FUNDAMENTALS

Terms

Attributes	HTML element
Client	HTTP (Hypertext Transfer Protocol)
CSS	HTTP request
CSS declaration	HTTP response
CSS property	JavaScript
CSS rule	Opening/closing tags
DOM (Document Object Model)	Server
HTML	URL (Uniform Resource Locator)
HTML document	Validation

Summary

- *HTML (Hypertext Markup Language)* is a markup language used to define the structure and contents of web pages.
- *CSS (Cascading Stylesheets)* are used for styling web pages.
- *JavaScript* is a programming language used for making web pages interactive.
- To view a web page, we enter its *URL (Uniform Resource Locator)* into the browser's address bar.
- To view the HTML behind a web page, we right-click on the page and select View Source or View Page Source.
- *HTTP (Hypertext Transfer Protocol)* is a set of rules that clients and servers use to communicate and exchange data.

- When you visit a website, your browser (client) sends an *HTTP request* to the server and receives an *HTTP response*. The response includes an *HTML document* that represents the page you're visiting. The browser reads the HTML document to construct a *Document Object Model* (an in-memory representation of the elements on the page). Once the DOM is ready, the browser *renders* (displays) the target page.

- We can use the Network tab of Chrome DevTools to inspect HTTP requests and responses exchanged between the browser and the server.

- An HTML document consists of a `DOCTYPE` declaration that specifies the version of HTML being used followed by a tree (hierarchy) of elements that define the structure and content of the document.

- The *root element* in an HTML document is the `html` element which always includes a `head` and a `body` element. The `head` element contains information about the page such as its title. The `body` element includes the structure and content of the page.

- Most HTML elements include an *opening and closing tag*.

- We can use *attributes* to provide additional information about an element (like the source of an image). Attributes are coded as part of the opening tag.

- To style an element using CSS, we write a *CSS rule* which consists of a *selector* and one or more *declarations*.

- The selector is used to target one or more elements. We can select elements by their type, class, ID and so on.

- A declaration consists of a property, a colon, a value and is terminated by a semicolon.

- Prettier is the most popular VSCode extension for formatting code.

- Using *validation* services we can ensure that our HTML/CSS code is valid and follows the official rules. This helps us get better rankings in search engines.

HTML BASICS

Terms

Absolute URLs
Block-level elements
Character sets
Container elements
Description lists
Entities
HTML boilerplate
Hyperlinks
Inline elements
Ordered lists
Relative URLs
Semantic HTML
Structural elements
Unordered lists
Viewport
Zen coding

Summary

- The `<head>` section is used to provide information about a webpage.
- The `<p>` element is used to represent a paragraph. A paragraph can be one or many lines of text.
- The `<em>` element is used to define emphasized text. By default, emphasized text is displayed in *italic*.
- The `<strong>` element is used to represent important content. Browsers, by default, render strong content in **bold**.

- The `<i>` and `<b>` elements are considered deprecated because HTML should not be used for styling. That's the role of CSS.

- Headings are represented using `<h1>, <h2>, <h3>, <h4>, <h5>, <h6>`. Every web page should have one and only one `<h1>` element. Headings should have a natural hierarchy and should not be skipped.

- Entities are used to display special characters such as angle brackets, copyright symbol, etc. The most important entities are: `&nbsp;` (non-breaking space), `&lt;` (less than sign), `&gt;` (greater than sign) and `&copy;` (copyright symbol).

- The `<a>` (anchor) element, with its `href` attribute, is used to create a hyperlink to web pages, locations in the same page, files and email addresses.

- A relative URL specifies the target resource *relative* to the current resource. An absolute URL specifies the location of a resource irrespective of the current resource. It can start with a / to indicate the root of the website or a protocol (eg http://) to represent a resource on a different website.

- The `<img>` element is used to display an image. It's a common best practice to set the `alt` (alternative text) attribute. This helps visually impaired people understand the page content. Also, if the image cannot be loaded, the alternative text is displayed.

- The `<video>` and `<audio>` elements are used to display video and audio. These elements have boolean attributes such as `controls, autoplay` and `loop.`

- The `<ul>` element is used to represent a list where the order of items doesn't matter. The `<ol>` element is used to represent an ordered list of items. The `<dl>` (description list) element is used to implement a glossary or to display metadata.

- The `<table>` element should only be used to represent tabular data. A table can have zero or more `<tr>` (table row) elements. Each `<tr>` element can have zero or more cells. Cells can be data cells (`<td>`) or header cells (`<th>`).

- The `<div>` and `<span>` elements are generic containers used for styling purposes. Divs are block-level elements, spans are inline elements. A block-level element starts on a new line and takes up the entire available horizontal space.

- Semantic elements help us write markup that is more meaningful and descriptive to search engines, screen readers and other software. So, use `<div>` and `<span>` elements when no other semantic element is appropriate.

- The semantic elements in HTML5 are: `<header>, <footer>, <nav>, <main>, <aside>, <article>, <section>, <figure>, <time>` and `<mark>`.

CSS BASICS

Terms

Embedded stylesheets	Pseudo-class selectors
External stylesheets	Pseudo-element selectors
Hexadecimal colors	Radial gradients
HSL colors	Relational selectors
Inheritance	RGB colors
Inline styles	Selectors
Linear gradients	Selectors specificity
Normalizing CSS	Separation of concerns

Summary

- CSS styles can be embedded in an HTML document, written in a separate file (as an external stylesheet) or written inline in an HTML element using the `style` attribute.

- Inline styles overwrite embedded styles which in turn overwrite external styles.

- External stylesheets provide the best separation of HTML and CSS code and result in more maintainable code. Plus, an external stylesheet can be used in many HTML documents.

- We can select elements by their type, class, attribute or ID.
- Relational selectors help us select elements without the need to assign them a specific ID or class. This, however, can result in fragile styles. If we move elements around, our CSS rules may break. We can still use them in situations where we are certain about the location of elements.
- We can take advantage of pseudo-classes to target elements without the need to give them a specific class. The most common pseudo-classes are: `first-child, first-of-type, last-child, last-of-type` and `nth-child`. Pseudo-classes start with a single colon.
- With pseudo-elements we can style a part of an element. The most common pseudo-elements are: `first-letter, first-line, selection, before` and `after`. Pseudo-elements start with double colons.
- Selectors specificity determines the weight of a selector. When multiple selectors target the same element, the browser applies the selector with the higher specificity (weight). If two selectors have the same specificity, the one that comes last is the winner.
- ID selectors are the most specific selectors because we cannot have multiple elements with the same ID. Class and attribute selectors are less specific because we can have many elements with the same class and/or attributes. Element selectors are the least specific selectors.
- In VSCode, we can see the specificity of a rule by hovering our mouse over it. The specificity is represented using three numbers (x, y, z) where x represents the number of ID selectors, y represents the number of class/attribute selectors and z represents the number of element selectors.
- Some CSS properties inherit their value from their parent element. Typically, properties that are used for styling text such as text color, font, font size, etc are inherited. We can stop the inheritance by setting the value of a property to `initial`. To enforce inheritance, we should set the value of a property to `inherit`.
- We can specify colors by their name, hexadecimal value, RGB/RGBA value or HSL/HSLA value.

- RGBA and HSLA values include an alpha channel used for transparency. The value for the alpha channel is a decimal point number between 0 (completely transparent) and 1 (completely opaque).

- Using the `linear-gradient()` and `radial-gradient()` functions we can create gradients in CSS. Gradients are images so they cannot be used as the value of `background-color` property. We can use them as the value of `background-image` or `background` properties.

- The `border` property is a shorthand property for `border-top, border-right, border-bottom` and `border-left`. It takes three values: the thickness of the border, its style and its color.

- We also have specific properties like `border-width, border-style` and `border-color`. These properties take four values for the top, right, bottom and left borders.

- Using the `box-shadow` and `text-shadow` properties we can apply a shadow to elements and text. These properties take a few values. The first two values determine the horizontal and vertical distance of the shadow from the element. The third value (called blur radius) determines the softness of the border. We can specify the color as the fourth value.

CSS Cheat Sheet

Basic Selectors

`article`	All article elements
`.product`	Elements with the product class
`#products`	The element with the ID of products
`a[href="…"]`	Anchors with the given href
`a[href*="google"]`	Anchors whose href contains google
`a[href^="https"]`	Anchors whose href starts with https
`a[href$=".com"]`	Anchors whose href ends with .com

Relational Selectors

`#products p`	All p elements inside #products
`#products > p`	All p elements that are direct children of #products
`#products + p`	The p element immediately after #products (sibling)
`#products ~ p`	All p elements after #products (siblings)

Pseudo-class Selectors

`article :first-child`	The first child of article elements
`article :first-of-type`	The first occurrence of elements of different type
`article p:first-of-type`	The first p element inside article elements
`article :last-child`	
`article :last-of-type`	
`article :nth-child(odd)`	
`article :nth-child(even)`	

Pseudo-element Selectors

`p::first-letter`	The first letter of every p element
`p::first-line`	The first line of every p element
`::selection`	Any selected element
`p::before`	To insert content before the content of p elements
`p::after`	To insert content after the content of p elements

Colors

`#fcba03`	Hexadecimal value
`rgb(252, 186, 3)`	RGB value
`rgba(252, 186, 3, 0.5)`	Semi-transparent RGB value
`hsl(44, 98%, 50%)`	HSL value
`hsla(44, 98%, 50%, 0.5)`	Semi-transparent HSL value

Gradients

```
background: linear-gradient(blue, yellow);
background: linear-gradient(to bottom right, blue, yellow);
background: linear-gradient(45deg, blue, yellow);
background: linear-gradient(45deg, blue, yellow 30%);

background: radial-gradient(white, yellow);
background: radial-gradient(circle, white, yellow);
background: radial-gradient(circle at top left, white, yellow);
```

Borders

```
border: 10px solid blue;
border-width: 10px 20px 30px 40px;  /* top right bottom left */

border-radius: 5px;
border-radius: 100%; /* full circle */
```

Shadows

```
box-shadow: 10px 10px;
box-shadow: 10px 10px grey;
box-shadow: 10px 10px 5px grey;

text-shadow: 3px 3px 5px rgba(0, 0, 0, 0.2);
```