Web Development

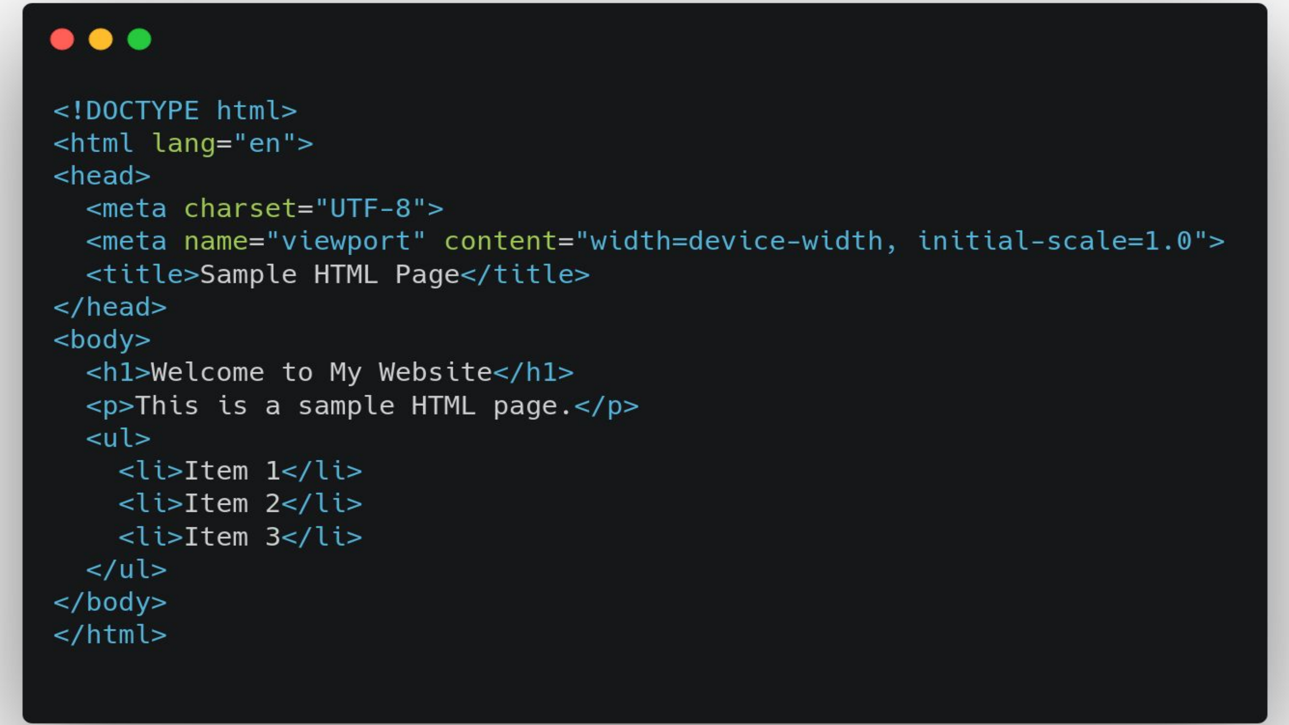
Introduction to HTML and CSS

Khyari hamza

What is HTML ?

- **HTML** stands for **HyperText Markup Language**
- It is used to create the **structure** and content of web pages
- Consists of a series of elements (**tags**) that are used to define the structure and display of content

What is HTML ?



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sample HTML Page</title>
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is a sample HTML page.</p>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
  </ul>
</body>
</html>
```

HTML document structure

- An HTML document consists of a series of nested elements (tags)
- The structure is hierarchical, with the `<html>` element being the root element
- The document is divided into two main sections: `<head>` and `<body>`

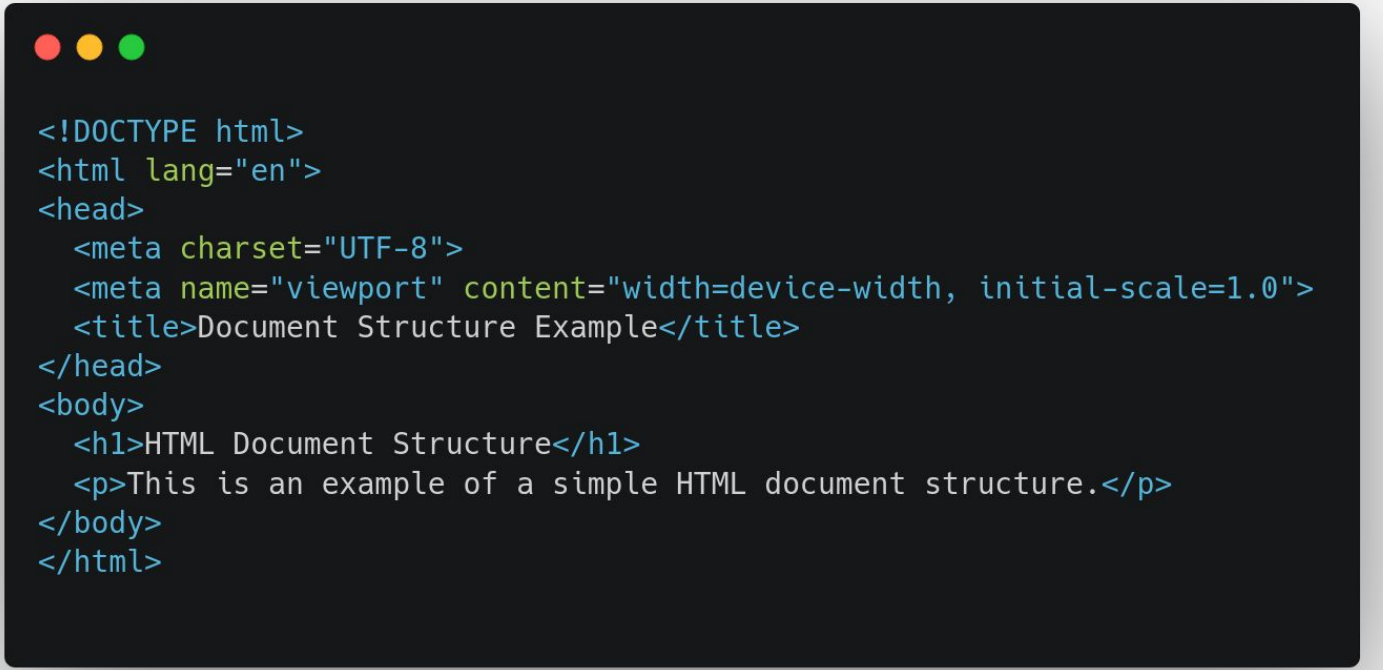
HTML document structure

Basic Structure

1. `<!DOCTYPE html>`: Declaration that defines the document type and version of HTML
2. `<html>`: The root element that contains all other elements
3. `<head>`: Contains metadata about the document, such as the title and character encoding
4. `<body>`: Contains the main content of the web page, such as text, images, and links

HTML document structure

Basic Structure



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document Structure Example</title>
</head>
<body>
  <h1>HTML Document Structure</h1>
  <p>This is an example of a simple HTML document structure.</p>
</body>
</html>
```

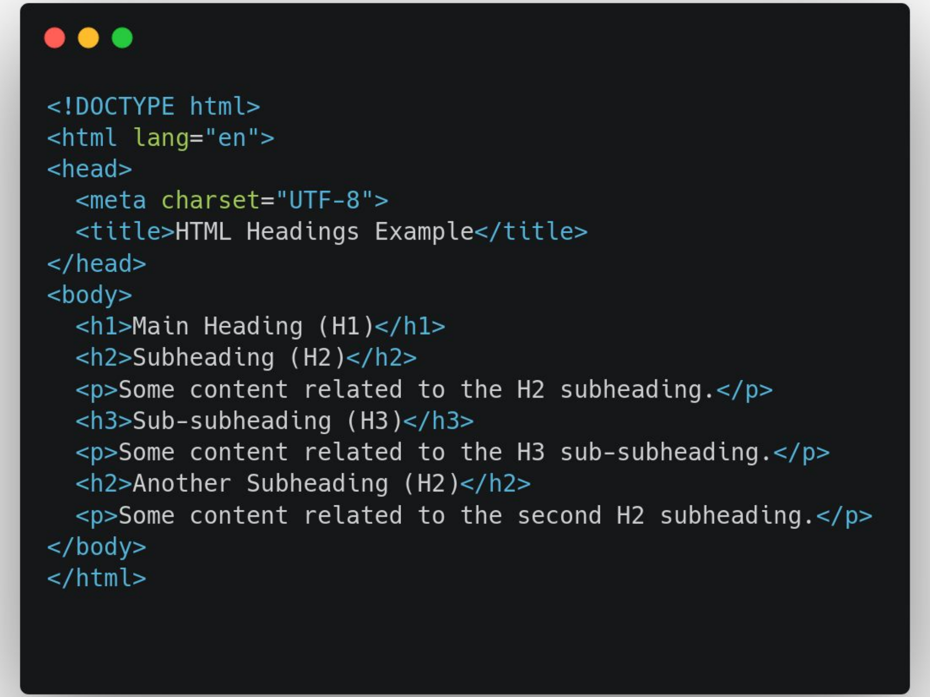
Basic HTML tags

- **HTML** tags are used to define the structure and content of a web page
- Tags are written in **pairs**, with an **opening** tag and a **closing** tag
- Some tags are self-closing and don't require a closing tag

Basic HTML tags

Headings:

<h1> to **<h6>**: Define headings, with **<h1>** being the largest and **<h6>** the smallest

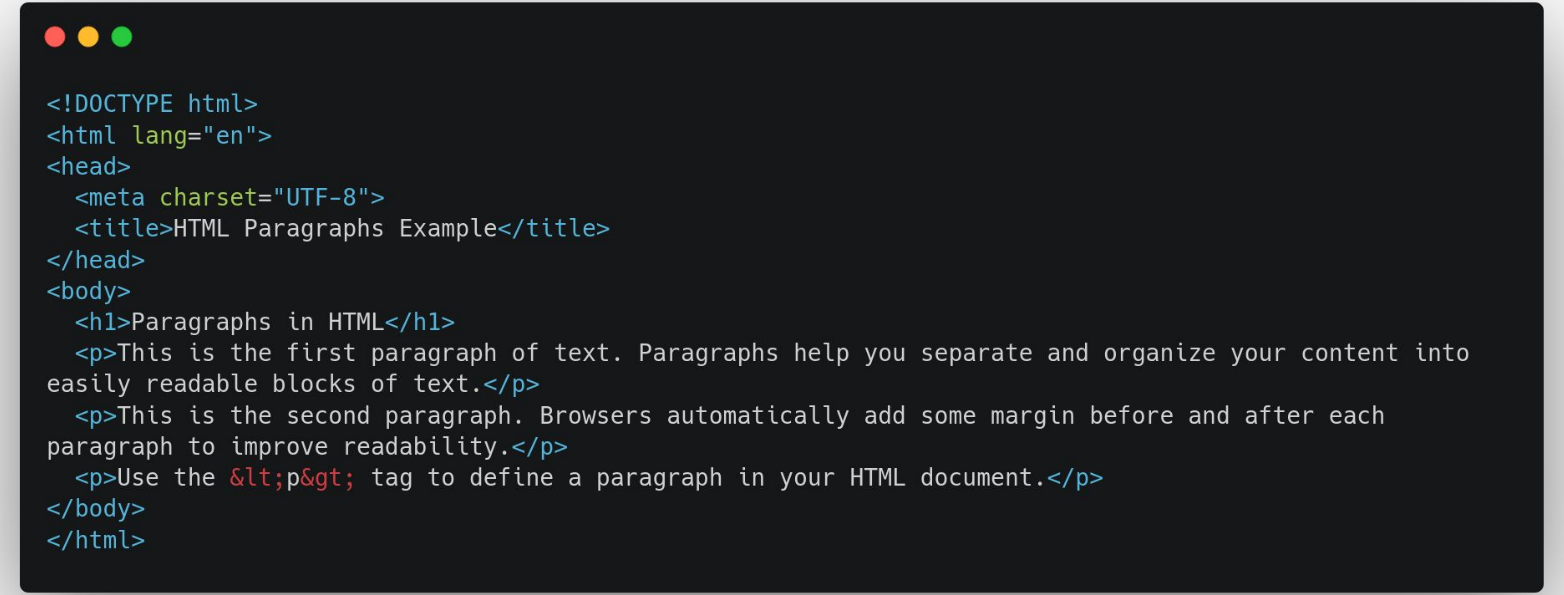


```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>HTML Headings Example</title>
</head>
<body>
  <h1>Main Heading (H1)</h1>
  <h2>Subheading (H2)</h2>
  <p>Some content related to the H2 subheading.</p>
  <h3>Sub-subheading (H3)</h3>
  <p>Some content related to the H3 sub-subheading.</p>
  <h2>Another Subheading (H2)</h2>
  <p>Some content related to the second H2 subheading.</p>
</body>
</html>
```

Basic HTML tags

Paragraph:

<p>: Define a paragraph of text



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>HTML Paragraphs Example</title>
</head>
<body>
  <h1>Paragraphs in HTML</h1>
  <p>This is the first paragraph of text. Paragraphs help you separate and organize your content into
easily readable blocks of text.</p>
  <p>This is the second paragraph. Browsers automatically add some margin before and after each
paragraph to improve readability.</p>
  <p>Use the &lt;p&gt; tag to define a paragraph in your HTML document.</p>
</body>
</html>
```

Basic HTML tags

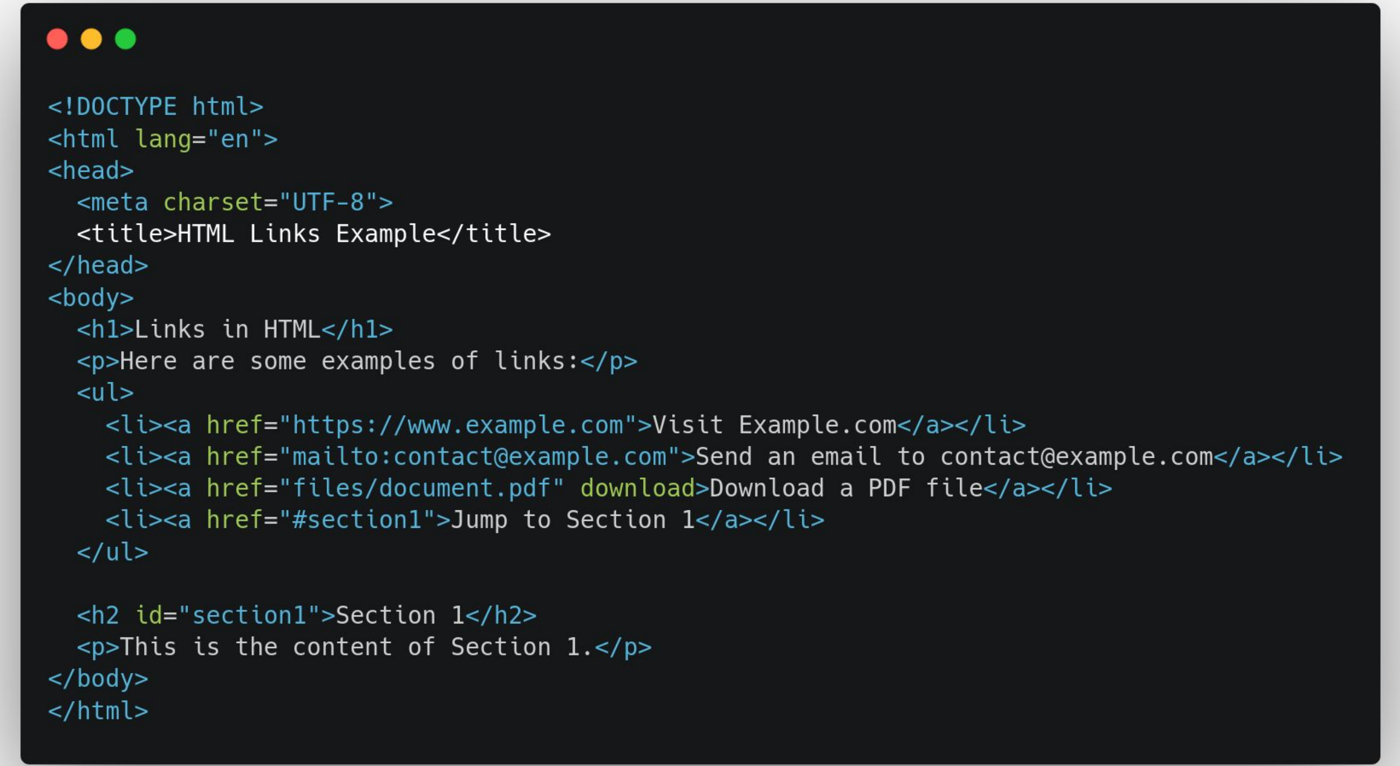
Links:

`<a href="URL">`: Create a hyperlink to another page or website

- Links, also known as hyperlinks, allow users to navigate between web pages
- The `<a>` tag is used to create links, with the `href` attribute specifying the destination URL
- Links can point to other web pages, files, email addresses, or even specific parts of the same page

Basic HTML tags

Links:



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>HTML Links Example</title>
</head>
<body>
  <h1>Links in HTML</h1>
  <p>Here are some examples of links:</p>
  <ul>
    <li><a href="https://www.example.com">Visit Example.com</a></li>
    <li><a href="mailto:contact@example.com">Send an email to contact@example.com</a></li>
    <li><a href="files/document.pdf" download>Download a PDF file</a></li>
    <li><a href="#section1">Jump to Section 1</a></li>
  </ul>

  <h2 id="section1">Section 1</h2>
  <p>This is the content of Section 1.</p>
</body>
</html>
```

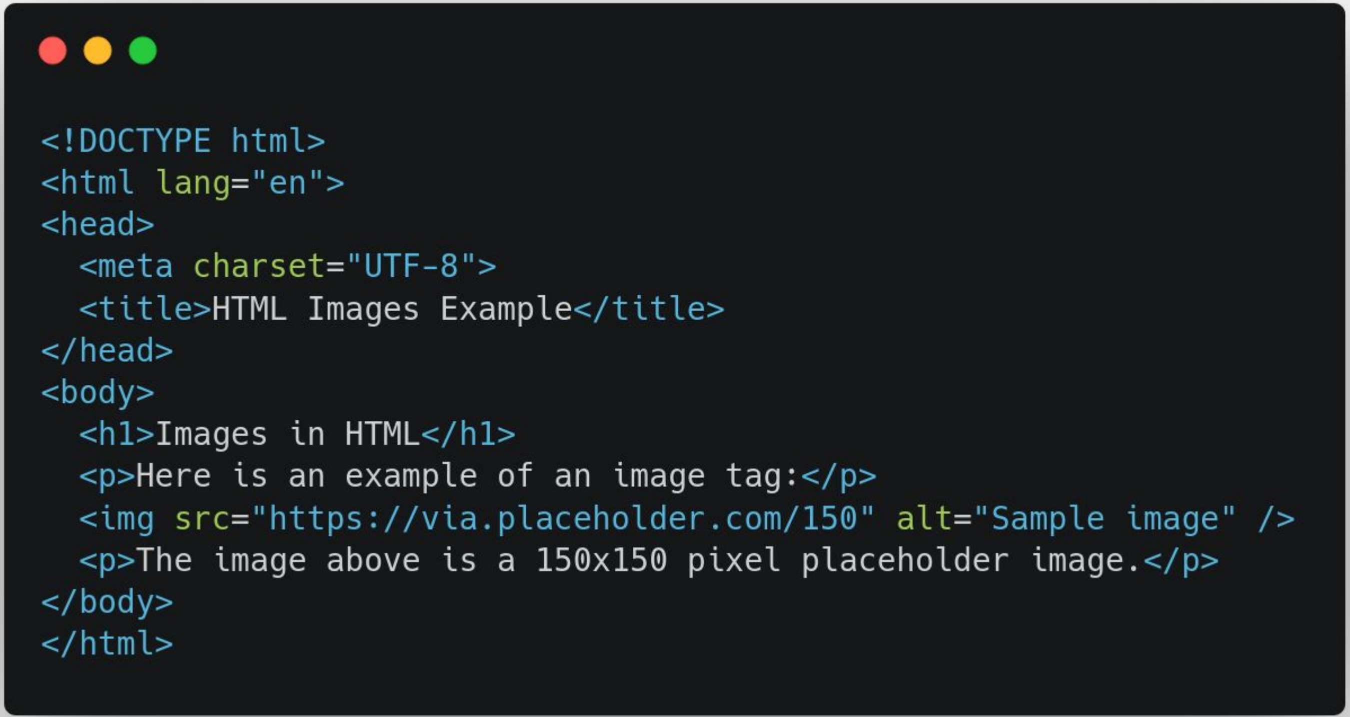
Basic HTML tags

Images:

- **<img src="image_url" alt="image_description">**: Add an image to the page
- The **<img>** tag is used to display images on a web page
- The **src** attribute specifies the image URL or path
- The **alt** attribute provides a text description for the image, which is important for accessibility and SEO
- The **<img>** tag is self-closing and does not require a closing tag

Basic HTML tags

Images:



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>HTML Images Example</title>
</head>
<body>
  <h1>Images in HTML</h1>
  <p>Here is an example of an image tag:</p>
  <img src="https://via.placeholder.com/150" alt="Sample image" />
  <p>The image above is a 150x150 pixel placeholder image.</p>
</body>
</html>
```

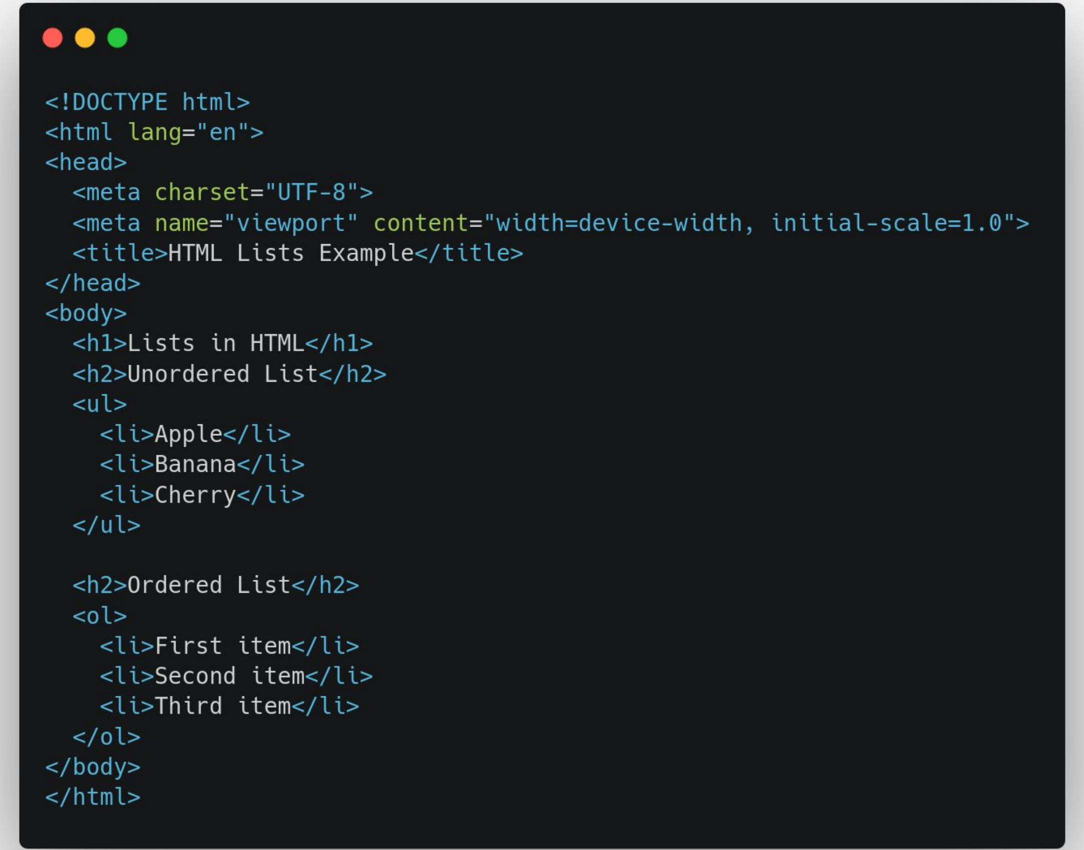
Basic HTML tags

Lists:

- `<ul>`: Create an unordered (bulleted) list
- `<ol>`: Create an ordered (numbered) list
- `<li>`: Define a list item of **<ul>** and **<ol>**

Basic HTML tags

Lists:



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>HTML Lists Example</title>
</head>
<body>
  <h1>Lists in HTML</h1>
  <h2>Unordered List</h2>
  <ul>
    <li>Apple</li>
    <li>Banana</li>
    <li>Cherry</li>
  </ul>

  <h2>Ordered List</h2>
  <ol>
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
  </ol>
</body>
</html>
```

What is CSS ?

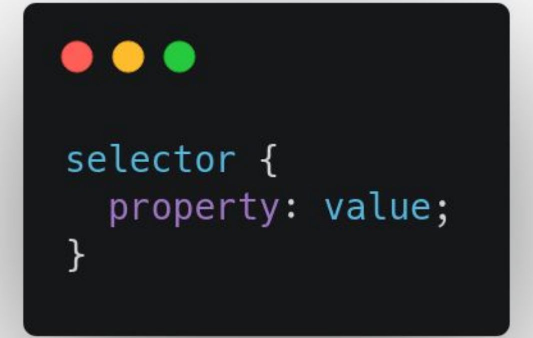
- **CSS** stands for **Cascading Style Sheets**
- It is a stylesheet language used to control the **presentation** and **styling** of web pages
- CSS works alongside HTML, which defines the structure and content of a web page

Role of CSS

- Separate presentation from content, making it easier to **maintain** and update web pages
- Control layout, colors, fonts, and other visual aspects of web pages
- Enable **responsive** design for different devices and screen sizes

CSS syntax

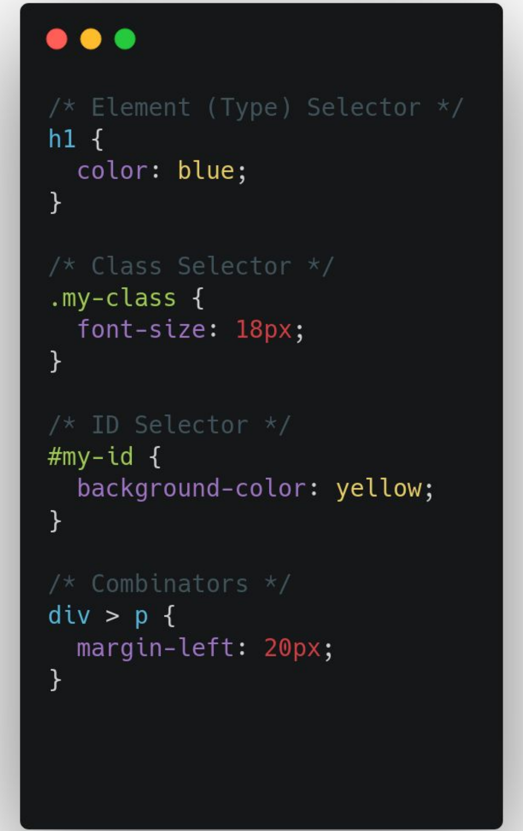
- CSS rules consist of a **selector** and a **declaration block**
- The declaration block contains one or more **property-value** pairs



```
selector {
  property: value;
}
```

CSS selectors

- **Element (Type) Selector**: Targets all elements of a specific type
- **Class Selector**: Targets elements with a specific class attribute
- **ID Selector**: Targets a single element with a specific ID attribute
- **Combinators**: Combine different selectors to target elements based on their relationships



```css
/* Element (Type) Selector */
h1 {
  color: blue;
}

/* Class Selector */
.my-class {
  font-size: 18px;
}

/* ID Selector */
#my-id {
  background-color: yellow;
}

/* Combinators */
div > p {
  margin-left: 20px;
}
```

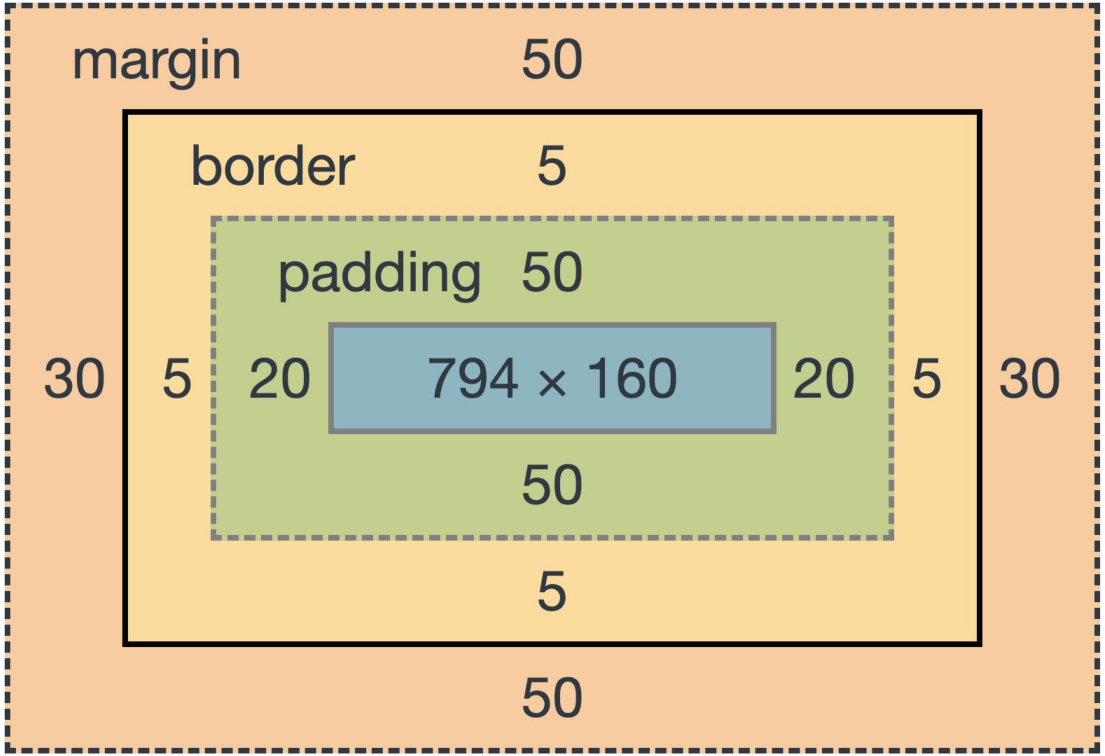
CSS Box model

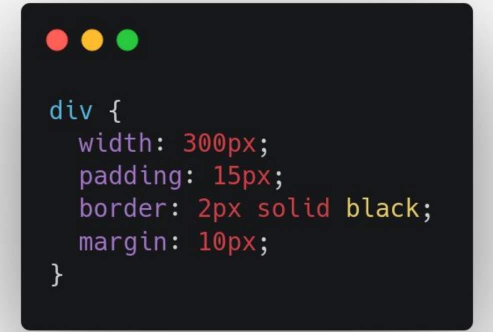
- The box model is a fundamental concept in CSS that describes how **layout**, **spacing**, and **borders** work for every HTML element
- Each element is represented as a **rectangular box**, with various properties defining the dimensions and appearance of the box

CSS Box model: Components

1. **Content**: The actual content of the element, such as text or images
2. **Padding**: The space between the content and the border
3. **Border**: The line that surrounds the content and padding
4. **Margin**: The space outside the border, separating the element from other elements

CSS Box model: Components

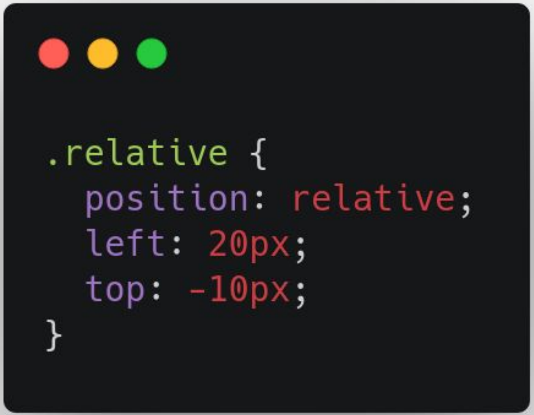




```css
div {
  width: 300px;
  padding: 15px;
  border: 2px solid black;
  margin: 10px;
}
```

CSS Position

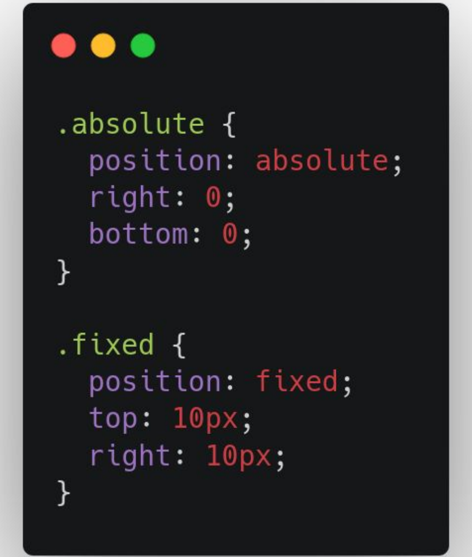
1. **Static (Default)**: Elements are positioned according to the normal flow of the document
2. **Relative**: Elements are positioned relative to their normal position, without affecting the position of other elements



```
.relative {
  position: relative;
  left: 20px;
  top: -10px;
}
```

CSS Position

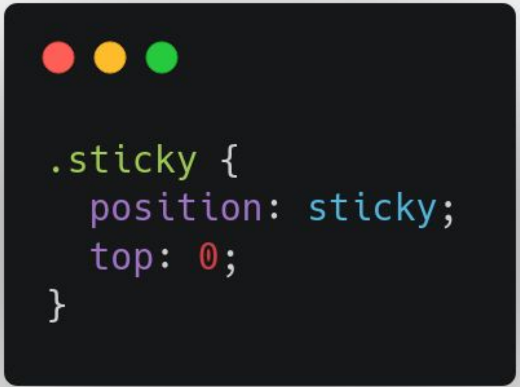
1. **Absolute**: Elements are positioned relative to their nearest positioned ancestor or the initial containing block, and removed from the normal document flow
2. **Fixed**: Elements are positioned relative to the browser window, and remain fixed when scrolling



```css
.absolute {
  position: absolute;
  right: 0;
  bottom: 0;
}

.fixed {
  position: fixed;
  top: 10px;
  right: 10px;
}
```

CSS Position

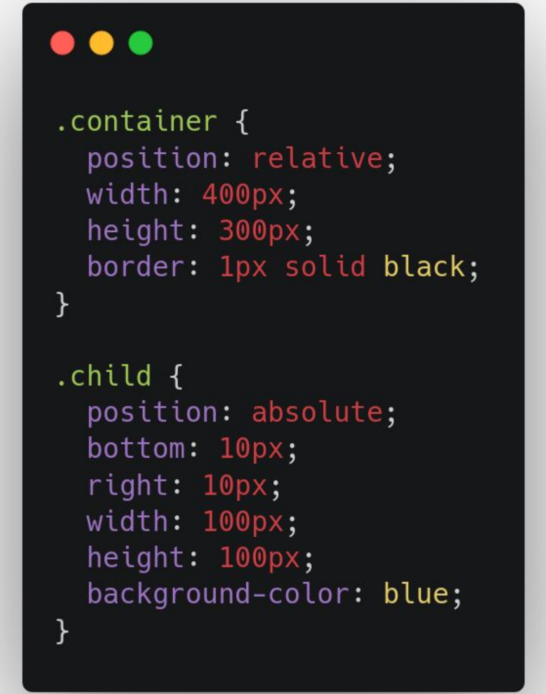
1. **Sticky**: Elements are positioned based on the user's scroll position, switching between relative and fixed positioning



```
.sticky {
  position: sticky;
  top: 0;
}
```

CSS Position

Example:



```css
.container {
  position: relative;
  width: 400px;
  height: 300px;
  border: 1px solid black;
}

.child {
  position: absolute;
  bottom: 10px;
  right: 10px;
  width: 100px;
  height: 100px;
  background-color: blue;
}
```

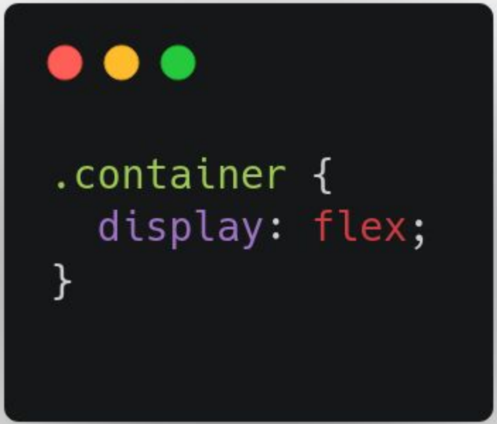
CSS FlexBox

- **Flexbox** (Flexible Box) is a CSS layout module that provides an efficient way to distribute space among items in a container and to align those items
- It is particularly useful for building responsive and fluid layouts with changing screen sizes

CSS FlexBox

Flex Container

To create a flex container, set the **display** property of an element to **flex** or **inline-flex**



```
.container {
  display: flex;
}
```

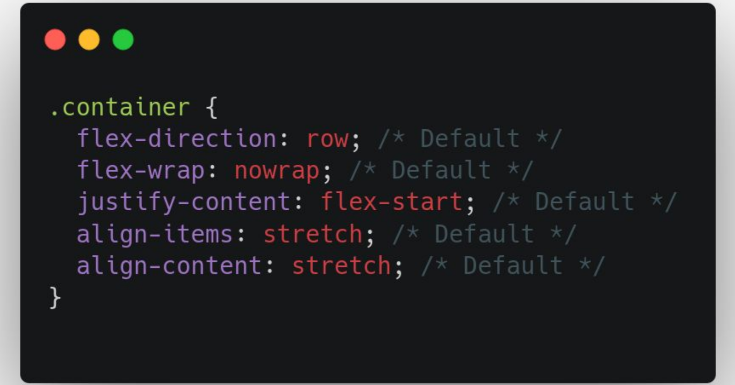
CSS FlexBox

Flex Items

- The direct children of a flex container automatically become flex items
- Flex items can be resized, aligned, and ordered inside the container

CSS FlexBox

Flex Items: Main properties

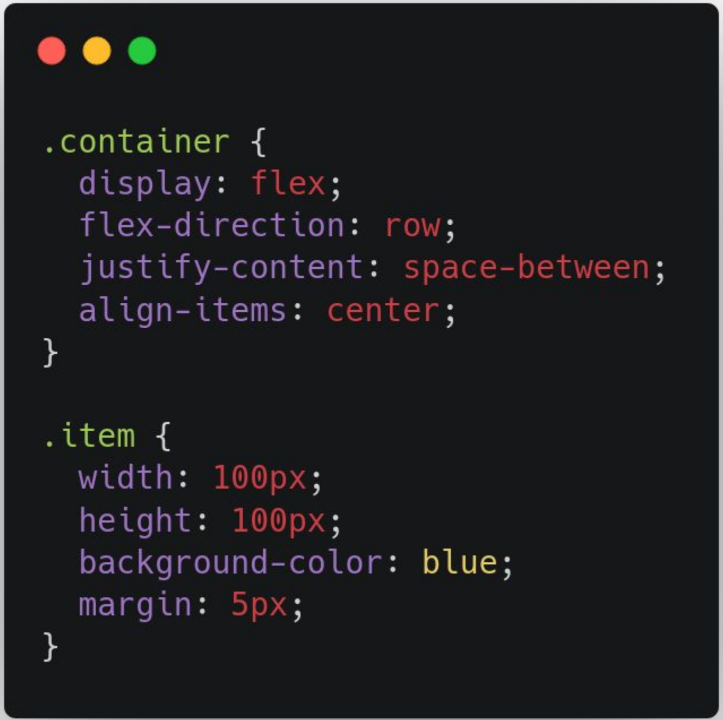


```
.container {
  flex-direction: row; /* Default */
  flex-wrap: nowrap; /* Default */
  justify-content: flex-start; /* Default */
  align-items: stretch; /* Default */
  align-content: stretch; /* Default */
}
```

1. **flex-direction**: Determines the direction of the main axis (row or column)
2. **flex-wrap**: Controls whether items wrap onto multiple lines or stay on a single line
3. **justify-content**: Aligns items along the main axis (horizontal or vertical, depending on flex-direction)
4. **align-items**: Aligns items along the cross axis (perpendicular to the main axis)
5. **align-content**: Aligns wrapped lines of items along the cross axis

CSS FlexBox

Example:



```css
.container {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
}

.item {
  width: 100px;
  height: 100px;
  background-color: blue;
  margin: 5px;
}
```

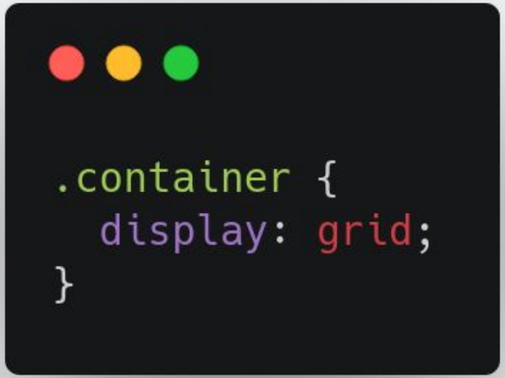
CSS Grid

- **Grid** is a CSS layout module that allows you to create complex, responsive, and flexible two-dimensional layouts
- It is particularly useful for building layouts with rows and columns, such as grids and tables

CSS Grid

Grid Container

To create a grid container, set the **display** property of an element to **grid** or **inline-grid**



```
.container {
  display: grid;
}
```

CSS Grid

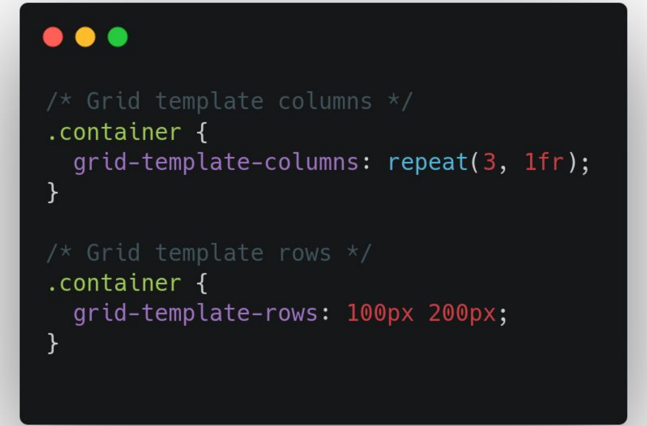
Grid Items

- The direct children of a grid container automatically become grid items
- Grid items can be placed and aligned within the container using grid lines, tracks, and areas

CSS Grid

Defining Grid Structure

- **grid-template-columns**: Defines the columns in the grid layout
- **grid-template-rows**: Defines the rows in the grid layout

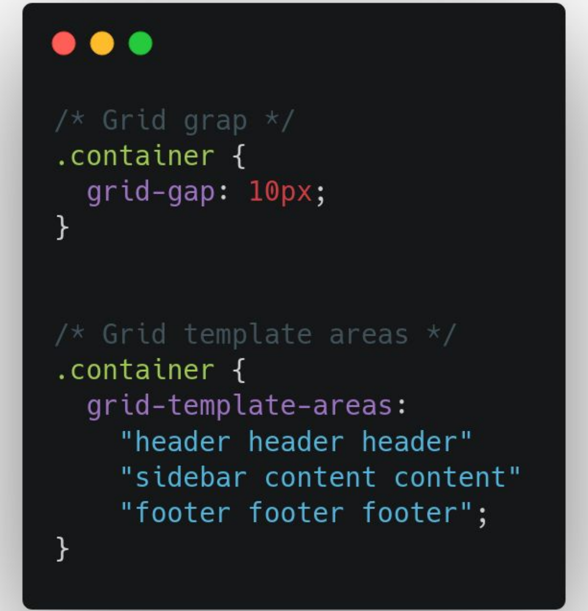


```css
/* Grid template columns */
.container {
  grid-template-columns: repeat(3, 1fr);
}

/* Grid template rows */
.container {
  grid-template-rows: 100px 200px;
}
```

CSS Grid

Defining Grid Structure

- **grid-gap**: Defines the space between grid items (both rows and columns)
- **grid-template-areas**: Defines named grid areas for easy placement of grid items

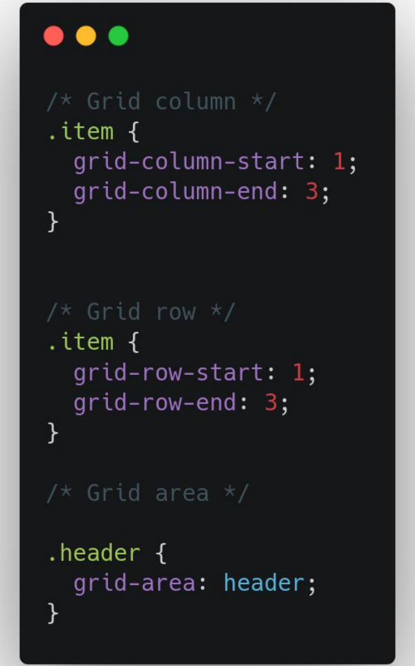


```css
/* Grid grap */
.container {
  grid-gap: 10px;
}


/* Grid template areas */
.container {
  grid-template-areas:
    "header header header"
    "sidebar content content"
    "footer footer footer";
}
```

CSS Grid

Placing Grid Items

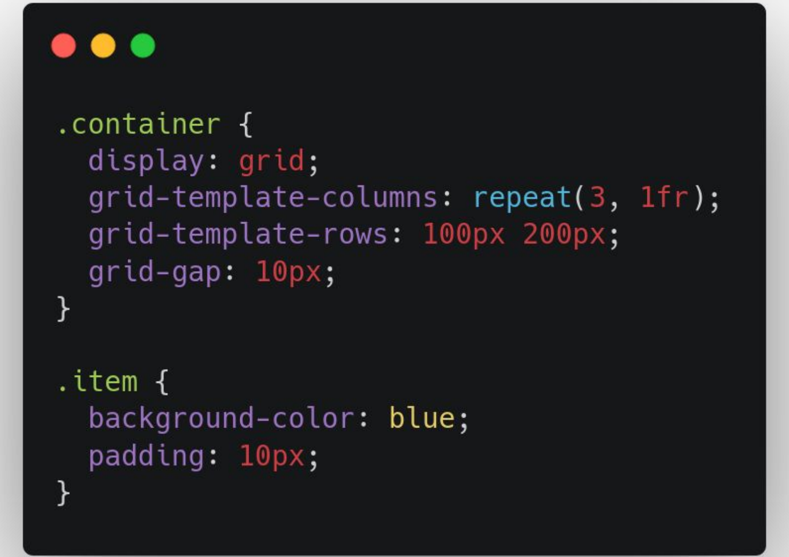
- **grid-column-start / grid-column-end**: Specifies the start and end grid lines for a grid item's column placement
- **grid-row-start / grid-row-end**: Specifies the start and end grid lines for a grid item's row placement
- **grid-area**: Specifies the named grid area for a grid item's placement



```css
/* Grid column */
.item {
  grid-column-start: 1;
  grid-column-end: 3;
}


/* Grid row */
.item {
  grid-row-start: 1;
  grid-row-end: 3;
}

/* Grid area */

.header {
  grid-area: header;
}
```

CSS Grid

CSS Grid Example:



```css
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: 100px 200px;
  grid-gap: 10px;
}

.item {
  background-color: blue;
  padding: 10px;
}
```

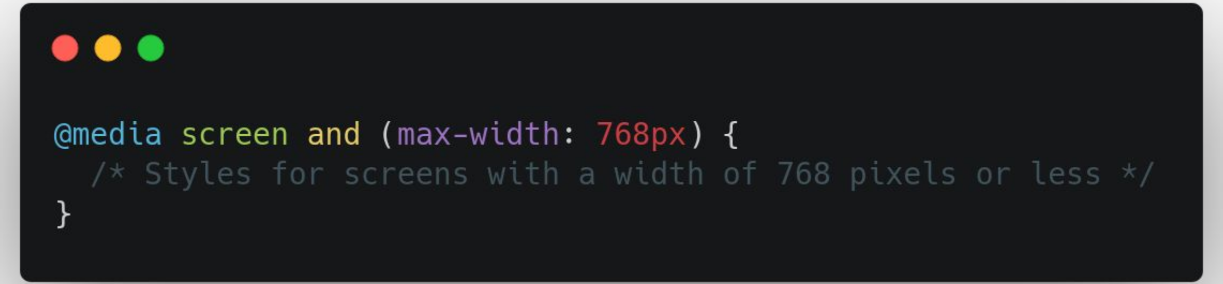
CSS Media Queries

- **Media queries** are used to apply **different** styles based on various conditions, such as device type, screen size, and orientation
- They are essential for creating **responsive web designs** that adapt to different devices and viewing environments

CSS Media Queries

Syntax

Media queries use the **@media** rule followed by a condition

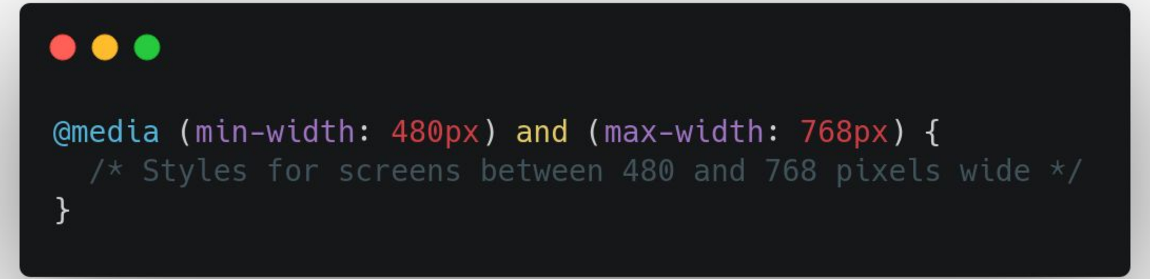


```css
@media screen and (max-width: 768px) {
  /* Styles for screens with a width of 768 pixels or less */
}
```

CSS Media Queries

Common Media Features

- **width / height**: Specifies the width or height of the viewport

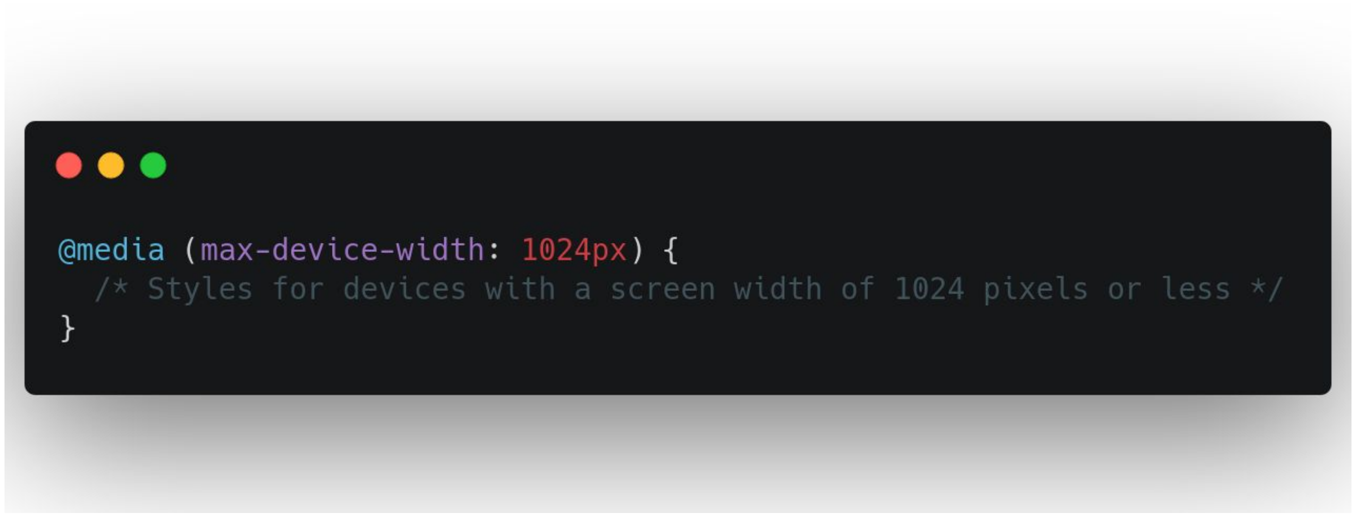


```css
@media (min-width: 480px) and (max-width: 768px) {
  /* Styles for screens between 480 and 768 pixels wide */
}
```

CSS Media Queries

Common Media Features

- **device-width / device-height**: Specifies the width or height of the device screen

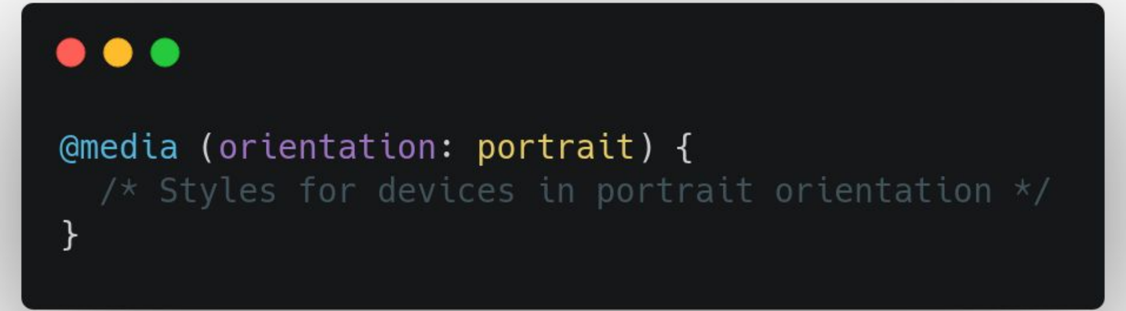


```css
@media (max-device-width: 1024px) {
  /* Styles for devices with a screen width of 1024 pixels or less */
}
```

CSS Media Queries

Common Media Features

- **orientation**: Specifies the orientation of the device (portrait or landscape)

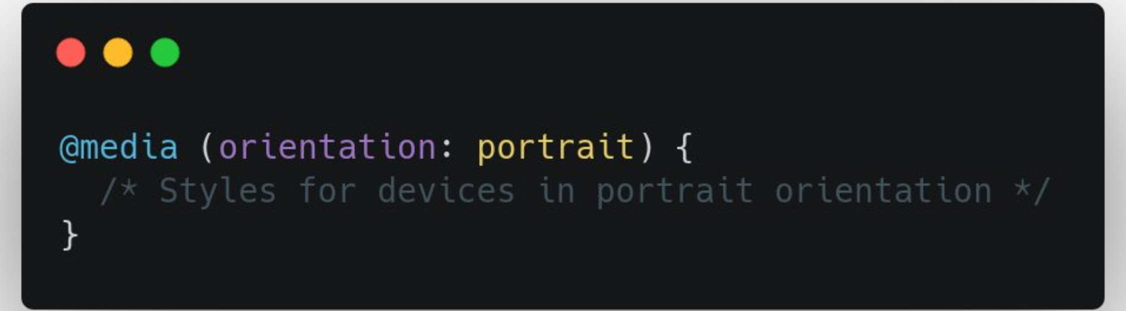


```css
@media (orientation: portrait) {
  /* Styles for devices in portrait orientation */
}
```

CSS Media Queries

Common Media Features

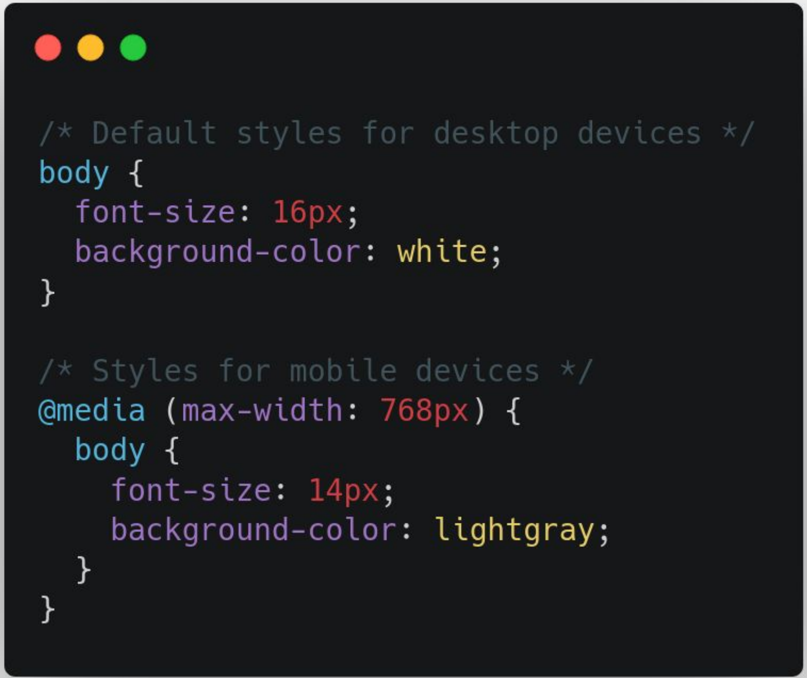
- **resolution**: Specifies the pixel density of the device (dots per inch or dots per pixel)



```css
@media (orientation: portrait) {
  /* Styles for devices in portrait orientation */
}
```

CSS Media Queries

Media Example:



```
/* Default styles for desktop devices */
body {
  font-size: 16px;
  background-color: white;
}

/* Styles for mobile devices */
@media (max-width: 768px) {
  body {
    font-size: 14px;
    background-color: lightgray;
  }
}
```

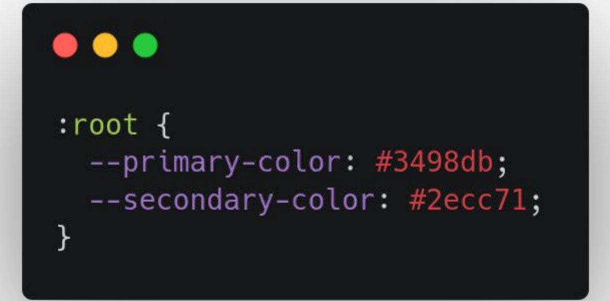
CSS Variables

- CSS variables, also known as custom properties, are used to **store** values that can be **reused** throughout a stylesheet
- They enable easier **maintenance**, more consistent design, and better scalability of CSS code

CSS Variables

Defining Variables

- Variables are defined using a double hyphen (`--`) followed by the variable name
- They are usually declared inside a selector, most commonly the `:root` selector for global scope

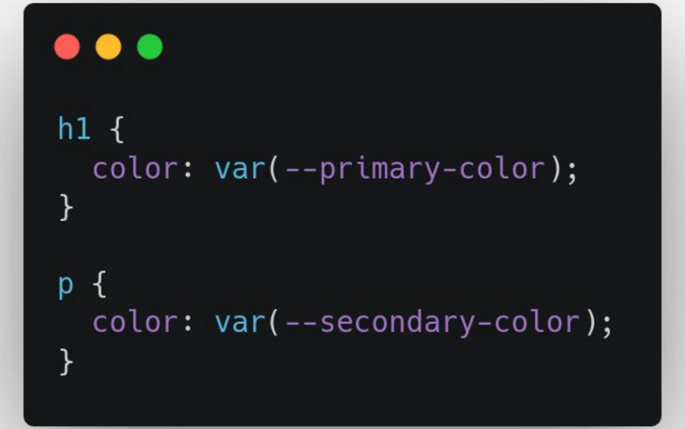


```css
:root {
  --primary-color: #3498db;
  --secondary-color: #2ecc71;
}
```

CSS Variables

Using Variables

To use a variable, reference its name with the **var()** function

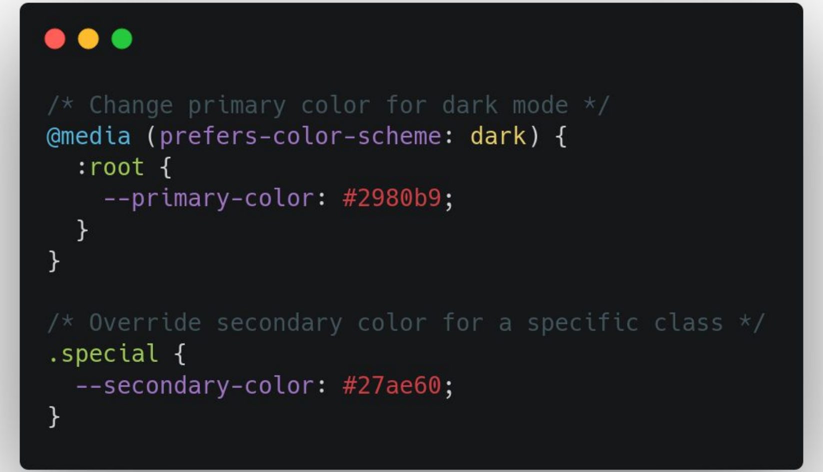


```css
h1 {
  color: var(--primary-color);
}

p {
  color: var(--secondary-color);
}
```

CSS Variables

Modifying Variables

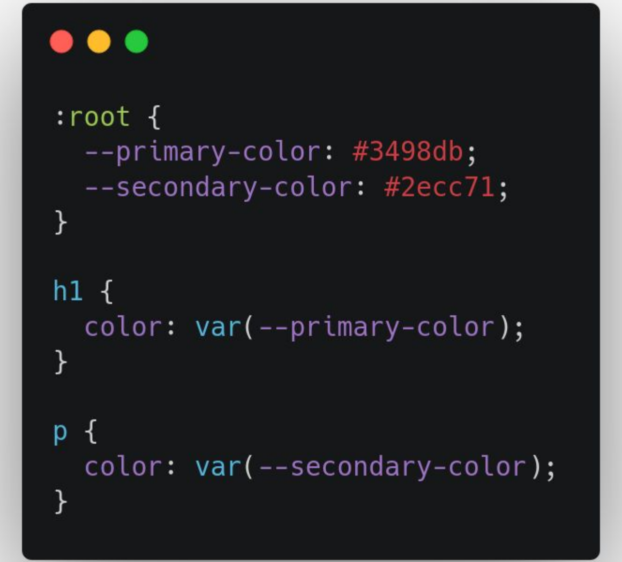
Variables can be modified within different scopes, such as inside media queries or within specific selectors



```
/* Change primary color for dark mode */
@media (prefers-color-scheme: dark) {
  :root {
    --primary-color: #2980b9;
  }
}

/* Override secondary color for a specific class */
.special {
  --secondary-color: #27ae60;
}
```

CSS Variables

Variables Example:



```css
:root {
  --primary-color: #3498db;
  --secondary-color: #2ecc71;
}

h1 {
  color: var(--primary-color);
}

p {
  color: var(--secondary-color);
}
```

Recap and Resources

1. **HTML Basics**: Elements, attributes, and syntax
2. **HTML Document Structure**: DOCTYPE, head, body, and common elements
3. **CSS Basics**: Selectors, properties, and syntax
4. **CSS Positioning**: Static, relative, absolute, fixed, and sticky
5. **CSS Flexbox**: Container, items, and properties for flexible layouts
6. **CSS Grid**: Container, items, and properties for two-dimensional layouts
7. **CSS Media Queries**: Responsive design and common media features
8. **CSS Variables**: Defining, using, and modifying custom properties

Recap and Resources

Recommended Resources

- MDN Web Docs: Comprehensive guides and documentation for web developers
- CSS-Tricks: Tips, tricks, and tutorials for CSS and web development
- W3Schools: Tutorials and references for web technologies
- Codecademy: Interactive coding courses for HTML, CSS, and more
- freeCodeCamp: Free coding curriculum and projects for web development

Practice and Experiment

- CodePen: Online code editor for HTML, CSS, and JavaScript
- JSFiddle: Another online code editor for HTML, CSS, and JavaScript
- Replit: Online code editor and hosting platform for various languages and technologies