1. CSS का परिचय (Introduction to CSS)

1.1 CSS क्या है? (What is CSS?)

CSS (Cascading Style Sheets) एक स्टाइलिंग लैंग्वेज है जिसका उपयोग वेबपेज को आकर्षक और यूजर-फ्रेंडली बनाने के लिए किया जाता है। यह HTML एलिमेंट्स को **रंग, लेआउट, फॉन्ट और अन्य विजुअल प्रभाव** देने में मदद करता है।

CSS की विशेषताएँ:

- **प्रेजेंटेशन को कंट्रोल करता है**: HTML केवल वेबपेज की संरचना तय करता है, जबकि CSS उसकी स्टाइलिंग को नियंत्रित करता है।
- **कोड को अलग करता है**: CSS वेबपेज की डिज़ाइन को HTML से अलग रखता है जिससे कोड साफ और प्रबंधनीय बनता है।
- **रिस्पॉन्सिव डिजाइन बनाना आसान होता है**: CSS मीडिया क्वेरीज़ का उपयोग करके मोबाइल-फ्रेंडली डिज़ाइन बनाने में मदद करता है।
- **रीयूज़ेबल स्टाइलिंग**: CSS का उपयोग करके एक ही स्टाइल को कई वेबपेज में आसानी से लागू किया जा सकता है।

1.2 CSS के प्रकार (Types of CSS)

CSS को वेबपेज में जोड़ने के तीन मुख्य तरीके होते हैं:

1. Inline CSS

इसमें CSS को HTML टैग के अंदर `style` एट्रिब्यूट में लिखा जाता है।

Example:

YOGI ACADEMY
Struggle and Achieve your Goal

```
<p style=”color: blue; font-size: 20px;”>यह एक इनलाइन स्टाइलिंग है।</p>
```

फायदे:

- **छोटे बदलावों के लिए उपयोगी।**

नुकसान:

- **पूरा कोड जटिल और अनमैनेजेबल हो सकता है।**
- **एक ही स्टाइल को बार-बार दोहराना पड़ता है।**

---

2. Internal CSS

इसमें CSS को HTML फाइल के <head> सेक्शन में <style> टैग के अंदर लिखा जाता है।

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 { color: red; text-align: center; }
    p { font-size: 18px; }
  </style>
</head>
<body>
  <h1>यह एक इंटर्नल CSS स्टाइल है।</h1>
  <p>यह टेक्स्ट स्टाइल किया गया है।</p>
</body>
```

YOGI ACADEMY
Struggle and Achieve your Goal

```
</html>
```

फायदे:

- **पूरी वेबपेज के लिए स्टाइल को व्यवस्थित रूप से लागू कर सकते हैं।**

नुकसान:

- **प्रत्येक HTML पेज में CSS दोहराना पड़ता है जिससे कोड लंबा हो जाता है।**

---

3. External CSS

इसमें CSS को एक अलग .css **फ़ाइल में लिखा जाता है और उसे HTML पेज में** <link> **टैग के द्वारा जोड़ा जाता है।**

Example:

CSS फ़ाइल (style.css)

```
body {
   background-color: lightgray;
   font-family: Arial, sans-serif;
}
h1 {
   color: blue;
   text-align: center;
}
```

HTML फ़ाइल

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

YOGI ACADEMY
Struggle and Achieve your Goal

```
<head>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <h1>यह एक एक्सटर्नल CSS स्टाइल है।</h1>
</body>
</html>
```

फायदे:

- **पूरा वेबपेज एक ही स्टाइलशीट से नियंत्रित किया जा सकता है।**
- **कोड साफ और व्यवस्थित रहता है।**
- **एक ही CSS फाइल को कई HTML पेज में उपयोग किया जा सकता है।**

नुकसान:

- अगर CSS फ़ाइल लोड नहीं हुई तो पेज बिना स्टाइलिंग के दिखेगा।

1.3 CSS कैसे काम करता है?

जब कोई वेबपेज लोड होता है, ब्राउज़र HTML को पढ़ता है और अगर उसमें **CSS जोड़ी गई हो**, तो उसे लागू करता है। CSS तीन मुख्य स्टेप्स में काम करता है:

1. **CSS फाइल लोड होती है**: एक्सटर्नल, इंटर्नल या इनलाइन स्टाइल को लोड किया जाता है।
2. **ब्राउज़र CSS को पढ़ता है**: CSS सेलेक्टर्स के अनुसार स्टाइलिंग लागू करता है।
3. **फाइनल रेंडरिंग होती है**: ब्राउज़र CSS स्टाइल के अनुसार वेबपेज दिखाता है।

YOGI ACADEMY
Struggle and Achieve your Goal

1.4 CSS का उपयोग क्यों करें?

1. वेबसाइट की सुंदरता बढ़ाने के लिए

CSS से वेबपेज को अधिक आकर्षक और यूज़र-फ्रेंडली बनाया जा सकता है।

2. वेबसाइट को Responsive बनाने के लिए

CSS का उपयोग करके वेबसाइट को **मोबाइल, टैबलेट और डेस्कटॉप** के लिए अनुकूलित किया जा सकता है।

3. SEO (Search Engine Optimization) के लिए

CSS का उपयोग करके **क्लीन और ऑप्टिमाइज़ कोड** लिखा जाता है, जिससे पेज लोडिंग स्पीड तेज होती है और वेबसाइट सर्च इंजन में अच्छी रैंक करती है।

4. ब्राउज़र कम्पैटिबिलिटी सुधारने के लिए

CSS का उपयोग वेबपेज को **सभी ब्राउज़र्स (Chrome, Firefox, Edge, Safari)** में एक जैसा दिखाने के लिए किया जाता है।

YOGI ACADEMY
Struggle and Achieve your Goal

1.5 CSS का विकास (Evolution of CSS)

CSS को W3C (World Wide Web Consortium) द्वारा विकसित किया गया था और यह कई संस्करणों में **उपलब्ध है:**

CSS Version	Year	Features

CSS1	1996	**बेसिक स्टाइलिंग, फ़ॉन्ट, कलर, टेक्स्ट प्रॉपर्टीज**
CSS2	1998	**पेज लेआउट, पोजिशनिंग, मीडिया टाइप्स**
CSS3	2011	**बॉक्स शैडो, एनिमेशन, ग्रेडिएंट, ट्रांज़िशन**
CSS4	**(आने वाला)**	**एडवांस्ड लेआउट्स और नए सेलेक्टर्स**

1.6 CSS और HTML का संबंध

HTML वेबपेज की **संरचना (Structure)** को परिभाषित करता है, जबकि CSS उसे **डिज़ाइन (Design)** देता है। उदाहरण के लिए:

HTML Code:

```
<p>Hello, World!</p>
```

CSS Code:

```
p {
  color: blue;
  font-size: 20px;
}
```

Final Output:

👉 "Hello, World!" ब्लू रंग और बड़े फॉन्ट में दिखाई देगा।

YOGI ACADEMY
Struggle and Achieve your Goal

2. CSS Selectors & Properties

2.1 CSS Selectors (CSS सेलेक्टर्स)

CSS Selectors का उपयोग HTML एलिमेंट्स को स्टाइल देने के लिए किया जाता है। यह ब्राउज़र को बताता है कि किस HTML एलिमेंट पर कौन सी CSS प्रॉपर्टी लागू करनी है।

Basic Selectors (मूल सेलेक्टर्स)

Element Selector: किसी विशेष HTML टैग को स्टाइल करने के लिए।

```
p {
    color: blue;
```

1. }

   यह सभी <p> टैग्स का टेक्स्ट नीले रंग में बदल देगा।

ID Selector: # के साथ किसी एलिमेंट के ID को टारगेट करता है।

```
#header {
    background-color: gray;
```

2. }

   यह id="header" वाले एलिमेंट का बैकग्राउंड ग्रे कर देगा।

Class Selector: . (डॉट) का उपयोग करके किसी भी क्लास वाले एलिमेंट को स्टाइल किया जाता है।

```
.button {
    font-size: 18px;
    padding: 10px;
```

3. }

   यह class="button" वाले सभी एलिमेंट्स पर लागू होगा।

YOGI ACADEMY
Struggle and Achieve your Goal

Group Selector: एक साथ कई एलिमेंट्स को स्टाइल देने के लिए।

h1, h2, h3 {

text-align: center;

4. **}**

   यह सभी <h1>, <h2>**, और** <h3> **को सेंटर में अलाइन करेगा।**

Advanced Selectors (एडवांस सेलेक्टर्स)

Descendant Selector ( ): किसी विशेष एलिमेंट के अंदर मौजूद अन्य एलिमेंट्स को टारगेट करता है।

div p {

color: green;

5. **}**

   यह केवल उन <p> **टैग्स को प्रभावित करेगा जो** <div> **के अंदर होंगे।**

Child Selector (>): केवल डायरेक्ट चाइल्ड एलिमेंट्स को टारगेट करता है।

div > p {

font-weight: bold;

6. **}**

   यह केवल <div> **के अंदर के डायरेक्ट** <p> **चाइल्ड्स पर लागू होगा।**

Adjacent Sibling Selector (+): किसी एलिमेंट के तुरंत बाद आने वाले भाई-बहन एलिमेंट को स्टाइल करता है।

h1 + p {

font-style: italic;

7. **}**

   यह केवल उन <p> **एलिमेंट्स को प्रभावित करेगा जो** <h1> **के तुरंत बाद आते हैं।**

General Sibling Selector (~): किसी एलिमेंट के सभी भाई-बहन एलिमेंट्स को प्रभावित करता है।

h1 ~ p {

YOGI ACADEMY
Struggle and Achieve your Goal

```
color: red;
```

8. }

यह सभी <p> एलिमेंट्स को प्रभावित करेगा जो <h1> के बाद होंगे।

Attribute Selector: किसी HTML एलिमेंट के विशेष एट्रिब्यूट को टारगेट करता है।

input[type="text"] {

```
border: 2px solid blue;
```

9. }

यह सभी type="text" वाले <input> टैग्स को स्टाइल करेगा।

2.2 CSS Properties (CSS प्रॉपर्टीज)

CSS में कई प्रकार की प्रॉपर्टीज होती हैं, जो एलिमेंट्स की स्टाइलिंग के लिए उपयोग की जाती हैं।

1. Color & Background (रंग और बैकग्राउंड)

Text Color:

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

- }

Background Color:

```
body {
  background-color: lightgray;
```

- }

Background Image:

```
body {
  background-image: url('background.jpg');
```

- }

YOGI ACADEMY
Struggle and Achieve your Goal

2. Text & Font (टेक्स्ट और फ़ॉन्ट)

Font Size:
p {
  font-size: 20px;

- }

Font Family:
body {
  font-family: Arial, sans-serif;

- }

Text Alignment:
h1 {
  text-align: center;

- }

---

3. Box Model (बॉक्स मॉडल)

Margin (बाहरी स्पेस):

div {
  margin: 10px;

- }

Padding (भीतरी स्पेस):

div {
  padding: 15px;

- }

Border (बॉर्डर जोड़ना):

div {
  border: 2px solid black;

- }

YOGI ACADEMY
Struggle and Achieve your Goal

4. Positioning (स्थिति निर्धारण)

Static (डिफ़ॉल्ट): एलिमेंट अपने सामान्य स्थान पर रहेगा।

div {

position: static;

- }

Relative: एलिमेंट अपनी डिफ़ॉल्ट स्थिति के अनुसार मूव करेगा।

div {

position: relative;
left: 20px;

- }

Absolute: एलिमेंट अपने पेरेंट के अनुसार पोज़िशन लेता है।

div {

position: absolute;
top: 50px;
left: 50px;

- }

Fixed: एलिमेंट हमेशा एक निश्चित स्थान पर रहेगा।

div {

position: fixed;
bottom: 10px;
right: 10px;

- }

Sticky: एलिमेंट एक निश्चित स्थान पर स्क्रॉल होने तक स्थिर रहेगा।

div {

position: sticky;
top: 0;

- }

YOGI ACADEMY
Struggle and Achieve your Goal

5. Display & Visibility (प्रदर्शन और दृश्यता)

Display:
div {
 display: none;

- }

Visibility:
div {
 visibility: hidden;

- }

Block & Inline:
span {
 display: block;

- }

---

3. CSS Box Model (CSS बॉक्स मॉडल)

3.1 CSS Box Model क्या है?

CSS Box Model वेबपेज पर प्रत्येक HTML एलिमेंट को एक **बॉक्स** के रूप में ट्रीट करता है। इसमें चार मुख्य भाग होते हैं:

1. **Content (कंटेंट)** - असली टेक्स्ट, इमेज, या अन्य एलिमेंट जो बॉक्स के अंदर होता है।
2. **Padding (पैडिंग)** - कंटेंट और बॉर्डर के बीच का स्पेस।
3. **Border (बॉर्डर)** - बॉक्स के चारों ओर की बॉर्डर।

4. **Margin (मार्जिन)** - **बॉक्स के बाहर का स्पेस, जिससे अन्य एलिमेंट्स के साथ गैप बनाया जाता है।**

Box Model का चित्रात्मक प्रतिनिधित्व:

```
| Margin |
|----------------|
| Border         |
|----------------|
| Padding        |
|----------------|
| Content        |
|----------------|
```

3.2 Box Model का उपयोग

Example:

```
.box {
   width: 200px;
   height: 100px;
   padding: 10px;
   border: 2px solid black;
   margin: 20px;
}
```

<div class="box">यह एक बॉक्स है</div>

YOGI ACADEMY
Struggle and Achieve your Goal

यहाँ क्या होगा?

- **Width** = 200px
- **Height** = 100px
- **Padding = 10px (कंटेंट के चारों ओर)**
- **Border = 2px (बॉक्स के चारों ओर)**
- **Margin = 20px (अन्य एलिमेंट्स से दूरी बनाएगा)**

3.3 Padding, Border और Margin के उपयोग

1. Padding (भीतर की जगह बढ़ाने के लिए)

```
div {
    padding: 20px;
    background-color: lightgray;
}
```

2. Border (बॉक्स के चारों ओर बॉर्डर जोड़ने के लिए)

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

3. Margin (अन्य एलिमेंट्स से दूरी बढ़ाने के लिए)

```
div {
    margin: 30px;
}
```

3.4 Box-Sizing Property (बॉक्स का साइज मैनेज करना)

Box Model में **width और height** में सिर्फ कंटेंट शामिल होता है, लेकिन अगर हम चाहते हैं कि **padding और border** को भी इस साइज में जोड़ा जाए, तो हम `box-sizing` प्रॉपर्टी का उपयोग कर सकते हैं।

Example:

```
div {
    width: 200px;
    padding: 20px;
    border: 5px solid black;
    box-sizing: border-box;
}
```

`box-sizing: border-box;` लगाने से **width में padding और border शामिल हो जाएगा।**

3.5 Negative Margin (नेगेटिव मार्जिन का उपयोग)

YOGI ACADEMY
Struggle and Achieve your Goal

CSS में हम नेगेटिव मार्जिन का उपयोग करके एलिमेंट को ऊपर या बाईं ओर खिसका सकते हैं।

```
div {
    margin-top: -10px;
}
```

यह बॉक्स को 10px ऊपर ले जाएगा।

3.6 Auto Margin (सेंटर एलाइनमेंट के लिए)

अगर किसी बॉक्स को सेंटर में रखना है, तो `margin: auto;` का उपयोग किया जाता है।

```
div {
    width: 300px;
    margin: auto;
}
```

यह बॉक्स को हॉरिजॉन्टल सेंटर में लाएगा।

3.7 Border Radius (गोल बॉर्डर बनाने के लिए)

अगर बॉक्स को गोल करना है, तो border-radius का उपयोग करें।

YOGI ACADEMY
Struggle and Achieve your Goal

```
div {
    border-radius: 10px;
}
```

अगर बॉक्स को पूरी तरह गोल करना हो, तो:

```
div {
    border-radius: 50%;
}
```

---

4. CSS Positioning & Layout (CSS पोजीशनिंग और लेआउट)

4.1 CSS में पोजीशनिंग क्या होती है? (What is CSS Positioning?)

CSS पोजीशनिंग का उपयोग वेबपेज पर **HTML एलिमेंट्स की स्थिति** को नियंत्रित करने के लिए किया जाता है। CSS हमें यह निर्धारित करने की सुविधा देता है कि कोई एलिमेंट स्क्रीन पर कैसे दिखाई देगा और उसका स्थान अन्य एलिमेंट्स के सापेक्ष क्या होगा।

CSS में पोजीशनिंग के मुख्य प्रकार:

1. **Static (डिफ़ॉल्ट)**
2. **Relative (अपेक्षाकृत)**
3. **Absolute (पूर्ण)**
4. **Fixed (स्थिर)**
5. **Sticky (स्टिकी)**

4.2 CSS Position Property (पोजीशन प्रॉपर्टी)

YOGI ACADEMY
Struggle and Achieve your Goal

1. Static Position (डिफ़ॉल्ट स्थिति)

Static पोजीशन CSS की डिफ़ॉल्ट स्थिति होती है। इसमें एलिमेंट HTML के सामान्य प्रवाह के अनुसार ही दिखता है।

```
div {
  position: static; /* यह डिफ़ॉल्ट मान है */
}
```

विशेषताएँ:

- यह किसी भी अन्य प्रॉपर्टी से प्रभावित नहीं होता।
- पेज पर एलिमेंट अपने **सामान्य क्रम** में दिखाई देता है।

---

2. Relative Position (अपेक्षाकृत स्थिति)

Relative पोजीशन वाले एलिमेंट को **उसकी डिफ़ॉल्ट स्थिति से हटकर** रखा जा सकता है।

```
div {
  position: relative;
  top: 20px;
  left: 30px;
}
```

विशेषताएँ:

- यह एलिमेंट **अपने मूल स्थान से हटकर** नई स्थिति में जाता है।
- अन्य एलिमेंट्स पर कोई प्रभाव नहीं पड़ता।

---

3. Absolute Position (पूर्ण स्थिति)

Absolute पोजीशनिंग वाले एलिमेंट को सबसे नज़दीकी पोजीशन्ड पैरेंट के सापेक्ष रखा जाता है। यदि कोई पैरेंट position: relative; नहीं रखता है, तो यह पूरे वेबपेज के अनुसार काम करेगा।

YOGI ACADEMY
Struggle and Achieve your Goal

```
div {
    position: absolute;
    top: 50px;
    right: 30px;
}
```

विशेषताएँ:

- यह एलिमेंट **नॉर्मल डोक्यूमेंट फ्लो से बाहर** हो जाता है।
- इसे पेज पर किसी भी स्थान पर **सटीक रूप से रखा जा सकता है।**

---

4. Fixed Position (स्थिर स्थिति)

Fixed पोजीशन वाले एलिमेंट्स **स्क्रॉल करने पर भी अपनी स्थिति नहीं बदलते।**

```
div {
    position: fixed;
    bottom: 10px;
    right: 10px;
}
```

विशेषताएँ:

- यह हमेशा एक निश्चित स्थान पर रहेगा।
- इसे वेबसाइट के **हेडर और फुटर** जैसे एलिमेंट्स के लिए उपयोग किया जाता है।

YOGI ACADEMY
Struggle and Achieve your Goal

---

5. Sticky Position (स्टिकी स्थिति)

Sticky पोजीशन वाले एलिमेंट को **स्क्रॉल होने पर स्थिर** रखा जाता है, जब तक कि वह एक निश्चित सीमा तक स्क्रॉल न हो जाए।

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

विशेषताएँ:

- यह **स्क्रॉलिंग के आधार पर स्थिति बदलता है।**
- **यह नेविगेशन बार जैसे एलिमेंट्स के लिए उपयोग किया जाता है।**

4.3 CSS Layout Techniques (लेआउट तकनीकें)

CSS में **लेआउट डिज़ाइन** करने के लिए विभिन्न तकनीकों का उपयोग किया जाता है।

1. CSS Float Layout (फ्लोट लेआउट)

Float का उपयोग एलिमेंट्स **को बाएँ (left) या दाएँ (right)** स्थान देने के लिए किया जाता है।

```
div {
    float: left;
    width: 50%;
}
```

विशेषताएँ:

- **इसे टेक्स्ट और इमेज के चारों ओर कंटेंट को व्यवस्थित करने के लिए** उपयोग किया जाता है।
- **फ्लोट क्लियर करने के लिए** `clear: both;` **का उपयोग किया जाता है।**

YOGI ACADEMY
Struggle and Achieve your Goal

---

2. CSS Flexbox Layout (फ्लेक्सबॉक्स लेआउट)

Flexbox का उपयोग **रिस्पॉन्सिव वेब डिज़ाइन** के लिए किया जाता है। यह एलिमेंट्स को **स्ट्रेच, अलाइन और डायरेक्शन** देने के लिए उपयोगी है।

```
.container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 200px;
```

```
    border: 1px solid black;
}
```

विशेषताएँ:

- फ्लेक्सबॉक्स **हॉरिजॉन्टल और वर्टिकल** दोनों प्रकार के लेआउट के लिए उपयोगी है।
- `justify-content` और `align-items` से एलिमेंट्स को संरेखित किया जा सकता है।

---

3. CSS Grid Layout (ग्रिड लेआउट)

CSS Grid Layout का उपयोग **कंप्लेक्स वेब लेआउट** को बनाने के लिए किया जाता है।

```
.container {
    display: grid;
    grid-template-columns: 1fr 2fr;
    gap: 10px;
}
```

विशेषताएँ:

- यह **2D लेआउट सिस्टम** है।
- कॉलम और रो **को नियंत्रित करने** के लिए `grid-template-columns` और `grid-template-rows` का उपयोग किया जाता है।

YOGI ACADEMY
Struggle and Achieve your Goal

4.4 CSS Overflow Property (ओवरफ्लो प्रॉपर्टी)

जब कंटेंट किसी बॉक्स से बाहर निकलता है, तो इसे नियंत्रित करने के लिए `overflow` प्रॉपर्टी का उपयोग किया जाता है।

```
div {
    overflow: scroll;
}
```

मूल्य (Values):

- **visible: कंटेंट बॉक्स से बाहर दिखेगा।**
- **hidden: अतिरिक्त कंटेंट छिपा दिया जाएगा।**
- scroll: स्क्रॉलबार जोड़ दिया जाएगा।
- auto: आवश्यकता होने पर स्क्रॉलबार दिखाया जाएगा।

5. CSS Flexbox & Grid (CSS फ्लेक्सबॉक्स और ग्रिड)

5.1 CSS Flexbox क्या है? (What is CSS Flexbox?)

CSS Flexbox (Flexible Box) एक लेआउट मॉडल है जो वेबपेज के एलिमेंट्स को **हॉरिजॉन्टल (row) और वर्टिकल (column)** दिशा में **संरेखित (align)** करने के लिए उपयोग किया जाता है। यह **रिस्पॉन्सिव डिज़ाइन** बनाने में मदद करता है।

YOGI ACADEMY
Struggle and Achieve your Goal

Flexbox लागू करने के लिए:

```
.container {
  display: flex; /* फ्लेक्सबॉक्स सक्रिय करें */
}
```

5.2 Flexbox के मुख्य गुण (Important Flexbox Properties)

Flexbox दो भागों में बंटा होता है:

1. **Parent (Flex Container) – जिसमें `display: flex;` लगाया जाता है।**

2. **Children (Flex Items) – जो Flexbox के अंदर होते हैं।**

1. Flex Direction (दिशा निर्धारित करना)

Flexbox के एलिमेंट्स किस दिशा में होंगे, यह `flex-direction` से तय किया जाता है।

```
.container {
  display: flex;
  flex-direction: row; /* Default (एलिमेंट्स क्षैतिज होंगे) */
}
```

अन्य वैल्यूज़:

- `row-reverse`: **उल्टी दिशा में क्षैतिज रूप से।**
- `column`: **एलिमेंट्स को वर्टिकल (स्तंभ) रूप में संरेखित करता है।**
- `column-reverse`: **उल्टी दिशा में वर्टिकल रूप से।**

---

2. Justify Content (मुख्य दिशा में संरेखण)

एलिमेंट्स को मुख्य दिशा (horizontal) में संरेखित करने के लिए `justify-content` का उपयोग करें।

```
.container {
  justify-content: center; /* सभी आइटम्स को सेंटर में रखें */
}
```

अन्य वैल्यूज़:

- `flex-start`: **आइटम्स बाएँ ओर।**
- `flex-end`: **आइटम्स दाएँ ओर।**
- `space-between`: **पहला और आखिरी आइटम किनारों पर, बीच में समान स्पेस।**
- `space-around`: **आइटम्स के चारों ओर समान स्पेस।**

YOGI ACADEMY
Struggle and Achieve your Goal

3. Align Items (क्रॉस एक्सिस में संरेखण)

`align-items` का उपयोग आइटम्स को **क्रॉस-एक्सिस** (vertical) में संरेखित करने के लिए किया जाता है।

```
.container {
    align-items: center;
}
```

अन्य वैल्यूज़:

- `flex-start`: आइटम्स शीर्ष पर।
- `flex-end`: आइटम्स नीचे की ओर।
- `stretch`: आइटम्स पूरा स्थान भर देंगे।

---

4. Flex Wrap (एलिमेंट्स को दूसरी लाइन में ले जाना)

यदि आइटम्स ज्यादा हों और एक लाइन में न आ रहे हों, तो `flex-wrap` का उपयोग करें।

```
.container {
   flex-wrap: wrap; /* एलिमेंट्स अगली लाइन में चले जाएंगे */
}
```

अन्य वैल्यूज़:

- `nowrap` (डिफ़ॉल्ट): सभी आइटम्स एक ही लाइन में रहेंगे।
- `wrap-reverse`: आइटम्स को उलटी दिशा में दूसरी लाइन में ले जाता है।

---

5. Align Self (किसी विशेष आइटम के लिए अलग संरेखण)

```
.item {
    align-self: flex-end;
}
```

YOGI ACADEMY
Struggle and Achieve your Goal

5.3 CSS Grid क्या है? (What is CSS Grid?)

CSS Grid एक 2D लेआउट सिस्टम है जो पेज लेआउट को अधिक नियंत्रित करता है। यह Flexbox से अधिक शक्तिशाली है और कॉलम व रो दोनों में एलिमेंट्स को नियंत्रित करता है।

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

5.4 CSS Grid के मुख्य गुण (Important Grid Properties)

1. Grid Template Columns (कॉलम्स सेट करना)

```
.container {
  display: grid;
  grid-template-columns: 100px 200px auto;
}
```

अन्य तरीके:

- `repeat(3, 1fr);` **(तीन बराबर कॉलम)**
- `1fr 2fr 1fr;` **(पहला और तीसरा बराबर, दूसरा दुगुना)**

YOGI ACADEMY
Struggle and Achieve your Goal

---

2. Grid Template Rows (पंक्तियाँ सेट करना)

```
.container {
  grid-template-rows: 100px 200px auto;
}
```

---

3. Grid Gap (कॉलम्स और रो के बीच गैप)

```
.container {
  grid-gap: 10px;
}
```

---

4. Grid Item Positioning (ग्रिड में एलिमेंट्स को रखना)

Grid Column & Row Span

```
.item {
  grid-column: 1 / 3; /* पहला और दूसरा कॉलम कवर करेगा */
  grid-row: 1 / 2; /* पहली रो कवर करेगा */
}
```

---

5. Justify Items & Align Items

Grid में एलिमेंट्स को `justify-items` और `align-items` का उपयोग करके संरेखित किया जाता है।

```
.container {
  justify-items: center;
  align-items: center;
}
```

5.5 CSS Grid vs Flexbox (Flexbox और Grid का अंतर)

Feature	Flexbox	Grid
लेआउट प्रकार	1D (एक दिशा)	2D (कॉलम और रो)
उपयोगिता	छोटे UI तत्वों के लिए	पूरा वेब लेआउट
प्राथमिकता	कंटेंट-बेस्ड लेआउट	लेआउट-बेस्ड डिज़ाइन

YOGI ACADEMY
Struggle and Achieve your Goal

6. CSS Animations & Transitions (CSS एनिमेशन और ट्रांज़िशन)

6.1 CSS Transitions (CSS ट्रांज़िशन)

CSS ट्रांज़िशन का उपयोग **एलिमेंट्स के बीच स्मूथ इफेक्ट जोड़ने** के लिए किया जाता है।

Transition लागू करने के लिए:

```
div {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: background-color 0.5s ease-in-out;
}

div:hover {
  background-color: red;
}
```

इसका प्रभाव: जब यूज़र माउस को डिव पर ले जाएगा, तो बैकग्राउंड कलर **ब्लू से रेड** स्मूथली बदलेगा।

Transition Properties (मुख्य प्रॉपर्टीज़)

प्रॉपर्टी	विवरण
`transition-property`	**कौन-सी CSS प्रॉपर्टी बदलनी है (जैसे** `color`, `width` **आदि)**
`transition-duration`	**एनिमेशन का समय (जैसे** `0.5s`, `2s` **आदि)**
`transition-timing-function`	**इफेक्ट की गति (जैसे** `ease-in`, `ease-out`, `linear`)

YOGI ACADEMY
Struggle and Achieve your Goal

`transition-delay`	**कितनी देर बाद इफेक्ट लागू होगा**

6.2 CSS Animations (CSS एनिमेशन)

CSS एनिमेशन का उपयोग **कस्टम और मल्टी-स्टेप इफेक्ट्स** बनाने के लिए किया जाता है।

Animation लागू करने के लिए:

```
@keyframes example {
   0% { background-color: red; }
   50% { background-color: yellow; }
   100% { background-color: green; }
}

div {
   width: 100px;
   height: 100px;
   animation: example 3s infinite;
}
```

इसका प्रभाव:

- **0%: बैकग्राउंड रेड होगा।**
- **50%: बैकग्राउंड येलो होगा।**
- **100%: बैकग्राउंड ग्रीन होगा।**
- **Infinite: यह इफेक्ट लगातार चलता रहेगा।**

Animation Properties (मुख्य प्रॉपर्टीज़)

प्रॉपर्टी	**विवरण**
`animation-name`	`@keyframes` **में दिए गए नाम को सेट करता है**
`animation-duration`	**एनिमेशन को पूरा होने में लगने वाला समय**

YOGI ACADEMY
Struggle and Achieve your Goal

`animation-iteration-count`	**एनिमेशन कितनी बार चलेगा (`infinite` हमेशा के लिए)**
`animation-timing-function`	**गति का प्रकार (`ease-in`, `ease-out`, `linear`)**
`animation-delay`	**एनिमेशन कितनी देर बाद शुरू होगा**
`animation-direction`	`normal`, `reverse`, `alternate`

6.3 Hover, Focus, और Active State Animation

इन स्टेट्स का उपयोग इंटरएक्टिव एनिमेशन के लिए किया जाता है।

Hover Animation:

```
button {
  background-color: blue;
  transition: background-color 0.5s;
}
button:hover {
  background-color: green;
}
```

जब बटन पर माउस जाएगा, तो इसका रंग ब्लू से ग्रीन हो जाएगा।

Focus Animation:

```
input:focus {
  border: 2px solid red;
  transition: border 0.3s ease;
}
```

जब यूज़र इनपुट फील्ड पर क्लिक करेगा, तो इसका बॉर्डर रेड हो जाएगा।

Active Animation:

```
a:active {
  color: orange;
```

```
}
```

जब यूज़र किसी लिंक को क्लिक करेगा, तो वह ऑरेंज रंग में दिखेगा।

6.4 Transform Property (CSS ट्रांसफॉर्म)

Transform का उपयोग **एलिमेंट्स को घुमाने, स्केल करने, और ट्रांसलेट करने** के लिए किया जाता है।

Scale (साइज़ बढ़ाना या घटाना):

```
div {
  transform: scale(1.2); /* 1.2 गुना बड़ा */
}
```

Rotate (एलिमेंट घुमाना):

```
div {
  transform: rotate(45deg); /* 45 डिग्री घुमाना */
}
```

Translate (स्थान बदलना):

```
div {
  transform: translate(50px, 20px); /* 50px दाएँ, 20px नीचे */
}
```

Skew (एलिमेंट झुकाना):

```
div {
  transform: skew(20deg); /* 20 डिग्री झुकाना */
}
```

6.5 Combining Transforms & Animations

एनिमेशन और ट्रांसफॉर्म को एक साथ उपयोग किया जा सकता है।

YOGI ACADEMY
Struggle and Achieve your Goal

```
@keyframes moveBox {
    0% { transform: translateX(0); }
    100% { transform: translateX(200px); }
}

div {
    animation: moveBox 2s infinite alternate;
}
```

इसका प्रभाव: बॉक्स 0px से 200px तक **दाएँ-बाएँ मूव** करेगा।

---

7. CSS Variables & Functions (CSS वेरिएबल्स और फंक्शन्स)

7.1 CSS Variables क्या हैं? (What are CSS Variables?)

CSS Variables (Custom Properties) का उपयोग CSS कोड को **डायनामिक और रीयूज़ेबल** बनाने के लिए किया जाता है। यह कोड को मैनेज करने में मदद करता है और भविष्य में बदलाव करना आसान बनाता है।

CSS Variable डिफाइन और उपयोग करने का तरीका:

```
:root {
    --primary-color: blue;
    --secondary-color: green;
    --font-size: 16px;
}

body {
    background-color: var(--primary-color);
    color: white;
```

```
    font-size: var(--font-size);
}
```

`--primary-color` और `--font-size` जैसी कस्टम प्रॉपर्टीज़ को `var()` फ़ंक्शन के ज़रिए इस्तेमाल किया जाता है।

CSS Variables के फायदे:

- पूरे CSS में **रिपीट होने वाली वैल्यूज़** को मैनेज करना आसान हो जाता है।
- **थीमिंग** के लिए उपयोगी (जैसे **डार्क मोड** और **लाइट मोड**)।
- किसी भी **CSS प्रॉपर्टी के साथ उपयोग किया जा सकता है।**

7.2 CSS Variables का स्कोप (Scope of CSS Variables)

CSS Variables दो प्रकार के होते हैं:

Global Variables (वैश्विक वेरिएबल्स): `:root` में डिफाइन होते हैं और पूरे डॉक्यूमेंट में उपलब्ध होते हैं।

```
:root {
    --main-bg: lightgray;
}
body {
    background-color: var(--main-bg);
```

1. }

Local Variables (स्थानीय वेरिएबल्स): किसी विशेष सेलेक्टर के अंदर डिफाइन होते हैं और उसी सेलेक्टर के अंदर उपयोग किए जा सकते हैं।

```
.container {
    --container-padding: 20px;
    padding: var(--container-padding);
```

2. }

YOGI ACADEMY
Struggle and Achieve your Goal

7.3 CSS Functions (CSS फंक्शन्स)

CSS में कई उपयोगी **बिल्ट-इन फंक्शन्स** होते हैं जो **कलर, साइज़ और अन्य स्टाइलिंग प्रॉपर्टीज़** में डायनामिक बदलाव करने में मदद करते हैं।

1. calc() – गणना करने के लिए

यह फ़ंक्शन गणितीय ऑपरेशंस (`+`, `-`, `*`, `/`) का उपयोग करके डायनामिक साइज़ सेट करने की सुविधा देता है।

```
.box {
   width: calc(100% - 50px);
}
```

इसका प्रभाव: बॉक्स की चौड़ाई **विंडो की चौड़ाई से 50px कम** होगी।

---

2. max() और min() – अधिकतम और न्यूनतम वैल्यू सेट करने के लिए

```
.container {
   width: max(50%, 300px);
}
```

इसका प्रभाव: `container` की चौड़ाई **कम से कम 300px** रहेगी, लेकिन अगर 50% इससे बड़ा होगा तो वही लिया जाएगा।

```
.box {
   height: min(80vh, 500px);
}
```

इसका प्रभाव: बॉक्स की ऊँचाई अधिकतम 500px या 80vh में से जो भी छोटा होगा, वह रहेगा।

---

3. clamp() – डायनामिक रिस्पॉन्सिव साइज़िंग के लिए

```
h1 {
   font-size: clamp(16px, 5vw, 40px);
```

}

इसका प्रभाव:

- **फ़ॉन्ट साइज़ कम से कम 16px, अधिकतम 40px, और नॉर्मल व्यू में 5vw रहेगा।**

---

4. rgb() और rgba() – कलर को पारदर्शिता के साथ सेट करना

```
p {
  color: rgba(255, 0, 0, 0.5); /* 50% ट्रांसपेरेंसी वाला रेड */
}
```

---

5. hsl() और hsla() – कलर सेट करने का एक और तरीका

```
div {
  background-color: hsl(240, 100%, 50%); /* ब्लू कलर */
}
```

7.4 CSS Variables & Functions को एक साथ उपयोग करना

YOGI ACADEMY
Struggle and Achieve your Goal

```
:root {
  --base-size: 10px;
  --dynamic-width: calc(var(--base-size) * 5);
}

.box {
  width: var(--dynamic-width);
  height: var(--dynamic-width);
  background-color: rgba(0, 0, 255, 0.7);
}
```

इसका प्रभाव: बॉक्स की चौड़ाई और ऊँचाई 50px (10px * 5) होगी और इसका बैकग्राउंड 50% ट्रांसपेरेंट ब्लू होगा।

8. CSS Media Queries & Responsive Design (CSS मीडिया क्वेरीज़ और रिस्पॉन्सिव डिज़ाइन)

8.1 CSS Media Queries क्या हैं? (What are CSS Media Queries?)

Media Queries का उपयोग **डिवाइस के स्क्रीन साइज के आधार पर अलग-अलग CSS स्टाइल लागू करने** के लिए किया जाता है। यह वेबसाइट को **मोबाइल-फ्रेंडली और रिस्पॉन्सिव** बनाता है।

Basic Media Query Example:

```
@media (max-width: 600px) {
    body {
        background-color: lightblue;
    }
}
```

इसका प्रभाव: जब स्क्रीन की चौड़ाई **600px या उससे कम** होगी, तो बैकग्राउंड का रंग लाइट ब्लू हो जाएगा।

8.2 Media Query Syntax (मीडिया क्वेरी का सिंटैक्स)

```
@media (media-type) and (condition) {
    /* CSS Rules */
}
```

Example:

```
@media screen and (max-width: 768px) {
    p {
        font-size: 14px;
    }
}
```

Explanation:

- `screen`: स्क्रीन डिवाइसेस के लिए लागू होगा।
- `(max-width: 768px)`: जब स्क्रीन 768px या उससे छोटी होगी, तब CSS लागू होगी।

8.3 Common Media Query Breakpoints (आम तौर पर उपयोग होने वाले ब्रेकप्वाइंट्स)

डिवाइस टाइप	ब्रेकप्वाइंट (max-width)
मोबाइल	600px
टैबलेट	768px
लैपटॉप	1024px
डेस्कटॉप	1200px

YOGI ACADEMY
Struggle and Achieve your Goal

8.4 Responsive Typography (रिस्पॉन्सिव फ़ॉन्ट साइज)

फॉन्ट साइज को **व्यूपोर्ट की चौड़ाई** के अनुसार सेट करना:

```
h1 {
    font-size: calc(1rem + 2vw);
}
```

इसका प्रभाव: फ़ॉन्ट साइज स्क्रीन की चौड़ाई के अनुसार ऑटोमैटिक एडजस्ट होगा।

8.5 Flexbox के साथ Responsive Design

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

.item {
    flex: 1 1 300px; /* न्यूनतम 300px साइज़ रहेगा */
}
```

इसका प्रभाव: जब स्क्रीन छोटी होगी, तो **फ्लेक्स आइटम्स** स्वतः नई लाइन में चले जाएंगे।

8.6 Grid Layout के साथ Responsive Design

YOGI ACADEMY
Struggle and Achieve your Goal

```
.container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
}
```

इसका प्रभाव: ग्रिड कॉलम्स **स्क्रीन के अनुसार ऑटोमैटिक एडजस्ट** होंगे।

8.7 Dark Mode Implementation (डार्क मोड लागू करना)

```
@media (prefers-color-scheme: dark) {
    body {
        background-color: black;
        color: white;
    }
}
```

इसका प्रभाव: यदि यूज़र का सिस्टम डार्क मोड पर सेट है, तो वेबसाइट का **बैकग्राउंड ब्लैक और टेक्स्ट व्हाइट** हो जाएगा।

YOGI ACADEMY
Struggle and Achieve your Goal

8.8 Mobile-First Approach (मोबाइल-फर्स्ट डिज़ाइन)

मोबाइल-फर्स्ट डिज़ाइन का मतलब है **पहले छोटी स्क्रीन के लिए डिज़ाइन बनाना**, फिर बड़ी स्क्रीन के लिए **एडजस्ट करना।**

/* मोबाइल के लिए डिफ़ॉल्ट स्टाइल */

```
p {
```

```
    font-size: 14px;
}


/* बड़ी स्क्रीन के लिए */
@media (min-width: 768px) {
    p {
        font-size: 18px;
    }
}
```

इसका प्रभाव:

- **मोबाइल पर फ़ॉन्ट साइज 14px रहेगा।**
- **टैबलेट/डेस्कटॉप पर यह 18px हो जाएगा।**

YOGI ACADEMY
Struggle and Achieve your Goal

9. CSS Best Practices & Performance Optimization (CSS बेस्ट प्रैक्टिसेज़ और परफॉर्मेंस ऑप्टिमाइज़ेशन)

9.1 CSS Best Practices (CSS के बेहतरीन अभ्यास)

CSS को प्रभावी ढंग से लिखने और मेंटेन करने के लिए कुछ बेस्ट प्रैक्टिसेस अपनाई जानी चाहिए।

1. External Stylesheets का उपयोग करें

इंटरनल या इनलाइन स्टाइल के बजाय **एक्सटर्नल CSS फाइल** का उपयोग करें।

```
<link rel="stylesheet" href="styles.css">
```

फायदा: यह कोड को साफ और मेंटेनेबल बनाता है।

---

2. CSS Reset या Normalize का उपयोग करें

अलग-अलग ब्राउज़र में एक समान लुक पाने के लिए `CSS Reset` या `Normalize.css` का उपयोग करें।

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

फायदा: यह सभी ब्राउज़र में स्टाइलिंग को एक समान करता है।

---

3. DRY Principle (Don't Repeat Yourself) अपनाएँ

बार-बार कोड लिखने के बजाय, **CSS Variables और Classes** का उपयोग करें।

```
:root {
    --primary-color: #3498db;
}
.button {
    background-color: var(--primary-color);
```

YOGI ACADEMY
Struggle and Achieve your Goal

}

फायदा: इससे कोड छोटा और रीयूज़ेबल होता है।

---

4. Semantic CSS Naming Convention अपनाएँ

क्लास नाम को अर्थपूर्ण रखें।

```
/* गलत */
.red-text { color: red; }

/* सही */
.error-message { color: red; }
```

फायदा: कोड समझने और अपडेट करने में आसानी होती है।

9.2 CSS Performance Optimization (CSS की परफॉर्मेंस को बेहतर बनाना)

वेबसाइट की लोडिंग स्पीड बढ़ाने और परफॉर्मेंस सुधारने के लिए निम्नलिखित तकनीकों का उपयोग करें:

1. Minify और Compress करें

CSS फाइल को Minify करके लोडिंग स्पीड बढ़ाएँ।

```
# Minify CSS using a tool like UglifyCSS
uglifycss styles.css --output styles.min.css
```

फायदा: छोटे साइज की CSS फाइल तेज़ी से लोड होती है।

YOGI ACADEMY
Struggle and Achieve your Goal

2. Unused CSS हटाएँ

CSS में अनावश्यक स्टाइल्स को हटाने के लिए PurifyCSS या UnCSS टूल्स का उपयोग करें।

```
purifycss styles.css index.html --out cleaned-styles.css
```

फायदा: यह CSS फाइल का आकार कम करता है।

---

3. Critical CSS लोड करें

सबसे पहले ज़रूरी CSS लोड करें और बाकी बाद में लोड करें।

```
<style>
   /* Critical CSS */
   body { font-family: Arial, sans-serif; }
</style>
<link rel="stylesheet" href="styles.css" media="print" onload="this.onload=null;this.media='all';">
```

फायदा: वेबपेज तेज़ी से लोड होगा।

---

4. CDN (Content Delivery Network) का उपयोग करें

CSS फाइल्स को CDN से लोड करें।

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.1.3/css/bootstrap.min.css">
```

फायदा: वेबसाइट तेज़ी से लोड होती है क्योंकि फ़ाइल नज़दीकी सर्वर से लोड होती है।

---

5. Lazy Loading लागू करें

वेबसाइट पर इमेज और फॉन्ट्स के लिए Lazy Load का उपयोग करें।

YOGI ACADEMY
Struggle and Achieve your Goal

<img src="image.jpg" loading="lazy" alt="Example Image">

फायदा: इससे पेज की लोडिंग स्पीड बेहतर होती है।

---

6. Fewer HTTP Requests बनाएँ

- `@import` का उपयोग न करें, क्योंकि यह अधिक HTTP अनुरोध उत्पन्न करता है।
- CSS Sprites का उपयोग करें जिससे **एक ही इमेज में कई आइकन स्टोर किए जा सकें।**

```
.icon {
   background: url('sprite.png') no-repeat;
   width: 50px;
   height: 50px;
   background-position: -10px -20px;
}
```

फायदा: यह वेबसाइट की स्पीड को तेज़ करता है।

---

7. Prefetching और Preloading का उपयोग करें

वेबसाइट को तेज़ी से लोड करने के लिए प्रीलोडिंग टेक्नोलॉजी का उपयोग करें।

<link rel="preload" href="styles.css" as="style">

फायदा: यह ब्राउज़र को पहले से ज़रूरी संसाधन लोड करने की अनुमति देता है।

10. CSS Frameworks (Bootstrap, Tailwind CSS)

YOGI ACADEMY
Struggle and Achieve your Goal

10.1 CSS Frameworks क्या हैं? (What are CSS Frameworks?)

CSS Frameworks ऐसे प्री-डिफाइंड CSS कोड का सेट होते हैं जो डेवलपर्स को तेज़ी से और कुशलता से **रिस्पॉन्सिव वेब डिज़ाइन** बनाने में मदद करते हैं। ये फ्रेमवर्क **रेडीमेड CSS क्लासेस** प्रदान करते हैं, जिससे कोड लिखने की आवश्यकता कम हो जाती है।

CSS Frameworks के फायदे:

- **तेज़ वेब डेवलपमेंट** – रेडीमेड क्लासेस का उपयोग करके UI जल्दी डिज़ाइन किया जा सकता है।
- **रिस्पॉन्सिव डिज़ाइन** – मोबाइल-फ्रेंडली वेबपेज आसानी से बनाए जा सकते हैं।
- **क्रॉस-ब्राउज़र सपोर्ट** – सभी मुख्य ब्राउज़र्स में एक समान लुक मिलता है।
- **कम कोडिंग आवश्यक** – पहले से बने हुए स्टाइल्स का उपयोग किया जाता है।

10.2 Bootstrap (बूटस्ट्रैप)

Bootstrap सबसे लोकप्रिय CSS फ्रेमवर्क है, जिसे Twitter द्वारा विकसित किया गया था। यह **CSS, JavaScript, और jQuery** का उपयोग करके आसानी से **रिस्पॉन्सिव वेब डिज़ाइन** बनाने की सुविधा देता है।

Bootstrap को जोड़ने का तरीका:

```
<!-- Bootstrap CDN से जोड़ें -->

<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
```

Bootstrap की मुख्य सुविधाएँ:

YOGI ACADEMY
Struggle and Achieve your Goal

1. **Grid System (ग्रिड सिस्टम)** – 12-कॉलम लेआउट सिस्टम का उपयोग करता है।
2. **Pre-styled Components (बटन, कार्ड, अलर्ट आदि)** – रेडीमेड UI एलिमेंट्स।
3. **Form Styling** – फॉर्म इनपुट्स, चेकबॉक्स, रेडियो बटन आदि को स्टाइल करने की सुविधा।
4. **Utility Classes** – मार्जिन, पैडिंग, बैकग्राउंड आदि के लिए शॉर्टकट क्लासेस।

Bootstrap Grid System (लेआउट बनाने का तरीका)

```
<div class="container">
  <div class="row">
    <div class="col-md-6">Column 1</div>
    <div class="col-md-6">Column 2</div>
  </div>
</div>
```

इसका प्रभाव:

- **Desktop पर:** दोनों कॉलम बराबर होंगे।
- **Mobile पर:** कॉलम **स्वतः स्टैक** हो जाएंगे।

YOGI ACADEMY
Struggle and Achieve your Goal

Bootstrap Buttons (बूटस्ट्रैप बटन)

```
<button class="btn btn-primary">Primary Button</button>
<button class="btn btn-danger">Danger Button</button>
```

इसका प्रभाव: स्टाइलिश बटन बिना किसी कस्टम CSS के।

10.3 Tailwind CSS (टेलविंड CSS)

Tailwind CSS एक Utility-First CSS Framework है, जो डेवलपर्स को कस्टम डिज़ाइन बनाने की पूरी स्वतंत्रता देता है।

Tailwind CSS को जोड़ने का तरीका:

```
<!-- Tailwind CSS CDN -->

<script src="https://cdn.tailwindcss.com"></script>
```

Tailwind CSS की विशेषताएँ:

1. **Utility-First Approach – हर एलिमेंट को स्टाइल देने के लिए अलग-अलग क्लासेस।**
2. **Highly Customizable – CSS को पूरी तरह से कस्टमाइज़ किया जा सकता है।**
3. **No Predefined Components – Bootstrap की तरह रेडीमेड बटन या कार्ड नहीं होते।**

Tailwind Grid System (ग्रिड सिस्टम)

```
<div class="grid grid-cols-2 gap-4">

   <div class="bg-blue-500 p-4">Column 1</div>

   <div class="bg-green-500 p-4">Column 2</div>

</div>
```

इसका प्रभाव:

- **2 कॉलम समान आकार के होंगे।**
- **Mobile पर कॉलम स्टैक हो सकते हैं।**

Tailwind Buttons (टेलविंड बटन स्टाइलिंग)

```
<button class="bg-blue-500 text-white px-4 py-2 rounded">Primary Button</button>
```

YOGI ACADEMY
Struggle and Achieve your Goal

इसका प्रभाव: बूटस्ट्रैप की तरह प्री-डिफाइंड बटन नहीं है, लेकिन **CSS क्लासेस** को जोड़कर स्टाइल किया जा सकता है।

10.4 Bootstrap vs Tailwind CSS (Bootstrap और Tailwind CSS की तुलना)

विशेषता	Bootstrap	Tailwind CSS
स्टाइलिंग अप्रोच	**रेडीमेड क्लासेस**	**कस्टमाइज़ेबल यूटिलिटी क्लासेस**
डिजाइन फ्लेक्सिबिलिटी	सीमित	बहुत अधिक
परफॉर्मेंस	भारी (Extra CSS लोड)	हल्का (Minimal CSS)
सीखने की जटिलता	आसान	थोड़ी कठिन
प्रोजेक्ट टाइप	**तेज़ी से तैयार डिज़ाइन**	**पूरी तरह से कस्टम डिज़ाइन**

YOGI ACADEMY
Struggle and Achieve your Goal

10.5 CSS Frameworks कब और क्यों उपयोग करें?

स्थिति	Bootstrap उपयुक्त है	Tailwind उपयुक्त है
तेज़ी से UI डिज़ाइन बनाना हो	✅	❌
पूरा कस्टम डिज़ाइन चाहिए	❌	✅
वेबसाइट को हल्का रखना हो	❌	✅
रेडीमेड UI चाहिए	✅	❌

YOGI ACADEMY
Struggle and Achieve your Goal

11. Dark Mode & Theming in CSS (डार्क मोड और थीमिंग)

11.1 Dark Mode क्या है? (What is Dark Mode?)

Dark Mode एक UI फीचर है जो वेबसाइट या ऐप के बैकग्राउंड को **डार्क** और टेक्स्ट को **लाइट** बना देता है। यह **आंखों की थकान कम करने** और **बैटरी की बचत** के लिए उपयोग किया जाता है।

डार्क मोड के फायदे:

- **कम रोशनी में आंखों के लिए आरामदायक।**
- **OLED और AMOLED स्क्रीन पर बैटरी की बचत होती है।**
- **स्टाइलिश और आधुनिक लुक प्रदान करता है।**

11.2 CSS में Dark Mode लागू करना (Implementing Dark Mode in CSS)

1. Prefers Color Scheme से ऑटोमैटिक डार्क मोड

CSS में `prefers-color-scheme` **मीडिया क्वेरी का उपयोग करके डार्क मोड को ऑटोमैटिकली लागू किया जा सकता है।**

YOGI ACADEMY
Struggle and Achieve your Goal

```
@media (prefers-color-scheme: dark) {
   body {
      background-color: black;
      color: white;
   }
}
```

इसका प्रभाव: अगर यूज़र के सिस्टम में **डार्क मोड ऑन है**, तो वेबसाइट का बैकग्राउंड **ब्लैक** और टेक्स्ट **व्हाइट** हो जाएगा।

---

2. Manual Dark Mode Toggle (मैन्युअल थीम टॉगल बटन)

अगर हम चाहते हैं कि यूज़र खुद डार्क और लाइट मोड को स्विच कर सके, तो हमें JavaScript का उपयोग करना होगा।

Step 1: HTML में टॉगल बटन जोड़ें

```
<button id="theme-toggle">Toggle Dark Mode</button>
```

Step 2: CSS में डार्क मोड क्लास जोड़ें

```
body.dark-mode {
    background-color: black;
    color: white;
}
```

Step 3: JavaScript से थीम टॉगल करें

```
document.getElementById("theme-toggle").addEventListener("click", function() {
    document.body.classList.toggle("dark-mode");
});
```

इसका प्रभाव: बटन क्लिक करने पर **डार्क और लाइट मोड के बीच स्विच** होगा।

YOGI ACADEMY
Struggle and Achieve your Goal

11.3 CSS Variables के साथ Theming (Custom Themes using CSS Variables)

CSS Variables का उपयोग करके **मल्टीपल थीम** (जैसे डार्क, लाइट, कस्टम) आसानी से बनाई जा सकती हैं।

Step 1: CSS Variables डिफाइन करें

```
:root {
    --bg-color: white;
    --text-color: black;
}
```

```
body {
    background-color: var(--bg-color);
    color: var(--text-color);
}
```

Step 2: Dark Mode के लिए CSS Variables बदलें

```
body.dark-mode {
    --bg-color: black;
    --text-color: white;
}
```

Step 3: JavaScript से थीम बदलें

```
document.getElementById("theme-toggle").addEventListener("click", function() {
    document.body.classList.toggle("dark-mode");
});
```

11.4 Advanced Theming with Multiple Themes (एक से अधिक थीम लागू करना)

अगर हम अलग-अलग थीम (जैसे ब्लू थीम, रेड थीम) जोड़ना चाहते हैं, तो हम अलग-अलग CSS क्लासेस बना सकते हैं।

Step 1: CSS में थीम डिफाइन करें

```
:root {
    --primary-color: blue;
    --secondary-color: white;
}

body.red-theme {
    --primary-color: red;
    --secondary-color: yellow;
}
```

Step 2: JavaScript से थीम बदलें

```
function changeTheme(theme) {
```

YOGI ACADEMY
Struggle and Achieve your Goal

```
    document.body.className = theme;
}
```

Step 3: HTML में थीम बटन जोड़ें

```
<button onclick="changeTheme('red-theme')">Red Theme</button>
<button onclick="changeTheme('')">Default Theme</button>
```

इसका प्रभाव:

- रेड थीम बटन दबाने पर **रेड और येलो कलर** थीम सेट होगी।
- डिफ़ॉल्ट थीम बटन दबाने पर **ब्लू और व्हाइट कलर** थीम सेट होगी।

12. CSS Grid vs Flexbox - कब क्या उपयोग करें?

12.1 CSS Grid और Flexbox क्या हैं? (What are CSS Grid & Flexbox?)

CSS Grid और Flexbox दोनों ही **लेआउट सिस्टम** हैं, लेकिन इनके उपयोग के तरीके अलग-अलग होते हैं।

- **CSS Grid:** दो-आयामी (2D) लेआउट सिस्टम, जिसमें **कॉलम और रो** दोनों को नियंत्रित किया जाता है।
- **CSS Flexbox:** एक-आयामी (1D) लेआउट सिस्टम, जिसमें **केवल एक दिशा (row या column)** में कंट्रोल किया जाता है।

12.2 CSS Grid और Flexbox में अंतर (Key Differences Between CSS Grid & Flexbox)

विशेषता	CSS Grid	CSS Flexbox
लेआउट प्रकार	2D (कॉलम + रो)	1D (केवल row या column)
उपयोगिता	संपूर्ण वेबपेज लेआउट के लिए	छोटे UI घटकों के लिए
संरेखण (Alignment)	ग्रिड सेल्स को नियंत्रित करता है	केवल एक दिशा में आइटम्स संरेखित करता है
Flexibility (लचीलापन)	फिक्स्ड लेआउट के लिए बढ़िया	डायनामिक कंटेंट के लिए बढ़िया
Responsive Design	कॉलम और रो दोनों एडजस्ट होते हैं	केवल एक दिशा में एडजस्ट होते हैं
सीखने की कठिनाई	थोड़ा कठिन	आसान

YOGI ACADEMY
Struggle and Achieve your Goal

12.3 CSS Grid कब उपयोग करें? (When to Use CSS Grid?)

- जब आपको **पूरा वेबपेज** डिज़ाइन करना हो।
- जब कॉलम और रो दोनों की जरूरत हो।
- जब एक **कंप्लेक्स लेआउट** बनाना हो।

Example:

```
.container {
   display: grid;
   grid-template-columns: repeat(3, 1fr);
   grid-template-rows: auto;
```

```
    gap: 10px;
}
<div class="container">
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
</div>
```

इसका प्रभाव: यह 3 कॉलम का **ग्रिड सिस्टम** बनाएगा।

12.4 CSS Flexbox कब उपयोग करें? (When to Use CSS Flexbox?)

- जब **केवल एक दिशा (row या column)** में एलिमेंट्स को व्यवस्थित करना हो।
- जब एलिमेंट्स को **डायनामिक रूप से एडजस्ट** करना हो।
- जब **छोटे UI घटकों (नवबार, बटन ग्रुप, कार्ड)** को डिजाइन करना हो।

Example:

```
.container {
    display: flex;
    justify-content: space-between;
}
<div class="container">
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
</div>
```

इसका प्रभाव: सभी आइटम्स को **स्पेस के साथ बराबर दूरी पर** व्यवस्थित करेगा।

12.5 CSS Grid और Flexbox को एक साथ उपयोग करना (Using Grid & Flexbox Together)

हम Grid और Flexbox को मिलाकर भी उपयोग कर सकते हैं।

Example:

```
.container {
   display: grid;
   grid-template-columns: repeat(2, 1fr);
}

.item {
   display: flex;
   justify-content: center;
   align-items: center;
}
<div class="container">
   <div class="item">Flexbox Inside Grid</div>
   <div class="item">Another Item</div>
</div>
```

इसका प्रभाव:

- `container` **Grid का उपयोग करेगा।**
- `item` **Flexbox का उपयोग करेगा।**

YOGI ACADEMY
Struggle and Achieve your Goal

12.6 CSS Grid vs Flexbox - अंतिम निर्णय (Final Decision)

परिस्थिति	Grid उपयुक्त है	Flexbox उपयुक्त है
संपूर्ण वेबपेज लेआउट	✅	❌
केवल एक दिशा में कंटेंट	❌	✅
जटिल डिज़ाइन की जरूरत	✅	❌
डायनामिक कंटेंट	❌	✅

---

13. CSS for SEO & Accessibility (CSS का SEO और एक्सेसिबिलिटी में उपयोग)

13.1 CSS और SEO (CSS for Search Engine Optimization)

SEO (Search Engine Optimization) वेबसाइट को सर्च इंजनों में बेहतर रैंक दिलाने में मदद करता है। सही तरीके से लिखी गई CSS SEO को प्रभावित कर सकती है।

1. Page Load Speed (पेज लोडिंग स्पीड बढ़ाएँ)

गूगल पेज स्पीड को SEO रैंकिंग फैक्टर मानता है। CSS परफॉर्मेंस को बेहतर बनाने के लिए:

- **CSS Minification करें:** CSS फाइल का साइज छोटा करें।
- **Critical CSS Load करें:** पहले ज़रूरी CSS लोड करें, बाकी बाद में।
- **CSS का Unused Code हटाएँ:** PurifyCSS या UnCSS टूल्स का उपयोग करें।

```
<link rel="stylesheet" href="styles.css" media="print" onload="this.onload=null;this.media='all';">
```

2. Mobile-First Design (मोबाइल-फ्रेंडली डिज़ाइन)

Google Mobile-First Indexing को प्राथमिकता देता है, इसलिए CSS को मोबाइल के लिए ऑप्टिमाइज़ करें।

```
@media (max-width: 600px) {
   body {
      font-size: 14px;
   }
```

YOGI ACADEMY
Struggle and Achieve your Goal

```
}
```

3. Proper Heading Structure (सही हेडिंग स्ट्रक्चर)

सही <h1>, <h2>, <h3> का उपयोग SEO को बेहतर करता है।

```
<h1>Main Title</h1>

<h2>Subheading</h2>

<h3>Another Section</h3>
```

4. Readability & UX (पढ़ने में आसानी और उपयोगकर्ता अनुभव)

सही फोंट और कलर कॉन्ट्रास्ट का उपयोग करें:

```
body {
    font-size: 16px;
    color: #333;
    line-height: 1.6;
}
```

13.2 CSS और Accessibility (CSS for Web Accessibility)

वेबसाइट को एक्सेसिबल बनाना ज़रूरी है ताकि दिव्यांग उपयोगकर्ता भी इसका उपयोग कर सकें।

1. High Contrast Colors (उच्च कंट्रास्ट रंग)

कम विज़न वाले उपयोगकर्ताओं के लिए कंट्रास्ट सही रखें।

```
body {
    background-color: #fff;
    color: #000;
```

YOGI ACADEMY
Struggle and Achieve your Goal

```
}
```

2. Focus Styles (फोकस इंडिकेटर)

वेब एक्सेसिबिलिटी के लिए फोकस इंडिकेटर बनाएँ:

```
a:focus {
    outline: 2px solid blue;
}
```

3. Responsive Typography (रिस्पॉन्सिव टाइपोग्राफी)

EM और REM यूनिट्स का उपयोग करें ताकि फॉन्ट साइज स्क्रीन के अनुसार एडजस्ट हो सके।

```
body {
    font-size: 1rem;
}
```

4. Skip Navigation Links (नेविगेशन स्किप करने का ऑप्शन दें)

स्क्रीन रीडर उपयोगकर्ताओं के लिए "Skip to Main Content" लिंक जोड़ें।

```
<a href="#maincontent" class="skip-link">Skip to Main Content</a>
.skip-link {
    position: absolute;
    top: -40px;
    left: 10px;
}
.skip-link:focus {
    top: 10px;
}
```

YOGI ACADEMY
Struggle and Achieve your Goal

5. Use ARIA Attributes (ARIA एट्रिब्यूट्स का उपयोग करें)

ARIA (`Accessible Rich Internet Applications`) **से स्क्रीन रीडर्स को मदद मिलती है।**

```
<button aria-label="Close">X</button>
```

13.3 CSS & Semantic HTML (Semantic HTML के साथ सही CSS)

Semantic HTML (`<article>`, `<section>`, `<nav>`, `<header>`, `<footer>`) **SEO और एक्सेसिबिलिटी दोनों के लिए फायदेमंद होता है।**

```
<header>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
        </ul>
    </nav>
</header>
```

---

14. CSS for Print Media (Print Stylesheets)

14.1 Print Stylesheets क्या हैं? (What are Print Stylesheets?)

YOGI ACADEMY
Struggle and Achieve your Goal

Print Stylesheets वेबपेज को **प्रिंटिंग के लिए ऑप्टिमाइज़** करने में मदद करता है। यह सुनिश्चित करता है कि जब कोई वेबपेज को प्रिंट करे, तो केवल आवश्यक जानकारी ही अच्छी तरह से दिखे और **बेकार के एलिमेंट्स (जैसे नेविगेशन बार, विज्ञापन आदि) हट जाएं।**

14.2 Print Stylesheet को जोड़ने का तरीका (How to Add a Print Stylesheet)

```
<link rel="stylesheet" href="print.css" media="print">
```

इसका प्रभाव: `print.css` केवल तब लोड होगा जब कोई वेबपेज को प्रिंट करेगा।

14.3 Print-Specific CSS Rules (प्रिंटिंग के लिए उपयोगी CSS नियम)

1. Hidden Elements (अनावश्यक तत्वों को छुपाएँ)

कई बार कुछ एलिमेंट्स (जैसे बटन, वीडियो, साइडबार) प्रिंटिंग में जरूरी नहीं होते। इन्हें छुपाने के लिए `display: none;` का उपयोग करें।

```
@media print {
    nav, footer, .ads, .sidebar {
        display: none;
    }
}
```

2. Text Formatting (टेक्स्ट को प्रिंटिंग के लिए बेहतर बनाना)

YOGI ACADEMY
Struggle and Achieve your Goal

प्रिंट में स्पष्ट टेक्स्ट के लिए सही फॉन्ट्स और कलर का उपयोग करें।

```
@media print {
    body {
        font-family: Times, serif;
        font-size: 12pt;
        color: black;
        background: white;
    }
}
```

3. Links को दिखाना या छुपाना (Showing or Hiding Links in Print)

वेबपेज के हाइपरलिंक्स प्रिंट में नज़र नहीं आते, इसलिए उन्हें टेक्स्ट के रूप में दिखाने के लिए `content` प्रॉपर्टी का उपयोग करें।

```
@media print {
    a::after {
        content: " (" attr(href) ")";
        font-size: 10pt;
    }
}
```

इसका प्रभाव: लिंक (www.example.com) के रूप में प्रिंट होगा।

4. Page Breaks (प्रिंट पेज ब्रेक सेट करना)

अगर कोई सेक्शन बहुत बड़ा है और आप चाहते हैं कि वह नए पेज से शुरू हो, तो `page-break-before` और `page-break-after` का उपयोग करें।

YOGI ACADEMY
Struggle and Achieve your Goal

```
@media print {
    h1, h2 {
        page-break-before: always;
    }
    .no-break {
        page-break-inside: avoid;
    }
}
```

5. Adjusting Image Sizes (इमेज साइज को सही करना)

इमेजेज़ को सही साइज में लाने के लिए `max-width` का उपयोग करें।

```
@media print {
    img {
        max-width: 100%;
        height: auto;
    }
}
```

14.4 Print Stylesheet का पूरा उदाहरण (Complete Print Stylesheet Example)

```
@media print {
    body {
        font-family: Arial, sans-serif;
```

```
        font-size: 12pt;
        color: black;
        background: white;
    }
    nav, footer, .sidebar, .ads {
        display: none;
    }
    a::after {
        content: " (" attr(href) ")";
        font-size: 10pt;
    }
    h1, h2 {
        page-break-before: always;
    }
}
```

YOGI ACADEMY
Struggle and Achieve your Goal

15. Custom Scrollbars & Styling Forms in CSS (कस्टम स्क्रॉलबार और फॉर्म स्टाइलिंग)

15.1 Custom Scrollbars in CSS (CSS में कस्टम स्क्रॉलबार बनाना)

डिफ़ॉल्ट ब्राउज़र स्क्रॉलबार को कस्टमाइज़ करने के लिए `::-webkit-scrollbar` प्रॉपर्टी का उपयोग किया जाता है।

Basic Custom Scrollbar (साधारण स्क्रॉलबार स्टाइलिंग)

```
::-webkit-scrollbar {
    width: 10px;
}
::-webkit-scrollbar-track {
    background: #f1f1f1;
}
::-webkit-scrollbar-thumb {
    background: #888;
    border-radius: 5px;
}
::-webkit-scrollbar-thumb:hover {
    background: #555;
}
```

इसका प्रभाव:

- **स्क्रॉलबार की चौड़ाई 10px होगी।**
- **ट्रैक (पीछे का भाग) लाइट ग्रे (#f1f1f1) होगा।**
- **स्क्रॉलबार हैंडल गहरे ग्रे (#888) रंग का होगा।**

Rounded & Transparent Scrollbar (गोल और पारदर्शी स्क्रॉलबार)

```
::-webkit-scrollbar {
```

YOGI ACADEMY
Struggle and Achieve your Goal

```
    width: 8px;
}
::-webkit-scrollbar-thumb {
    background: rgba(0, 0, 0, 0.5);
    border-radius: 10px;
}
```

15.2 Styling Forms in CSS (CSS में फॉर्म को स्टाइल करना)

HTML फॉर्म को सुंदर और उपयोगकर्ता-अनुकूल बनाने के लिए CSS का उपयोग किया जाता है।

1. Basic Input Field Styling (साधारण इनपुट फील्ड स्टाइलिंग)

```
input {
    width: 100%;
    padding: 10px;
    border: 2px solid #ccc;
    border-radius: 5px;
    font-size: 16px;
}
input:focus {
    border-color: #007bff;
    outline: none;
}
```

इसका प्रभाव: इनपुट बॉक्स में फोकस करने पर बॉर्डर **ब्लू (#007bff)** हो जाएगा।

2. Custom Button Styling (बटन को स्टाइल करना)

```
button {
    background-color: #007bff;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
}
button:hover {
    background-color: #0056b3;
}
```

इसका प्रभाव: बटन का **डिफ़ॉल्ट रंग ब्लू (#007bff)** होगा और होवर करने पर यह **डार्क ब्लू (#0056b3)** हो जाएगा।

YOGI ACADEMY
Struggle and Achieve your Goal

---

3. Checkbox & Radio Button Customization (चेकबॉक्स और रेडियो बटन को कस्टमाइज़ करना)

```
input[type="checkbox"] {
    width: 18px;
    height: 18px;
    accent-color: #007bff;
```

```
}
input[type="radio"] {
    width: 18px;
    height: 18px;
    accent-color: #007bff;
}
```

इसका प्रभाव:

- चेकबॉक्स और रेडियो बटन **ब्लू (#007bff)** रंग में दिखेंगे।

---

4. Custom Placeholder Styling (प्लेसहोल्डर टेक्स्ट को स्टाइल करना)

```
input::placeholder {
    color: #888;
    font-style: italic;
}
```

इसका प्रभाव: प्लेसहोल्डर टेक्स्ट **गहरे ग्रे (#888)** और italic स्टाइल में होगा।

---

6. Real-World CSS Projects & Case Studies (वास्तविक दुनिया के CSS प्रोजेक्ट्स और केस स्टडीज़)

16.1 CSS का वास्तविक दुनिया में उपयोग (Practical Applications of CSS)

CSS का उपयोग सिर्फ़ वेबपेज को स्टाइल करने के लिए ही नहीं, बल्कि **प्रोफेशनल वेब डिज़ाइन, मोबाइल ऐप इंटरफेस, और इंटरएक्टिव वेब एप्लिकेशन** बनाने के लिए किया जाता है। नीचे कुछ **रियल-वर्ल्ड प्रोजेक्ट्स** दिए गए हैं जो CSS के प्रभावी उपयोग को दर्शाते हैं।

---

1. E-Commerce Website (ई-कॉमर्स वेबसाइट डिज़ाइन)

E-commerce वेबसाइट में CSS का उपयोग **रिस्पॉन्सिव ग्रिड, फ्लेक्सबॉक्स, और एनीमेशन** के लिए किया जाता है।

```
.product-card {
    display: flex;
    flex-direction: column;
    align-items: center;
    border: 1px solid #ddd;
    padding: 10px;
}
```

केस स्टडी: Amazon और Flipkart जैसी वेबसाइटें CSS Grid और Flexbox का उपयोग करके अपने प्रोडक्ट पेज डिज़ाइन करती हैं।

---

2. Portfolio Website (पोर्टफोलियो वेबसाइट डिज़ाइन)

```
.hero-section {
    background-image: url('background.jpg');
```

YOGI ACADEMY
Struggle and Achieve your Goal

```
    background-size: cover;
    text-align: center;
    padding: 50px 20px;
}
```

केस स्टडी: Behance और Dribbble जैसी पोर्टफोलियो साइट्स आकर्षक CSS डिज़ाइन का उपयोग करती हैं।

---

3. CSS Animations in Web Applications (वेब ऐप्स में CSS एनीमेशन)

```
@keyframes fadeIn {
    from { opacity: 0; }
    to { opacity: 1; }
}
.modal {
    animation: fadeIn 0.5s ease-in-out;
}
```

केस स्टडी: Facebook, Twitter, और Instagram जैसी सोशल मीडिया साइट्स CSS ट्रांज़िशन और एनीमेशन का उपयोग करती हैं।