Codex Website Notes by Cornelius kandulu

. Creating your own website using *HTML* (HyperText Markup Language) and *CSS* (Cascading Style Sheets) is an essential skill for web development. Below is a comprehensive guide to help you learn HTML and CSS to create websites.

Introduction to HTML and CSS

- *HTML* is the backbone of any website. It defines the structure and content of a webpa

ge.

- *CSS* is used to style and lay out the page, making it visually appealing. It controls the presentation of the website, including colors, fonts, spacing, and positioning.

HTML Notes:

1. Basic Structure of an HTML Document

An HTML document typically consists of the following structure:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Your Website Title</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is your first webpage!</p>
</body>
</html>
```

- *`<!DOCTYPE html>`*: Declares the document type (HTML5).

- *`<html>`*: Root element of an HTML document.

- *`<head>`*: Contains metadata about the page (like the title, links to stylesheets, etc.).

- *`<body>`*: Contains the actual content that will appear on the webpage.

2. Common HTML Tags

- *Headings*: Define the structure and importance of titles.<h2>Blog Post Title</h2>

    This is the content of a blog post.

  </article>

  ```

- *`<section>`*: Represents a section of the document, such as a group of content that fo rms a thematic block.

  ```html

  <section>

    <h2>About Us</h2>

    <p>We are a web development company...</p>

  </section>

  ```

- *`<nav>`*: Defines a block of navigation links.

  ```html

  <nav>

    <ul>

      <li><a href="#">Home</a></li>

      <li><a href="#">Services</a></li>

      <li><a href="#">Contact</a></li>

    </ul>

```
</nav>
```

2. Forms in HTML

Forms are used to collect user input such as text, selections, and file uploads. Here's a m
ore detailed breakdown of form elements:

- *Input Elements*:

```html
<form action="/submit" method="POST">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name">

    <label for="email">Email:</label>
    <input type="email" id="email" name="email">

    <label for="password">Password:</label>
    <input type="password" id="password" name="password">

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

- *Checkboxes*:

```html
<form>
    <label><input type="checkbox" name="subscribe"> Subscribe to our newsletter
```

```
    </label>
</form>
```

- *Radio Buttons*:
```html
<form>
    <label>
        <input type="radio" name="gender" value="male"> Male
    </label>
    <label>
        <input type="radio" name="gender" value="female"> Female
    </label>
</form>
```

- *Select Dropdown*:
```html
<form>
    <label for="country">Country:</label>
    <select id="country" name="country">
        <option value="usa">United States</option>
        <option value="canada">Canada</option>
        <option value="uk">United Kingdom</option>
    </select>
</form>
```

- *Textareas*: Used for longer user input, such as a message or comment.

  ```html
  <form>
    <label for="message">Message:</label>
    <textarea id="message" name="message"></textarea>
  </form>
  ```

3. HTML Media Elements

- *Video*:

  ```html
  <video width="600" controls>
    <source src="movie.mp4" type="video/mp4">
    Your browser does not support the video tag.
  </video>
  ```

- *Audio*:

  ```html
  <audio controls>
    <source src="audio.mp3" type="audio/mp3">
    Your browser does not support the audio element.</audio>
  ```

CSS Advanced Concepts

1. CSS Flexbox Layout

Flexbox is a powerful one-dimensional layout system used to align and distribute spac
e within a container. It works well for both rows and columns.

- *Setting up Flexbox*:
  ```css
  .container {
      display: flex;
      justify-content: space-between; /* Align items horizontally */
      align-items: center; /* Align items vertically */
  }
  ```

- *Common Flexbox Properties*:
  - *`justify-content`*: Controls horizontal alignment.
  - *`align-items`*: Controls vertical alignment.
  - *`flex-direction`*: Determines the direction of items (row, column).
  - *`flex-wrap`*: Determines if the items should wrap onto the next line.

Example:
```css
.container {
  display: flex;
  justify-content: space-between;
```

```
    flex-wrap: wrap;
}

.item {
    flex: 1 1 30%; /* Flex-grow, flex-shrink, and basis */
}
```

2. CSS Grid Layout

CSS Grid is a two-dimensional system for creating grid-based layouts. It's more powerful than Flexbox when it comes to creating complex layouts.

- *Basic Grid Layout*:
  ```css
  .grid-container {
      display: grid;
      grid-template-columns: repeat(3, 1fr); /* 3 equal-width columns */
      gap: 10px; /* Space between grid items */CSS rules are written in a "selector" and "declaration" block:

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

Example:

```css
```
```

```
h1 {
    color: blue;
    font-size: 30px;
}
```

- *Selector*: The element to which the styles are applied (`h1` in this case).
- *Property*: The style you want to apply (`color` or `font-size`).
- *Value*: The value for that property (e.g., `blue` or `30px`).

2. CSS Selectors

- *Element Selector*: Targets HTML elements.
  ```css
  p {
      color: green;
  }
  ```

- *Class Selector*: Targets elements with a specific class.
  ```css
  .myClass {
      background-color: yellow;
  }
  ```

- *ID Selector*: Targets elements with a specific ID.
  ```css
  ```

```
#myId {
    border: 1px solid black;
}
```

- *Universal Selector*: Targets all elements on the page.

```css
* {
    margin: 0;
}
```

3. CSS Box Model

The CSS box model controls the layout of elements:

- *Content*: The actual content of the box (e.g., text).
- *Padding*: Space around the content inside the box.
- *Border*: The border surrounding the padding.
- *Margin*: Space outside the box.

Example of CSS box model usage:

```css
div {
    width: 200px;
    padding: 20px;
    border: 5px solid black;
    margin: 10px;
}
```

```*4. CSS Layout Techniques*

- *Flexbox*: A modern layout technique for creating responsive designs.
  ```css
  .container {
      display: flex;
      justify-content: center;
      align-items: center;
  }
  ```

- *Grid*: A two-dimensional layout system.
  ```css
  .grid-container {
      display: grid;
      grid-template-columns: 1fr 1fr 1fr;
  }
  ```

5. CSS Positioning

- *Static*: Default positioning for elements (normal flow).
- *Relative*: Positioned relative to its normal position.
  ```css
  .relative-box {
      position: relative;
      top: 10px;
      left: 10px;
```
```

```
}
```

- *Absolute*: Positioned relative to the nearest positioned ancestor.
- *Fixed*: Positioned relative to the browser window.
- *Sticky*: Elements stick at a defined point during scroll.

6. CSS Colors and Backgrounds

- *Color*: Use `color` for text color and `background-color` for background.

```css
body {
    color: #333;
    background-color: lightgray;
}
```

- *Gradient*: Create smooth transitions between colors.

```css
background: linear-gradient(to right, red, yellow);
```

- *Images as Background*: You can set background images using CSS.

```css
body {
    background-image: url('background.jpg');background-size: cover;
}
```

7. CSS Transitions and Animations

- *Transition*: Used to change properties smoothly over time.

```css
.box {
    transition: all 0.3s ease;
}

.box:hover {
    background-color: red;
}
```

- *Animation*: Used to create keyframe-based animations.

```css
@keyframes move {
    0% {
        left: 0;
    }
    100% {
        left: 100px;
    }
}

.animated-box {
    animation: move 2s infinite;
}
```

Building Your First Website:

1. *Create an HTML Structure*:
   - Begin with a simple HTML structure with headings, paragraphs, images, and links.
   - Use semantic HTML tags like `<header>`, `<footer>`, `<article>`, and `<section>` to structure your content.

2. *Apply CSS*:
   - Style your webpage using CSS.
   - Use colors, fonts, margins, paddings, and layouts to design your page.
   - Make the site responsive by using media queries.

3. *Add Interactivity with JavaScript (Optional)*:
   - Enhance user experience with JavaScript by adding interactivity such as form validation, animations, or dynamic content.

---

*Best Practices for HTML and CSS*1. *Semantic HTML*: Use proper HTML tags that describe the content, e.g., `<header>`, `<footer>`, `<article>`, and `<section>`.

2. *Use External CSS*: Instead of using inline styles, link external CSS files for better maintainability.

3. *Responsive Design*: Make sure your website looks good on all devices using techniques like media queries.

4. *Keep Code Clean*: Write readable and organized code, using comments and consistent indentation.

5. *Testing*: Always test your website in different browsers and devices to ensure compa
tibility.

Additional Resources to Learn HTML & CSS

1. *MDN Web Docs*: Comprehensive and detailed documentation.
   - [MDN HTML Documentation](https://developer.mozilla.org/en-US/docs/Web/HTML)
   - [MDN CSS Documentation](https://developer.mozilla.org/en-US/docs/Web/CSS)

2. *W3Schools*: Offers tutorials and references for HTML and CSS.
   - [W3Schools HTML](https://www.w3schools.com/html/)
   - [W3Schools CSS](https://www.w3schools.com/css/)

3. *FreeCodeCamp*: Offers free interactive coding lessons.
   - [FreeCodeCamp](https://www.freecodecamp.org/)

---

*Conclusion*Absolutely! Let's dive deeper into HTML and CSS with more advanced conc
epts and additional topics that will help you master web development and create profess
ional websites.

HTML Advanced Concepts

1. Semantic HTML

Semantic HTML tags describe the structure and content of the page in a meaningful wa
y, improving accessibility and SEO (Search Engine Optimization). These tags are importa
nt for creating well-organized, maintainable, and accessible websites.

- *`<header>`*: Represents the introductory section of a webpage or a section. It usually c
ontains navigation links, branding, and heading information.

```html
<header>
  <h1>My Website</h1>
  <nav>
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
      <li><a href="#">Services</a></li>
    </ul>
  </nav>
</header>
```

- *`<footer>`*: Represents the footer of a document or section. It typically contains inform
ation like copyright, contact details, and links.

```html
<footer>
  <p>&copy; 2025 My Website. All rights reserved.</p>
</footer>
```

- *`<article>`*: Represents a self-contained piece of content that could stand alone (e.g., b
log post, news article).

  ```html
  <article>}
  ```

- *Placing Items*:

  ```css
  .grid-item {
      grid-column: span 2; /* Span two columns */
      grid-row: span 2; /* Span two rows */
  }
  ```

Example of a 3x3 grid:

```css
.grid-container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: 100px 100px 100px;
    gap: 10px;
}

.grid-item {
    background-color: lightgray;
}
```

3. Advanced CSS Selectors

- *Child and Descendant Selectors*:
  - *`>`*: Selects direct children.
    ```css
    div > p {
        color: red;
    }
    ```

  - *` `* (space): Selects any descendant.
    ```css
    div p {
        color: blue;
    }
    ```

- *Adjacent Sibling Selector*:
  ```css
  h1 + p {
      color: green;
  }
  ```

- *Attribute Selectors*:
  - *`[attribute="value"]`*: Selects elements with a specific attribute.

```css
a[href="https://www.example.com"] {
    color: purple;
}
```

- *Pseudo-classes*:
  - *`:hover`*: When the user hovers over an element.
    ```css
    a:hover {
        text-decoration: underline;
    }
    ```
  - *`:focus`*: When an element has focus (like an input field).
    ```css
    input:focus {border: 2px solid blue;
    }
    ```

- *Pseudo-elements*:
  - *`::before`*: Adds content before an element.
  - *`::after`*: Adds content after an element.
  ```css
  p::before {
      content: "n  ";
      color: green;
```

}
```

---

*4. CSS Responsive Design*

Responsive design ensures that a website looks good on all screen sizes (desktops, table ts, smartphones). It is commonly achieved using media queries.

- *Basic Media Query Example*:
  ```css
  @media (max-width: 768px) {
      .container {
          flex-direction: column;
      }
  }
  ```

- *Common Breakpoints*:
  - *Mobile*: `max-width: 600px`
  - *Tablet*: `max-width: 768px`
  - *Desktop*: `min-width: 1024px`

5. CSS Transitions and Animations

CSS allows you to animate elements, which can make your website interactive and visua lly engaging.
  ```

- *Transition*:

  ```css
  .box {
    transition: background-color 0.3s ease;
  }

  .box:hover {
    background-color: blue;
  }
  ```

- *Animation*:

  ```css
  @keyframes fadeIn {
    0% { opacity: 0; }
    100% { opacity: 1; }
  }

  .box {
    animation: fadeIn 2s ease-in;
  }
  ```

---

*6. CSS Variables*CSS variables (also known as custom properties) allow you to define r

eusable values for properties, making it easier to maintain your stylesheets.

- *Define Variables*:
  ```css
  :root {
    --primary-color: #3498db;
    --font-size: 16px;
  }

  body {
    color: var(--primary-color);
    font-size: var(--font-size);
  }
  ```

Additional Resources for HTML & CSS Learning

- *CSS-Tricks*: Offers in-depth tutorials and guides.
  - [CSS-Tricks](https://css-tricks.com/)

- *W3C CSS Validator*: A tool to check for syntax errors in your CSS.
  - [W3C CSS Validator](https://jigsaw.w3.org/css-validator/)

- *WebAIM*: Learn about web accessibility and how to make websites usable for everyon
e.

- [WebAIM](https://webaim.org/)

- *CSS Layout Generator*: A helpful tool to visually generate layout code (Flexbox and Grid).
  - [CSS Layout Generator](https://www.layoutit.com/)

Conclusion

By mastering both *HTML* and *CSS*, you can create stunning, functional, and responsive websites. Start with simple projects like a personal portfolio or a landing page, and gradually incorporate advanced CSS techniques like Flexbox, Grid, and animations. Combine that with semantic HTML to improve accessibility and SEO.

Happy coding!