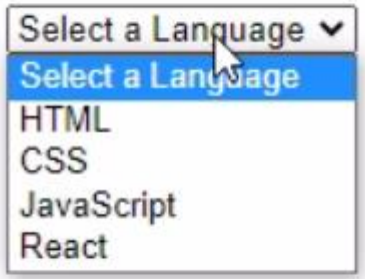
Other Input Types

type=“password”
type=“email”
type=“number”
type=“radio”
type=“checkbox”
type=“submit”
type=“button”
type=“file”
type=“color”
type=“search”
type=“url”
type=“date”
type=“datetime-local”
type=“textarea”

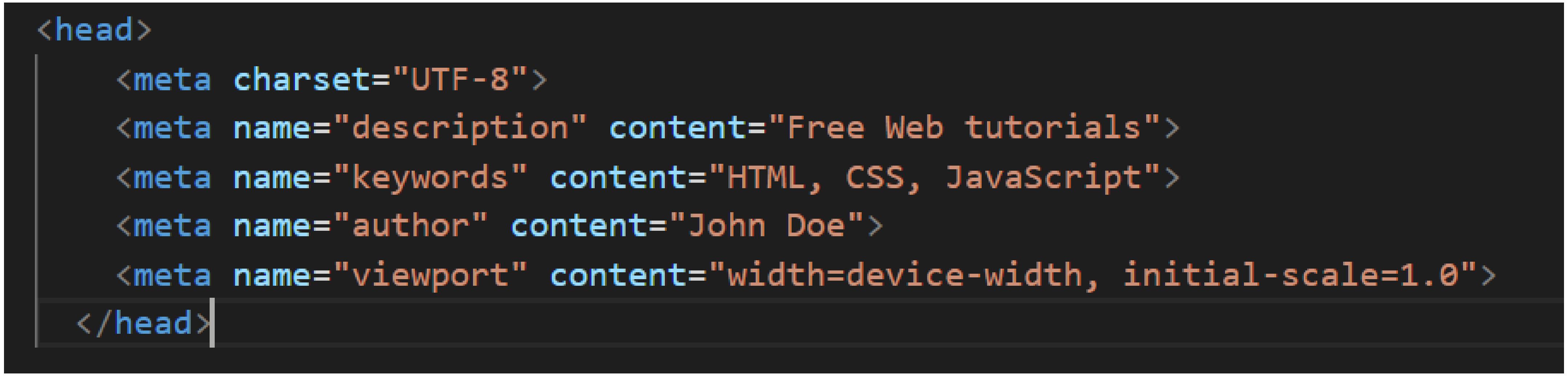
Multiple Select Box



```
<select>
    <option value="HTML">Select a Language</option>
    <option value="HTML">HTML</option>
    <option value="CSS">CSS</option>
    <option value="JavaScript">JavaScript</option>
    <option value="React">React</option>
</select>
```

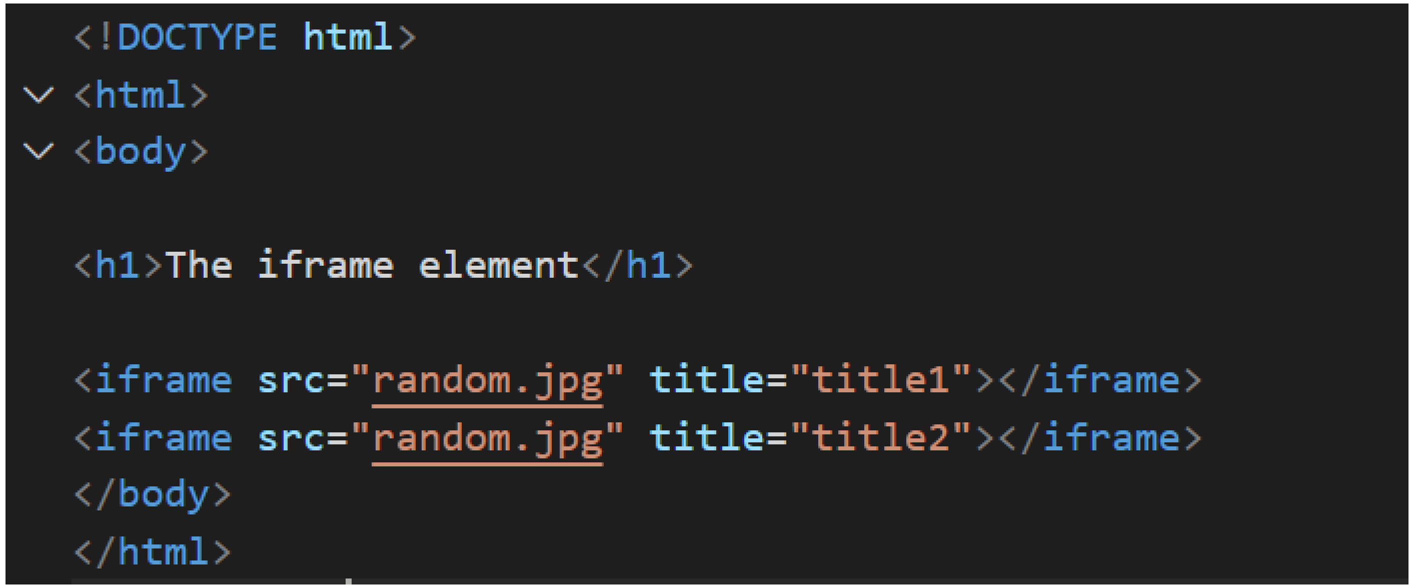


Meta

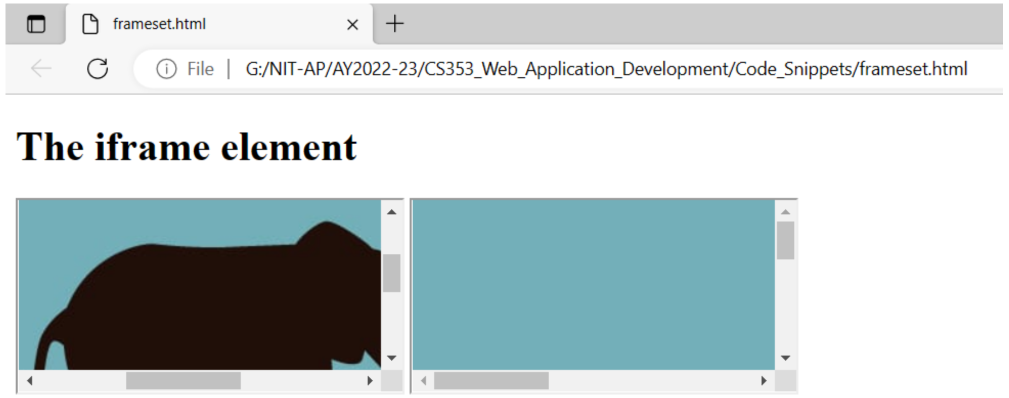


```html
<head>
    <meta charset="UTF-8">
    <meta name="description" content="Free Web tutorials">
    <meta name="keywords" content="HTML, CSS, JavaScript">
    <meta name="author" content="John Doe">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
```

Frameset



```
<!DOCTYPE html>
<html>
<body>

<h1>The iframe element</h1>

<iframe src="random.jpg" title="title1"></iframe>
<iframe src="random.jpg" title="title2"></iframe>
</body>
</html>
```


frameset.html
File | G:/NIT-AP/AY2022-23/CS353_Web_Application_Development/Code_Snippets/frameset.html

The iframe element

HTML Evolution

Influenced by browser implementation quirks

What to do if you see "`<p>`Some text" (missing closing `</p>`)?

1. Complain bitterly about malformed HTML.
2. Figure out there was a missing `</p>`, add it, and continue processing.

Forked into HTML and XHTML (XML-based HTML)

XHTML is more strict about adhering to proper syntax

Users came to depend on browser quirks, so browsers couldn't change

Example XHTML document

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

Basic Syntax rules for XHTML

Document: hierarchical collection of **elements**, starting with `<html>`

- Element: start tag, contents, end tag
- Elements may be nested
- Every element must have an explicit start and end
    - Can use <foo /> as shorthand for <foo></foo>
- Start tags can contain **attributes**:

```
<img src="face.jpg">
<input type="text" value="94301" name="zip">
<div class="header">
```

Need to handle markup characters in content

To display a literal < or > in a document, use entities:

`&lt;`	Displays <
`&gt;`	Displays >
`&amp;`	Displays &
`&quot;`	Displays "
`&nbsp;`	Nonbreaking space (won't insert a line break at this space)

Many other entities are defined for special characters.

Whitespace is not significant except in a few cases (e.g. `textarea`, `pre` tags)

Newer HTML - HTML5

- Additions tags to allow content definition
  - `<article>, <section>, <header>, <footer>, <summary>, <aside>, <details>`
  - `<mark>, <figcaption>, <figure>`
  - `<nav>, <menuitem>`
- Drawing
  - `<svg>` – Scalable Vector Graphics - Draw shapes
  - `<canvas>` - Draw from JavaScript - 3D with WebGL
- Timed media playback: `<video>` and `<audio>`

Cascading Style Sheet (CSS)

Driving problem behind CSS

What font type and size does `<h1>Introduction</h1>` generate?

Answer: Some default from the browser (HTML tells **what** browser **how**)

Early HTML - Override defaults with attributes

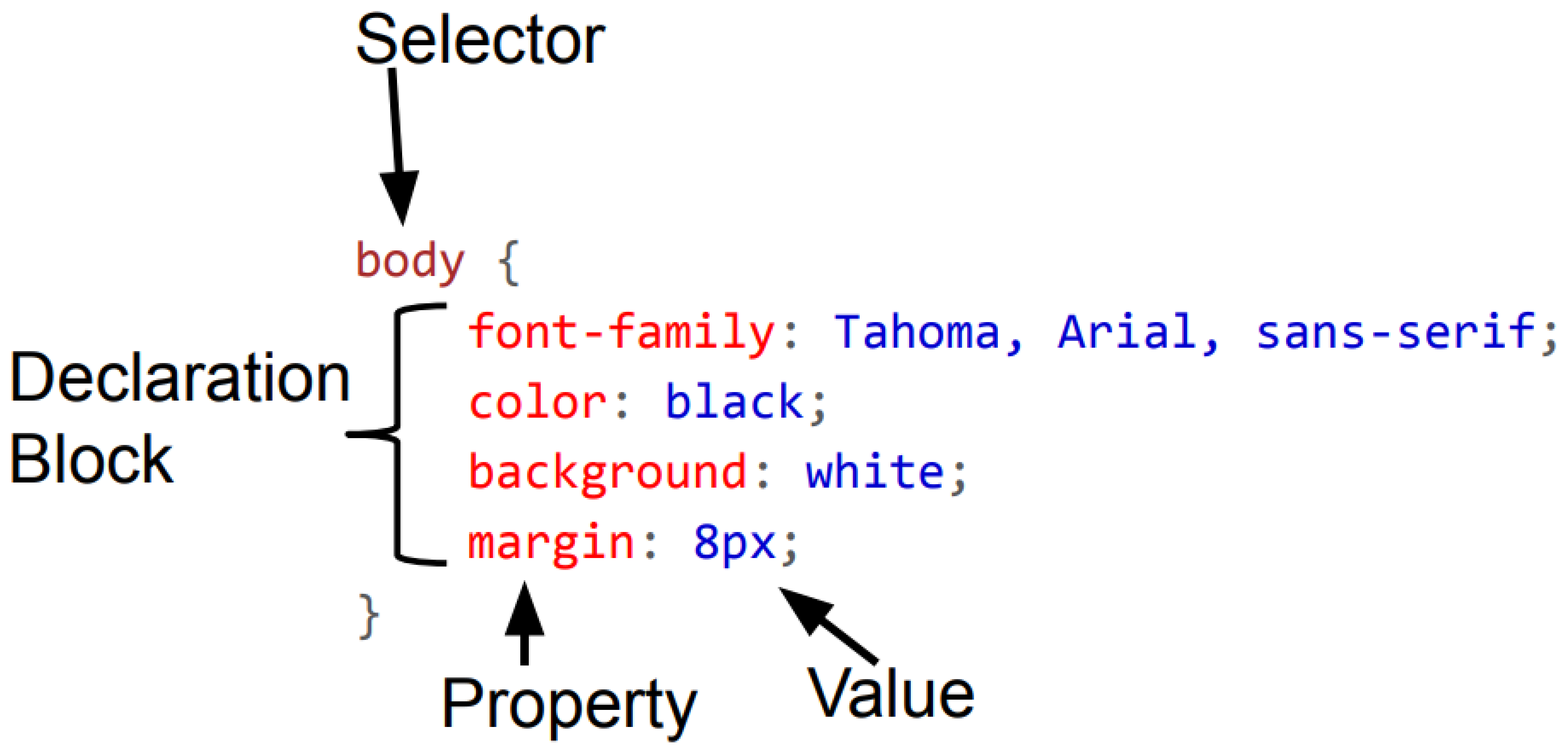
```
<table border="2" bordercolor="black">
```

Style sheets were added to address this:

Specify style to use rather than browser default

Not have to code styling on every element

Style sheet contain one or more **CSS Rules**



```
body {
    font-family: Tahoma, Arial, sans-serif;
    color: black;
    background: white;
    margin: 8px;
}
```

CSS Selector	CSS	HTML

Tag name	`h1 {`
	`  color: red;`
	`}`
		`<h1>Today’s Specials</h1>`

Class attribute	`.large {`
	`  font-size: 16pt;`
	`}`
		`<p class="large">...`

Tag and Class	`p.large {...}`	`<p class="large">...`

Element id	`#p20 {`
	`  font-weight: bold;`
	`}`
		`<p id="p20">...`

CSS Pseudo Selectors

hover - Apply rule when mouse is over element (e.g. tooltip)

```
p:hover, a:hover {
  background-color: yellow;
}
```

`a:link, a:visited` - Apply rule when link has been visited or not visited (`link`)

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

```
a:link {
  color: blue;
}
```

CSS Properties

Control many style properties of an element:

- Coloring
- Size
- Position
- Visibility
- Many more: (e.g. `p: { text-decoration: line-through; }`)

- Also used in animation

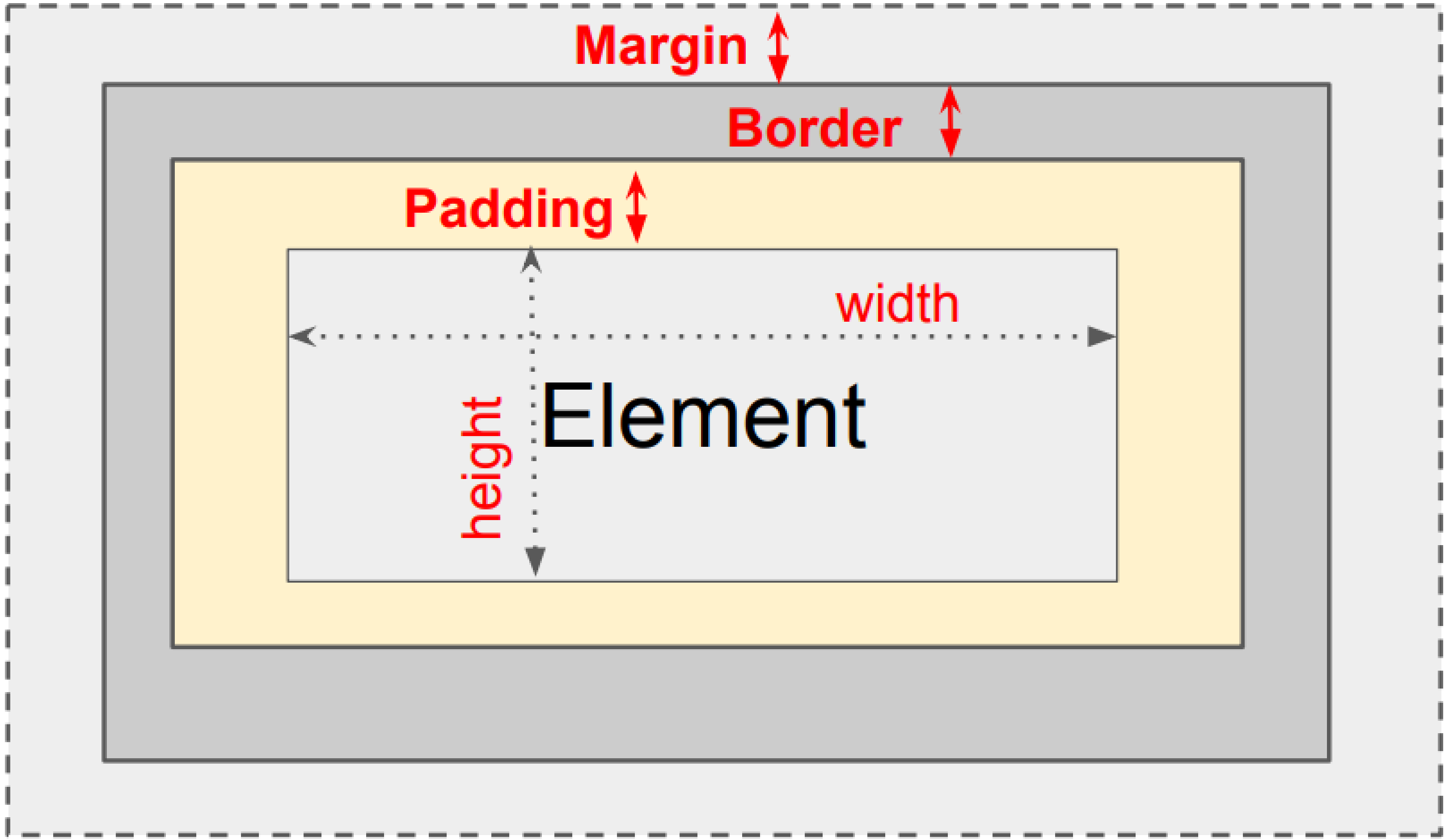
Color - Properties: `color` & `background_color`

Must ultimately turn into red, green, and blue intensities between 0 and 255:

- Predefined names: `red`, `blue`, `green`, `white`, etc. (140 standard names)
- 8-bit hexadecimal numbers for red, green, blue: $\texttt{\#}\underbrace{\texttt{ff}}_{R}\underbrace{\texttt{00}}_{G}\underbrace{\texttt{00}}_{B} \longrightarrow$ 🟥
- 0-255 decimal intensities: $\texttt{rgb(}\underbrace{\texttt{255}}_{R}\texttt{,}\underbrace{\texttt{255}}_{G}\texttt{,}\underbrace{\texttt{0}}_{B}\texttt{)} \longrightarrow$ 🟨
- Percentage intensities: $\texttt{rgb(}\underbrace{\texttt{80\%}}_{R}\texttt{,}\underbrace{\texttt{80\%}}_{G}\texttt{,}\underbrace{\texttt{100\%}}_{B}\texttt{)} \longrightarrow$ 🟪

Example: `h1: { color: red; }`

CSS Box Model



Total element width =
width +
left padding +
right padding +
left border +
right border +
left margin +
right margin

Margin & Padding
Transparent

CSS distance units

Absolute	
2px	pixels
1mm	millimeters
2cm	centimeters
0.2in	inches
3pt	printer point 1/72 inch
Relative	
2em	2 times the element’s current font size
3rem	3 times the root element’s current font size

Size Properties - Element, pad, margin, border

width - Override element defaults
height

padding-top
padding-right
padding-bottom
padding-left

margin-top
margin-right
margin-bottom
margin-left

```
border-bottom-color
border-bottom-style
border-bottom-width
border-left-color
border-left-style
border-left-width
border-right-color
border-right-style
border-right-width
```
etc.

```
p {
  border: 5px solid red;
}
```

`position` property

`position: static;`	(default) - Position in document flow
`position: relative;`	Position relative to default position via `top`, `right`, `bottom`, and `left` properties
`position: fixed;`	Position to a fixed location on the screen via `top`, `right`, `bottom`, and `left` properties
`position: absolute;`	Position relative to ancestor `absolute` element via `top`, `right`, `bottom`, and `left` properties

Fixed position (0,0) is `top left` corner

Some more common properties

`background-image`: image for element's background

`background-repeat`: should background image be displayed in a repeating pattern (versus once only)

`font`, `font-family`, `font-size`, `font-weight`, `font-style`: font information for text

`text-align`, `vertical-align`: `Alignment`: `center`, `left`, `right`

`cursor` - Set the cursor when over element (e.g. `help`)

Element visibility control properties

`display: none;` - Element is not displayed and takes no space in layout.

`display: inline;` - Element is treated as an inline element.
`display: block;` - Element is treated as a block element.

`display: flex;` - Element is treated as a flex container.
`display: grid;` - Element is treated as a grid container.

`visibility: hidden;` - Element is hidden but space still allocated.
`visibility: visible;` - Element is normally displayed

Flexbox and Grid layout

- `display: flex;` (Flexbox)
- `display: grid;` (Grid) newer layout method
    - Items flex to fill additional space and shrink to fit into smaller spaces.
    - Useful for web app layout:
        - Divide up the available space equally among a bunch of elements
        - Align of different sizes easily
        - Key to handling different window and display sizes
- Flexbox - Layout one dimension (row or column) of elements
- Grid - Layout in two dimensions (rows and columns) of elements

Some other CSS issues

- Inheritance
  - Some properties (e.g. `font-size`) are inherited from parent elements
  - Others (`border`, `background`) are not inherited.
- Multiple rule matches
  - General idea: most specific rule wins

```
<span>Text1</span>                 span.test { color: green }
<span class="test">Text2</span>  span { color: red }
```

Adding Styles to HTML

Separate style sheet (best way)

```
<head>
  <link rel="stylesheet" type="text/css" href="myStyles.css" />
  <style type="text/css">
    body {
        font-family: Tahoma, Arial, sans-serif;
    }
  </style>
</head>
<body>
  <div style="padding:2px; ... ">
</body>
```

Page-specific styles

Element-specific styles

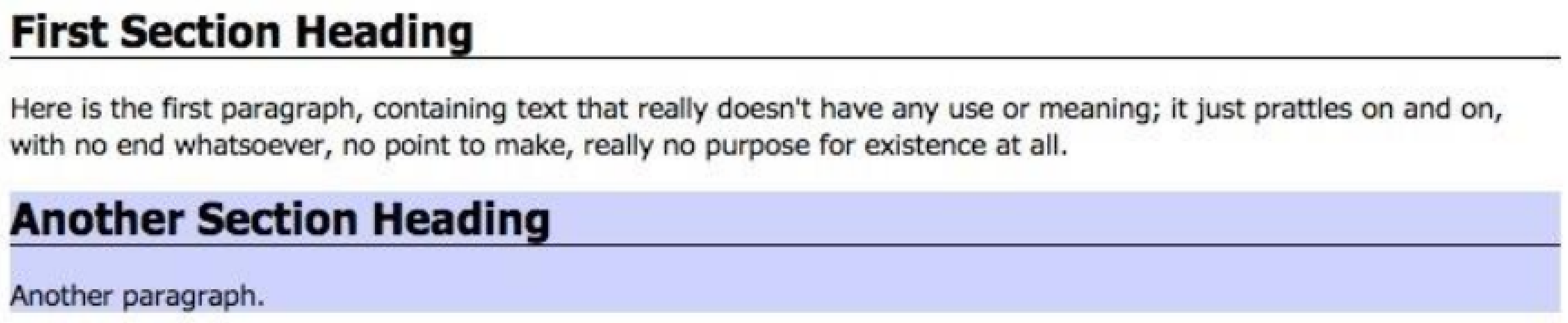
```css
body {
  font-family: Tahoma, Arial, sans-serif;
  font-size: 13px;
  color: black;
  background: white;
  margin: 8px;
}
h1 {
  font-size: 19px;
  margin-top: 0px;
  margin-bottom: 5px;
  border-bottom: 1px solid black
}
.shaded {
  background: #d0d0ff;
}
```

CSS:

```html
<body>
  <h1>First Section Heading</h1>
  <p>
    Here is the first paragraph, containing
    text that really doesn't have any use
    or meaning; it just prattles on and on,
    with no end whatsoever, no point to
    make, really no purpose for existence
    at all.
  </p>
  <div class="shaded">
    <h1>Another Section Heading</h1>
    <p>
      Another paragraph.
    </p>
  </div>
</body>
```

HTML:

Example Output



First Section Heading

Here is the first paragraph, containing text that really doesn't have any use or meaning; it just prattles on and on, with no end whatsoever, no point to make, really no purpose for existence at all.

Another Section Heading

Another paragraph.

Universal Resource Locator (URL)

Hypertext

- Text with **links** to other text
    - Click on links takes you somewhere else
- Web adapted the idea, link specification:
    - Uniform Resource Locators (URL) - Provided **names** for web content

```
<a href="https://en.wikipedia.org/wiki/URL">URL</a>
```

Parts of an URL

`http://host.company.com:80/a/b/c.html?user=Alice&year=2008#p2`

Scheme (**`http:`**): identifies protocol used to fetch the content.

Host name (**`host.company.com`**): name of a machine to connect to.

Server's port number (**`80`**): allows multiple servers to run on the same machine.

Hierarchical portion (**`/a/b/c.html`**): used by server to find content.

Query parameters (**`?user=Alice&year=2008`**): provides additional parameters

Fragment (**`#p2`**): Have browser scroll page to fragment (html: p2 is anchor tag)

Used on the browser only; not sent to the server.

URL: schemes (e.g. `http`)

`http`: is the most common scheme; it means use the HTTP protocol

`https`: is similar to http: except that it uses SSL encryption

`file`: means read a file from the local disk

`websocket:` means create a TCP connection

`mailto`: means open an email program composing a message

There are many (~350) other schemes: https://www.iana.org/assignments/uri-schemes/

Example: mongodb: points to a MongoDB database

URL: Hierarchical portion (/a/b/c.html)

- Passed to the web server for interpretation. Early web servers:
  - Path name for a static HTML file.
  - Path name of a program that will generate the HTML content (e.g., `foo.php`).
- Web server programmed with **routing** information
  - Map hierarchical position to function to be performed and possibly the function's parameters
- Application Programming Interface (API) design, Example:
  - `/user/create`
  - `/user/list`
  - `/user/0x23490`
  - `/user/0x23433`
  - `/user/delete/0x23433`

Links

- Browser maintains a notion of current location (i.e. URL)
- Links: content in a page which, when clicked on, causes the browser to go to URL
- Links are implemented with the <a> tag:

```
<a href="http://www.company.com/news/2009.html">2009 News</a>
```

Different types of links

Full URL: `<a href="http://www.xyz.com/news/2009.html">2009 News</a>`

```
Absolute URL: <a href="/stock/quote.html">
    same as http://www.xyz.com/stock/quote.html
```

Relative URL (intra-site links): `<a href="2008/March.html">`

    same as `http://www.xyz.com/news/2008/March.html`

Define an anchor point (a position that can be referenced with # notation):

`<a name="sec3">`

Go to a different place in the same page: `<a href="#sec3">`

Uses of URLs

- Loading a page: type the URL into your browser
- Load a image:
  `<img src="..." />`
- Load a `stylesheet`:
  `<link rel="stylesheet" type="text/css" href="...">`
- Embedded a page:
  `<iframe src="http://www.google.com">`