HTML & CSS REVISION

<meta name="robots"> tag HTML ke head section mein use hota hai, aur iska kaam search engine crawlers (jaise Googlebot) ko batana hota hai ke wo page ko kaise handle karein aur index karein. Yeh ek **SEO** (Search Engine Optimization) related tag hai.

Syntax

```html
<meta name="robots" content="value">
```

Common content Values aur Unka Matlab

content attribute mein aap crawlers ke liye specific instructions dete hain. Common values aur unka meaning:

1. **index**
   - Page ko search engines ke index mein include karna hai.
   - Default behavior hai (agar robots tag na ho tab bhi).

```html
<meta name="robots" content="index">
```

2.noindex

- Search engines ko page ko apne index mein add na karne ke liye kehta hai.
- Page search results mein nahi dikhega.

```html
<meta name="robots" content="noindex">
```

3. **follow**

- Crawlers ko page ke links ko follow karne ki permission deta hai.
- Default behavior hai.

```html
<meta name="robots" content="follow">
```

nofollow

- Crawlers ko page ke links ko follow karne se rokta hai.
- Links ka "link juice" pass nahi hoga.

```html
<meta name="robots" content="nofollow">
```

5.Combine Values

- Aap ek se zyada values ek saath use kar sakte hain.
- Example

```html
<meta name="robots" content="noindex, nofollow">
```

- □
  - Page ko index nahi karna aur links ko follow nahi karna.

□ **none**

- Equivalent to noindex, nofollow.

```html
<meta name="robots" content="none">
```

Example Use Cases

1. Search Engines ko Page Index karne se Rokna
   - Agar aap chahte hain ke ek specific page search results mein na aaye:

   ```html
   <meta name="robots" content="noindex">
   ```

2. Private Pages
   - Jo pages private hain ya publicly searchable nahi hone chahiye:

   ```html
   <meta name="robots" content="noindex, nofollow">
   ```

3. SEO Optimization
   - Agar aap chahte hain ke sirf links ko follow kiya jaye lekin page ko index na kiya jaye:

   ```html
   <meta name="robots" content="noindex, follow">
   ```

Notes

- Agar aapko ek specific search engine ke liye instructions deni ho (e.g., Googlebot), to aap `name="googlebot"` ka use kar sakte hain:

```html
<meta name="googlebot" content="noindex">
```

- **Important**: Robots meta tag sirf us case mein kaam karega jab search engine crawler us page ko access kar sake. Agar `robots.txt` file mein crawling disable hai, toh yeh meta tag kaam nahi karega.

HTML mein <meta> tags ka use web pages ke metadata define karne ke liye hota hai. Metadata wo information hoti hai jo web browsers aur search engines ke liye hoti hai, lekin wo user ko page par directly nahi dikhai deti. Yahan kuch commonly used <meta> tags aur unka purpose diya gaya hai:

---

1. <meta charset>

- **Purpose**: Web page ki character encoding set karta hai.
- **Example**

```html
<meta charset="UTF-8">
```

Explanation:

- `UTF-8` ek common character encoding hai jo almost saare languages aur special characters ko support karti hai.
- Iske bina, characters galat dikh sakte hain.

2. `<meta name="viewport">`

- **Purpose**: Responsive design ke liye use hota hai. Yeh tag browsers ko batata hai ke page kaise scale hona chahiye, especially mobile devices par.
- **Example**:

```html
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- **Explanation**:
  - `width=device-width`: Page ka width device ki width ke equal hoga.
  - `initial-scale=1.0`: Default zoom level set karta hai (100% zoom).

3. `<meta name="description">`

- **Purpose**: Page ka short description provide karta hai jo search engine results mein dikhaya jata hai.
- **Example**:

```html
<meta name="description" content="This is a practice HTML and CSS project for learning
```

- **Explanation**:
  - Search engine results ke snippets ke liye use hota hai.
  - Description concise aur relevant honi chahiye.

4. `<meta name="keywords">`

- **Purpose**: Page ke related keywords define karta hai (SEO ke liye helpful hota tha, ab itna relevant nahi hai).
- **Example**:

```html
<meta name="keywords" content="HTML, CSS, Practice, Web Development">
```

- **Explanation**:
  - Keywords separate by commas likhe jate hain.
  - Modern search engines is par depend nahi karte.

5. `<meta name="author">`

- **Purpose**: Page ka author specify karta hai.
- **Example**:

```html
<meta name="author" content="John Doe">
```

- **Explanation**:
  - Useful hai jab aapko page ke creator ko mention karna ho.

6. `<meta http-equiv>`

- **Purpose**: HTTP headers ko simulate karta hai.
- **Common Uses**:
  - **Redirect to another URL**:

    ```html
    <meta http-equiv="refresh" content="5; url=https://example.com">
    ```

    - Iska matlab hai ke page 5 seconds ke baad automatically redirect ho jayega.
  - **Content-Type**:

    ```html
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    ```

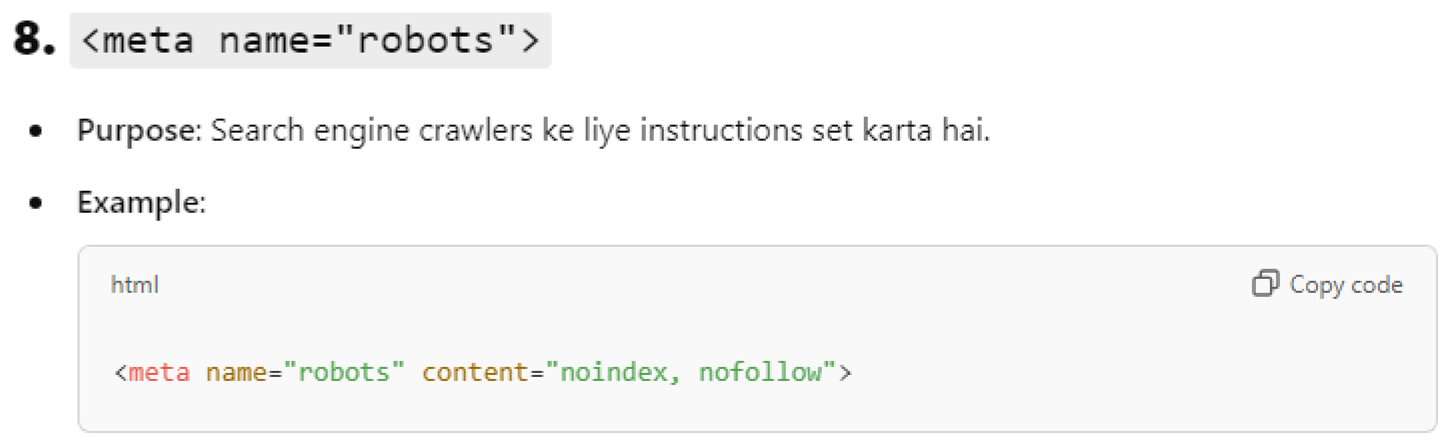
    - Yeh page ki content type aur encoding set karta hai.

7. `<meta name="theme-color">`

- **Purpose**: Browser toolbar ke background color ko set karta hai (mostly mobile devices ke liye).
- **Example**:

```html
<meta name="theme-color" content="#4CAF50">
```

- **Explanation**:
  - Android devices ke Chrome browser mein isse status bar ka color change hota hai.



8. `<meta name="robots">`

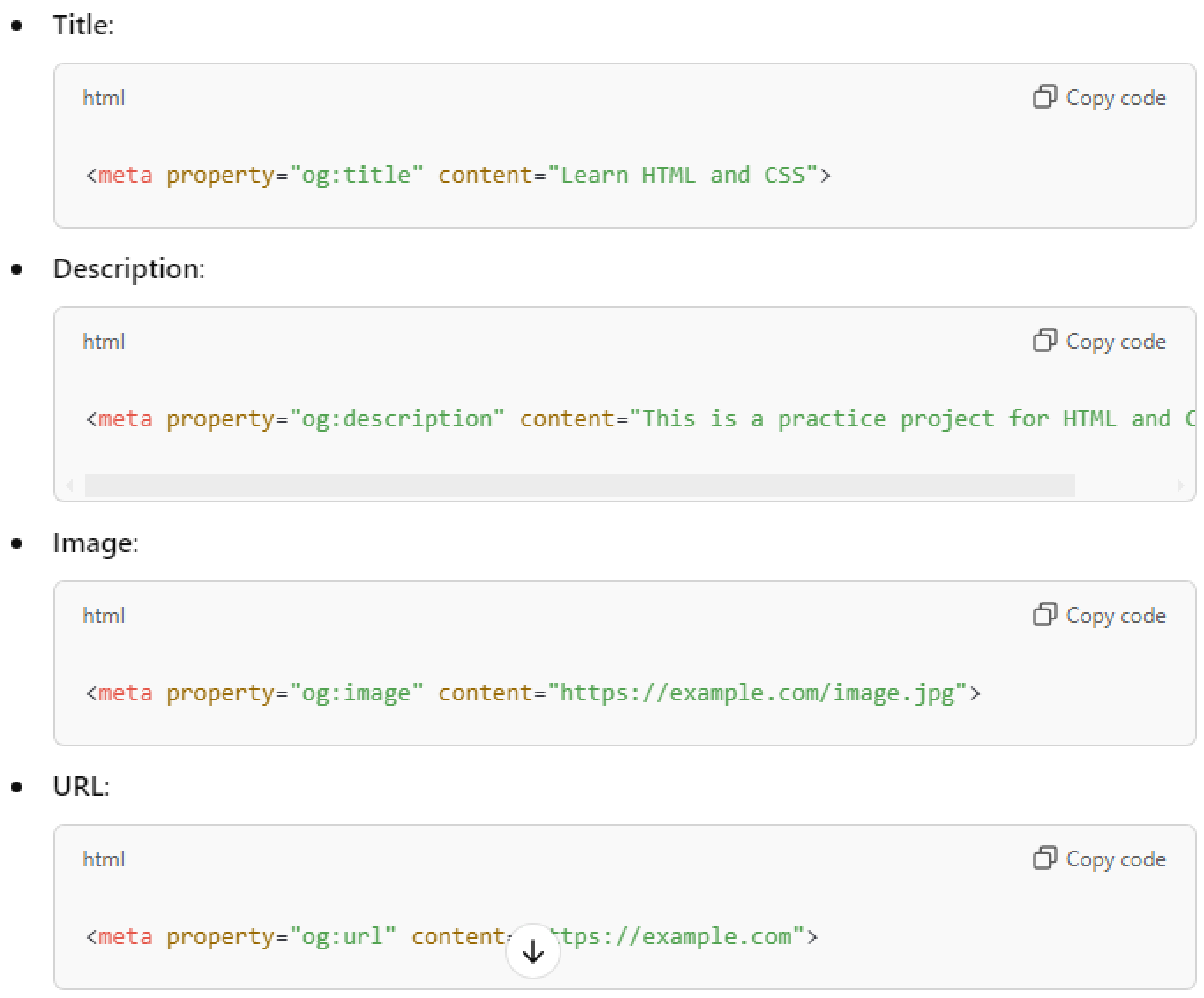
- **Purpose**: Search engine crawlers ke liye instructions set karta hai.
- **Example**:

html Copy code

```html
<meta name="robots" content="noindex, nofollow">
```

9. <meta property="og:*"> (Open Graph)

- **Purpose**: Social media platforms (jaise Facebook, Twitter) ke liye metadata set karta hai.
- **Common Properties**



- Title:

html Copy code

```html
<meta property="og:title" content="Learn HTML and CSS">
```

- Description:

html Copy code

```html
<meta property="og:description" content="This is a practice project for HTML and C
```

- Image:

html Copy code

```html
<meta property="og:image" content="https://example.com/image.jpg">
```

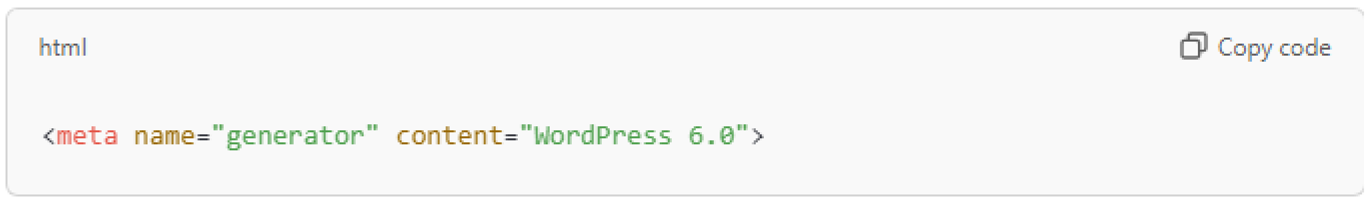
- URL:

html Copy code

```html
<meta property="og:url" content= ttps://example.com">
```

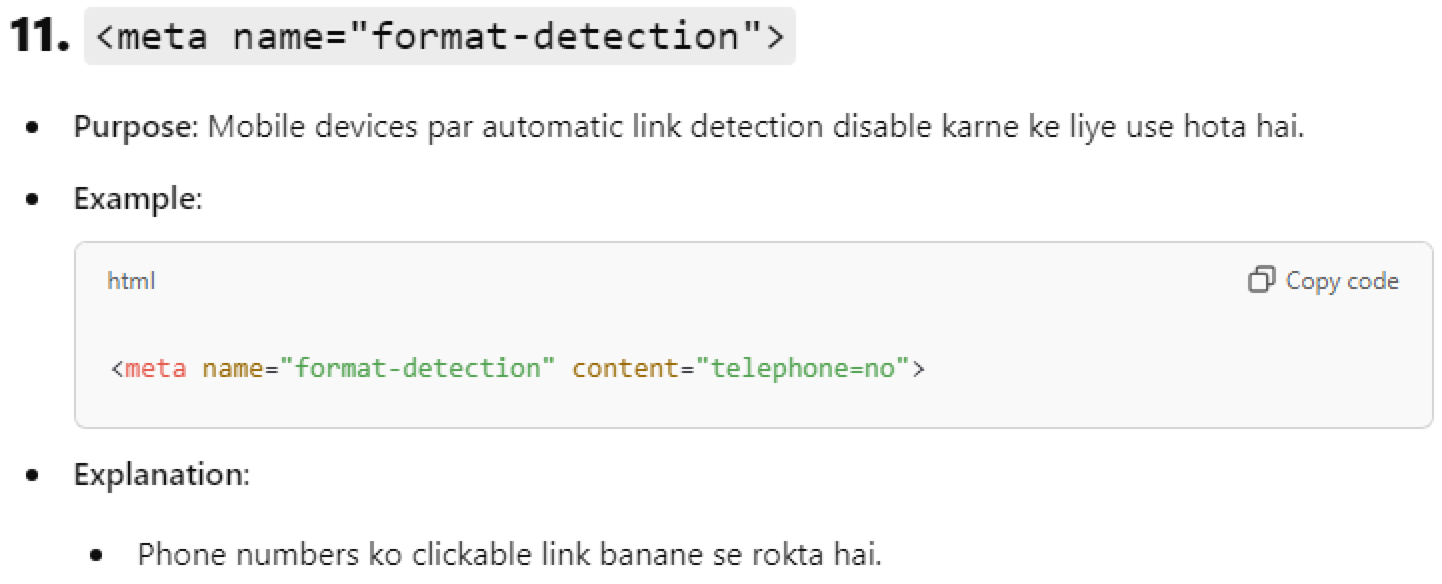
10. <meta name="generator">

- **Purpose**: Batata hai ki page kaunse tool ya platform se banaya gaya hai.
- **Example**:



html Copy code

```html
<meta name="generator" content="WordPress 6.0">
```



11. `<meta name="format-detection">`

- **Purpose:** Mobile devices par automatic link detection disable karne ke liye use hota hai.
- **Example:**

```html
<meta name="format-detection" content="telephone=no">
```

- **Explanation:**
  - Phone numbers ko clickable link banane se rokta hai.

Conclusion

<meta> tags kaafi useful hain web pages ke SEO aur usability ko improve karne ke liye. Yeh browsers aur search engines ko page ke bare mein essential information dete hain

****

CSS mein html { font-size: 62.5%; } dene ka maksad **font-size ko easier and more consistent units mein convert karna** hota hai, specifically **rem units** ke liye. Iska use karne ka reason aur iska impact yeh hai:

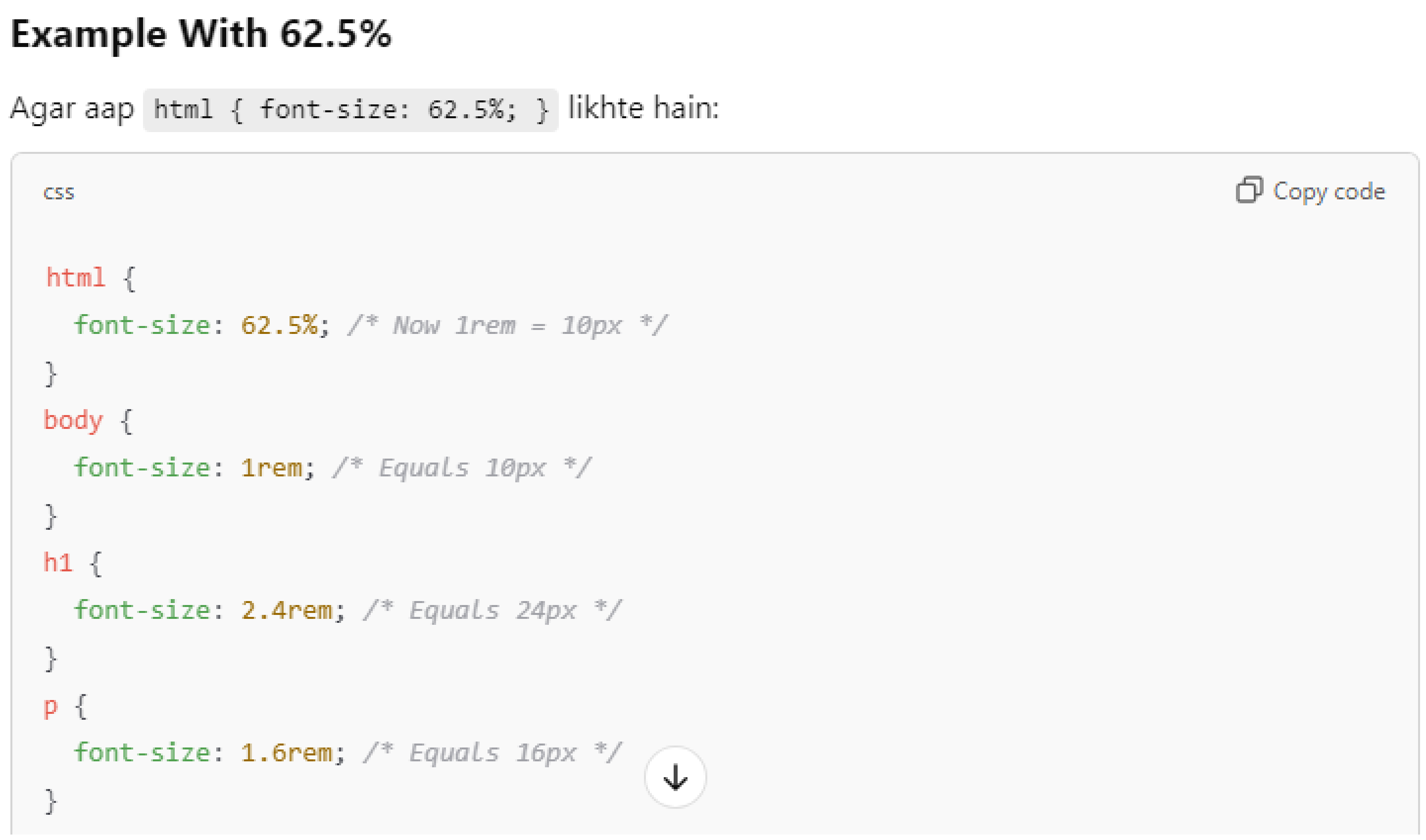
Default Font Size

- Web browsers ka default font size hota hai **16px**.
- Agar aap html { font-size: 62.5%; } likhte hain, to aap font size ko default ka **62.5%** set kar rahe hote hain: 16 px×62.5%=10 px16 \, \text{px} \times 62.5\% = 10 \, \text{px}16px×62.5%=10px

---

Benefits of Setting 62.5%

1. **Simplified Calculation**:
   - rem (Root Em) units ko calculate karna easy ho jata hai.
   - 1 rem = 10px, 2 rem = 20px, 1.5 rem = 15px, etc.
   - Iske bina, rem ke values ko calculate karna thoda complex hota hai:
     - Default setting pe 1 rem = 16px, 1.5 rem = 24px, etc.
2. **Consistency Across Design**:
   - Responsive designs mein har element ka size proportional rakhna asaan ho jata hai.
   - Aap poore project mein rem ka use karke consistent typography aur spacing maintain kar sakte hain.
3. **Scalability**:
   - Agar future mein font size change karna ho, toh sirf html ke font-size ko update karna padega, aur saare rem-based units automatically adjust ho jayenge.



Example With 62.5%

Agar aap `html { font-size: 62.5%; }` likhte hain:

```css
html {
  font-size: 62.5%; /* Now 1rem = 10px */
}
body {
  font-size: 1rem; /* Equals 10px */
}
h1 {
  font-size: 2.4rem; /* Equals 24px */
}
p {
  font-size: 1.6rem; /* Equals 16px */
}
```

Responsive Design Advantage

Agar aap font-size ko rem ke saath set karte hain aur user browser ke settings se font size ko increase ya decrease karta hai, toh rem units automatically adjust ho jate hain. Isse aapka design **accessibility-friendly** ban jata hai.

When to Use This Approach

- Jab aap rem units extensively use karna chahte hain.
- Jab aapko responsive designs ke saath consistent typography aur spacing maintain karni ho.
- Projects mein jahan scalable and maintainable CSS ki zarurat ho.

html { font-size: 62.5%; } ka use karne se **responsive font sizes** ko implement karna asaan ho jata hai, aur chhoti screens par font sizes ko adjust karne mein flexibility milti hai. Iska core concept yeh hai ke aap **rem units** ke saath kaam karte hain, jo root font size ke proportional hoti hain. Agar aap root font size ko media queries ke zariye change karte hain, toh saare rem-based font sizes automatically adjust ho jate hain.

Kaise Responsive Font Size Banai Jaye?

Step-by-Step Guide:

1. **Root Font Size Define Karein (Base Font Size):**

```css
html {
    font-size: 62.5%; /* Base font size = 10px */
}
```

2. Font Sizes Ko `rem` Mein Set Karein:

css Copy code

```css
body {
    font-size: 1.6rem; /* Equals 16px */
}
h1 {
    font-size: 2.4rem; /* Equals 24px */
}
p {
    font-size: 1.4rem; /* Equals 14px */
}
```

3. **Media Queries Ke Zariye Font Size Adjust Karein:** Small screens ke liye root font size reduce karein:

css Copy code

```css
@media (max-width: 768px) {
    html {
        font-size: 50%; /* Base font size = 8px */
    }
}

@media (max-width: 480px) {
    html {
        font-size: 45%; /* Base font size = 7.2px */
    }
}
```

   - Isse saare rem-based font sizes automatically proportionally chhote ho jayenge.

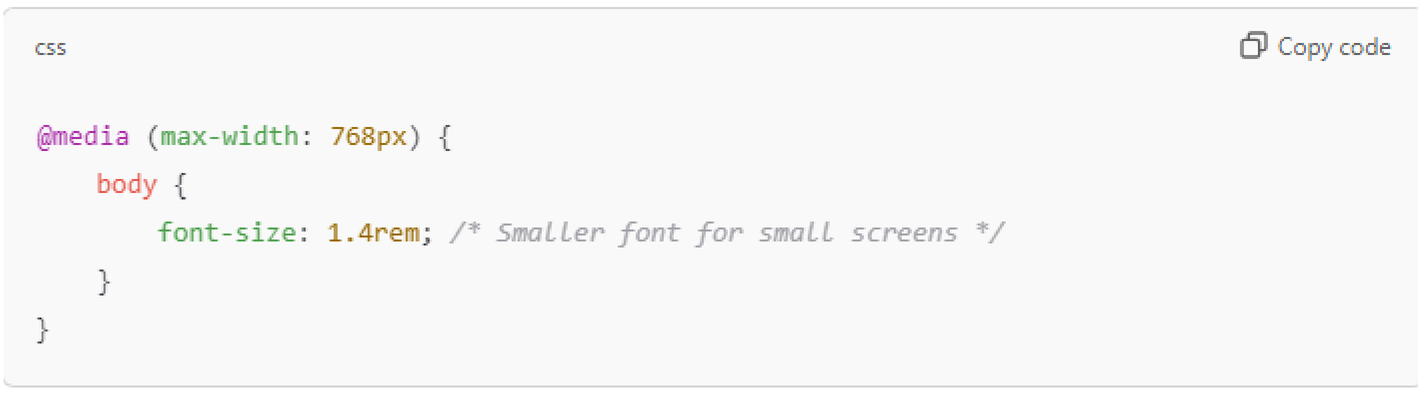
2. **Resultant Responsive Font Sizes:** Agar h1 ka size 2.4rem hai:
   - **Desktop (62.5%)**:
     - 2.4rem × 10px = 24px
   - **Tablet (50%)**:
     - 2.4rem × 8px = 19.2px
   - **Mobile (45%)**:
     - 2.4rem × 7.2px = 17.28px

Responsive Font Size Ke Advantages

1. **Automatic Scaling**:
   - Root font size ko change karne se saare rem units adjust ho jate hain.
   - Media queries ka use karke alag-alag devices ke liye consistent scaling milti hai.
2. **Accessibility**:
   - Agar user browser settings se font size badhata ya ghatata hai, toh rem units us change ko respect karte hain.
3. **Simplified Maintenance**:
   - Har font size ko individually adjust karne ki zarurat nahi hoti. Sirf root font size change karke aap pura layout modify kar sakte hain.

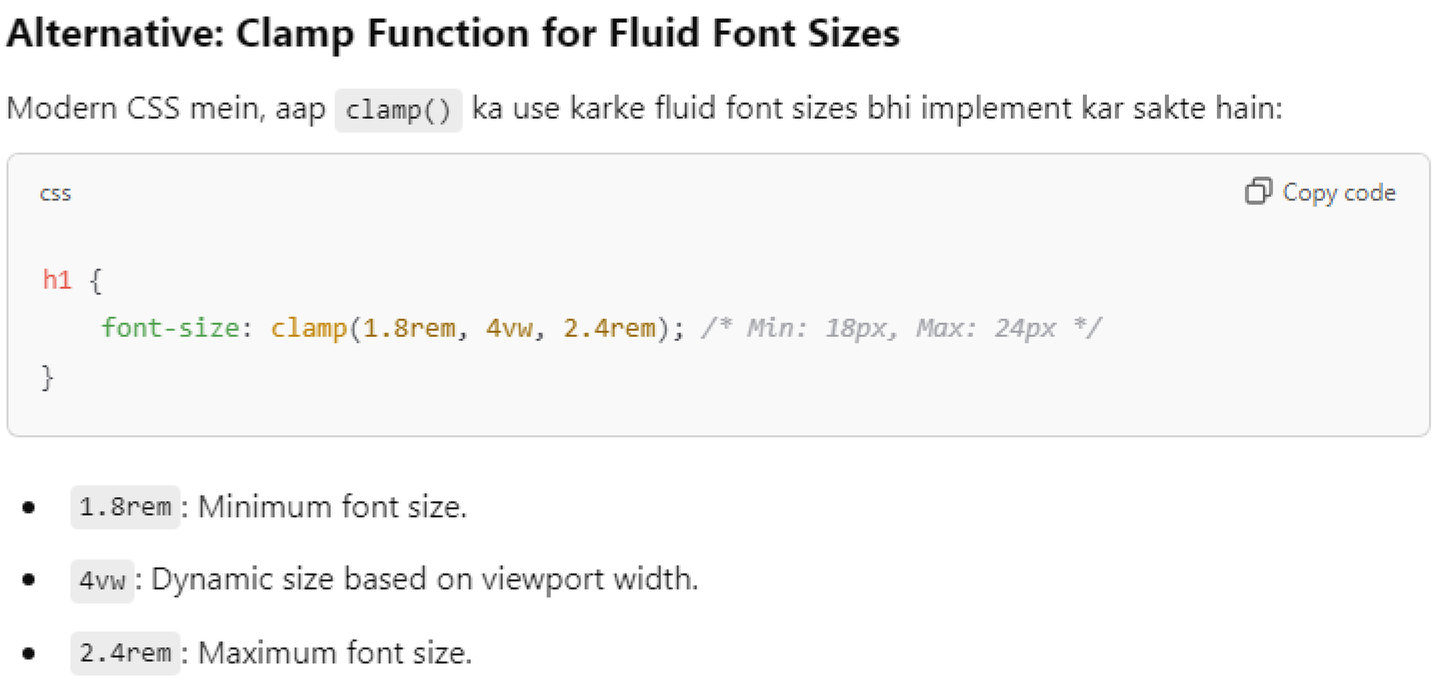
Chhoti Screens Par Font Size Adjust Karna:

Chhoti screens par text ko readable banane ke liye aap specific media queries laga sakte hain, jaise:



css Copy code

```css
@media (max-width: 768px) {
    body {
        font-size: 1.4rem; /* Smaller font for small screens */
    }
}
```



Alternative: Clamp Function for Fluid Font Sizes

Modern CSS mein, aap `clamp()` ka use karke fluid font sizes bhi implement kar sakte hain:

css Copy code

```css
h1 {
    font-size: clamp(1.8rem, 4vw, 2.4rem); /* Min: 18px, Max: 24px */
}
```

- `1.8rem` : Minimum font size.
- `4vw` : Dynamic size based on viewport width.
- `2.4rem` : Maximum font size.

Conclusion

- **font-size: 62.5%;** ka use responsive designs aur rem units ke saath kaafi helpful hai.
- Small screens par font size ko chhota karne ke liye sirf root font size ko adjust karna padta hai.

Examples of Viewport-Based Calculations

Suppose viewport width alag-alag ho, toh yeh 4vw ka calculation kaisa hoga:

- **Viewport width = 400px**:
  - 4vw = 4% of 400px = 16px.
  - Lekin minimum 1.8rem (18px) set hai, isliye font size 18px hoga.
- **Viewport width = 800px**:
  - 4vw = 4% of 800px = 32px.
  - Lekin maximum 2.4rem (24px) set hai, isliye font size 24px hoga.
- **Viewport width = 600px**:
  - 4vw = 4% of 600px = 24px.
  - Yeh preferred size hai aur allowed range mein hai, isliye font size 24px hoga.

Advantages of Using clamp()

1. **Fluid and Responsive Design**:
   - Font size dynamically adjust hota hai viewport ke according.
   - Yeh manual media queries ki zarurat kam karta hai.
2. **Readability Across Devices**:
   - Minimum aur maximum size set karne se text chhoti screens par readable aur badi screens par aesthetic lagta hai.
3. **Less Code, More Control**:
   - Ek hi line mein aap font size ka complete behavior define kar sakte hain.

Comparison with Media Queries

Traditional approach:

```css
@media (max-width: 768px) {
    h1 {
        font-size: 18px;
    }
}
@media (min-width: 768px) {
    h1 {
        font-size: 24px;
    }
}
```

`clamp()` ke saath:

```css
h1 {
    font-size: clamp(1.8rem, 4vw, 2.4rem);
}
```

****

Ji haan, aap bilkul sahi keh rahe hain! **rem** aur **em** dono **relative units** hain, lekin in dono ka kaam thoda alag hota hai. Chaliye, dono ka comparison aur unke working ko samajhte hain:

---

1. rem (Root em) Units

- **rem** ka matlab **Root em** hai. Yeh unit **html element ke font size ke proportional** hota hai.
- Jab aap rem use karte hain, toh uska reference **html ke font size** ko consider kiya jata hai (jo by default 16px hota hai, jab tak aap html { font-size } set na karein).

Example:

Agar html ka font size 16px hai, toh:

- **1rem = 16px**
- **2rem = 32px**
- **0.5rem = 8px**

Agar aap html { font-size: 62.5%; } set karte hain, toh:

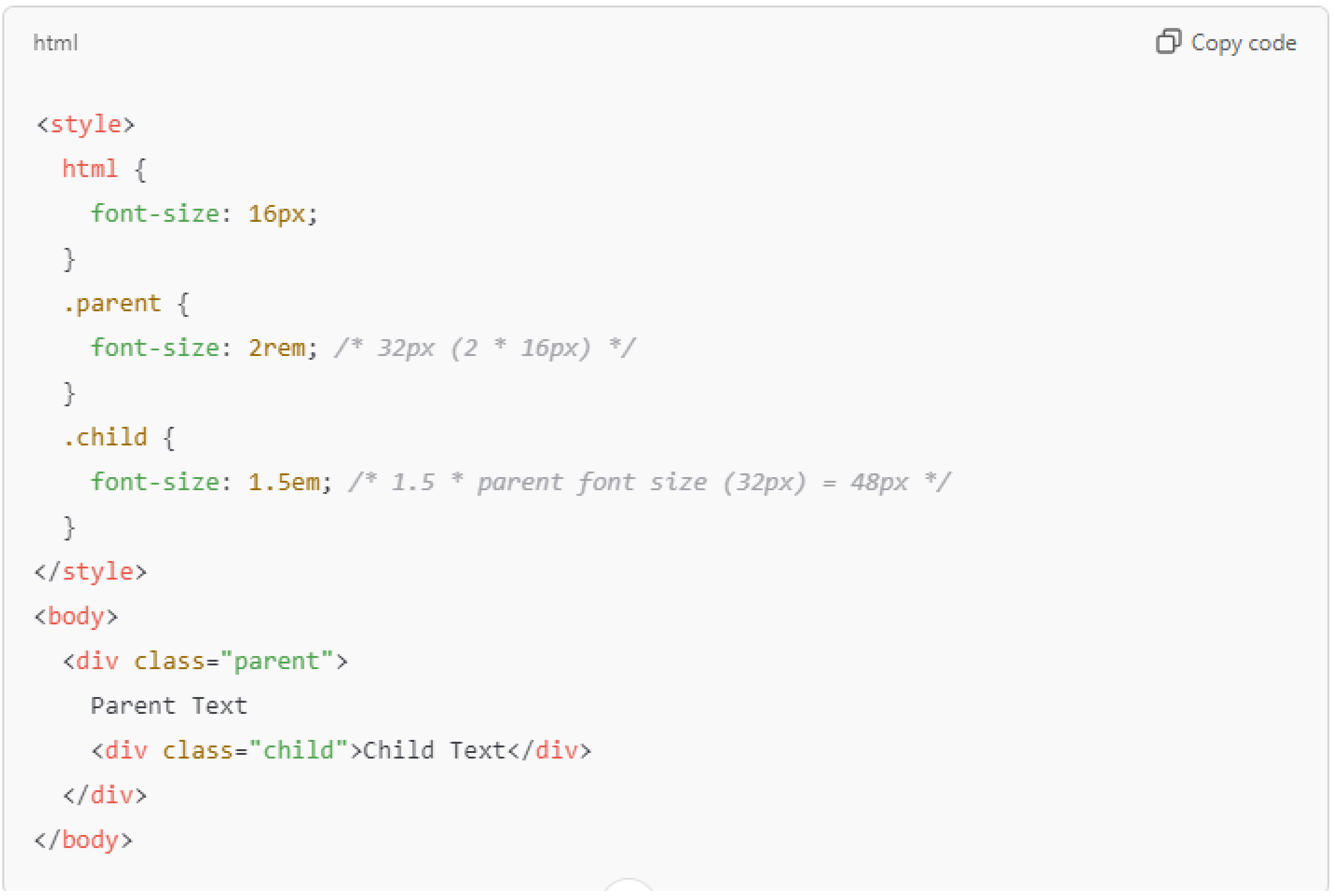
- 1rem = 10px (kyunki 16px ka 62.5% = 10px)
- Is case mein, **1rem = 10px** ho jata hai.

---

2. em Units

- **em** ka matlab **relative to the parent element's font size** hai.
- Jab aap em units ka use karte hain, toh uska reference **current element ke parent** ke font size ke according hota hai.
- Agar aap kisi element ko em value denge, toh us element ka font size parent ke font size ke proportion mein hoga.

Example 1: Parent aur Child ka Font Size



```html
<style>
  html {
    font-size: 16px;
  }
  .parent {
    font-size: 2rem; /* 32px (2 * 16px) */
  }
  .child {
    font-size: 1.5em; /* 1.5 * parent font size (32px) = 48px */
  }
</style>
<body>
  <div class="parent">
    Parent Text
    <div class="child">Child Text</div>
  </div>
</body>
```

- **.parent ka font size** 2rem = 32px (since html ka font size 16px hai).
- **.child ka font size** 1.5em = 48px (since parent ka font size 32px hai aur child 1.5 * 32px = 48px ho jayega).

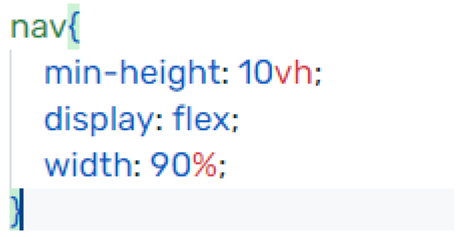
Key Differences: rem vs em

1. **rem**:
   - Root element (html) ke font size ke proportional hota hai.
   - Yeh predictable hota hai, kyunki yeh **global reference** pe based hai.
   - Example: Agar html { font-size: 16px; }, to **1rem = 16px**.
2. **em**:
   - Parent element ke font size ke proportional hota hai.
   - Yeh **local reference** pe based hota hai aur nested elements mein scale karte hue size change hota hai.
   - Example: Agar parent ka font size 16px hai, toh **1em = 16px** hoga, lekin agar parent ka font size 20px hai, toh **1em = 20px** hoga.

****

Set the height for nav div:

Generally ham height ko set nahi krty bal k uske content pe chor dety hen k jitna content ho wo default size le le, magar kabhi kabhi set krna par skta ha …..



```
nav{
  min-height: 10vh;
  display: flex;
  width: 90%;
}
```

 Relative to Parent: width: 90% ka size **parent** ke size ke relative hota hai. Agar parent ka size badalta hai, to nav ki width bhi usi ke proportional change hogi.

 Responsive Design: width: 90% ko responsive design mein use karna useful hota hai, jahan aap chahete hain ke element screen ya container ke size ke hisaab se adjust ho.

Agar nav element ka **parent element** nahi hai, ya agar aapne explicitly koi parent define nahi kiya hai, toh width: 90% ka matlab **viewport** ya **browser window** ke width ke **90%** hoga.

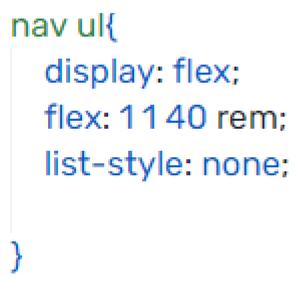
Iska matlab hai ki agar nav kisi container mein directly nahi hai, toh yeh apne **default parent**, jo ki body hai, ke width ke proportion mein adjust hoga. **body element** usually **100% width** occupy karta hai, unless aap uska size explicitly define karein.

Yahan, **body** element ka width default 100% hai. To jab aap nav ko width: 90% dete hain, toh:

- **nav** ki width body ke width ke **90%** ke barabar hogi.
- Agar viewport (browser window) ka width 1000px hai, toh nav ki width 900px hogi (1000px * 90% = 900px).

Agar aap body element ko completely remove kar den ya kisi aur container ka use na karein, toh nav ka width still **viewport** ke size ke 90% hoga, jo ki browser window ka width hai.

****



```
nav ul{
  display: flex;
  flex: 1 1 40 rem;
  list-style: none;

}
```

```css
flex: <flex-grow> <flex-shrink> <flex-basis>;
```

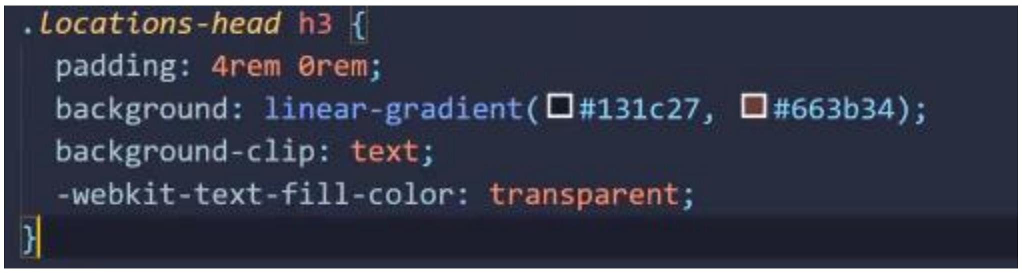
- `flex-grow`: Ye specify karta hai ki flex item kitna grow kar sakta hai relative to other items. Default value `0` hoti hai.
- `flex-shrink`: Ye specify karta hai ki flex item kitna shrink (chhota) ho sakta hai agar container ka size kam ho. Default value `1` hoti hai.
- `flex-basis`: Ye define karta hai initial size of the flex item before any growing or shrinking happens. Yeh ek fixed value bhi ho sakti hai (e.g., `px`, `rem`, etc.) ya percentage bhi ho sakti hai.

****

Background image on hero section with overlay effect:

```css
.hero {
  min-height: 90vh;
  background: linear-gradient(rgba(0, 0, 0, 0.4), transparent), url('assets/imgs/landing-page.jpg');
  background-repeat: no-repeat;
  background-size: cover;
  background-position: center;
}
```

Text ko gradient color dena:



```css
.locations-head h3 {
  padding: 4rem 0rem;
  background: linear-gradient(#131c27, #663b34);
  background-clip: text;
  -webkit-text-fill-color: transparent;
}
```

****