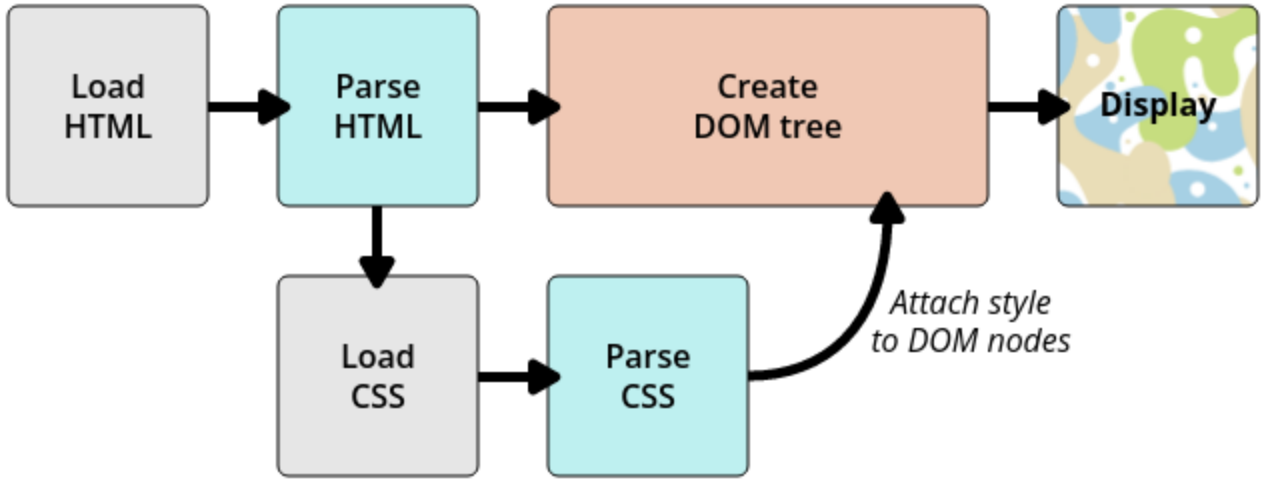
CSS

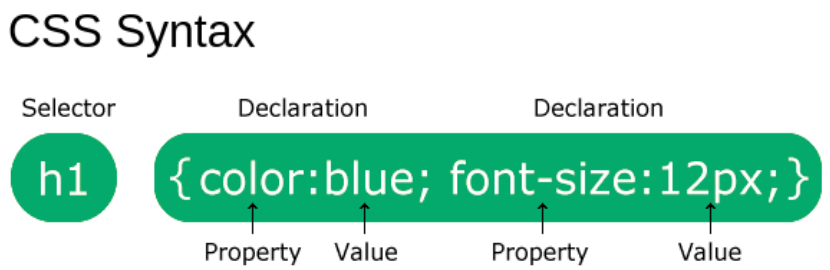
What is CSS

- CSS stands for Cascading Style Sheets. While HTML is used to define the structure and semantics of your content, CSS is used to style it and lay it out.



Syntax

- CSS is a rule-based language — you define rules specifying groups of styles that should be applied to particular elements or groups of elements on your web page. For example "I want the main heading on my page to be shown as large red text."



Adding CSS to our document

There are three ways of inserting a style sheet:

External CSS

```
<head>
  <link rel="stylesheet"  href="mystyle.css">
</head>
```

Inline CSS

```
<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>
```

Internal CSS

```
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
```

Selectors

Selectors are used to target the HTML elements on our web pages that we want to style.

It is a pattern of elements and other terms that tell the browser which HTML elements should be selected to have the CSS property values inside the rule applied to them.

Selector Lists

If more than one element uses the same CSS then the individual selectors can be combined into a selector list so that the rule is applied to all of the individual selectors.

Read more

Selector	Example
Type selector	`h1 { }`
Universal selector	`* { }`
Class selector	`.box { }`
id selector	`#unique { }`
Attribute selector	`a[title] { }`
Pseudo-class selectors	`p:first-child { }`
Pseudo-element selectors	`p::first-line { }`
Descendant combinator	`article p`
Child combinator	`article > p`
Adjacent sibling combinator	`h1 + p`
General sibling combinator	`h1 ~ p`

Colors

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values. Color the text, border and background.


```css
background-color:rgb(255, 99, 71);

color:#ff6347;

border-color:hsl(9, 100%, 64%);

background-color:hsla(9, 100%, 64%,0.8);
```

rgb(255, 99, 71)

#ff6347

hsl(9, 100%, 64%)

hsl(9, 100%, 64%, 0.8)

Convert from decimal to hex

- Divide the number by 16.
- Get the integer quotient for the next iteration.
- Get the remainder for the hex digit.
- Repeat the steps until the quotient is equal to 0.

Example #1

Convert 7562_{10} to hex:

Division by 16	Quotient (integer)	Remainder (decimal)	Remainder (hex)	Digit #
7562/16	472	10	A	0
472/16	29	8	8	1
29/16	1	13	D	2
1/16	0	1	1	3

RGB To HEX Conversion

Example #1

Convert red color (255,0,0) to hex color code:

$R = 255_{10} = FF_{16}$
$G = 0_{10} = 00_{16}$
$B = 0_{10} = 00_{16}$
So the hex color code is:
Hex = FF0000

Example #2

Convert gold color (255,215,0) to hex color code:

$R = 255_{10} = FF_{16}$
$G = 215_{10} = D7_{16}$
$B = 0_{10} = 00_{16}$
So the hex color code is:
Hex = FFD700

Backgrounds

- **background-color** defines the background color. A background-color extends underneath the content and padding box of the element.
- The **background-image** enables the display of an image in the background of an element.
- The **background-repeat** property is used to control the tiling behavior of images.
- **background-size** property can take length or percentage values, to size the image to fit inside the background. You can also use keywords: cover, contain
- The **background-position** property allows you to choose the position in which the background image appears on the box it is applied to. The default background-position value is (0,0). Can also use keywords or percentages.
- It is also possible to have **multiple background images** — you specify multiple background-image values in a single property value, separating each one with a comma.
- The **background** shorthand CSS property sets all background style properties at once, such as color, image, origin and size, or repeat method.

```css
body {
  background-color:grey;
  background-image: url("image.gif");
  background-repeat: no-repeat;
  background-position: center top;
  background-attachment: scroll;
}
```

Borders

- The CSS border properties allow you to specify the style, width, and color of an element's border.
- The **border-style** property specifies what kind of border to display.
- The **border-width** property specifies the width of the four borders. border-{individual}-width
- The **border-color** property is used to set the color of the four borders.
- Individual sides: border-{top|right|bottom|left}-{style|width|color}
- The **border** property is a shorthand property for the following individual border properties:
- border: {border-width} {border-style} {border-color}
- The **border-radius** property is used to add rounded borders to an element:


```css
box {
    border-style: dotted solid double dashed;
    border-color: blue;
    border-width: 2px;
    border-radius: 15px;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Exercises

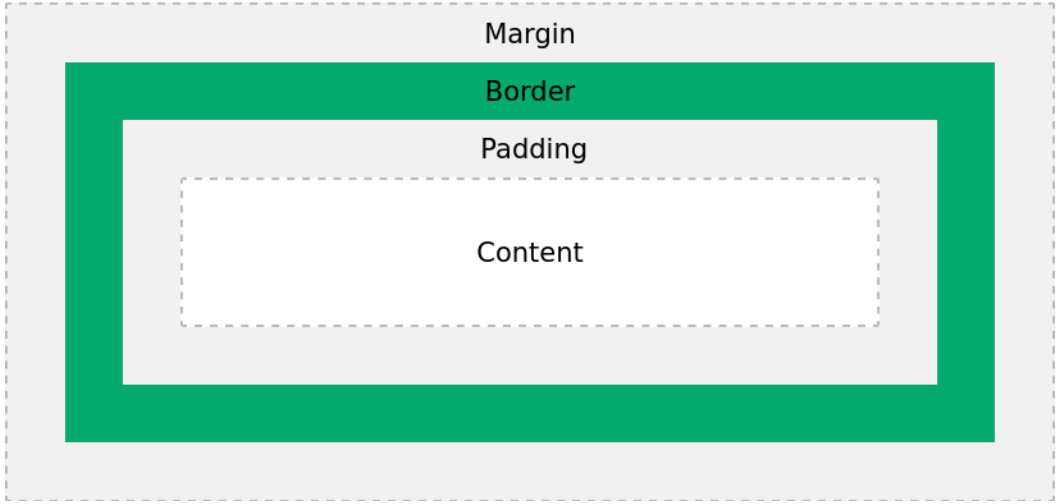
1)Style & Color the index.html page now, apply borders, backgrounds and text colors.

2) Create the following patterns.

Spacing Items

With Html we can have a few line breaks **
 vertically** and use ** ** to create space **horizontally** between two elements which is cumbersome; with CSS we have lot of flexibility in spacing the items using margins and paddings.

Box Model



- **Content box:** The area where your content is displayed, which can be sized using properties like width and height.
- **Padding box:** The padding sits around the content as white space; its size can be controlled using padding and related properties.
- **Border box:** The border box wraps the content and any padding. Its size and style can be controlled using border and related properties.
- **Margin box:** The margin is the outermost layer, wrapping the content, padding, and border as whitespace between this box and other elements. Its size can be controlled using margin and related properties.

Standard vs Border Box

In the standard box model, if you give a box a width and a height attribute, this defines the width and height of the *content box*.

Any padding and border is then added to that width and height to get the total size taken up by the box.

By default, browsers use the standard box model.

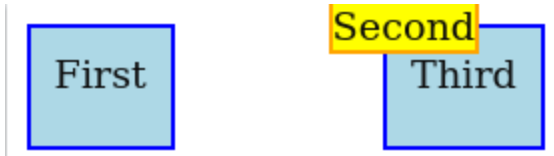
box-sizing: border-box

Any width is the width of the visible box on the page, therefore the content area width is that width minus the width for the padding and border.

Margins

- Margins can have positive or negative values. Setting a negative margin on one side of your box can cause it to overlap other things on the page.
- We can control all margins of an element at once using the margin property, or each side individually using the equivalent longhand properties
- **Whether you are using the standard or alternative box model, the margin is always added after the size of the visible box has been calculated.**

```
.second {
  margin-top: -10px;
  margin-right: -40px;
  margin-bottom: 40px;
  margin-left: 4em;
  /*Styling*/
  position: relative;
  border: 2px solid orange;
  background-color:yellow;
}
```



Margin Collapsing

- If you have two elements whose margins touch, and both margins are positive, those margins will combine to become one margin, and its size will be equal to the largest individual margin. If one margin is negative, its value will be subtracted from the total. Where both are negative, the margins will collapse and the largest value will be used.

Padding

- Unlike margins, you cannot have negative amounts of padding, so the value must be 0 or a positive value. Padding is typically used to push the content away from the border.
- padding property, or on each side individually using the equivalent longhand properties:

```
.box {
  padding-top: 0px;
  padding-right: 30px;
  padding-bottom: 40px;
  padding-left: 4em;
}
```

Inline vs Block elements

The display property specifies if/how an element is displayed. Every element has default value, that can be overridden.

Hide Element

visibility:hidden or display:none? visibility:hidden; the element will still take up the same space as before.

Block	Inline
Width and height respected	Width and Height not respected
Will break into a new line	Will not break into new line
Padding, margin borders are respected causing the other elements to move away from the content	Vertical padding, margins, and borders will apply but will not cause other inline boxes to move away from the box. Horizontal padding, margins, and borders will apply and will cause other inline boxes to move away from the box.
<h1> , ,<p>...	<a>, , ,<label>,...

Exercise

1) Lets add margins and padding for the paragraphs and sections to create a decent look.

2) Create a standard box and border box of width 400px and height 300px.

3) Add to the box:

A 5px, black, dotted border.

A top margin of 20px.

A right margin of 1em.

A bottom margin of 40px.

A left margin of 2em.

Padding on all sides of 1em.

Styling Texts,Lists & Links

Lets see how we can enhance the page look easily by adding lot of in-built styles.

Texts

font-family — allows you to specify a font (or list of fonts) for the browser to apply to the selected element.

Web safe fonts are generally available across all systems.

Generic fonts: serif, sans-serif, monospace, cursive, and fantasy.

font-size can take values measured in most of the available units. Px,em,rem.

font-stlye: Used to turn italic text on or off. Rarely used. **font-weight:** Sets how bold the text is.

text-transform:Allows you to set your font to be transformed.

text-decoration: Sets/unsets text decorations on fonts

word-spacing properties allow you to set the spacing between letters and words in your text.

text-align property is used to control how text is aligned. line-height property sets the height of each line of text. You can apply drop shadows to your text using the **text-shadow**

```
p{
 text-align: center;
 color: green;
 text-decoration: underline;
 text-transform: capitalize;
 letter-spacing: 3px;
 line-height: 1.8;
 word-spacing: -5px;
 text-shadow: 2px 2px 5px red;
 font-style: italic;
}
```

Paragraph With So Many Style Applied

Please Bear With Me!

Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.).

When setting the style for several link states, there are some order rules:

- **a:hover MUST come after a:link and a:visited**
- **a:active MUST come after a:hover**

Links can also be styled as buttons

```
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

/* mouse over link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
```

Lists

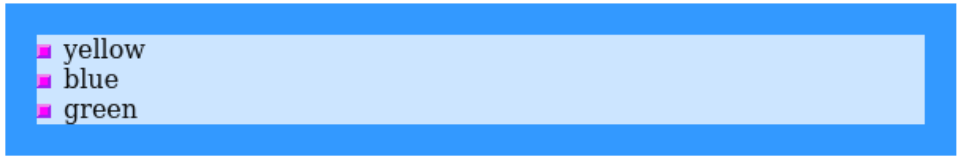
Set different list item markers/images/add background colors for lists

```
ol.a {
  list-style-type: circle;
  list-style-position: outside;
  background: #ff9999;
  padding: 20px;
}

ol.a li {
  background: #ffe5e5;
  padding: 5px;
  margin-left: 35px;
}
```


```
ul.b {
  list-style-image: url('sqpurple.gif');
  list-style-position: inside;
  background: #3399ff;
  padding: 20px;
}

ul.b li {
  background: #cce5ff;
}
```



Exercise

Style the text, links and lists in the page.

- Style the links based on states, style some links like buttons
- Style the texts eg: shadowing, weights, different fonts...
- Style the lists using different markers and other CSS properties.

Positioning Elements

Floats, Position, Images

Images

- Images and video are described as **replaced elements.** This means that CSS cannot affect the internal layout of these elements — only their position on the page amongst other elements.
- Images and video, are also described as having an **aspect ratio**. This means that it has a size in both the horizontal (x) and vertical (y) dimensions, and will be displayed using the intrinsic dimensions of the file by default.
- Replaced elements inside a grid or flex layout, have different default behaviors, essentially to avoid them being stretched strangely by the layout.

Sizing Images

If you place an image inside a box that is smaller or larger than the intrinsic dimensions of the image file in either direction, it will either appear smaller than the box, or overflow the box.

Setting the max-width to 100% of the image will enable the image to become smaller in size than the box but not larger.

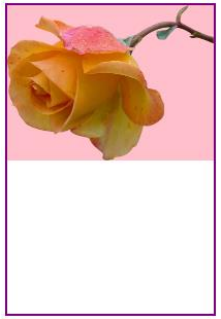
Without max-width

max-width:100%

object-fit: cover - sizes the image down, maintaining the aspect ratio so that it neatly fills the box. As the aspect ratio is maintained, some parts of the image will be cropped by the box.

object-fit:contain - the image will be scaled down until it is small enough to fit inside the box. This will result in "letterboxing" if it is not the same aspect ratio as the box.

Object-fit:contain



object-fit:cover

Exercises

- Resize the image to fit into a box without loosing any part of the image and maintaining its aspect ratio
- Create a horizontal image gallery with description at the bottom of each image.
- Create image sprite

floats

- The float property was introduced to allow web developers to implement simple layouts involving an image floating inside a column of text, with the text wrapping around the left or right of it. *The kind of thing you might get in a newspaper layout.*
- The element with the float set on it (the element in this case) is taken out of the normal layout flow of the document.

float:none;

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor.

float:left;

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor.

float:right;

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor.

Clearing floats

- If we want to stop the following element from moving up, we need to clear it; this is achieved with the clear property.

Lorem ipsum
dolor sit amet0

Second paragraph

Third paragraph

Lorem ipsum
dolor sit amet0

Second paragraph

Third paragraph

The clearfix hack

This involves first inserting some generated content after the box that contains both the float and content wrapped around it, then setting that generated content to clear both.

```
.wrapper::after {
  content: "";
  clear: both;
  display: block;
}
```

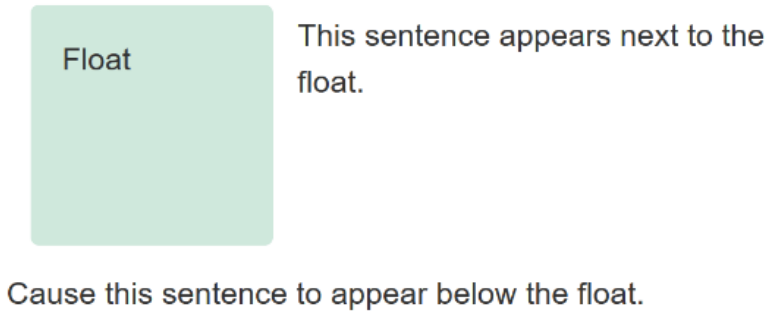
Using overflow : `overflow:auto;`

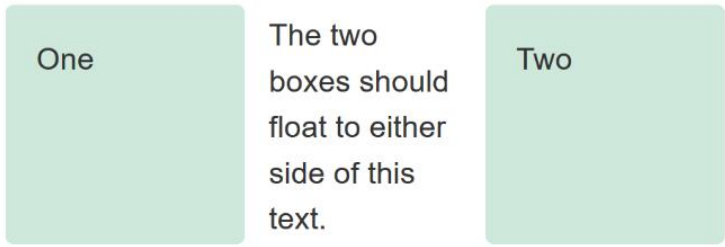
An alternative method is to set the overflow property of the wrapper to a value other than visible.

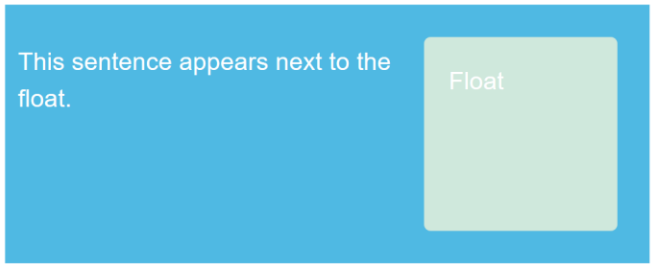
```
display: flow-root;
```

The modern way of solving this problem is to use the value flow-root of the display property.

Exercises


Float
This sentence appears next to the float.
Cause this sentence to appear below the float.


One
The two boxes should float to either side of this text.
Two

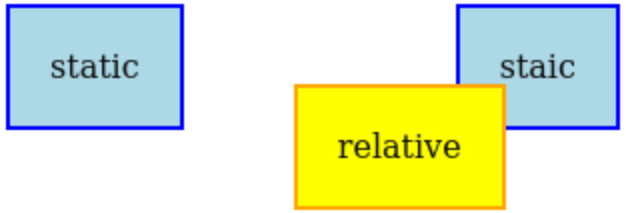

This sentence appears next to the float.
Float

Positioning

Positioning allows you to take elements out of normal document flow and make them behave differently, for example, by sitting on top of one another or by always remaining in the same place inside the browser viewport.

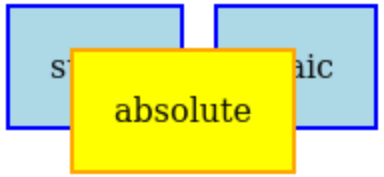
- **static**: every element has a static position by default, so the element will stick to the normal page flow. So if there is a left/right/top/bottom/z-index set then there will be no effect on that element.
- **relative**: an element's original position remains in the flow of the document, just like the static value. But now left/right/top/bottom/z-index will work. The positional properties "nudge" the element from the original position in that direction.

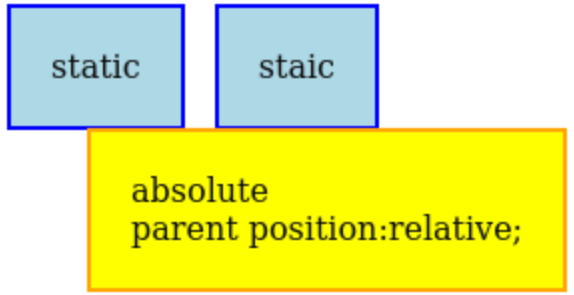
```
.second{
  position: relative;
  top: 40px;left:40px;
  border: 2px solid orange;
  background-color:yellow;
}
```



Absolute & Fixed

- **absolute**: the element is removed from the flow of the document and other elements will behave as if it's not even there whilst all the other positional properties will work on it.
- To make the child element positioned absolutely from its parent element we need to set this on the parent element itself:



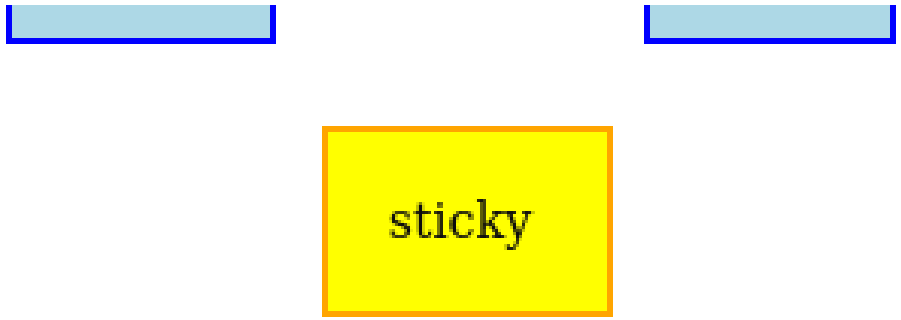


- **fixed**: the element is removed from the flow of the document like absolutely positioned elements. In fact they behave almost the same, only fixed positioned elements are always relative to the document, not any particular parent, and are unaffected by scrolling.

Sticky

- **sticky** (experimental): the element is treated like a relative value until the scroll location of the viewport reaches a specified threshold, at which point the element takes a fixed position where it is told to stick.

```
.second{
  position: sticky;
  top: 40px;left:40px;
  border: 2px solid orange;
  background-color:yellow;
}
```



Exercise

The Contents should be sidebar and should not move as the page scrolls.

Either right or left float the content around the images in each of the sections.

Cascade,Inheritance & Specificity

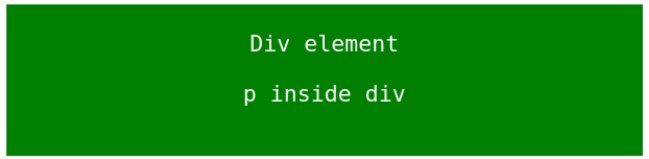
Inheritance

Inheritance also needs to be understood in this context — some CSS property values set on parent elements are inherited by their child elements, and some aren't.

Some styles, like font family, text-alignment etc., are automatically inherited by child elements from their parent element.

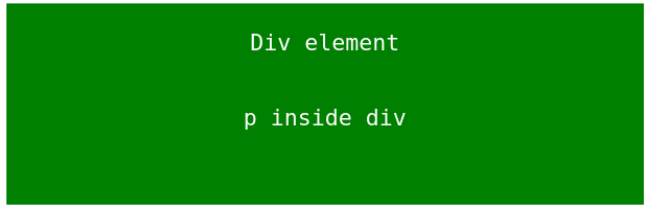
Others are not automatically inherited (margin, padding).

```
div{
    padding:20px;
    font-family:monospace;
    text-align:center;
    color:white;
    background-color:green;
    font-size:1.5em;
}
```



Add the **padding to p** and notice the difference!

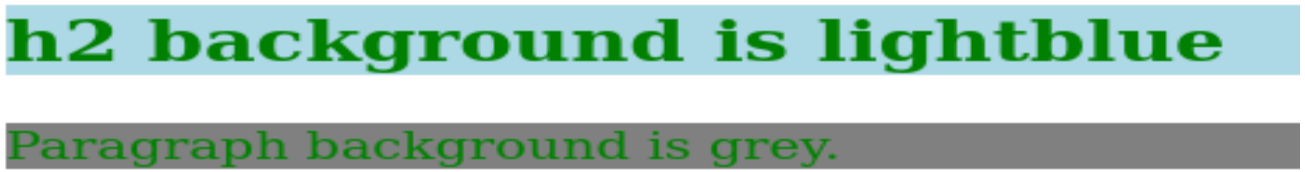
```
p{
    padding:20px;
}
```



Cascade

- **Stylesheets cascade** — at a very simple level, this means that the order of CSS rules matter; when two rules apply that have equal specificity the one that comes last in the CSS is the one that will be used.

```
/* select multiple elements separated by commas */
p, h1, h2 {
   color: green;
   background-color: grey;
}
/* when two styles set conflicting values for the same
property, the latter style takes precedence */
h2 {
   background-color: blue;
}
```



Specificity

- **Specificity** is how the browser decides which rule applies if multiple rules have different selectors, but could still apply to the same element. It is basically a measure of how specific a selector's selection will be:
- An **element selector is less specific** — it will select all elements of that type that appear on a page — so will get a lower score.
- A **class selector is more specific** — it will select only the elements on a page that have a specific class attribute value — so will get a higher score.

```
.special{
    background-color: grey;
}
p {
    color:white;
    background-color: lightblue;
}
```

p background is lightblue

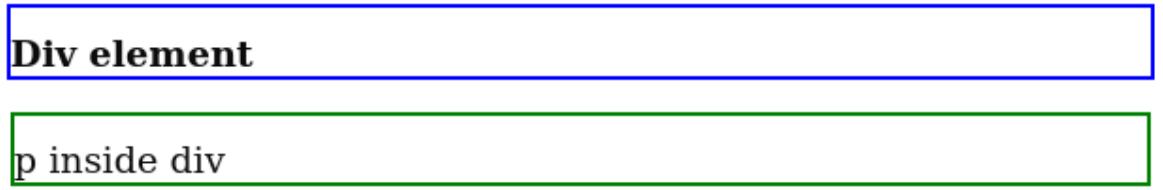
p with class special

!important

- This is a special piece of CSS that you can use to overrule all of the above calculations, however, you should **be very careful** with using it**.**

```
div{
    font-weight:bold !important;
    border:2px solid blue !important;
    height:20px;
}

p,.myDiv{
     padding-top:10px;
     font-weight:lighter;
     border:2px solid green;
}
```



- **important!** Took over cascade, specificity and inheritance!!

Values & Units

Every property used in CSS has a value type defining the set of values that are allowed for that property. CSS value types referred to as data types.

The term value refers to any particular expression supported by a value type.

Value Types are <color>,<length>....

Lengths

Absolute Length Units: **px**, cm,mm, Q..

Relative Length Units

em Font size of the parent, in the case of typographical properties like font-size, and font size of the element itself, in the case of other properties like width.

rem Font size of the root element.

vw 1% of the viewport's width.

vh 1% of the viewport's height.

vmin 1% of the viewport's smaller dimension.

vmax 1% of the viewport's larger dimension.

Units

Percentages

Always set relative to some other value. For example, if you set an element's font-size as a percentage, it will be a percentage of the font-size of the element's parent. If you use a percentage for a width value, it will be a percentage of the width of the parent.

Numbers

Some value types accept numbers, without any unit added to them. An example of a property which accepts a unitless number is the opacity property, which controls the opacity of an element (how transparent it is). This property accepts a number between 0 (fully transparent) and 1 (fully opaque).

Functions

The final type of value we will take a look at is the group of values known as functions. For example, below we are using calc() to make the box 20% + 100px wide. The 20% is calculated from the width of the parent container .wrapper and so will change if that width changes. We can't do this calculation beforehand because we don't know what 20% of the parent will be,

Readings

- CSS Tutorial (w3schools.com)
- Selectors, Backgrounds, borders, Margins, Padding, Links, Lists, Tables , Display, Position, Combinators, pseudo-class, Pseudo-element, opacity, Attr selectors
- Do the Exercises at the end of each these Topics

More Detailed Readings

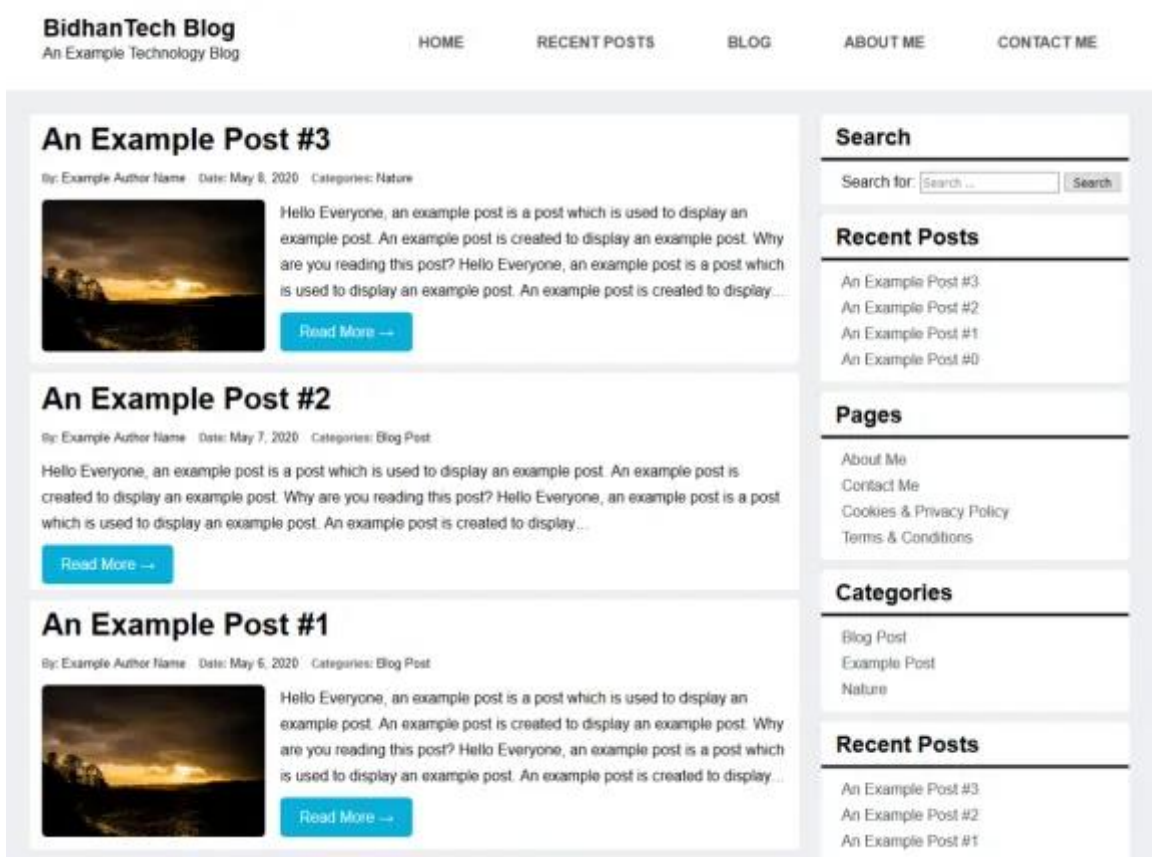
CSS selectors - Learn web development | MDN (mozilla.org)

Backgrounds and borders

The box model - Learn web development | MDN (mozilla.org)

Assignment

- Create blog page as shown below



Interview Questions

- https://www.interviewbit.com/css-interview-questions/#progressive-rendering
- https://css-tricks.com/interview-questions-css/