MŰEGYETEM 1782

Mobile and Web Development

Department of Automation and Applied Informatics

CSS

Created: 1/2024/2025

Dr. Mohammad Saleem

msaleem@aut.bme.hu

CSS2

Introduction

CSS (Cascading Style Sheets) allows you to create great-looking web pages, but how does it work under the hood? This article explains what CSS is with a simple syntax example and also covers some key terms about the language.

Using CSS, you can control exactly how HTML elements look in the browser, presenting your markup using whatever design you like.

A **document** is usually a text file structured using a markup language — HTML is the most common markup language, but you may also come across other markup languages such as SVG or XML.

Presenting a document to a user means converting it into a form usable by your audience. Browsers, like Firefox, Chrome, or Edge, are designed to present documents visually, for example, on a computer screen, projector, or printer.

CSS can be used for very basic document text styling — for example, for changing the color and size of headings and links. It can be used to create a layout — for example, turning a single column of text into a layout with a main content area and a sidebar for related information. It can even be used for effects such as animation.

Why CSS?

- At the beginning it was enough to format the document with simple styles
  - bold/underlined/italic
  - images and blue links
  - nothing else was needed
- Than expectations for styling began to grow
  - The color of the link could be set in the <body>
  - The 	align attribute turned up to align text
  - Then hell broke loose after <font> tag turned up which

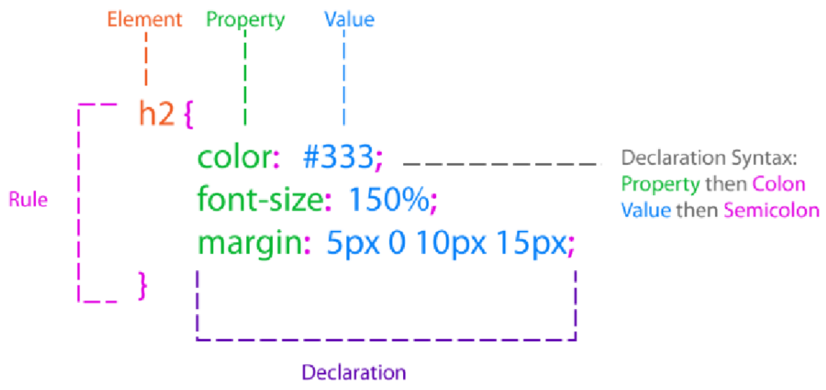
could precisely specify the font style of the text

- If all <h1> elements are red and 26 pixel high
  - It should be defined for all <h1> tags
  - Hard to modify
- Not flexible
  - printing
  - mobile device

<h1 style="color: red; font-size: 26px">

CSS - Cascading Style Sheets:

- Rules start with a **selector**
    - > Specifies the elements the rule applies to
- Inside of the selector a rule contains key-value pairs for styling properties and values
- An HTML page may refer to one or more CSS files
- There are higher priority selectors that can overwrite rules with lower priority selectors.



Example:

```
h1 {
color: blue;  font-size: 16px;
}
a {
/* Every link is green */  color: green;
}
/* Except the ones in the nav block, they are red*/
nav a {
/*More specific selectors are stronger */  color: red;
}
```

Most important selectors

Tag, .classname. #identifier, tag .descendantclassname

```
tag {
/* <tag>... </tag> */
}
.classname {
/* <any class="classname">...</any> */
}
#identifier {
/* <any id="identifier">...</any> */
}
tag .descendantclassname {
/* <tag> ... <any1> ... <any2  class="descendantclassname">...</any2> ... </any1> ... </tag>
*/
}
```

Rules overwriting each other

- Any inline stylesheet takes the highest priority. Therefore, it will override any rule defined in <style>…</style> tags or rules defined in an external style sheet file.
- Any rule defined in <style>…</style> tags will override rules defined in any external style sheet file.
- Any rule defined in external style sheet file takes the lowest priority, and rules defined in this file will be applied only when above two rules are not applicable.

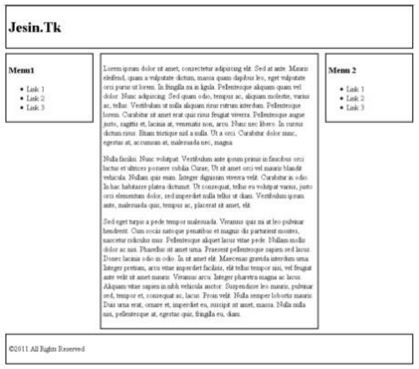
Example: https://www.w3schools.com/css/tryit.asp?filename=trycss_specificity_inline

CSS layout

- Define the layout of the page in CSS.
- Early CSS versions support document-like layout
- CSS-based layout is difficult and is full of hacks

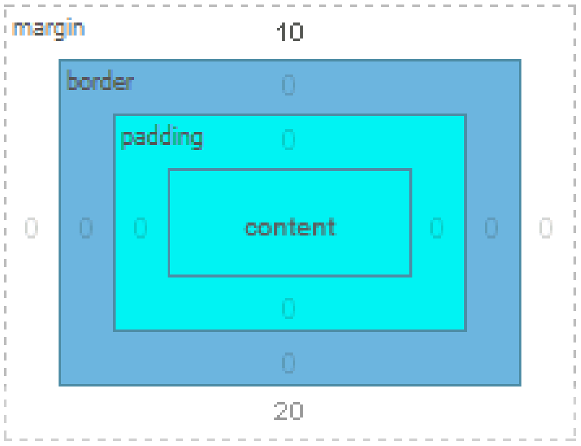
Layout example

```
#sidebar-left,
#sidebar-right {
    margin-top: 10px;
    float: left;
    width: 20%;
}
#content {
    margin: 10px 5px;
    width: 52%;
    float: left;
}
#header,
#footer {
    clear: both;
}
```



The boxing model

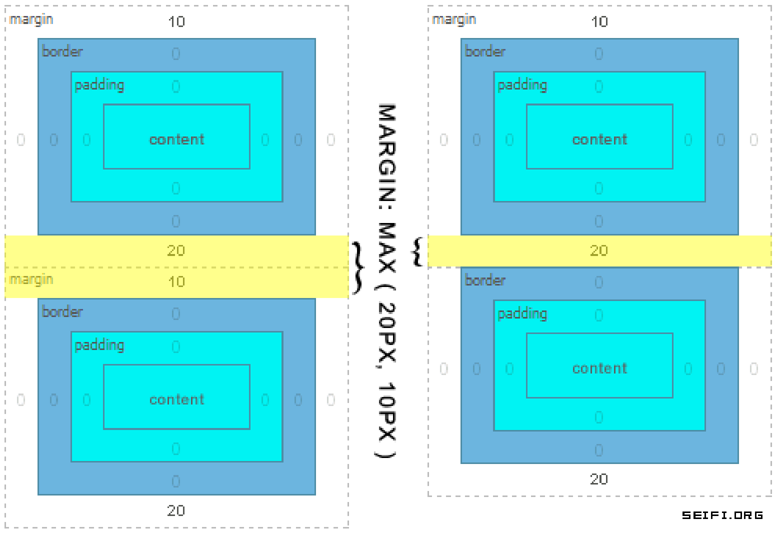
- Non-”inline” contents usually have a dedicated box in the layout
  - > that are placed under each other
  - > or in case of **float** are placed next to each other
  - > or we can also set their **absolute positions**
    - – In this case we have to solve text wrapping around the area
- We can set the following properties for the box: width, height and margin, border, padding



Content-box sizing

- The standard hasn't been clear about what width x height refers to.
- According to older IE versions the width and height properties includes only the content. Border, padding, or margin are not included – this is called the **border-box** sizing
- Later W3C made clarified it. The width and height properties *includes content, padding and border, but not the margin* – this is called the **content-box** sizing.

We cannot always rely on margins:



*"While margin collapsing is **great for written text such as paragraphs and headings etc.**, it can get somewhat tricky **if you're trying to get pixel perfect spacing between your boxes**, so there might come a time when you want to disable margin collapsing.„*

- But there is no margin-collapsing CSS property.
  - > "disable“ here means that we have to trigger some side effect that prevents margin collapsing.
- https://seifi.org/css/understanding-taming-collapsing-margins-in-css.html/

Creating layout with CSS2 is not easy

- Strong coupling between the DOM and the style sheets
  - > The place of on element in the DOM strongly influences its display
- Hard to make it adaptive
  - > Recent usual web layouts: optimized for 1024 pixels, on wider screens there is big empty space on both sides.
  - > The content is not long enough, hard to position the footer to the bottom of the screen.
- Fragile UI – every modification is dangerous
  - > E.g. if you increase the padding you shouldn't forget to decrease the content. That may easily ruin a float-based layout.

CSS3

CSS2 is primarily used to provide styling to web pages, including color, layout, background, font, and border properties. The main objective of CSS is to improve content accessibility, provide enhanced flexibility and control, and specify presentation characteristics.

CSS3 stands for Cascading Style Sheets Level 3. It is an advanced version of CSS, used for structuring, styling, and formatting web pages. CSS3 introduces several new features and is supported by all modern web browsers. One of the most significant advancements in CSS3 is the splitting of CSS standards into separate modules, making it simpler to learn and use.

CSS	**CSS3**
Capable of positioning texts and objects.	Capable of making web pages more attractive and takes less time to create. It is backward compatible with CSS.
Does not support responsive design.	Supports responsive design.
Cannot be split into modules.	Can be broken down into modules.
Cannot build 3D animation and transformation.	Supports animation and 3D transformations.
Slower compared to CSS3.	Faster than CSS.
Uses a set of standard colors and basic color schemes.	Has a good collection of HSL, RGBA, HSLA, and gradient colors.
Supports only single text blocks.	Supports multi-column text blocks.
Does not support media queries.	Supports media queries.
Not supported by all types of modern browsers.	Supported by all modern browsers.
Requires manual development of rounded gradients and corners.	Provides advanced codes for setting rounded gradients and corners.
No special effects like shadowing text or text animation; requires jQuery and JavaScript for animations.	Supports text shadows, visual effects, and a wide range of font styles and colors.
Can add background colors to list items and lists, and set images for list items.	Lists have a special display property and list items have counter reset properties.
Developed in 1996.	Released in 2005.
Memory intensive.	Consumes less memory compared to CSS.

New Features of CSS3

1. **Combinator**: CSS3 introduces a new general sibling combinator, which matches sibling elements using the tilde (~) combinator.
2. **CSS Selectors**: CSS3 selectors are more advanced than the simple selectors offered by CSS, providing a sequence of easy-to-use and simple selectors.
3. **Pseudo-elements**: CSS3 adds many new pseudo-elements for easier and more detailed styling. The new convention of double colons (::) is also introduced.
4. **Border Style**: CSS3 includes new border styling features like border-radius, border-image-slice, border-image-source, and values for “width stretch”.
5. **Background Style Properties**: CSS3 introduces new background style properties such as background-clip, background-size, background-style, and background-origin.

Boxes and texts

- New / extended background and border properties
  - Pl. rounded border corner, image border, etc.
  - Finally we have box-sizing: border-box!
- Gradients
- Webfonts
- Multi-column layout
- New properties for texts
  - E.g. text wrapping conditions

Transitions, transformations and animation

- **2D and 3D transformations:** translate, rotate, scale, skew, matrix transform
- **Transitions:** transitions allows you to change property values smoothly (from one value to another) E.g. transition: width 2s, height 2s, color 2s;
  - Easy to use, quite spectacular
- **Animations:** An animation lets an element gradually change from one style to another. You can change as many CSS properties you want. Keyframes hold what styles the element will have at certain times.

New options in creating the layout

- Media Query
  - Different CSS rules depending on the size of the screen
  - Mobil web support (responsive design) – see later
- New layout models (display: ...)
  - Flexbox
    - Supported in newer browser versions
  - Grid
    - Not mature enough, W3C keeps changing it
  - The goal is to have a layout option that is more robust than the float-based solutions for creating web app layouts

The display property in CSS determines how an element is displayed on a web page. It controls the layout of an element, influencing how it interacts with other elements around it. Here's a breakdown of some of the most commonly used display values:

- **display: block**, An element with display: block takes up the full width available, starting on a new line, and pushing the next element down.
- **display: inline**, An inline element does not start on a new line and only takes up as much width as necessary. It does not respect the width or height properties.
- **display: inline-block**, Combines the characteristics of inline and block. Elements are displayed inline (side-by-side) but allow setting width and height.
- **display: none**, Hides an element completely. It is removed from the document flow, so it doesn't take up any space.

Inline-flex, flex, table, list-item, ...etc

CSS3 new layouts

1. **Flexbox (Flexible Box Layout):** Flexbox is designed for one-dimensional layouts, meaning it allows you to control the layout of items along a single row or column.

- **Flex-direction**: Controls the direction of flex items (row, column, etc.).
- **Justify-content**: Aligns items along the main axis (e.g., space-between, center).
- **Align-items & align-self**: Aligns items along the cross axis.
- **Order**: Changes the order of items without altering the HTML structure.
- **Wrap**: Allows items to wrap onto multiple lines if they don't fit in a single row/column.

Try this out: https://flexboxfroggy.com/

Check the code example at the bottom of this document

2. **CSS Grid Layout:** CSS Grid is a two-dimensional layout system, which means it can handle both rows and columns simultaneously. It provides greater control for building complex and responsive grid layouts.

- **Grid-template-rows & grid-template-columns**: Define the number of rows and columns in a grid.
- **Grid-gap (gap)**: Defines the space between rows and columns.
- **Grid-template-areas**: Allows naming areas of the grid for easier placement of content.
- **Place-items**: Centers items horizontally and vertically within their grid cell.
- **Auto-fit and auto-fill**: Create responsive grids that adjust based on available space.

Check the code example at the bottom of this document

@supports block

The @supports rule in CSS, also known as the **CSS Feature Queries**, allows you to apply styles conditionally based on the browser's support for a specific CSS property or feature. This is useful for implementing **progressive enhancement**—where you add new CSS features only if the browser supports them, while providing a fallback for older browsers.

- If a propery: value pair is supported it will be applied, otherwise it won't.
- If a browser doesn't know the @supports block it will ignore its content

```css
@supports (property: value) {
  /* CSS rules for browsers that support the property/value combination */
}

@supports not (property: value) {
  /* CSS rules for browsers that do NOT support the property/value combination */
}
```

- When a browser supports a CSS3 feature it may not support its latest syntax and all the options
- It is an old problem in the world of CSS. Designers around CSS3 try to overcome it.

Solution: vendor prefixes

- When IE implemented the width and height CSS properties they included the padding in the width as the standard didn't have a clear guidance
  - > For mysterious reasons not this model was finally accepted in the standard – that made targeting multiple browsers very difficult
- When they started working on CSS3, web browser developer companies agreed to use browser specific prefixes for each newly introduced property until its specification gets finalized to avoid later naming conflicts with the standardized property name.

Vendor prefixes are a way for browser vendors (like Google, Mozilla, Apple, and Microsoft) to add support for new or experimental CSS properties before they become a part of the official CSS specifications. These prefixes are added directly before a CSS property or value, allowing developers to use new features while they are still in development or while they are being standardized.

- **-webkit-**: For **Chrome**, **Safari**, and newer versions of **Opera** (based on WebKit or Blink engine).
- **-moz-**: For **Mozilla Firefox**.
- **-ms-**: For **Internet Explorer** and **Microsoft Edge** (older versions before the Chromium-based version).
- **-o-**: For **older versions of Opera** (before it switched to WebKit/Blink).

```
body {
  /*Safari 5.1-6*/
  background: -webkit-linear-gradient(
    left, rgba(255,0,0,0), rgba(255,0,0,1));
  /*Opera 11.1-12*/
  background: -o-linear-gradient(
    right, rgba(255,0,0,0), rgba(255,0,0,1));
  /*Firefox 3.6-15*/
  background: -moz-linear-gradient(
    right, rgba(255,0,0,0), rgba(255,0,0,1));
  /*Standard*/
  background: linear-gradient(
    to right, rgba(255,0,0,0), rgba(255,0,0,1));
}
```

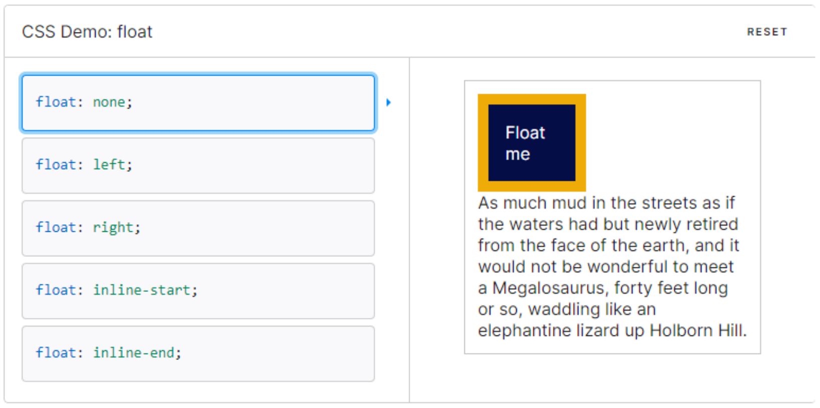
What is the problem with prefixes?

- Lots of redundant code, easy to misuse
  - > They are usually generate (e.g. from LESS)
- Very difficult to debug
  - > In many cases it is not only the syntax that is different
- Vendor prefixing hasn't solved the original problem
  - > Vendor prefixes will also stay with us for years just like the width implementation of IE
- New direction

> New properties won't work until their specification is finalized!

Float-based layout

- Original goal: wrap text around a table or an image
- The subsequent text wraps around the "floated" element
- It can be used to position items next to each other (by default they would be placed under each other)
- The float property specifies whether an element should float to the left, right, or not at all.
- **Note:** Absolutely positioned elements ignore the float property!
- **Note:** Elements next to a floating element will flow around it. To avoid this, use the clear property or the clearfix hack



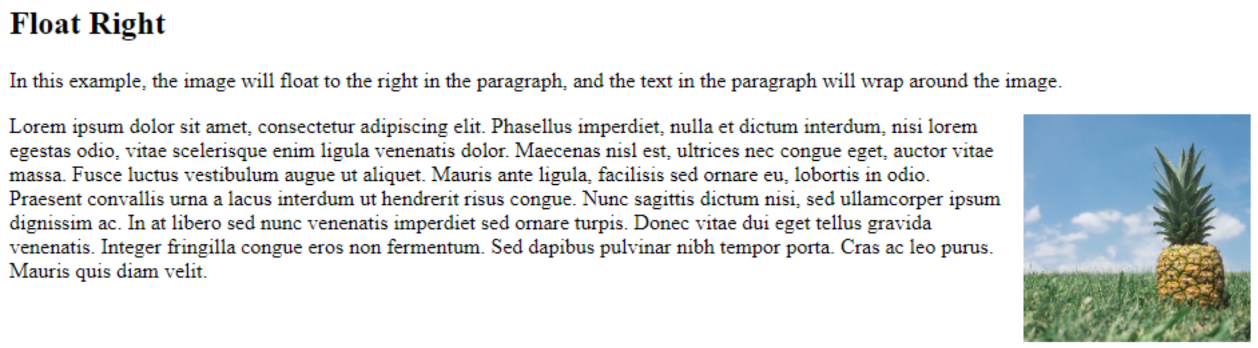
```
float: none;
float: left;
float: right;
float: inline-start;
float: inline-end;
```

As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

A code example:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: right;
}
</style>
</head>
<body>

<h2>Float Right</h2>

<p>In this example, the image will float to the right in the paragraph, and the text in the
paragraph will wrap around the image.</p>

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;margin-left:15px;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum
interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est,
ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante
ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut
hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero
sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis.
Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo
purus. Mauris quis diam velit.</p>

</body>
</html>
```

The results:



Float Right

In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

inline-block

The inline-block value in CSS is a **display** property that combines characteristics of both inline and block elements. When an element is set to display: inline-block, it behaves like an inline element in that it sits in line with other elements, but it also allows block-level features, such as setting a specific width, height, margin, and padding.

- An other option place block elements next to each other
  - display: inline-block;
- Advantages:
  - No clearfix, no container collapse
  - Reacts to the vertical-align property
- Disadvantages:
  - Sensitive for whitespaces

Code examples

CSS flexbox example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Flexbox Demo</title>
  <style>
    .flex-container {
      display: flex;
      background-color: #f1f1f1;
    }

    .flex-container > div {
      background-color: DodgerBlue;
      color: white;
      width: 100px;
      margin: 10px;
      text-align: center;
      line-height: 75px;
```

```html
      font-size: 30px;
    }
    </style>
</head>
<body>
    <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
      </div>
      <!-- The order property can change the order of the flex items: -->
       <h2> The order property can change the order of the flex items:</h2>
      <div class="flex-container">
        <div style="order: 3">1</div>
        <div style="order: 2">2</div>
        <div style="order: 4">3</div>
        <div style="order: 1">4</div>
      </div>
      <!-- The flex-grow Property
       The flex-grow property specifies how much a flex item will grow relative to the rest of the
flex items.-->
       <h2> The flex-grow Property</h2>
        <p>The flex-grow property specifies how much a flex item will grow relative to the rest of
the flex items.</p>
      <div class="flex-container">
        <div style="flex-grow: 1">1</div>
        <div style="flex-grow: 1">2</div>
        <div style="flex-grow: 8">3</div>
      </div>
      <!-- The flex-shrink Property:
       The flex-shrink property specifies how much a flex item will shrink relative to the rest of
the flex items. -->
       <h2>The flex-shrink Property</h2>
       <p>The flex-shrink property specifies how much a flex item will shrink relative to the rest
of the flex items.</p>
      <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div style="flex-shrink: 0">3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
        <div>8</div>
        <div>9</div>
        <div>10</div>
      </div>
      <!-- The flex-basis Property:
       The flex-basis property specifies the initial length of a flex item.-->
       <h2>The flex-basis Property</h2>
        <p>The flex-basis property specifies the initial length of a flex item.</p>
       <div class="flex-container">
        <div>1</div>
        <div>2</div>
        <div style="flex-basis: 200px">3</div>
        <div>4</div>
      </div>
      <!-- The flex Property
The flex property is a shorthand property for the flex-grow, flex-shrink, and flex-basis
properties.-->
<h2>The flex Property </h2>
    <p>The flex property is a shorthand property for the flex-grow, flex-shrink, and flex-basis
properties.</p>
<div class="flex-container">
```

```
    <div>1</div>
    <div>2</div>
    <div style="flex: 0 0 200px">3</div>
    <div>4</div>
  </div>
</body>
</html>
```

CSS grid example

```
<!DOCTYPE html>
<html>
<head>
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}
.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
</head>
<body>

<h1>Grid Elements</h1>

<p>A Grid Layout must have a parent element with the <em>display</em> property set to <em>grid</em>
or <em>inline-grid</em>.</p>

<p>Direct child element(s) of the grid container automatically becomes grid items.</p>

<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>

</body>
</html>
```

Mobile and Web Development

Useful links and references:

https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS
https://www.geeksforgeeks.org/difference-between-css-and-css3/
flexbox: https://www.youtube.com/watch?v=fYq5PXgSsbE&ab_channel=WebDevSimplified
https://www.w3schools.com/css