

CSS

Saba sadat Faghih Imani

2023

CSS Introduction

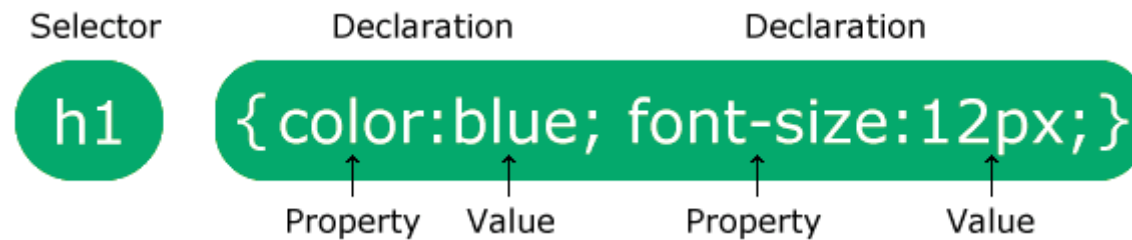
- CSS is the language we use to style an HTML document.
- CSS describes how HTML elements should be displayed.
- Cascading Style Sheet (CSS) defines the presentation and style of web pages
 - How elements are **presented** in web pages!!
- Override the default presentation of elements
 - If presentation is not specified, browser uses its own default presentation for HTML elements

CSS Introduction

- Advantages:
- Reuse styling rules in multiple pages.
 - Make easier site management
 - Save time
- More control over layout
- Style+JavaScript => dynamic presentation
- Multiple presentation of the same content
 - Media dependent presentation

CSS Syntax

- A CSS rule consists of a selector and a declaration block.
- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.



Sample Properties

- Font:
 - font-family
 - font-size
 - font-style
 - font-weight
- Text:
 - text-align
 - color
 - letter-spacing
 - word-spacing
- Background:
 - background-color
 - background-image

Sample Properties

- Table:
 - color
 - Text-align
 - border
 - width , height
- List:
 - list-style-type
 - list-style-image
- Box Model:
 - border-color, border-style, border-width
 - width, ...
 - background-color, ...
- Position:
 - bottom, left, right, ...

CSS Style Types: Inline Style


- Add styles to each tag within HTML file!!!
 - Used to format a single HTML element
 - Selector is implicitly specified
 - Style is given as an attribute

```
<h1 style="color: red; font-family: sans-serif">Test Heading 1</h1>
```

- Element based
 - Hard to update
 - Violates structure-style separation
 - (Only for) Styling exceptional elements

CSS Style Types: Internal Style

- A style is used in the entire HTML file
- Used to control style of elements (e.g., all h1) in a single web page

```
<head>  
  <style type="text/css">  
    h1 {color: red;  
    font-family: sans-serif;}  
  </style>  
</head>
```


CSS Style Types: External Style

- A text file containing style rules
- Used to control style in multiple web pages
- Example
 - A text document with .css extension contains:

```
h1, h2, h3, h4, h5, h6 {  
    color: #FF0000;  
    font-family: sans-serif;  
}
```

External style file is used in HTML web page through linking it to the web page

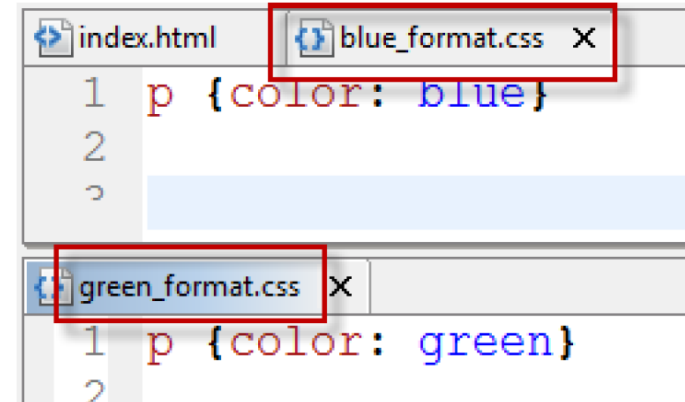
```
<head>  
  <title>External CSS</title>  
  <link href="external_css.css" rel="stylesheet" type="text/css" />  
</head>
```

CSS Style Types: External Style Advantages

- Avoid repeating styles for each page
 - It is easier to update whole site
- CSS can be cached independent of HTML
- Different style sheets can be attached to the same document
 - Personalized & Customization & Media dependent
- A style sheet can import and use styles from other style sheets

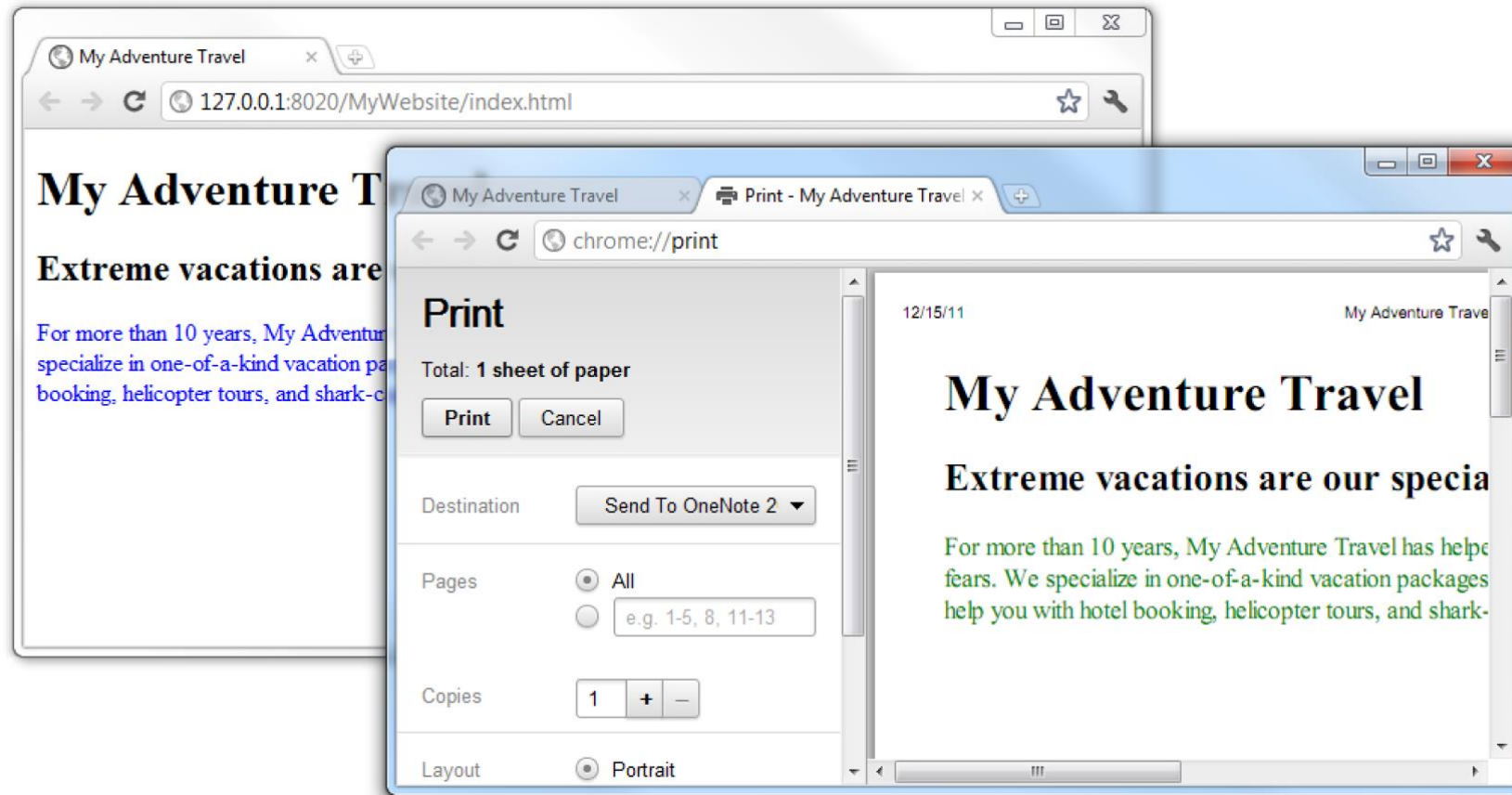
Media Dependent Presentation

- Web page presentation can be depended on media



```
<!DOCTYPE html>
<html>
  <head>
    <title>My Adventure Travel</title>
    <link rel="stylesheet" type="text/css" href="green_format.css" media="print" />
    <link rel="stylesheet" type="text/css" href="blue_format.css" media="screen" />
  </head>
  <body>
    <h1 class="myClass1">My Adventure Travel</h1>
    <h2>Extreme vacations are our specialty</h2>
    <p>For more than 10 years, My Adventure Travel has helped customers fulfill their
dreams and conquer their fears. We specialize in one-of-a-kind vacation packages to the
most exciting destinations in the world. Let us help you with hotel booking, helicopter
tours, and shark-cage rentals, either online or in person.</p>
  </body>
</html>
```

Media Dependent Presentation



Cascading Order

- What style will be used when there is more than one style specified for an HTML element?
- All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:
 1. Inline style (inside an HTML element)
 2. External and internal style sheets (in the head section)
 3. Browser default
- So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

Inheritance

- XHTML document is interpreted as a tree
 - DOM tree
- Children inherit some styles from their parent
 - Not all properties, e.g. border

```
<head>
  <style type="text/css">
    p {font-style: italic;}
    code {color: red;}
  </style>
</head>
<body>
  <p>This is <code> a code </code> in paragraph </p>
</body>
```

This is a code in paragraph

Conflicting Styles & Overriding

- What happen if multiple style rules specify different values for the same property of an element?
 - External style: `p {color: red}`
 - Internal style: `p {color: blue}`
- They are conflicting
- What is the final rule?
 - It depends on rule types, order, ...
- Specified by the overriding algorithm

Overriding

- In general,
 - Priority 1: more specific
 - Inline > ID > Class > Element (Tag) > Div/Span
 - Priority 2: more closer to the element if they are specific as the same as each other (e.g., both are id selector)
 - Inline > Internal
 - Inline > External
 - Internal <?> External
 - *The style that comes after the other*
- styles override more general styles
 - Children's style overrides parent's style

Overriding

```
<head>
  <style type="text/css">
    p{color: ■ red;}
    p.c{color: ■ blue;}
    p#i{color: ■ green;}
  </style>
</head>
<body>
  <p class="c" style="color: ■ black;"> This is test </p>
  <p class="c" id="i">This is test</p>
  <p class="c">This is test </p>
  <p>This is test </p>
</body>
```

This is test

This is test

This is test

This is test

Overriding

- To prevent overriding, add **!important**

```
<head>
  <style type="text/css">
    h1{font-style: italic;
      color: blue !important}
  </style>
</head>
<body>
  <h1> Heading 1</h1>
  <h1 style="font-style: normal; color: red">Heading 1</h1>
</body>
```

Heading 1

Heading 1

CSS Selectors

CSS Selectors

- Simple selectors (select elements based on name, id, class)
- Attribute selectors (select elements based on an attribute or attribute value)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-element selectors (select and style a part of an element)
- We will discuss later in more details

CSS Selectors: Element Selector

- The element selector selects HTML elements based on the element name.

```
<head>
  <style type="text/css">
    * {color: ■blue}
    h1 {text-decoration: underline}
    p {font-style: italic}
  </style>
</head>
<body>
  <h1>Element Selector</h1>
  <p> This is a paragraph </p>
</body>
```

Element Selector

This is a paragraph

CSS Selectors: id Selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
<head>
  <style type="text/css">
    #blue_heading{color: blue;}
    #red_heading{color: red;}
  </style>
</head>
<body>
  <h2 id="blue_heading">This is blue heading </h2>
  <h2 id="red_heading">This is red heading </h2>
</body>
```

This is blue heading

This is red heading

CSS Selectors: Class Selector

- The class selector selects HTML elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the class name.
- You can also specify that only specific HTML elements should be affected by a class.
- HTML elements can also refer to more than one class.

```
<head>
  <style type="text/css">
    .red{color:red;}
    .bold{font-weight:bold;}
  </style>
</head>
<body>
  <p class="red">This is a red paragraph </p>
  <p class="bold">This is a bold paragraph </p>
  <p class="red bold"> This is a red-bold paragraph </p>
</body>
```

This is a red paragraph

This is a bold paragraph

This is a red-bold paragraph

CSS Grouping Selector

- The grouping selector selects all the HTML elements with the same style definitions.

```
<head>
  <style>
    h1, h2, p {
      color: red;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <h1>Hello World!</h1>
  <h2>Smaller heading!</h2>
  <p>This is a paragraph.</p>
</body>
```

Hello World!

Smaller heading!

This is a paragraph.

CSS Selectors: Attribute Selector

- It is possible to style HTML elements that have specific attributes or attribute values.

Selector	Example	Example description
[attribute]	[target]	Selects all elements with a target attribute
[attribute=value]	[target="_blank"]	Selects all elements with target="_blank"
[attribute~=value]	[title~="flower"]	Selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower"
[attribute =value]	[lang = "en"]	Selects all elements with a lang attribute value "en" or "en" followed by a hyphen (-)
[attribute^=value]	a[href^="https"]	Selects all <a> elements with a href attribute value starting with "https"
[attribute\$=value]	a[href\$=".pdf"]	Selects all <a> elements with a href attribute value ending with ".pdf"
[attribute*=value]	a[href*="w3schools"]	Selects all <a> elements with a href attribute value containing the substring "w3schools"

CSS Selectors: Attribute Selector

```
<head>
<style>
  [class="top"] {
    background: yellow;
  }
  [class^="top"] {
    text-decoration: underline;
  }
</style>
</head>
<body>

<h2>CSS Attribute Selector</h2>

<h1 class="top-header">Welcome</h1>
<p class="top-text">Hello world!</p>
<p class="topcontent">Are you learning CSS?</p>

</body>
```

CSS Attribute Selector

Welcome

Hello world!

Are you learning CSS?

CSS Selectors: Combinators

- A CSS selector can contain more than one simple selector.
- Between the simple selectors, we can include a combinator.
- There are four different combinators in CSS:
 - Descendant selector (space)
 - Child selector (>)
 - Adjacent sibling selector (+)
 - General sibling selector (~)

Descendant Selector

- The descendant selector matches all elements that are descendants of a specified element.

```
<head>
  <style>
    ol{color: ■red;}
    ol li{color: ■blue;}
  </style>
</head>
<body>
  <ol>
    <li> Item 1 </li>
    <ul> <li> Netsted1</li> </ul>
    <li> Item 2 </li>
    <dl> <dt> def: <dt> <dd>Definition </dd>
    </dl>
  </ol>
</body>
```

- 1. Item 1
 - Netsted1
- 2. Item 2

def:
Definition

Child Selector (>)

- The child selector selects all elements that are the children of a specified element.

```
<head>
  <style>
    ol{color: red;}
    ol > li{color: blue;}
  </style>
</head>
<body>
  <ol>
    <li> Item 1 </li>
    <ul> <li> Netsted1</li> </ul>
    <li> Item 2 </li>
    <dl> <dt> def: <dt> <dd>Definition </dd>
    </dl>
  </ol>
</body>
```

1. Item 1

◦ Netsted1

2. Item 2

def:



Definition

Adjacent Sibling Selector (+)

- The adjacent sibling selector is used to select an element that is directly after another specific element.
- Sibling elements must have the same parent element, and "adjacent" means "immediately following".

General Sibling Selector (~)

- The general sibling selector selects all elements that are next siblings of a specified element.

```
<head>
  <style>
    h2+p{color:  red;}
    h3~p{color:  green;}
  </style>
</head>
<body>
  <h2>Heading 2 </h2>
  <p>Next sibling of h2 </p>
  <p>Another sibling of h2 </p>
  <h3>Heading 3 </h3>
  <p>Next sibling of h3 </p>
  <p>Another sibling of h3 </p>
</body>
```

Heading 2

Next sibling of h2

Another sibling of h2

Heading 3

Next sibling of h3

Another sibling of h3

CSS Selectors: Pseudo-class selectors

- A pseudo-class is used to define a special state of an element.
- first-child matches a specified element that is the first child of another element.
 - `ul li:first-child {color: blue;}`
- unvisited link
 - `a:link {color: #FF0000;}`
- visited link
 - `a:visited { color: #00FF00;}`
- mouse over link
 - `a:hover { color: #FF00FF;}`
- selected link
 - `a:active { color: #0000FF;}`
- hover on <div>
 - `div:hover {background-color: blue;}`

CSS Selectors: Pseudo-element selectors

- A CSS pseudo-element is used to style specified parts of an element.
- `::first-letter` (in heading & paragraph)
 - `p::first-letter {font-size: 200%;}`
- `::first-line` (in paragraph)
 - `p::first-line { color: red;}`
- `::before` can be used to insert some content before the content of an element.
 - `h1::before {content: url(smiley.gif);}`
- `::after` can be used to insert some content after the content of an element.
 - `h1::after {content: url(smiley.gif);}`
- `::selection` matches the portion of an element that is selected by a user.
 - `::selection {color: red; background: yellow;}`

CSS Selectors: Pseudo-class/element selectors

```
<head>
  <style>
    p code:first-child {color: blue;}
    p::first-letter {font-size: 200%;}
    p::first-line {color: red;}
    ::selection {color: red; background: yellow;}
    h1:hover {color: blueviolet;}
  </style>
</head>
<body>
  <h1>This is heading</h1>
  <p>This is the first <code> code
  </code>, this is second <code>
  code</code>. <br/>
</p>
</body>
```

This is heading

This is the first code , this is second code.

CSS Color

- Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.
- Same as color name "Tomato":

`rgb(255, 99, 71)`

`#ff6347`

`hsl(9, 100%, 64%)`

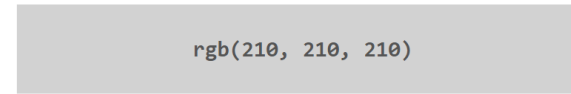
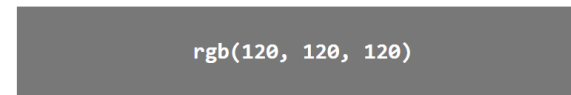
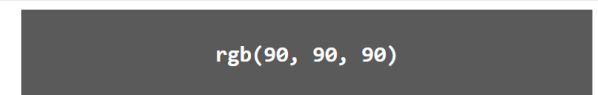
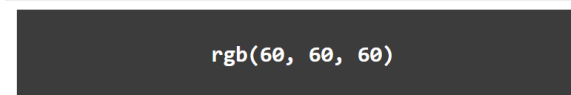
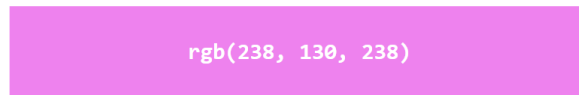
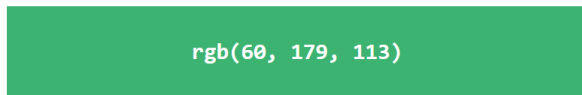
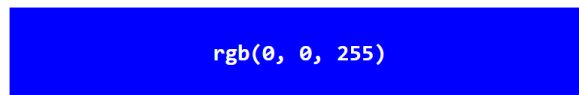
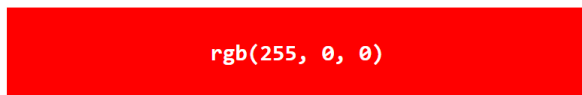
- Same as color name "Tomato", but 50% transparent:

`rgba(255, 99, 71, 0.5)`

`hsla(9, 100%, 64%, 0.5)`

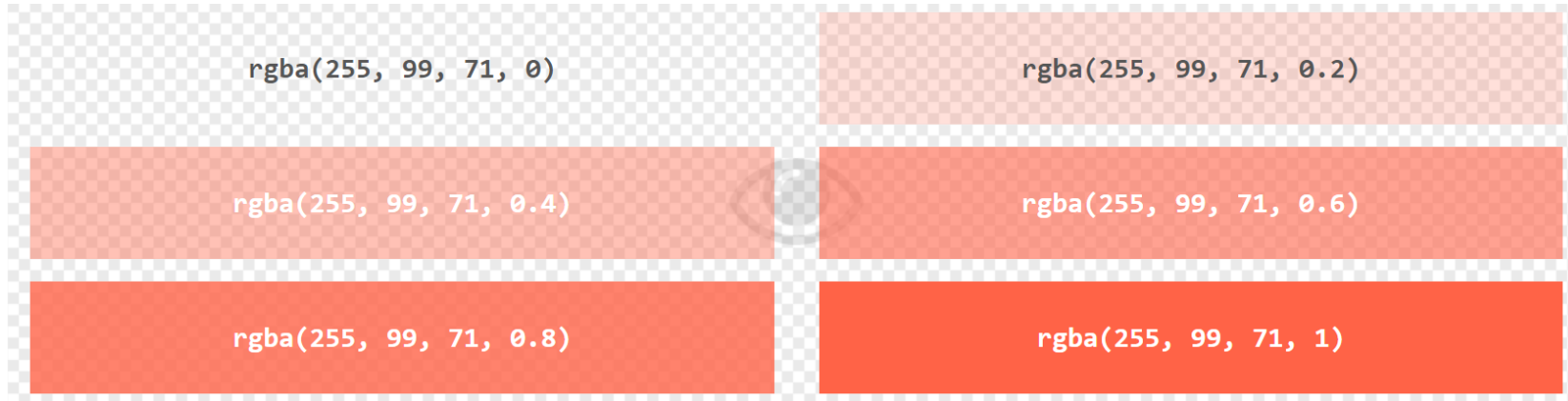
CSS Color: RGB

- `rgb(red, green, blue)`
- Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.
- For example, `rgb(255, 0, 0)` is displayed as red, because red is set to its highest value (255) and the others are set to 0.
- To display black, set all color parameters to 0, like this: `rgb(0, 0, 0)`.
- To display white, set all color parameters to 255, like this: `rgb(255, 255, 255)`.



CSS Color: RGBA

- RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
- An RGBA color value is specified with:
 - `rgba(red, green, blue, alpha)`
- The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all).



CSS Color: HEX

- `#rrggbb`
 - Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).
 - For example, `#ff0000` is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).
 - To display black, set all values to 00, like this: `#000000`.
 - To display white, set all values to ff, like this: `#ffffff`.
 - 3 Digit HEX Value (`#rgb`): The 3-digit hex code is a shorthand for some 6-digit hex codes. The 3-digit hex code can only be used when both the values (RR, GG, and BB) are the same for each component
 - `fc9 => ffcc99`
 - `f0f => ff00ff`

CSS Color: HSL

- `hsl(hue, saturation, lightness)`
- Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
- Saturation is a percentage value. 0% means a shade of gray, and 100% is the full color.
- Lightness is also a percentage. 0% is black, 50% is neither light or dark, 100% is white

<code>hsl(0, 100%, 50%)</code>	<code>hsl(240, 100%, 50%)</code>
<code>hsl(147, 50%, 47%)</code>	<code>hsl(300, 76%, 72%)</code>
<code>hsl(39, 100%, 50%)</code>	<code>hsl(248, 53%, 58%)</code>

CSS Text

- Text Color: The `color` property is used to set the color of the text
- Text Shadow: The `text-shadow` property adds shadow to text.
 - `text-shadow: 2 2 3px #ff0000;`
 - `text-shadow`: The horizontal shadow, the vertical shadow, a blur effect, color;
- Text Alignment
 - `text-align`: is used to set the horizontal alignment of a text. A text can be left or right aligned, centered, or justified.
 - `text-align-last`: specifies how to align the last line of a text.
- Text Decoration
- Text Transformation
- Text Spacing

CSS Text: Text Decoration

Property	Description
text-decoration	Sets all the text-decoration properties in one declaration
text-decoration-line	Specifies the kind of text decoration to be used (underline, overline, etc.)
text-decoration-color	Specifies the color of the text-decoration.
text-decoration-style	Specifies the style of the text decoration (solid, dotted, wavy, dashed, double, etc.)
text-decoration-thickness	Specifies the thickness of the text decoration line

```
<head>
<style>
  p {
    text-decoration: underline red wavy 5px;
  }
</style>
</head>
<body>
  <p>This is nice.</p>
</body>
```

This is nice.

CSS Text: Text Transformation

- The `text-transform` property is used to specify uppercase and lowercase letters in a text.

```
<head>
<style>
  p.uppercase {
    text-transform: uppercase;
  }

  p.lowercase {
    text-transform: lowercase;
  }

  p.capitalize {
    text-transform: capitalize;
  }
</style>
</head>
<body>
<h1>Using the text-transform property</h1>
<p class="uppercase">This text is transformed to uppercase.</p>
<p class="lowercase">This text is transformed to lowercase.</p>
<p class="capitalize">This text is capitalized.</p>
</body>
```

Using the text-transform property

THIS TEXT IS TRANSFORMED TO UPPERCASE.

this text is transformed to lowercase.

This Text Is Capitalized.

CSS Text: Text Spacing

Property	Description
letter-spacing	Specifies the space between characters in a text
line-height	Specifies the line height
text-indent	Specifies the indentation of the first line in a text-block
word-spacing	Specifies the space between words in a text

```
<head>
<style>
p.one {
  text-indent: 10px;
  letter-spacing: 7px;
  word-spacing: 10px;
}
</style>
</head>
<body>
<p>This is a paragraph.</p>
<p class="one">This is a paragraph.</p>
</body>
```

This is a paragraph.

T h i s i s a p a r a g r a p h .

CSS Lists

- The CSS list properties allow you to:
 - Set different list item markers for ordered lists.
 - Set different list item markers for unordered lists
 - Set an image as the list item marker
 - Add background colors to lists and list items
- **list-style-type**: Specifies the type of list item marker.
 - Values: circle, square, upper-roman, lower-alpha, ...
 - Some of the values are for unordered lists, and some for ordered lists.
- **list-style-image**: specifies an image as the list item marker.

Styling List With Colors

```
<head>
<style>
ol {
  background: #ff9999;
  padding: 20px;
  width: 20%;
}
ul {
  background: #3399ff;
  padding: 20px;
  width: 20%
}

ol li {
  background: #ffe5e5;
  color: darkred;
  padding: 5px;
  margin-left: 35px;
}
ul li {
  background: #cce5ff;
  color: darkblue;
  margin: 5px;
}
</style>
</head>
<body>
<h1>Styling Lists With Colors</h1>
<ol>
  <li>Coffee</li> <li>Tea</li> <li>Coca Cola</li>
</ol>

<ul>
  <li>Coffee</li> <li>Tea</li> <li>Coca Cola</li>
</ul>
</body>
```

Styling Lists With Colors

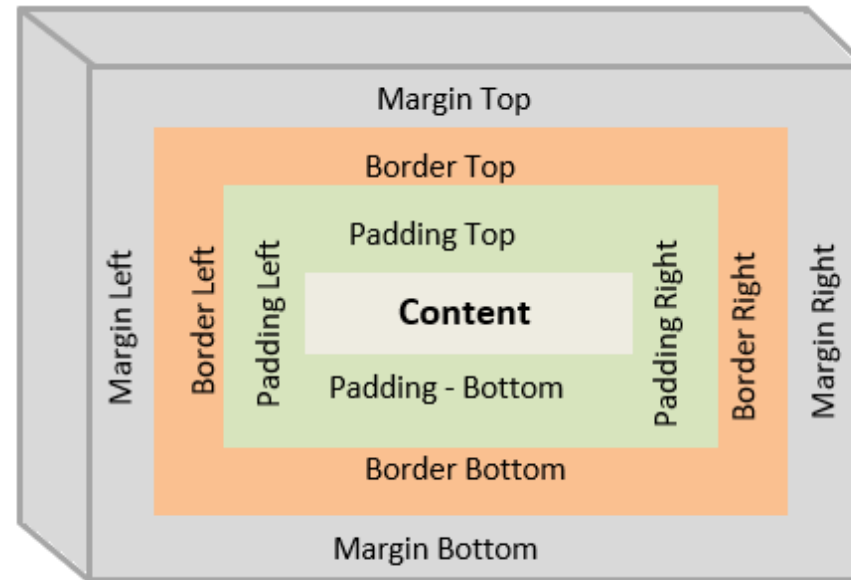
1. Coffee
2. Tea
3. Coca Cola

- Coffee
- Tea
- Coca Cola

CSS Box Model

Box Model

- The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.
 - Content - The content of the box, where text and images appear
 - Padding - Clears an area around the content. The padding is transparent
 - Border - A border that goes around the padding and content
 - Margin - Clears an area outside the border. The margin is transparent



Box Model

- In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.
- Important: When you set the width and height properties of an element with CSS, you just set the width and height of the content area. To calculate the full size of an element, you must also add padding, borders and margins.
- This <div> element will have a total width of 350px:
- ```
div {
 width: 320px;
 padding: 10px;
 border: 5px solid gray;
 margin: 0;
}
```
- **Total element width** = width + left padding + right padding + left border + right border + left margin + right margin
- **Total element height** = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin



# Box Model

- Border
  - `border-left`, `border-right`, `border-bottom`, `border-top`
    - `border`: All borders
  - `border-X-color`, `border-X-style`, `border-X-width`
- Padding
  - `padding-left`, `padding-right`, `padding-top`, `padding-bottom`
    - `padding`: All padding
- Margin
  - `margin-left`, `margin-right`, `margin-top`, `margin-bottom`
    - `margin`: All margins
  - Centering block elements `margin-left: auto; margin-right: auto;`
- Content (dimensions)
  - `width`, `height`, `max-width`, ...

# Box Model

```
<head>
<style>
#first {
 border-style:solid;
 border-width:3px;
 border-color: blueviolet;
 margin-left:1cm;
 padding-left:3cm;
 margin-right:1cm;
 padding-right:0.5cm;
}
#second {
 border-style:dotted;
 border-width:3px;
 border-color: blue;
 margin-left:0.2cm;
 padding-left:1.5cm;
 margin-right:3cm;
 padding-right:1.5cm;
}
</style>
</head>
<body>
 This is the first box,
 and this is thesecond box
</body>
```

This is the

first box

, and this is the

second box

# Vertical Margin Collapse

- When a bottom margin of one element meets the top margin of another, only the larger of the two will show
  - This only applies to vertical margins; the same is not true for left and right margins
- If the <h1> element has a bottom margin of 50px and the <h2> element has a top margin set to 20px. Common sense would seem to suggest that the vertical margin between the <h1> and the <h2> would be a total of 70px (50px + 20px). But due to margin collapse, the actual margin ends up being 50px.

# CSS Background

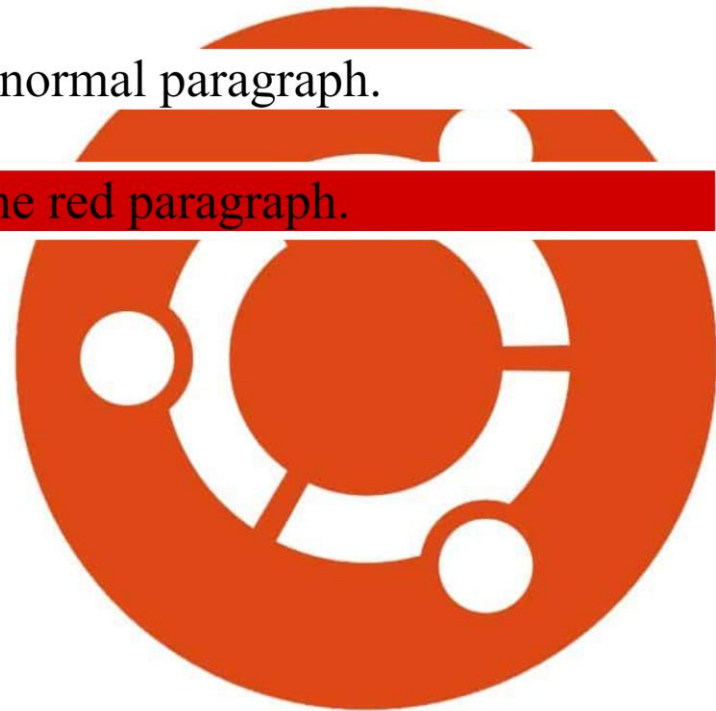
- **background-color**: specifies the background color of an element.
- **background-image**: specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element.
- **background-repeat**: **no-repeat**, **repeat-x**, **repeat-y**
- **background-attachment**: specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page)
- **background-position**: specifies the position of the background image.
- **background** (shorthand property): When using the shorthand property the order of the property values is:
  - background-color, background-image, background-repeat, background-attachment, background-position

# CSS Background

```
<head>
 <style>
 body{
 background-image: url("../img/ubuntu1.jpg");
 background-repeat: no-repeat;
 font-size: 3em;
 }
 p {background-color: white;}
 p#red{
 width: 50%;
 border-style: solid;
 border-width: 8px;
 border-color: white;
 background-color: #CF0000;
 }
 </style>
</head>
<body>
 <p> This is a normal paragraph. </p>
 <p id="red">This is the red paragraph. </p>
</body>
```

This is a normal paragraph.

This is the red paragraph.



# CSS Layout

# Layout Design = Element Positioning

- Three positioning schemes
  - Normal
    - Block-level elements flow from top to bottom
    - Inline elements flow from left to right
  - Specified position
    - Element is taken out from normal flow
    - It is placed in the specified position
  - Float
    - Elements is taken out from normal flow
    - It is placed as far right/left as possible to previous element

# The Position Property

- The `position` property specifies the type of positioning method used for an element.
  - `static`
  - `relative`
  - `fixed`
  - `absolute`
  - `sticky`
- Elements are then positioned using the `top`, `bottom`, `left`, and `right` properties. However, these properties will not work unless the `position` property is set first. They also work differently depending on the position value.



# position: static;

- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.
- An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page.

# position: relative;

- An element with `position: relative;` is positioned relative to its normal position.
- Position is specified by top, left, right, bottom properties, e.g.,
  - `top: +20px` => move the element 20px downward
  - `top: -20px` => move the element 20px upward
  - `bottom: +20px` => move the element 20px upward
  - `bottom: -20px` => move the element 20px downward
  - `left: +20px` => move the element 20px left
  - `left: -20px` => move the element 20px right

# position: relative;

```
<head>
<style>
p{
 border-style:solid; border-width:2px;
 border-color:□black; width:300px;
}
#r{
 position:relative;
 left:100px;
 top:150px;
 border-color:■red;
}
</style>
</head>
<body>
 <p>The paragraph 1</p>
 <p id="r">The paragraph 2</p>
 <p>The paragraph 3</p>
 <p>Theparagraph 4</p>
</body>
```

The paragraph 1

The paragraph 3

Theparagraph 4

The paragraph 2

# position: fixed;

- An element with fixed position is positioned relative to the browser window
  - Its position does not change by scrolling the page
    - Very useful for menus
- These elements are removed from the normal flow
  - The document and other elements behave like the fixed positioned element does not exist (no content, padding, margin)
- Fixed positioned elements can overlap other elements
- Position is specified by top, left, right, bottom properties

# position: fixed;

```
<head>
<style>
div.fixed {
 position: fixed;
 bottom: 300px;
 left: 0;
 width: 300px;
 border: 3px solid #73AD21;
}
</style>
</head>
<body>
<h2>position: fixed;</h2>
<p>An element with position: fixed; is positioned relative to the viewport,
 which means it always stays in the same place even if the page is scrolled:</p>
<div class="fixed">
This div element has position: fixed;
</div>
</body>
```

## position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:

This div element has position: fixed;

# position: absolute;

- An absolute position element is positioned relative to the first parent element that has a position other than static
  - If no such element is found, the containing block is <html>
- Absolute positioned elements are removed from the normal flow
  - The document and other elements behave like the positioned element does not exist (no space for content, padding, margin)
- The positioned elements can overlap other elements
- Position is specified by top, left, right, bottom properties

# position: absolute;

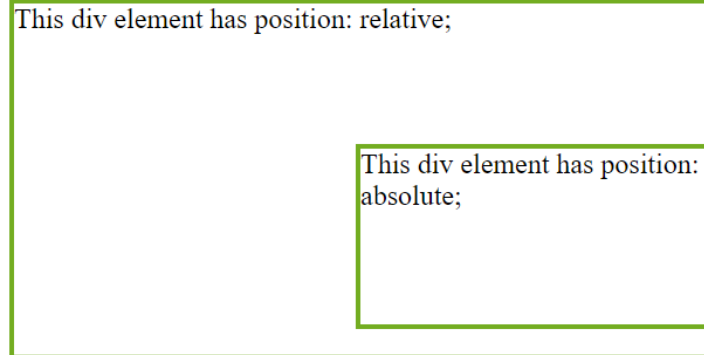
```
<head>
<style>
div.relative {
 position: relative;
 width: 400px;
 height: 200px;
 border: 3px solid #73AD21;
}
div.absolute {
 position: absolute;
 top: 80px;
 right: 0;
 width: 200px;
 height: 100px;
 border: 3px solid #73AD21;
}
</style>
</head>
<body>
<h2>position: absolute;</h2>
<p>An element with position: absolute; is positioned relative to

 the nearest positioned ancestor

 (instead of positioned relative to the viewport, like fixed):</p>
<div class="relative">This div element has position: relative;
 <div class="absolute">This div element has position: absolute;</div>
</div>
</body>
```

## position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):



# position: sticky;

- An element with `position: sticky;` is positioned based on the user's scroll position.
- A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like `position: fixed`).
- Internet Explorer does not support sticky positioning. Safari requires a `-webkit-` prefix. You must also specify at least one of top, right, bottom or left for sticky positioning to work.

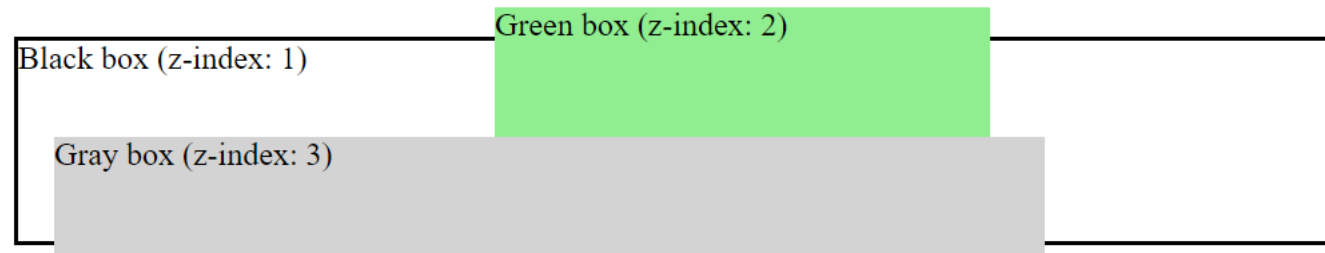


# Controlling Overlap Order

- Elements can overlap other elements
  - When positioned out of the normal flow
- The **z-index** specifies the stack order of an element
  - positive or negative stack order
  - Larger value => foreground

## Z-index Example

An element with greater stack order is always above an element with a lower stack order.



# Float Design

- The `float` property is used for positioning and formatting content e.g. let an image float left to the text in a container.
- The `float` property can have one of the following values:
  - `left` - The element floats to the left of its container
  - `right` - The element floats to the right of its container
  - `none` - The element does not float (will be displayed just where it occurs in the text). This is default
  - `inherit` - The element inherits the float value of its parent
- In its simplest use, the float property can be used to wrap text around images.

# Float Design

```
<head>
<style>
img {
 float: left;
}
p.c{
 width: 50%;
}
</style>
</head>
<body>
 <h2>Float Left</h2>
 <p class="c">
 Ubuntu is a Linux distribution based on Debian
 and composed mostly of free and open-source software.
 Ubuntu is officially released in multiple editions:
 Desktop, Server, and Core for Internet of things devices and robots.
 All of the editions can run on a computer alone, or in a virtual machine.
 </p>
</body>
```

## Float Left



Ubuntu is a Linux distribution based on Debian and composed mostly of free and open-source software. Ubuntu is officially released in multiple editions: Desktop, Server, and Core for Internet of things devices and robots. All of the editions can run on a computer alone, or in a virtual machine.

# Float Design: Float Next To Each Other

```
<head>
<style>
div {
 float: left;
 padding: 15px;}

.div1 {background: red;}

.div2 {background: yellow;}

.div3 {background: green;}
</style>
</head>
<body>
 <h2>Float Next To Each Other</h2>
 <p>In this example, the three divs will float next to each other.</p>
 <div class="div1">Div 1</div>
 <div class="div2">Div 2</div>
 <div class="div3">Div 3</div>
</body>
```

## Float Next To Each Other

In this example, the three divs will float next to each other.



# Clearing around Float Elements

- The `clear` property specifies what should happen with the element that is next to a floating element.
- The `clear` property can have one of the following values:
  - `none` - The element is not pushed below left or right floated elements. This is default
  - `left` - The element is pushed below left floated elements
  - `right` - The element is pushed below right floated elements
  - `both` - The element is pushed below both left and right floated elements
  - `inherit` - The element inherits the clear value from its parent

# Clear Example

## Without clear

div1 div2 - Notice that div2 is after div1 in the HTML code. However, since div1 floats to the left, the text in div2 flows around div1.

## With clear

div3

div4 - Here, clear: left; moves div4 down below the floating div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".

# Sample Application of Float & Clear

```
<head>
<style>
#head {border: 3px black solid; }
#left {border: 3px red solid; float: left; width: 50%; }
#right {border: 3px blue solid; float: right; width: 30%; }
#footer {border: 3px green solid; clear: both;}
</style>
</head>
<body>
 <div id = "head" style="text-align: center">
 <h2>Two Columns with Float</h2>
 <p> This is header </p>
 </div>
 <div id = "left">
 <h3>Left Column</h3>
 <p> This is a paragraph in the left column </p>
 <p> 1 ... 1 </p>
 </div>
 <div id = "right">
 <h3>RightColumn</h3>
 <p>Menu Item 1</p> <p>Menu Item 2</p>
 </div>
 <div id = "footer">
 <h3>Footer</h3>
 <p> This is footer. </p>
 </div>
</body>
```

Two Columns with Float		
This is header		
<b>Left Column</b> This is a paragraph in the left column 1 ... 1		<b>RightColumn</b> Menu Item 1 Menu Item 2
<b>Footer</b> This is footer.		

# CSS Layout: The display Property

- Displaying elements are mainly influenced by the “display” property.
  - inline: no break after or before, no width/height
  - block: break after and before, fill a line, width/height are modifiable
  - inline-block: inside is formatted as block-level box, the element formatted as an inline-level box
    - Are like inline but can have a width and a height
  - none: Hiding an element can be done by setting the display property to none.
- Some other values too
  - flex
  - grid



# Using inline-block to Create Navigation Links

- One common use for `display: inline-block` is to display list items horizontally instead of vertically.

```
<head>
<style>
.nav {
 background-color: yellow;
 list-style-type: none;
 text-align: center;
 margin: 0;
 padding: 0;
}

.nav li {
 display: inline-block;
 font-size: 20px;
 padding: 20px;
}
</style>
</head>
<body>
<h1>Horizontal Navigation Links</h1>
<p>By default, list items are displayed vertically.
 In this example we use display: inline-block
 to display them horizontally (side by side).</p>

<ul class="nav">
 Home
 About Us
 Our Clients
 Contact Us

</body>
```

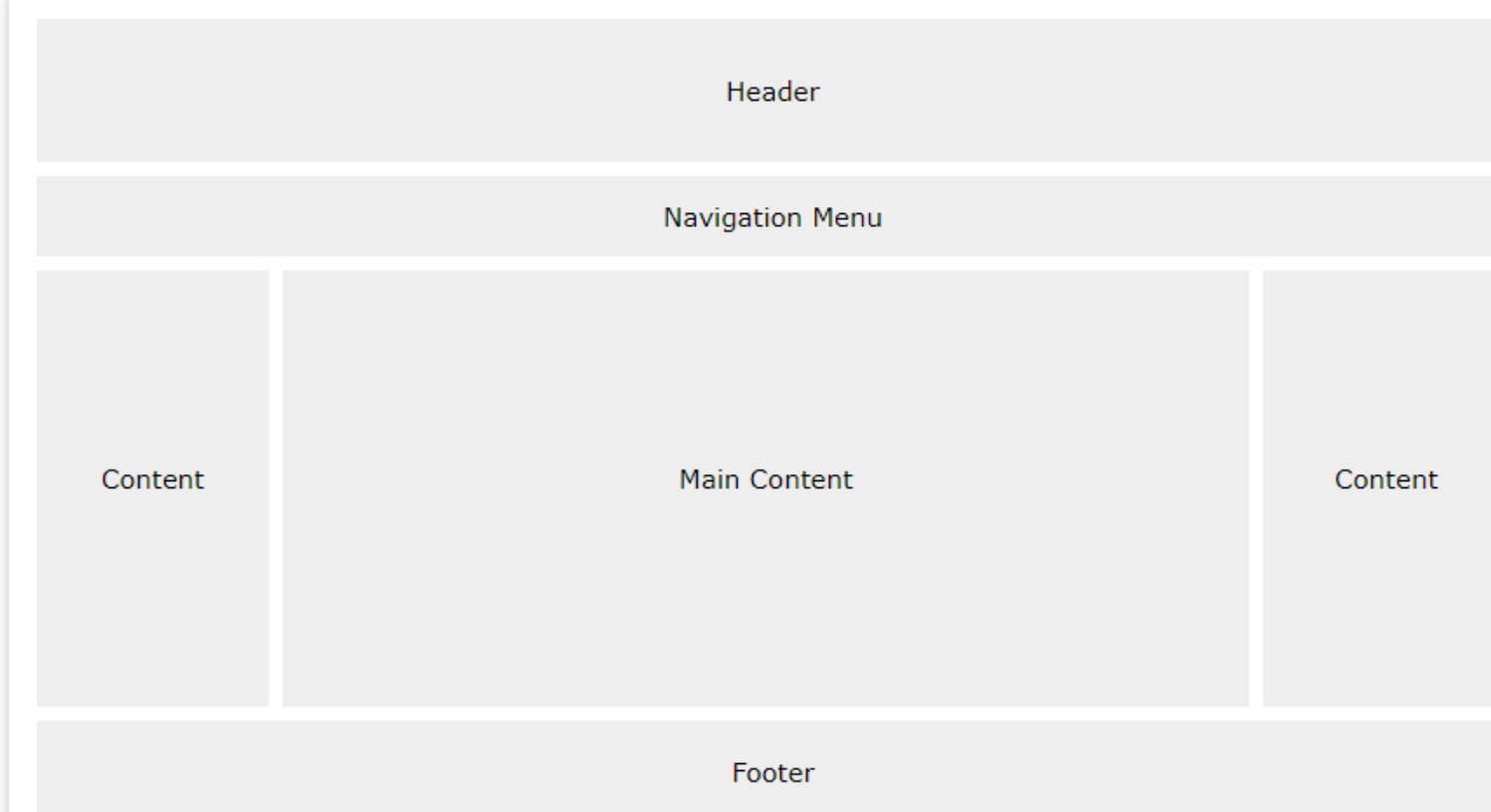
## Horizontal Navigation Links

By default, list items are displayed vertically. In this example we use display: inline-block to display them horizontally (side by side).

[Home](#) [About Us](#) [Our Clients](#) [Contact Us](#)

# Website Layout

- There are tons of different layout designs to choose from. However, the structure below, is one of the most common.

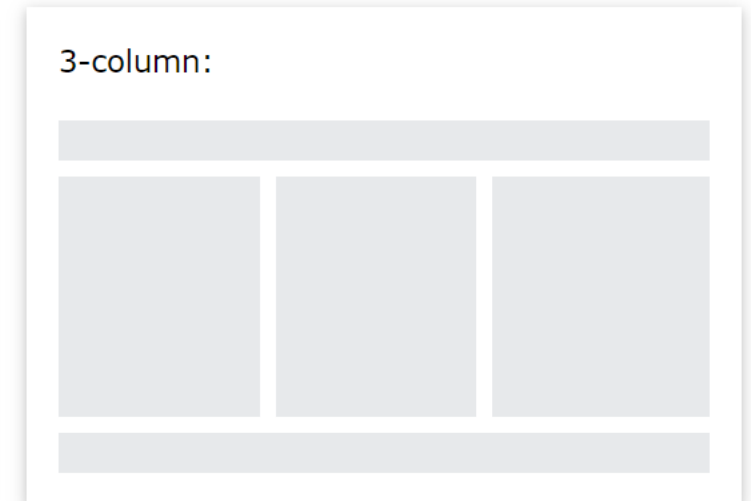
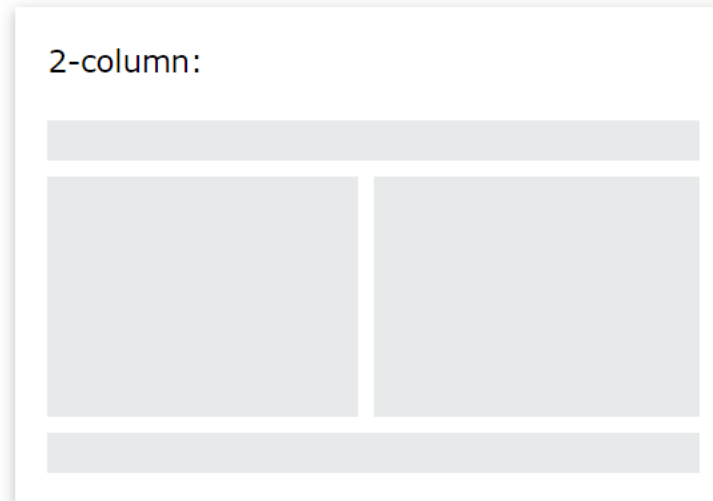
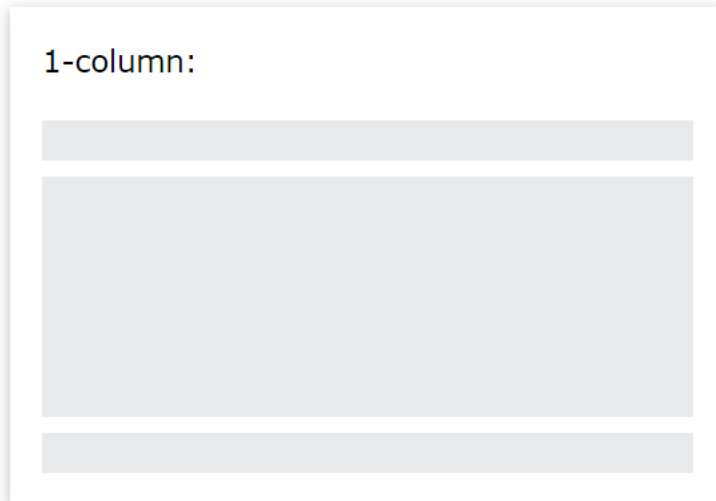


# Website Layout

- Header: A header is usually located at the top of the website (or right below a top navigation menu). It often contains a logo or the website name.
- Navigation Menu: A navigation bar contains a list of links to help visitors navigating through your website.
- Footer: The footer is placed at the bottom of your page. It often contains information like copyright and contact info.

# Website Layout

- Content: The layout in this section, often depends on the target users. The most common layout is one (or combining them) of the following:
  - 1-column (often used for mobile browsers)
  - 2-column (often used for tablets and laptops)
  - 3-column layout (only used for desktops)



# What is the Next: Responsive Design

- Reformat the presentation of the page based on
  - Browser window size, Browser type, ....
- Relative units: %, vw, vh, ...
  - max sizes specially for images
- Media Query [@media](#)
  - To include a block of CSS properties only if a certain condition is true
- Layout control by grid
- Bootstrap and other frameworks