Felix-ITs

Front-End

CSS

Felix - IT Systems

CSS math functions

CSS math functions let you do calculations directly in CSS to control size, position, and layout.

- **calc()** → Perform addition, subtraction, multiplication, division.
- **min()** → Picks the smallest value.
- **max()** → Picks the largest value.

Example - min

```
    <style>
        .box {
            width: min(50%, 100px);
            /* Takes 50% of the container or 300px, whichever is smaller */
            height: 100px;
            background-color: lightgreen;
        }
    </style>
</head>

<body>
    <div class="box"></div>
</body>
```

The .box will be either 50% of the screen's width or 300px, depending on which one is smaller.

Example - max

```
<style>
    .box {
      width: max(200px, 70%);
      /* Takes 200px or 30% of the container, whichever is larger */
      height: 100px;
      background-color: lightcoral;
    }
  </style>
</head>

<body>
  <div class="box"></div>
</body>
```

The .box will be at least 200px wide or 30% of the container, whichever is larger.

Example - Calc

```
<style>
      .container {
         width: 100%;
         height: 200px;
         background-color: lightblue;
      }
      .box {
         width: calc(100% - 70%); /* Width is 100% of the container minus 50px */
         height: 100px;
         background-color: coral;
         margin: 0 auto; /* Centers the box */
      }
   </style>
</head>
<body>
   <div class="container">
      <div class="box">30%</div>
   </div>
</body>
```

The .box inside the .container takes the full width of the container minus 50px.

Note: min max calc

In modern web development, **min()**, **max()**, and **calc()** are powerful tools for making responsive and adaptive designs.

They eliminate(skip) the need for hardcoded values, making layouts more scalable and ensuring a smoother user experience into all devices. ✅

Specificity in CSS

Specificity tells the browser which style to use when many styles try to change the same element.

The style that is more specific will be applied. ✅

Each selector has a different **strength (score):**

- Element selectors (h1, p, div) → 1 point
- Class selectors (.button, .header) → 10 points
- ID selectors (#main, #footer) → 100 points
- Inline styles (style="color:red;") → 1000 points (strongest)

✓ The style with the highest score is applied.

Example

```
<head>
  <style>
    p {
      color: blue;
      /* Specificity: 1 (type selector) */
    }

    .special {
      color: green;
      /* Specificity: 10 (class selector) */
    }

    #unique {
      color: red;
      /* Specificity: 100 (ID selector) */
    }
  </style>
</head>

<body>
  <p class="special" id="unique">Hello World</p>
</body>
```

Example

```
<head>
  <style>
    p {
      color: blue;
      /* Specificity: 1 (type selector) */
    }

    .special {
      color: green;
      /* Specificity: 10 (class selector) */
    }

    #unique {
      color: red;
      /* Specificity: 100 (ID selector) */
    }
  </style>
</head>

<body>
  <p class="special" id="unique" style="color: yellow;">Hello World</p>
</body>
```

Units

px (Pixels)

A fixed unit (tiny dot on the screen).

Stays the same size on all screens.

Doesn’t change with browser window size (absolute).

```
p {
  font-size: 16px;     /* Always 16px on any device */
}
```

☞ The text will look the same size on both phone and desktop.

% (Percentage)

A relative unit, based on the parent container.

Example: 50% means half the size of the parent.

```
.box {
  width: 50%;        /* Takes 50% of parent’s width */
}
```

☞ If the parent is 800px wide, the box will be 400px wide.

vh (Viewport Height)

Based on the screen’s height (viewport).

1vh = 1% of the viewport height.

```
.box {
  height: 50vh;        /* 50% of screen height */
}
```

☞ If the screen height is 1000px, then 50vh = 500px.

vw (Viewport Width)

Based on the screen’s width (viewport).

1vw = 1% of the viewport width.

```
.box {
  width: 50vw;    /* 50% of screen width */
}
```

☞ If the screen width is 1200px, then 50vw = 600px.

rem (Root em)

Relative to the root <html> font size.

Default root size = 16px in most browsers.

1rem = root font size.

```
html {
  font-size: 16px;      /* root size */
}

p {
  font-size: 2rem;      /* 2 × 16px = 32px */
}
```

☞ 2rem = 32px when root size is 16px.

em

Relative to the parent element’s font size.

Changes if the parent’s size changes.

```
.container {
  font-size: 20px;      /* parent */
}

.child {
  font-size: 1.5em;      /* 1.5 × 20px = 30px */
}
```

☞ 1em = parent size. Example: 20px parent → 1.5em = 30px.

Attribute Selectors

a[target]

```
<head>
   <style>
      a[target] {
         background-color: red;
      }
   </style>
</head>

<body>

   <a href="#">HTML</a>
   <a href="#" target="_blank">CSS</a>
   <a href="#" target="_blank">Bootstrap</a>

</body>
```

[attribute="value"] Selector

```
<head>
   <style>
      a[target="_blank"] {
         background-color: red;
      }
   </style>
</head>

<body>

   <a href="#">HTML</a>
   <a href="#" target="_blank">CSS</a>
   <a href="#" target="_top">Bootstrap</a>

</body>
```

[attribute~="value"] Selector

The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word.

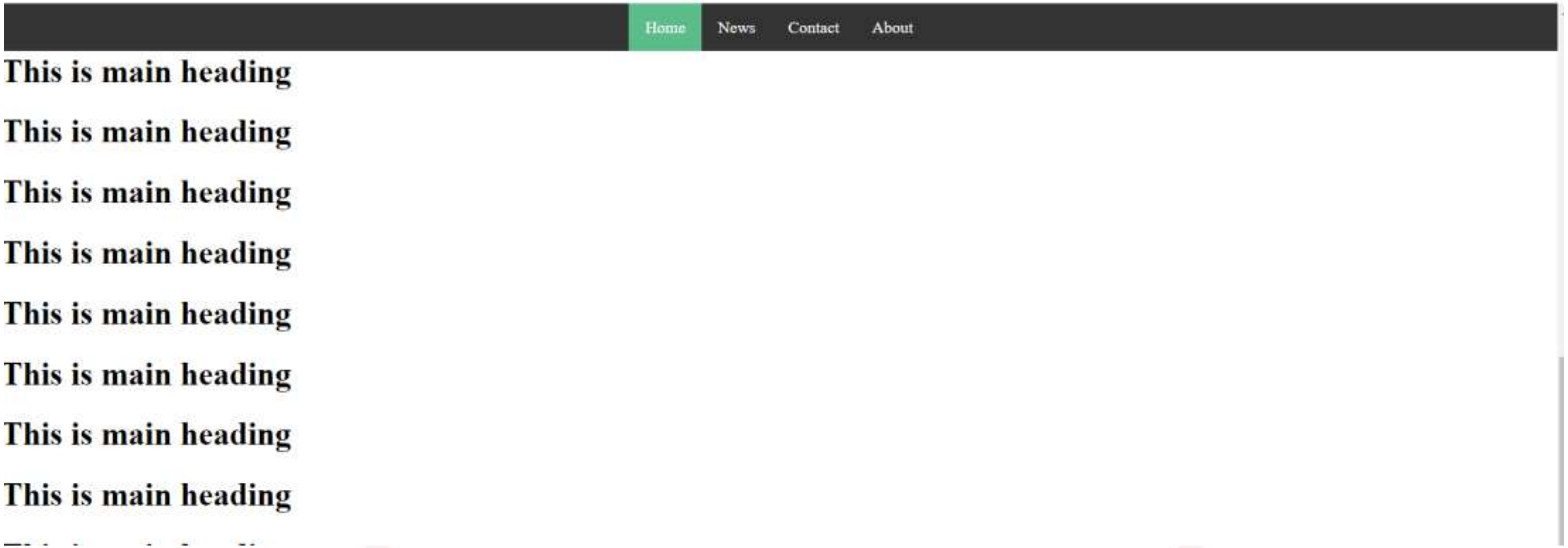
```
<head>
   <style>
      h1[title~="web"] {
         border: 5px solid red;
         padding: 8px;
      }
   </style>
</head>

<body>

   <h1>This is main heading without title</h1>
   <h1 title="web">This is main heading with title</h1>
   <h1 title="main">This is main heading with title</h1>

</body>
```

input[type="text"], input[type="button"]

```
<style>
    input[type="text"] {
        width: 250px;
        color: white;
        border-radius: 30px;
        background-color: red;
        text-align: center;
    }

    input[type="button"] {
        width: 150px;
        background-color: red;
        color: white;
        text-align: center;
        padding: 8px;
    }
  </style>

  <form>

    Firstname:<input type="text" name="Name" placeholder="John">
    Lastname:<input type="text" name="Name" placeholder="Doe">

    <input type="button" value="Example Button">
  </form>
```

Fixed Top Navigation Bar

Felix-ITs



This is main heading

This is main heading

This is main heading

This is main heading

This is main heading

This is main heading

This is main heading

This is main heading

```
<body>

    <nav>
        <ul>
            <li><a class="active" href="#home">Home</a></li>
            <li><a href="#news">News</a></li>
            <li><a href="#contact">Contact</a></li>
            <li><a href="#about">About</a></li>
        </ul>
    </nav>

    <br><br><br>
    <h1>This is main heading</h1>
    <h1>This is main heading</h1>
    …….

</body>

</html>
```

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

        ul {
            list-style-type: none;
            margin: 0;
            padding: 0;
            overflow: hidden;
            background-color: #333;
            position: fixed;
            top: 0;
            width: 100%;
            display: flex;
            justify-content: center;
        }
```

```css
li {
    display: inline;
    /* Make the list items inline */
}

li a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

.active {
    background-color: #23bf86;
    color: white;
}

li a:hover:not(.active) {
    background-color: #111;
    color: white;
}
```

Felix-ITs

Overlap some text on top of an image.

```
<body>

    <div class="container">
        <img src="https://via.placeholder.com/400x200" alt="Image"
class="image">
        <div class="text">Overlapping Text</div>
    </div>

</body>
```

```
<style>
        /* This is for the container that holds the image and the text */
        .container {
            position: relative;
            /* Makes the container a reference point for positioning */
            width: 50%;
            /* You can adjust the width of the image */
        }

        /* Style for the image */
        .image {
            width: 100%;
            /* Makes the image fit the container */
        }


        .text {
            position: absolute;
            /* Allows text to overlap the image */
            top: 30px;
            /* Distance from the top of the image */
            left: 30px;
            /* Distance from the left of the image */
            color: white;
            background-color: black;
            padding: 15px;
        }
    </style>
```

Styled Textarea

```
<body>

    <h2>Styled Textarea</h2>
    <form>
        <textarea class="styled-textarea" placeholder="Type something
here..."></textarea>
    </form>

</body>
```

```
<style>
        body {
            font-family: Arial, sans-serif;
            padding: 20px;
        }
        .styled-textarea {
            width: 100%;
            height: 150px;
            padding: 10px;
            border: 2px solid #fd0b0b;
            border-radius: 10px;
            background-color: #f9f9f9;
            font-size: 16px;
            color: #333;
            resize: none;
            /* Disable resizing */
        }

        .styled-textarea:focus {
            border-color: #333;
            outline: none;
        }
    </style>
```

Styling Select Menu

```
<body>

    <h2>Styling select menu</h2>

    <form>
        <select id="country" name="country">
            <option value="html">HTML</option>
            <option value="css">CSS</option>
            <option value="bootstrap">Bootstrap</option>
            <option value="java">Java</option>
        </select>
    </form>

</body>
```

```
<head>
    <style>
        select {
            width: 100%;
            padding: 16px 20px;
            border: none;
            border-radius: 4px;
            background-color: #f1f1f1;
        }
    </style>
</head>
```

Icon inside input box using HTML and CSS

```
<body>

    <div class="input-box">
        <!-- Emoji icon -->
        <span class="icon">🔍</span>
        <!-- Input field -->
        <input type="text" placeholder="Type here...">
    </div>

</body>
```

```
<style>
        /* Container for the input box and the icon */
        .input-box {
            position: relative;
            /* To allow positioning the icon inside */
            width: 300px;
            /* Set a fixed width for the input box */
            margin: 20px;
        }

        /* Style the input */
        input {
            width: 100%;
            /* Make input fill the width of the container */
            padding: 10px;
            /* Add padding inside the input */
            padding-left: 40px;
            /* Add space on the left for the icon */
            border: 1px solid #000000;
            /* Add a light border */
            border-radius: 5px;
            /* Rounded corners */
        }
```

```
     /* Style for the icon inside the input */
    .icon {
        position: absolute;
        /* Place it absolutely inside the container */
        left: 10px;
        /* Position it 10px from the left side */
        top: 50%;
        /* Center it vertically */
        transform: translateY(-50%);
        /* Perfect vertical centering */
        color: #888;
        /* Icon color */
    }
</style>
```

Pseudo-classes and Pseudo-elements in CSS

Pseudo-classes

Used to style an element in a special state.

Example: hover, first-child, focus, etc.

```
a:hover {
  color: red; /* when mouse is over link */
}

p:first-child {
  font-weight: bold; /* only first paragraph */
}
```

☞ :hover = when mouse is over.

☞ :first-child = only the first child of a parent.

Pseudo-elements

Used to style a specific part of an element.
Example: first letter, before/after content, etc.

```
p::first-letter {
  font-size: 2em; /* first letter bigger */
}

p::before {
  content: "Note: "; /* adds text before paragraph */
}
```

☞ ::first-letter = styles only the first letter.
☞ ::before = inserts content before the element.

Key Differences

Pseudo-classes → Style an element’s state (e.g., :hover, :focus, :first-child).

Pseudo-elements → Style a part of an element (e.g., ::first-letter, ::before, ::after).

☞ Example:

a:hover { color: red; } → changes style when hovered.

p::first-letter { font-size: 2em; } → styles only first letter.

Example

```
<body>

    <h2>Pseudo-class and Pseudo-element Example</h2>

    <a href="#">Hover over this link</a>

    <ul>
        <li>First item (styled by :first-child)</li>
        <li>Second item</li>
        <li>Third item</li>
    </ul>

    <p>This is a paragraph of text, and the first letter will be styled with a
pseudo-element.</p>

</body>
```

```
<style>
       /* Pseudo-class for hover effect */
       a {
           text-decoration: none;
       }
       a:hover {
           color: red;
           text-decoration: underline;
       }

       /* Pseudo-class for first child */
       li:first-child {
           color: green;
           font-weight: bold;
           font-size: 20px;
       }
```

```css
    /* Pseudo-element for first letter styling */
    p::first-letter {
        font-size: 3em;
        color: blue;
    }

    /* Pseudo-element to add content before a paragraph */
    p::before {
        content: "Important: ";
        font-weight: bold;
    }

    /* Pseudo-element to add content after a paragraph */
    p::after {
        content: "Demo: ";
        color: red;
    }
</style>
```

Example

```
<head>
    <style>
        h1::before {
            content: url('https://picsum.photos/200/300');
        }
        p::after {
            content: "Demo";
            color: red;
            font-size: 30px;
        }
    </style>
</head>

<body>

    <h1>This is a heading</h1>

    <p>The ::before ::after pseudo-elements inserts content before & after the content of an element.</p>

    <h1>This is a heading</h1>

</body>
```

Example - ::selection

```
<head>
    <style>
        ::selection {
            background-color: red;
            color: white;
        }
    </style>
</head>

<body>

    <h1>This is main heading</h1>

    <p>This is a paragraph.</p>

    <div>This is some text in a div element.</div>

</body>
```

Felix-ITs

Horizontally center a block element (like div)

```
<head>
   <style>
      .center {
         margin: auto;
         width: 60%;
         border: 3px solid red;
         padding: 8px;
         text-align: center;
      }
   </style>
</head>

<body>
   <h2>Center Align Elements</h2>

   <div class="center">
      <p> Hello World </p>
   </div>
</body>
```

Center an Image

```
<head>
  <style>
    img {
      display: block;
      margin-left: auto;
      margin-right: auto;
      width: 250px;
      height: 250px;
    }
  </style>
</head>

<body>

  <h2>Center an Image</h2>

  <p>To center an image, set left and right margin to auto, and make it into a block element.</p>

  <img src="https://picsum.photos/200/300"  alt="My image">

</body>
```

CSS display Property

display: inline

Elements are placed side by side.

They do not start on a new line.

Width and height cannot be set; they take only as much space as their content.

display: block

Elements start on a new line.

They occupy the full width of their parent container by default.

Width and height can be set.

display: inline-block

Elements are placed side by side (like inline elements).

You can set width and height (unlike inline).

They do not start on a new line, unless the container width is exceeded.

Example for display: inline

```
 <style>
        .inline {
            display: inline;
            background-color: lightblue;
            padding: 10px;
            border: 1px solid blue;
        }
    </style>
    <title>Display: Inline Example</title>
</head>

<body>

    <h2>Display: Inline</h2>
    <span class="inline">Item 1</span>
    <span class="inline">Item 2</span>
    <span class="inline">Item 3</span>

</body>
```

Example display: block

```
  <style>
    .block {
      display: block;
      background-color: lightgreen;
      padding: 10px;
      border: 1px solid green;
      margin-bottom: 10px;
    }
  </style>
  <title>Display: Block Example</title>
</head>

<body>

  <h2>Display: Block</h2>
  <div class="block">Item 1</div>
  <div class="block">Item 2</div>
  <div class="block">Item 3</div>

</body>
```

Example for display: inline-block

```
  <style>
    .inline-block {
      display: inline-block;
      background-color: lightcoral;
      padding: 10px;
      border: 1px solid red;
      width: 100px;
      height: 50px;
    }
  </style>
  <title>Display: Inline-block Example</title>
</head>

<body>

  <h2>Display: Inline-block</h2>
  <span class="inline-block">Item 1</span>
  <span class="inline-block">Item 2</span>
  <span class="inline-block">Item 3</span>

</body>
```

📚 Practice for Lecture 10 - 1

- ✅ **Watch the video carefully and write the code shown in it.**

💻 **Upload your code to GitHub** after completing the practice.

- ⍰ Don't worry if you don't fully understand some lines of code — just follow along with the video for now.

- Understanding will come with practice! 💪

Video Link

https://drive.google.com/drive/folders/1uuARQX6WKEdcOz66fCSuirox1xKMwpXg?usp=sharing

Practice for Lecture 10 - 2

Write answers to the following questions in your notebook in your own words.

- What are CSS math functions and how are they used?
- What is specificity in CSS and why is it important?
- What units can be used in CSS for styling elements?
- What are attribute selectors in CSS, and what are their benefits?
- How can you overlap text on an image using CSS?
- What are pseudo-classes and pseudo-elements in CSS, and how do they differ?
- Explain the CSS display property (inline, block, and inline-block) with examples.

📚 Practice for Lecture 10 - 3

🎥 Record Videos in English on the Following Topics

1. ***What do you know about our industry?***
2. ***Why did you choose this career path?***
3. ***How does this job align with your career goals?***
4. ***What are your long-term career plans?***
5. ***What do you want to achieve in the next 6 months?***

� Speak clearly and confidently in English.

This will help improve your communication and interview skills!

Felix-ITs

Thank You

Felix -ITs