CSS NOTES
CSS selectors including simple, combinator, pseudo- class, pseudo- element, and attribute selectors:

Simple Selectors:
1. *Type Selector:* Selects elements based on their element type. Example: div, p, span.
2. *Class Selector:* Selects elements based on their class attribute.
Example: .classname.
3. *ID Selector:* Selects elements based on their id attribute. Example: #idname.
4.*Universal Selector:*Selects all elements in webpage at once.
5.*Grouping Selector:* Selects multiple elements same time separated by comma.
Example: h1,p,div{//css}

Combinator Selectors:
1. *Descendant Selector:* Selects an element that is a descendant of another specified element. Example: div p.
2. *Child Selector:* Selects an element that is a direct child of another specified element. Example: ul > li.
3. *Adjacent Sibling Selector:* Selects an element that is immediately preceded by a sibling element. Example: h2 + p.
4. *General Sibling Selector:* Selects elements that are siblings of a specified element. Example: h2 ~ p.

Pseudo- class Selectors:
1. *:hover:* Selects an element when the mouse pointer is over it.
2. *:active:* Selects an element when it is being activated by the user.
3. *:focus:* Selects an element when it is in focus.
4. *:first- child:* Selects an element that is the first child of its parent.
5. *:last- child:* Selects an element that is the last child of its parent.
6. *:nth- child():* Selects elements based on their position within a parent.
7. *:not():* Selects elements that do not match a specific selector.
8. *:nth- of- type():* Selects elements based on their position within a parent, counting only elements of the same type.
9. *:checked:* Selects input elements that are checked.

Pseudo- element Selectors:
1. *::before:* Inserts content before the selected element.
2. *::after:* Inserts content after the selected element.
3. *::first- line:* Selects the first line of text within the selected element.
4. *::first- letter:* Selects the first letter of text within the selected element.
5. *::selection:* Selects the portion of an element that is selected by a user.

Attribute Selectors:
1. *[attribute]:* Selects elements that have the specified attribute.

2. *[attribute=value]:* Selects elements with the specified attribute and value.
3. *[attribute~=value]:* Selects elements with an attribute that includes the specified value as one of its space- separated values.
4. *[attribute|=value]:* Selects elements with an attribute that exactly matches the specified value or starts with the specified value followed by a hyphen.

notes on the priorities between inline CSS, internal CSS, external CSS, class selectors, ID selectors, tag name selectors.

** Inline CSS:**
- Applied directly to individual HTML elements using the style attribute.
- Highest specificity.
- Overrides external and internal styles for the targeted element.
- Example: .

** Selector Specificity:**
- Inline styles have the highest specificity, followed by ID selectors, class selectors, and tag name selectors.
- Specificity is calculated based on the combination of selectors used to target an element.
- The more specific selector takes precedence over less specific selectors.

** ID Selectors:**
- Defined using the id attribute in HTML elements.
- More specific than class and tag name selectors.
- Should be used for unique elements.
- Example: .

** Class Selectors:**
- Defined using the class attribute in HTML elements.
- Less specific than ID selectors but more specific than tag name selectors.
- Can be applied to multiple elements.
- Example: .

Tag Name Selectors:
- Targets all HTML elements of a specific type.

- Least specific selector.
- Example: p { color: green; }.

** Importance:**
- The !important declaration overrides normal specificity rules.
- Should be used sparingly as it can make styles harder to override and maintain.

CSS selectors including simple, combinator, pseudo- class, pseudo- element, and attribute selectors:

Simple Selectors:
1. *Type Selector:* Selects elements based on their element type. Example: div, p, span.
2. *Class Selector:* Selects elements based on their class attribute.
Example: .classname.
3. *ID Selector:* Selects elements based on their id attribute. Example: #idname.
4.*Universal Selector:*Selects all elements in webpage at once.
5.*Grouping Selector:* Selects multiple elements same time separated by comma.
Example: h1,p,div{//css}

Combinator Selectors:
1. *Descendant Selector:* Selects an element that is a descendant of another specified element. Example: div p.
2. *Child Selector:* Selects an element that is a direct child of another specified element. Example: ul > li.
3. *Adjacent Sibling Selector:* Selects an element that is immediately preceded by a sibling element. Example: h2 + p.
4. *General Sibling Selector:* Selects elements that are siblings of a specified element. Example: h2 ~ p.

Pseudo- class Selectors:

1. *:hover:* Selects an element when the mouse pointer is over it.
2. *:active:* Selects an element when it is being activated by the user.
3. *:focus:* Selects an element when it is in focus.
4. *:first- child:* Selects an element that is the first child of its parent.
5. *:last- child:* Selects an element that is the last child of its parent.
6. *:nth- child():* Selects elements based on their position within a parent.
7. *:not():* Selects elements that do not match a specific selector.
8. *:nth- of- type():* Selects elements based on their position within a parent, counting only elements of the same type.
9. *:checked:* Selects input elements that are checked.

Pseudo- element Selectors:
1. *::before:* Inserts content before the selected element.
2. *::after:* Inserts content after the selected element.
3. *::first- line:* Selects the first line of text within the selected element.
4. *::first- letter:* Selects the first letter of text within the selected element.
5. *::selection:* Selects the portion of an element that is selected by a user.

Attribute Selectors:
1. *[attribute]:* Selects elements that have the specified attribute.
2. *[attribute=value]:* Selects elements with the specified attribute and value.
3. *[attribute~=value]:* Selects elements with an attribute that includes the specified value as one of its space- separated values.
4. *[attribute|=value]:* Selects elements with an attribute that exactly matches the specified value or starts with the specified value followed by a hyphen.

CSS Colors

1. *Using Named Colors*
- *Description:* Use standard color names in CSS to style elements.
- *Example:*

```
html
<div style="color: red;">This text is red.</div>
<div style="color: blue;">This text is blue.</div>
<div style="color: green;">This text is green.</div>
```

```
 <div style="color: black;">This text is black.</div>
 <div style="color: white; background- color: black;">This text is white on black.</div>
 <div style="color: gray;">This text is gray.</div>
```

2. *Using Hexadecimal Colors*
- *Description:* Use hex values to specify colors in CSS. The format is #RRGGBB or #RGB.
- *Example:*

```
 html
 <div style="color: #FF0000;">This text is red.</div>
 <div style="color: #0000FF;">This text is blue.</div>
 <div style="color: #00FF00;">This text is green.</div>
 <div style="color: #000000;">This text is black.</div>
 <div style="color: #FFFFFF; background- color: #000000;">This text is white on black.</div>
 <div style="color: #808080;">This text is gray.</div>
```

3. *Using RGB Colors*
- *Description:* Use RGB values to define colors. The format is rgb(red, green, blue), where each value ranges from 0 to 255.
- *Example:*

```
 html
 <div style="color: rgb(255, 0, 0);">This text is red.</div>
 <div style="color: rgb(0, 0, 255);">This text is blue.</div>
 <div style="color: rgb(0, 255, 0);">This text is green.</div>
 <div style="color: rgb(0, 0, 0);">This text is black.</div>
 <div style="color: rgb(255, 255, 255); background- color: rgb(0, 0, 0);">This text is white on black.</div>
 <div style="color: rgb(128, 128, 128);">This text is gray.</div>
```

4. *Using RGBA Colors*
- *Description:* Use RGBA values to define colors with an alpha channel (transparency). The format is rgba(red, green, blue, alpha), where alpha ranges from 0 (completely transparent) to 1 (completely opaque).
- *Example:*

```
 html
 <div style="color: rgba(255, 0, 0, 1);">This text is fully opaque red.</div>
 <div style="color: rgba(0, 0, 255, 0.5);">This text is semi- transparent blue.</div>
 <div style="color: rgba(0, 255, 0, 0.3);">This text is more transparent green.</div>
```

```
  <div style="color: rgba(0, 0, 0, 0.8);">This text is mostly opaque black.</div>
  <div style="color: rgba(255, 255, 255, 1); background- color: rgba(0, 0, 0, 1);">This text is fully opaque white on black.</div>
  <div style="color: rgba(128, 128, 128, 0.6);">This text is semi- transparent gray.</div>
```

5. *Using HSL Colors*
- *Description:* Use HSL values to define colors. The format is hsl(hue, saturation%, lightness%). Hue ranges from 0 to 360, saturation and lightness from 0% to 100%.
- *Example:*

```
  html
  <div style="color: hsl(0, 100%, 50%);">This text is red.</div>
  <div style="color: hsl(240, 100%, 50%);">This text is blue.</div>
  <div style="color: hsl(120, 100%, 50%);">This text is green.</div>
  <div style="color: hsl(0, 0%, 0%);">This text is black.</div>
  <div style="color: hsl(0, 0%, 100%); background- color: hsl(0, 0%, 0%);">This text is white on black.</div>
  <div style="color: hsl(0, 0%, 50%);">This text is gray.</div>
```

6. *Using HSLA Colors*
- *Description:* Use HSLA values to define colors with an alpha channel. The format is hsla(hue, saturation%, lightness%, alpha).
- *Example:*

```
  html
  <div style="color: hsla(0, 100%, 50%, 1);">This text is fully opaque red.</div>
  <div style="color: hsla(240, 100%, 50%, 0.5);">This text is semi- transparent blue.</div>
  <div style="color: hsla(120, 100%, 50%, 0.3);">This text is more transparent green.</div>
  <div style="color: hsla(0, 0%, 0%, 0.8);">This text is mostly opaque black.</div>
  <div style="color: hsla(0, 0%, 100%, 1); background- color: hsla(0, 0%, 0%, 1);">This text is fully opaque white on black.</div>
  <div style="color: hsla(0, 0%, 50%, 0.6);">This text is semi- transparent gray.</div>
```

Summary of Color Formats

- *Named Colors:* Predefined color names like red, blue, green, etc.
- *Hexadecimal Colors:* #RRGGBB or #RGB
- *RGB Colors:* rgb(red, green, blue) where values are 0- 255
- *RGBA Colors:* rgba(red, green, blue, alpha) where alpha is 0- 1
- *HSL Colors:* hsl(hue, saturation%, lightness%) where hue is 0- 360, saturation and

lightness are 0- 100%
- *HSLA Colors:* hsla(hue, saturation%, lightness%, alpha) where alpha is 0- 1

CSS Units Notes

Absolute Units
Absolute units are fixed and not relative to other elements. They are useful when you need precise control over the size of elements.

1. **Pixels (px)**
   - *Description:* One pixel on the screen.
   - *Example:*
     html
     100px x 50px</
div>

2. **Centimeters (cm)**
   - *Description:* One centimeter.
   - *Example:*
     html
     5cm x 3cm</
div>

3. **Millimeters (mm)**
   - *Description:* One millimeter.
   - *Example:*
     html
     50mm x
30mm

4. **Inches (in)**
   - *Description:* One inch (1in = 2.54cm).
   - *Example:*
     html
     2in x 1in

5. **Picas (pc)**
   - *Description:* One pica (1pc = 12pt = 1/6in).
   - *Example:*
     html
     6pc x 3pc

6. **Points (pt)**
   - *Description:* One point (1pt = 1/72in).
   - *Example:*
     html
     This text is 12pt

Relative Units
Relative units are based on the size of other elements and the viewport. They are more flexible and useful for responsive design.

1. **Percentage (%)**
   - *Description:* A percentage of the parent element's size.
   - *Example:*
     html
     50% of parent width

2. **Viewport Width (vw)**
   - *Description:* 1% of the viewport's width.
   - *Example:*
     html
     50vw

3. **Viewport Height (vh)**
   - *Description:* 1% of the viewport's height.
   - *Example:*
     html
     50vh

4. **Viewport Minimum (vmin)**
   - *Description:* 1% of the smaller dimension of the viewport (width or height).
   - *Example:*
     html
     50vmin</

div>

5. **Viewport Maximum (vmax)**
  - *Description:* 1% of the larger dimension of the viewport (width or height).
  - *Example:*
    html
    50vmax</
div>

6. **Em (em)**
  - *Description:* Relative to the font- size of the element (2em means 2 times the size of the current font).
  - *Example:*
    html

      10em x 2em</
div>

7. **Rem (rem)**
  - *Description:* Relative to the font- size of the root element (<html>).
  - *Example:*
    html

      10rem x
2rem

Summary of Units

Absolute Units
- **Pixels (px): Fixed unit; one pixel on the screen.
- **Centimeters (cm): One centimeter.
- **Millimeters (mm): One millimeter.
- **Inches (in): One inch (2.54cm).
- **Picas (pc): One pica (1/6 inch).
- **Points (pt): One point (1/72 inch).

Relative Units
- **Percentage (%): Relative to the parent element.
- **Viewport Width (vw): 1% of the viewport's width.

- **Viewport Height (vh): 1% of the viewport's height.
- **Viewport Minimum (vmin): 1% of the smaller dimension of the viewport.
- **Viewport Maximum (vmax): 1% of the larger dimension of the viewport.
- **Em (em): Relative to the font- size of the element.
- **Rem (rem): Relative to the font- size of the root element.

CSS display: flex Notes

The display: flex property in CSS allows you to create a flexible layout structure. This is achieved by defining a flex container that can adapt the size and position of its child elements. Below are detailed notes on how to use display: flex effectively, along with examples.

1. Basic Flex Container
- *Description:* Setting an element to display: flex makes it a flex container, with its children becoming flex items.
- *Example:*

```
html
<div style="display: flex; background- color: lightgray;">
  <div style="background- color: lightblue; padding: 10px;">Item 1</div>
  <div style="background- color: lightgreen; padding: 10px;">Item 2</div>
  <div style="background- color: lightcoral; padding: 10px;">Item 3</div>
</div>
```

2. Flex Direction
- *Description:* Defines the direction in which the flex items are placed.
  - row (default): Items are placed in a row.
  - row- reverse: Items are placed in a row, but in reverse order.
  - column: Items are placed in a column.
  - column- reverse: Items are placed in a column, but in reverse order.
- *Example:*

```
html
<div style="display: flex; flex- direction: column; background- color: lightgray;">
  <div style="background- color: lightblue; padding: 10px;">Item 1</div>
  <div style="background- color: lightgreen; padding: 10px;">Item 2</div>
  <div style="background- color: lightcoral; padding: 10px;">Item 3</div>
</div>
```

3. Justify Content
- *Description:* Aligns the flex items along the main axis (horizontal for row).
  - flex- start: Items are aligned to the start of the container.
  - flex- end: Items are aligned to the end of the container.
  - center: Items are centered along the main axis.
  - space- between: Items are evenly spaced, with the first item at the start and the last item at the end.
  - space- around: Items are evenly spaced with equal space around them.
  - space- evenly: Items are evenly spaced with equal space between them.
- *Example:*

```
html
<div style="display: flex; justify- content: space- between; background- color: lightgray;">
  <div style="background- color: lightblue; padding: 10px;">Item 1</div>
  <div style="background- color: lightgreen; padding: 10px;">Item 2</div>
  <div style="background- color: lightcoral; padding: 10px;">Item 3</div>
</div>
```

4. Align Items
- *Description:* Aligns the flex items along the cross axis (vertical for row).
  - flex- start: Items are aligned to the start of the cross axis.
  - flex- end: Items are aligned to the end of the cross axis.
  - center: Items are centered along the cross axis.
  - baseline: Items are aligned based on their baseline.
  - stretch: Items stretch to fill the container (default).
- *Example:*

```
html
<div style="display: flex; align- items: center; height: 200px; background- color: lightgray;">
  <div style="background- color: lightblue; padding: 10px;">Item 1</div>
  <div style="background- color: lightgreen; padding: 10px;">Item 2</div>
  <div style="background- color: lightcoral; padding: 10px;">Item 3</div>
</div>
```

5. Flex Wrap
- *Description:* Specifies whether flex items should wrap or not.
  - nowrap (default): All items are on one line.
  - wrap: Items wrap onto multiple lines.
  - wrap- reverse: Items wrap onto multiple lines in reverse order.
- *Example:*

```
 html
 <div style="display: flex; flex- wrap: wrap; background- color: lightgray;">
   <div style="background- color: lightblue; padding: 10px; flex: 1 1 150px;">Item 1</
div>
   <div style="background- color: lightgreen; padding: 10px; flex: 1 1 150px;">Item 2</
div>
   <div style="background- color: lightcoral; padding: 10px; flex: 1 1 150px;">Item 3</
div>
   <div style="background- color: lightyellow; padding: 10px; flex: 1 1 150px;">Item 4</
div>
 </div>
```

CSS Text Properties

1. *Text Align*
The text- align property specifies the horizontal alignment of text within an element.

- *left*: Aligns text to the left.
- *right*: Aligns text to the right.
- *center*: Centers the text.
- *justify*: Stretches the lines so that each line has equal width (like in newspapers and magazines).

```
css
p {
   text- align: center;
}
```

2. *Text Decoration*
The text- decoration property adds decoration to text. It can be used as a shorthand or with individual properties.

- *text- decoration- line*: Sets the kind of text decoration (underline, overline, line-through).
- *text- decoration- color*: Sets the color of the text decoration.
- *text- decoration- style*: Sets the style of the text decoration (solid, wavy, dotted, dashed, double).
- *text- decoration- thickness*: Sets the thickness of the text decoration line.

Shorthand:
```
css
p {
   text- decoration: underline wavy red;
}
```

Individual properties:
```
css
p {
   text- decoration- line: underline;
   text- decoration- color: red;
   text- decoration- style: wavy;
   text- decoration- thickness: 2px;
}
```

3. *Text Transform*
The text- transform property controls the capitalization of text.

- *none*: Default value; no transformation.
- *capitalize*: Capitalizes the first letter of each word.
- *uppercase*: Transforms all characters to uppercase.
- *lowercase*: Transforms all characters to lowercase.

```
css
p {
   text- transform: uppercase;
}
```

4. *Text Shadow*

The text- shadow property adds shadow to text. You can specify multiple shadows separated by commas.

- *Horizontal offset*: Required. Positive values move the shadow to the right, negative values to the left.
- *Vertical offset*: Required. Positive values move the shadow down, negative values up.
- *Blur radius*: Optional. The higher the number, the more blurred the shadow.
- *Color*: Optional. Specifies the color of the shadow.

```
css
h1 {
   text- shadow: 2px 2px 5px grey;
}
```

5. *Text Wrap*
The white- space property controls how text is handled when it is too long to fit in its containing element.

- *normal*: Default value. Text will wrap when necessary.
- *nowrap*: Text will not wrap; it will continue on the same line.

```
css
p {
   white- space: nowrap;
}
```

Overflow Properties

1. *Overflow*
The overflow property specifies what happens if content overflows an element's box.

- *visible*: Default. Content is not clipped and may be rendered outside the element's box.
- *hidden*: Content is clipped and not visible outside the element's box.
- *scroll*: Content is clipped, but a scrollbar is added to see the rest of the content.
- *auto*: Similar to scroll, but scrollbars are added only when necessary.

```
css
div {
   overflow: auto;
}
```

2. *Overflow- X and Overflow- Y*
These properties specify what happens to the content when it overflows the element's box horizontally (overflow- x) or vertically (overflow- y).

css
div {
  overflow- x: hidden;
  overflow- y: scroll;
}

3. *Removing Scrollbar with Vendor Prefixes*
To remove the scrollbar in WebKit browsers (like Chrome and Safari), you can use the ::- webkit- scrollbar pseudo- element.

css
div {
  overflow: auto;
}

div::- webkit- scrollbar {
  display: none; /* Safari and Chrome */
}

CSS Vendor Prefixes in Short

Vendor prefixes allow you to use CSS features that are not yet standardized across all browsers. Each major browser has its own prefix:

- **- webkit- **: Chrome, Safari, newer Opera
- **- moz- **: Firefox
- **- ms- **: Internet Explorer, Edge
- **- o- **: Older Opera

Common Properties with Prefixes

1. **Transform**
```css
.element {
  - webkit- transform: rotate(45deg);
  - moz- transform: rotate(45deg);
```

```
   - ms- transform: rotate(45deg);
   - o- transform: rotate(45deg);
   transform: rotate(45deg);
}
```

2. **Transition**
```css
.element {
   - webkit- transition: all 0.3s ease;
   - moz- transition: all 0.3s ease;
   - ms- transition: all 0.3s ease;
   - o- transition: all 0.3s ease;
   transition: all 0.3s ease;
}
```

3. **Animation**
```css
.element {
   - webkit- animation: example 5s infinite;
   - moz- animation: example 5s infinite;
   - ms- animation: example 5s infinite;
   - o- animation: example 5s infinite;
   animation: example 5s infinite;
}
```

4. **Box Shadow**
```css
.element {
   - webkit- box- shadow: 10px 10px 5px #888;
   - moz- box- shadow: 10px 10px 5px #888;
   box- shadow: 10px 10px 5px #888;
}
```

5. **Border Radius**
```css
.element {
   - webkit- border- radius: 10px;
   - moz- border- radius: 10px;
   border- radius: 10px;
}
```

Usage Tips

- **Order**: Always declare the standard property last.
- **Tools**: Use Autoprefixer to automatically add prefixes.
- **Modern Use**: Many properties no longer need prefixes in modern browsers, but they are necessary for older browser support.

Example
```css
.element {
    - webkit- transform: scale(1.2);
    - moz- transform: scale(1.2);
    - ms- transform: scale(1.2);
    - o- transform: scale(1.2);
    transform: scale(1.2);

    - webkit- transition: transform 0.3s ease;
    - moz- transition: transform 0.3s ease;
    - ms- transition: transform 0.3s ease;
    - o- transition: transform 0.3s ease;
    transition: transform 0.3s ease;

    - webkit- animation: bounce 2s infinite;
    - moz- animation: bounce 2s infinite;
    - ms- animation: bounce 2s infinite;
    - o- animation: bounce 2s infinite;
    animation: bounce 2s infinite;

    - webkit- box- shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    - moz- box- shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    box- shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

    - webkit- border- radius: 8px;
    - moz- border- radius: 8px;
    border- radius: 8px;
}
```

Removing Scrollbars (WebKit)
```css
div {
    overflow: auto;
}
```

```
div::- webkit- scrollbar {
    display: none;
}
```

Using vendor prefixes ensures compatibility across different browsers and their versions.

CSS Font Properties

1. Font Family
The `font- family` property specifies the font for an element. You can specify multiple fonts as a fallback system. Fonts can be divided into several categories:

- **Serif**: Fonts with small strokes at the ends of characters.
  ```css
  body {
    font- family: 'Times New Roman', Times, serif;
  }
  ```
- **Sans- Serif**: Fonts without the small strokes at the ends of characters.
  ```css
  body {
    font- family: Arial, Helvetica, sans- serif;
  }
  ```
- **Monospace**: Fonts where each character takes up the same amount of width.
  ```css
  body {
    font- family: 'Courier New', Courier, monospace;
  ```

```
}
```
- **Cursive**: Fonts that mimic human handwriting.
```css
body {
  font- family: 'Brush Script MT', cursive;
}
```
- **Fantasy**: Decorative fonts.
```css
body {
  font- family: 'Impact', fantasy;
}
```

2. Font Size
The `font- size` property sets the size of the font. It can be specified in various units such as `px`, `em`, `rem`, `%`, `vw`, etc.
```css
p {
  font- size: 16px; /* Pixels */
}

h1 {
  font- size: 2em; /* Relative to the parent element's font- size */
}
```

3. Font Weight
The `font- weight` property sets the thickness of the font characters.
```css
p {
  font- weight: normal; /* Normal weight (400) */
}

strong {
  font- weight: bold; /* Bold weight (700) */
}

.light- text {
  font- weight: 300; /* Light weight */
}
```

4. Font Style
The `font- style` property specifies whether the font should be italic, oblique, or normal.
```css
em {
  font- style: italic; /* Italic text */
}

p {
  font- style: normal; /* Normal text */
}

.oblique- text {
  font- style: oblique; /* Oblique text */
}
```

Using Google Fonts via CDN
To use Google Fonts, you can link them in the `<head>` of your HTML file without using `@import` in CSS. Here's how to do it:

1. Go to [Google Fonts](https://fonts.google.com/).
2. Select the font you want to use.
3. Click on the `+` button to add it to your selection.
4. Click on the selection drawer at the bottom.
5. Copy the `<link>` tag provided.
6. Paste it into the `<head>` section of your HTML.

For example, to use the `Roboto` font:
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF- 8">
  <meta name="viewport" content="width=device- width, initial- scale=1.0">
  <link href="https://fonts.googleapis.com/css2?
family=Roboto:wght@400;700&display=swap" rel="stylesheet">
  <style>
    body {
      font- family: 'Roboto', sans- serif;
    }
  </style>
  <title>Document</title>
</head>
<body>
```

```
  <p>This is a paragraph with the Roboto font.</p>
</body>
</html>
```

Complete Example
Here's a complete example that incorporates various font properties:
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF- 8">
  <meta name="viewport" content="width=device- width, initial- scale=1.0">
  <link href="https://fonts.googleapis.com/css2?
family=Lobster&family=Roboto:wght@400;700&display=swap" rel="stylesheet">
  <style>
    body {
      font- family: 'Roboto', sans- serif;
      font- size: 16px;
    }

    h1 {
      font- family: 'Lobster', cursive;
      font- size: 3em;
      font- weight: normal;
    }

    p {
      font- size: 1em;
      font- weight: 400;
      font- style: normal;
    }

    .highlight {
      font- weight: 700; /* Bold */
      font- style: italic;
    }
  </style>
  <title>Font Properties Example</title>
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is a paragraph styled with the Roboto font.</p>
  <p class="highlight">This text is bold and italicized using CSS.</p>
</body>
```

</html>
```

In this example:
- The body text uses the `Roboto` font from Google Fonts.
- The heading `h1` uses the `Lobster` cursive font from Google Fonts.
- Different font sizes, weights, and styles are demonstrated.

### CSS Background Properties

#### 1. Background- Color
The `background- color` property sets the background color of an element.
```css
body {
  background- color: #f0f0f0; /* Light gray */
}

header {
  background- color: #0044cc; /* Dark blue */
}
```

2. Background- Image
The `background- image` property sets one or more background images for an element. The value is a URL to the image.
```css
body {
  background- image: url('background.jpg');
}
```
```

3. Background- Position
The `background- position` property sets the starting position of a background image.
```css
header {
  background- image: url('header- bg.jpg');
  background- position: center; /* Center the background image */
}

section {
  background- image: url('section- bg.jpg');
  background- position: top right; /* Top right corner */
}
```

4. Background- Repeat
The `background- repeat` property defines how background images are repeated.
```css
body {
  background- image: url('pattern.png');
  background- repeat: repeat; /* Repeats both horizontally and vertically */
}

header {
  background- image: url('banner.jpg');
  background- repeat: no- repeat; /* No repeat */
}

section {
  background- image: url('tile.png');
  background- repeat: repeat- x; /* Repeat horizontally */
}
```

5. Background- Size
The `background- size` property specifies the size of the background images.
```css
header {
  background- image: url('header- bg.jpg');
  background- size: cover; /* Scale the image to cover the element */
}

section {
  background- image: url('section- bg.jpg');
  background- size: contain; /* Scale the image to fit within the element */
```

```
}

div {
  background- image: url('small- pattern.png');
  background- size: 50px 50px; /* Custom size */
}
```

6. Background Shorthand
The `background` shorthand property can be used to set all the background properties at once.
```css
div {
  background: #ffcc00 url('pattern.png') no- repeat center/cover;
  /* Equivalent to:
  background- color: #ffcc00;
  background- image: url('pattern.png');
  background- repeat: no- repeat;
  background- position: center;
  background- size: cover;
  */
}
```

Complete Example
Here's a complete example combining the various background properties:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF- 8">
  <meta name="viewport" content="width=device- width, initial- scale=1.0">
  <style>
    body {
      background- color: #e0f7fa; /* Light cyan background color */
    }

    header {
      background- image: url('header- bg.jpg');
      background- position: center;
      background- repeat: no- repeat;
      background- size: cover;
      height: 200px;
    }
```

```
    section {
      background- color: #ffffff; /* White background color */
      background- image: url('section- bg.jpg');
      background- position: top right;
      background- repeat: no- repeat;
      background- size: contain;
      padding: 20px;
    }

    footer {
      background: #0044cc url('footer- pattern.png') repeat center/auto;
      /* Shorthand:
      background- color: #0044cc;
      background- image: url('footer- pattern.png');
      background- repeat: repeat;
      background- position: center;
      background- size: auto;
      */
      height: 100px;
    }
  </style>
  <title>Background Properties Example</title>
</head>
<body>
  <header></header>
  <section>
    <p>This is a section with a background image.</p>
  </section>
  <footer></footer>
</body>
</html>
```

In this example:
- The `body` has a simple background color.
- The `header` has a background image that covers the entire element.
- The `section` uses a background image that is positioned at the top right and is contained within the element.
- The `footer` uses the shorthand `background` property to set a background color, image, repeat, position, and size.

CSS Border Properties

1. Border- Color
The `border- color` property sets the color of the border.
```css
div {
  border- color: #ff0000; /* Red border color */
}
```

2. Border- Width
The `border- width` property sets the width of the border.
```css
div {
  border- width: 2px; /* 2 pixels border width */
}
```

3. Border- Style
The `border- style` property specifies the style of the border.
```css
div {
  border- style: solid; /* Solid border style */
}
```

/* Other styles include: none, hidden, dotted, dashed, double, groove, ridge, inset, outset */
```

#### 4. Border- Radius
The `border- radius` property defines the radius of the element's corners.
```css
div {
  border- radius: 10px; /* Rounded corners with 10px radius */
}
```

5. Border (Shorthand)
The `border` shorthand property allows you to set the width, style, and color in one declaration.
```css
div {
  border: 2px solid #ff0000; /* 2px solid red border */
}
```

6. Border- Top, Border- Right, Border- Bottom, Border- Left
These properties allow you to set the border properties for each side of an element.
```css
div {
  border- top: 2px solid #ff0000; /* Top border */
  border- right: 4px dashed #00ff00; /* Right border */
  border- bottom: 6px double #0000ff; /* Bottom border */
  border- left: 8px groove #ff00ff; /* Left border */
}
```

7. Border- Top- Left- Radius, Border- Top- Right- Radius, Border- Bottom- Right- Radius, Border- Bottom- Left- Radius
These properties allow you to set the border radius for each corner of an element.
```css
div {
  border- top- left- radius: 10px; /* Top- left corner radius */
  border- top- right- radius: 20px; /* Top- right corner radius */
  border- bottom- right- radius: 30px; /* Bottom- right corner radius */
  border- bottom- left- radius: 40px; /* Bottom- left corner radius */
}
```
```

Complete Example
Here's a complete example combining the various border properties:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF- 8">
  <meta name="viewport" content="width=device- width, initial- scale=1.0">
  <style>
    .container {
      width: 300px;
      padding: 20px;
      border: 2px solid #333; /* Solid border */
      border- radius: 10px; /* Rounded corners */
      margin: 20px auto;
      text- align: center;
    }

    .special- border {
      border- top: 5px dotted #ff0000; /* Top border */
      border- right: 5px dashed #00ff00; /* Right border */
      border- bottom: 5px double #0000ff; /* Bottom border */
      border- left: 5px groove #ff00ff; /* Left border */
      border- top- left- radius: 15px; /* Top- left corner */
      border- top- right- radius: 25px; /* Top- right corner */
      border- bottom- right- radius: 35px; /* Bottom- right corner */
      border- bottom- left- radius: 45px; /* Bottom- left corner */
      padding: 10px;
    }
  </style>
  <title>Border Properties Example</title>
</head>
<body>
  <div class="container">
    <p>This container has a solid border and rounded corners.</p>
  </div>
  <div class="special- border">
    <p>This container has different border styles and rounded corners for each side.</
p>
  </div>
</body>
</html>
```

In this example:
- The `.container` class has a simple solid border with rounded corners using `border-radius`.
- The `.special- border` class demonstrates different border styles, colors, and widths for each side, along with individual corner radii.

- Transition Property :- The transition property in css is used to make some transition effect.

  The transition effect can be defined in two states.(:hover and :active) using pseudo class selectors.

  The Transition property is the combination of 4 properties:-

  1. Transition- property
  2. Transition- duration
  3. Transition- timing- function
  4. Transition- delay

  Transition- Property : To specify on which property transition should be applied.

Values :- width , Height , all, Background, none.

Transition- duration : This property allows you to determine how long it will take to complete the transition from one CSS property to the other.

Or

Time to be taken to complete the transition. Value in Seconds(s) or Milliseconds (ms).

Example:- transition- duration : 2s;

Transition- timing- function : This property allows you to determine the speed

of change during the transition effect. Like, the change should be fast at the beginning and slow at the end, etc.

Example:-

transition- timing- function: ease|ease- in|ease- out|ease- in- out|linear| step- start|step- end;

1. Linear : (Default Value) Same speed throughout the transition.
2. Ease : Slow Start + Fast + Slow End
3. Ease- in : Slow Start
4. Ease- out : Slow End
5. Ease- in- out : Slow Start + Linear + Slow End

Transition- delay : This property allows you to determine the amount of time to wait before the transition actually starts to take place.

OR

Time to be taken to start the transition. Value in Seconds(s) or Milliseconds (ms).

Example:- transition- delay : 2s;

Shorthand Property :

transition: (property name) | (duration) | (timing function) | (delay);

- Animation Property :- The CSS animation property is used to specify the animation that should be applied to an element.

- Animation Name :
  - To animate an element we have to provide animation to the element.
  - Example: Animation- name : Box1 ;
  - By this name {Box1} @keyframe will target the element for animation.

- @Keyframes :- Keyframes defines the style to be applied for that moment with in the animation.

  We can apply style in two ways :-

  1. From and To :- It will target starting and ending of animation.

     @keyframes identifier {

     From{

     Here you have to mention box-name which is mentioned in Animation- name Property

     }

     To {

     }

     }

  2. Percentage :-

     We can use percentage. By using percentage (%) we can target any moment in the animation

     @keyframes identifier {

     0%{

     background- color : red;

     }

     50% {

     background- color : green;

     }

     100%{

     background- color : red;

     }

     }

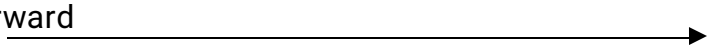
- animation- duration: It is used to specify the time duration and it takes animation to complete one cycle.
- animation- timing- function: It is used to specify how the animation makes transitions through keyframes.
- animation- delay: It is used to specify the delay when the animation starts.
- animation- iteration- count: It is used to specify the number of times the animation will repeat. It can specify as infinite to repeat the animation indefinitely.

Example :- animation- iteration- count: value; [value :- 1,2,3,infinite]

- animation- direction: It is used to specify the direction of the animation.

Value :-

1. Forward



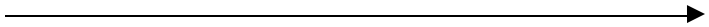
normal

2. Reverse

Reverse

3. Forward

Alternate



Reverse

4. Reverse

alternate- reverse

Forward