CS2204 Fundamentals of Internet Applications Development

Lecture 5 CSS – Part 3

Computer Science, City University of Hong Kong

Semester B 2024-25

Post-Lab Quiz Reminder

- **Post-Lab Quiz 2** is due this Friday, Feb 21st, 11:59 PM; no extension is allowed

- **Post-lab Quiz 3** will be available this Friday 7 PM, and due in next Friday, Feb 28th, 11:59 PM.

Reminder: CW1 is available now, due on Mar 7th

202502CS2204 › Assignments › CW1: A simple website for Hong Ko...

2024/25 Semester B

- Home
- Announcements
- Assignments
- Grades
- Pages
- Files
- Syllabus
- Quizzes
- Modules
- Collaborations
- Attendance
- Library Resources
- Class List (AIMS)
- LockDown Browser
- uReply
- Panopto Recordings

CW1: A simple website for Hong Kong Disneyland

Published | Assign To | Edit

You are required to build a simple website for Hong Kong Disneyland by going through the 3Ss: structure, style, and script. This CW1 is the first step focusing on structure and basic style only. The Website, therefore, would not be fully functional until the second part (CW2) using advanced CSS and JavaScript are completed.

The detailed instructions for this assignment can be found here: CS2204-2425B-CW1-Description.pdf with corresponding materials under file > Assignment > CW1 > materials.

To create HTML files, you are recommended to use professional code editors for better syntax checking/code management (e.g., VS Code).

Important notes:

- Image sources needed in this assignment can be found under files > Assignments > CW1
- For videos, please link them directly to https://personal.cs.cityu.edu.hk/~cs2204/video/. Please **DO NOT** include them in your submission.

Points	13
Submitting	a file upload
File Types	zip

Some Rubric

Criteria	Ratings		Pts
Structure Have three main pages, including index.html, design.html, and order.html, which are organized in proper folders (e.g., html, image). A shared theme.css is linked to all three pages.	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
Use of HTML tags Use appropriate HTML structural tags (e.g., <header>, <footer>, ,), and give as much structure as feasible	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
CSS basic styles Follow the styling instructions to apply at least one tag, class, and id selector(s); also apply at least one contextual or advanced selector. This information should be specified in the design page and the style should be as consistent as possible	**4 pts** **Full Marks**	**0 pts** **No Marks**	4 pts
Good web dev practices Accessibility (e.g., alt-text for images), code readability (e.g., indentation), and page validation	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
		Total Points: 13	

Question Time

- In adding external CSS stylesheet, how does `<link>` differ from `@import`? Which one is generally preferred and why?

- What is the primary difference between the following two CSS selectors?
  - `#text, a {…}`
  - `#text a {…}`

Agenda

CSS Layout

- Review: The box model
- Positioning
- Viewing pattern/typical layout

CSS3

- Effects
- Transform
- Transition
- Animation

Agenda

CSS Layout

- Review: The box model
- Positioning
- Viewing pattern/typical layout

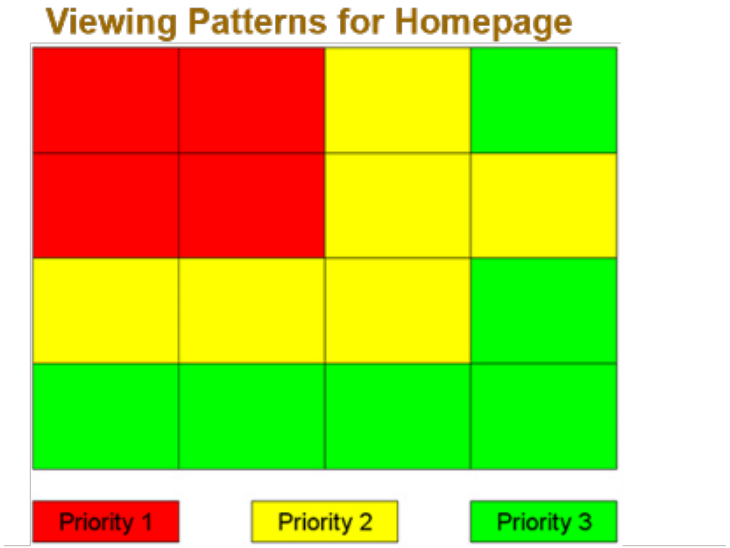
CSS3

- Effects
- Transform
- Transition
- Animation

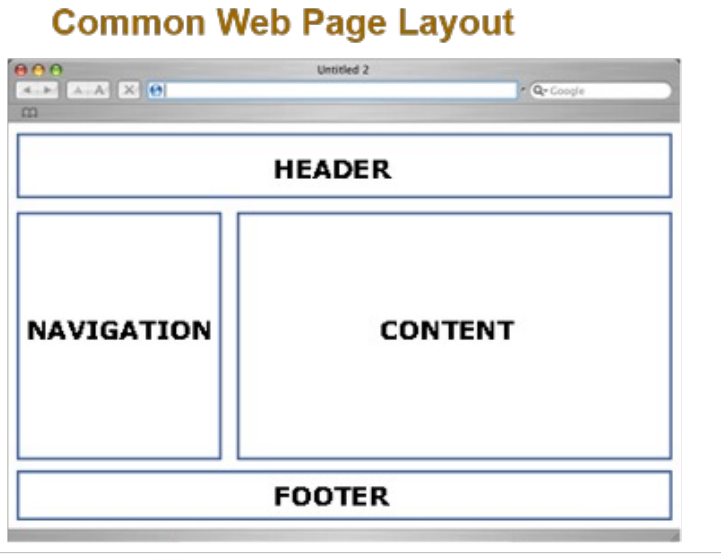
Layout

Layout refer to the arrangement/positioning of text and graphics.

Viewing pattern studies lead to some common, typical layouts



Ruel, L. and Outing, S. (2006). *Viewing Patterns for Homepages.*



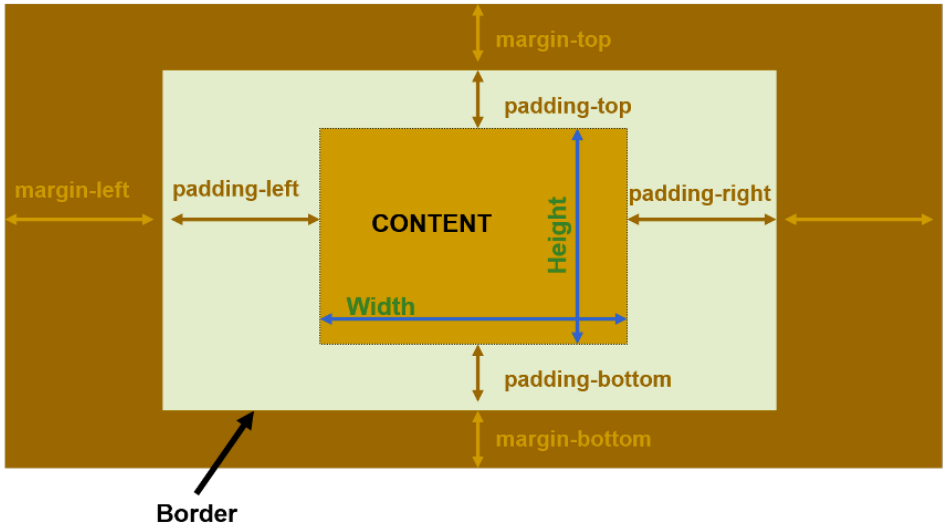
ADC. (2006). *Web Page Development: Best Practices.*

Review: Box Model

A box (or a block) can be any HTML tag depending on at which level we are looking

Set border: `color`, `width`, `border-style` (e.g., `solid`, `dotted`)

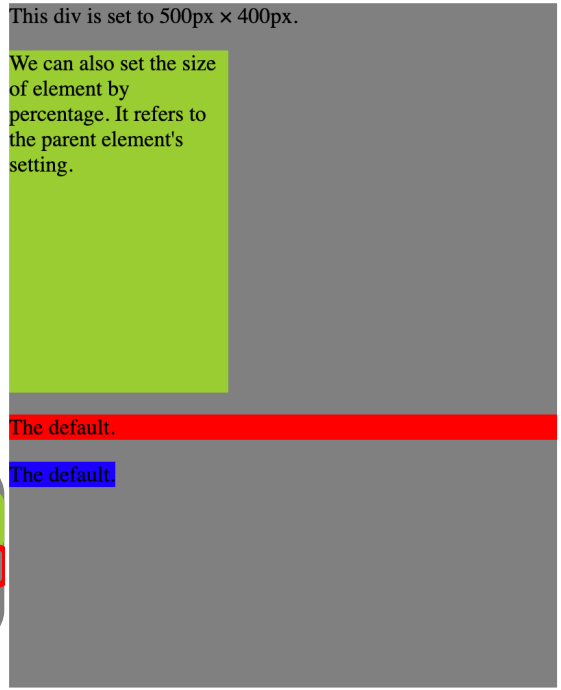
Differences in `width/height`, `padding`, and `margin`



Width & Height

Each element can be specified their size using the **width** and **height** properties

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CSS Width and Height</title>
    <style>
        .box {
            width: 200px;
            height: 200px;
            background-color: red;
        }
        .length {
            height: 500px;
            width: 400px;
            background-color: gray;
        }
        .percentage {
            height: 50%;
            width: 40%;
            background-color: yellowgreen;
        }
        .auto {
            background-color: red;
        }
        .inline {
            background-color: blue;
        }
    </style>
</head>
<body>

    <!-- <div class="box">
        this is a box
    </div> -->

    <div class="length">
        This div is set to 500px × 400px.
        <p class="percentage">
            We can also set the size of element by percentage. It refers to the parent element's
setting.
        </p>
        <p class="auto">
            The default.
        </p>
        <span class="inline">The default.</span>
        <!-- <span class="inline">This is another inline example.</span> -->
    </div>
</body>
</html>
```



Code Example: lec05-1-CSS-width-and-height.html

Border

Border can be useful in knowing the exact position of a block when we work with complicate layout

Different ways to specify border width

Border example

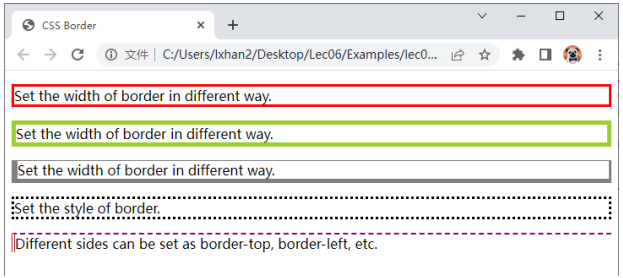
The width, color, and style of the border can be specified using border-width, border-color and border-style properties

```
12      .width-one {
13          border-style: solid;
14          border-color: red;
15          border-width: 3px;
16      }
17      .width-two {
18          border-style: solid;
19          border-color: yellowgreen;
20          border-width: thick;
21      }
22      .width-three {
23          border-style: solid;
24          border-color: gray;
25          border-width: 1px 3px 5px 7px;
26      }
27      .style {
28          border-style: dotted;
29          border-color: black;
30      }
31      .specify-side {
32          border-top: 2px dashed purple;
33          border-left: 4px double brown;
34      }
35    </style>
36  </head>
37
38  <body>
39      <p class="width-one">
40          Set the width of border in different way.</p>
41      <p class="width-two">
42          Set the width of border in different way.</p>
43      <p class="width-three">
44          Set the width of border in different way.</p>
45      <p class="style">
46          Set the style of border.
47      </p>
48      <p class="specify-side">
49          Different sides can be set as border-top, border-left, etc.
50      </p>
51  </body>
52
53  </html>
```

In order of top, right, bottom & left

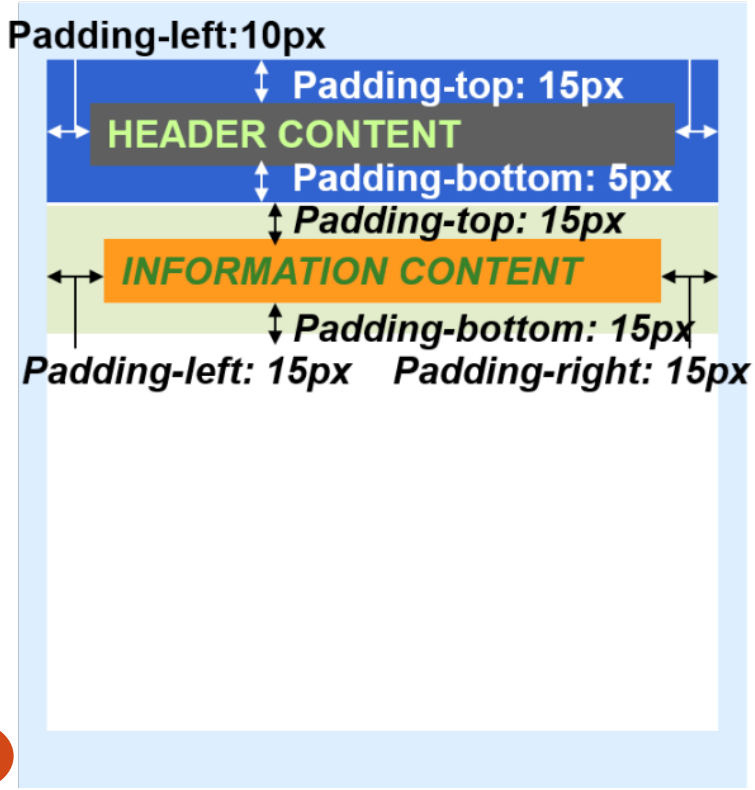
- border-width : | thin | medium | thick | inherit
- border-color : | transparent | inherit
- border-style : dotted | dashed | solid | double | groove | ridge | inset | outset | none | inherit

border-top, border-right, border-bottom, and border-left



Code Example: lec05-02-CSS-border.html

Padding

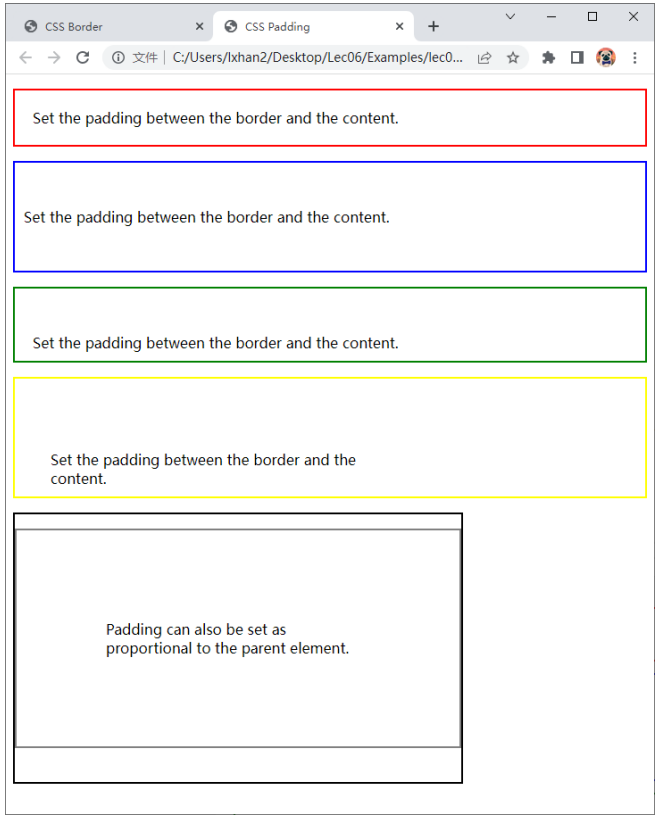


The padding property is used to add **space between the border** and the **content**

padding : <value for all sides>
padding : <top/bottom> <left/right>
padding : <top> <left/right> <bottom>
padding : <top> <right> <bottom> <left>

Padding example

```
9     <style>
10        .padding-one {
11            border: 2px solid red;
12            padding: 20px;
13        }
14        .padding-two {
15            border: 2px solid blue;
16            padding: 50px 10px;
17        }
18        .padding-three {
19            border: 2px solid green;
20            padding: 50px 20px 10px;
21        }
22        .padding-four {
23            border: 2px solid yellow;
24            padding: 80px 300px 10px 40px;
25        }
26        .parent {
27            width: 500px;
28            height: 300px;
29            border: 2px solid black;
30        }
31        .padding-percentage {
32            border: 2px solid gray;
33            padding: 20%
34
35    </style>
36</head>
37
38<body>
39    <p class="padding-one">
40        Set the padding between the border and the content.
41    </p>
42    <p class="padding-two">
43        Set the padding between the border and the content.
44    </p>
45    <p class="padding-three">
46        Set the padding between the border and the content.
47    </p>
48    <p class="padding-four">
49        Set the padding between the border and the content.
50    </p>
51    <div class="parent">
52        <p class="padding-percentage">
53            Padding can also be set as proportional to the parent
              element.
54        </p>
55    </div>
56</body>
57
58</html>
```



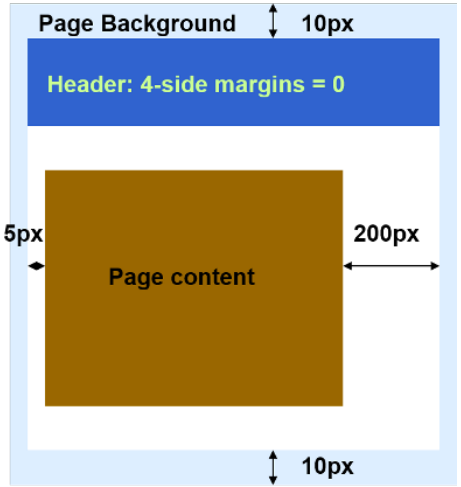
Code Example: lec05-03-CSS-padding.html

Margins

Value can be set in different ways:

- Length - in an **absolute** unit, such as px
- Percentage - with reference to the parent element's **width**
- Auto - leave to the browser's calculation, usually used for centering, e.g. `{margin: auto;}`
  - The element will then take up the specified width, and the remaining space will be split equally between the left and right margins

margin: <value for all sides>
margin: <top/bottom> <left/right>
margin: <top> <left/right> <bottom>
margin: <top> <right> <bottom> <left>

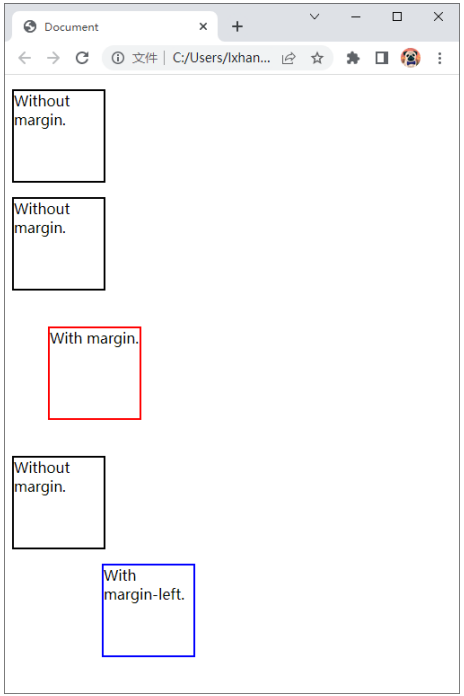


Margins Example

Use margin property to set the space **between** that element and other elements in the same container

The property can be a single "`margin`" or margins of 4 sides: "`margin-left`", "`margin-top`", etc.

```
8     <style>
9         p {
10            width: 100px;
11            height: 100px;
12            border: 2px solid;
13        }
14        .with-margin {
15            margin: 40px;
16            border-color: red;
17        }
18        .with-margin-left {
19            margin-left: 100px;
20            border-color: blue;
21        }
22    </style>
23</head>
24<body>
25    <p>
26        Without margin.
27    </p>
28    <p>
29        Without margin.
30    </p>
31    <p class="with-margin">
32        With margin.
33    </p>
34    <p>
35        Without margin.
36    </p>
37    <p class="with-margin-left">
38        With margin-left.
39    </p>
40</body>
41</html>
```

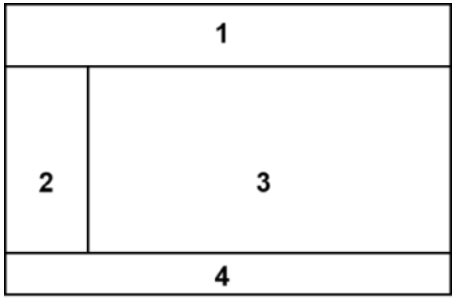


Code Example: lec05-04-CSS-margin.html

Structuring Your Page For Layout

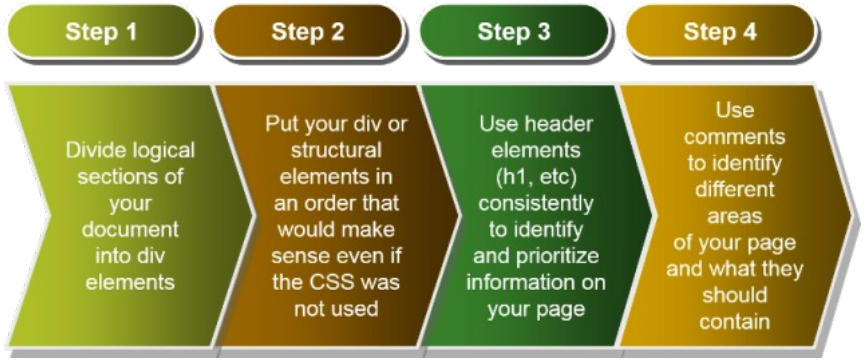
Consider a typical layout with 4 sections:

- Header
- Content
- Navigation
- Footer



In HTML5, new structural tags, such as `<header>`, `<section>`, `<nav>`, `<footer>`, can be used, similar to create different ids for each `<div>`

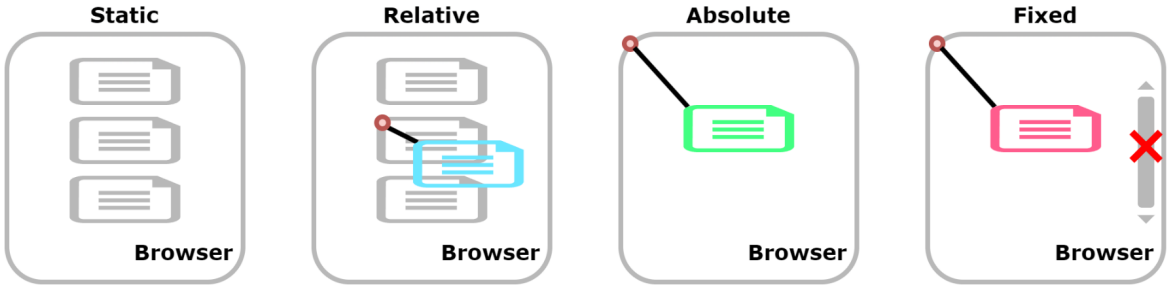
```
<div id="container">
 <div id="header>…</div>
 <div id="content">…</div>
 <div id="navigation">…</div>
 <div id="footer">…</div>
</div>
```



Positioning

Positioning is how the browser puts elements into containers. This determines the layout of a website

- Types of positioning: static | relative | absolute | fixed

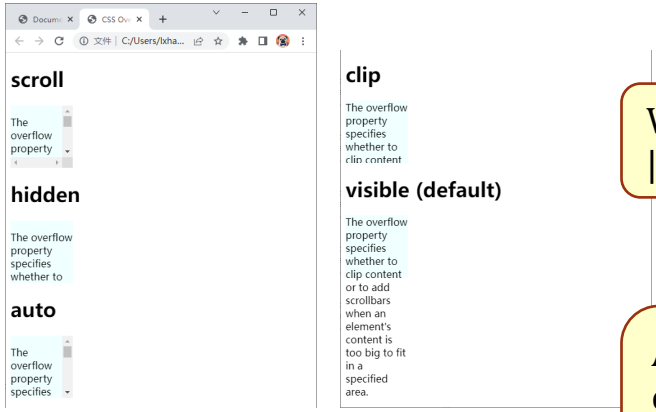


- All are positioned based on a base point, except static

The **Overflow** Property

The overflow property can be used to specify the behavior of the browser when the size of the element is larger than the container

```
    <style>
        div {
            height: 100px;
            width: 100px;
            background-color: azure;
        }
        .scroll {
            overflow: scroll;
        }
        .hidden {
            overflow: hidden;
        }
        .auto {
            overflow: auto;
        }
        .clip {
            overflow: clip;
        }
        .visible {
            overflow: visible;
        }
    </style>
</head>
<body>
    <h1>scroll</h1>
    <div class="scroll">
        <p>The overflow property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.</p>
    </div>
    <h1>hidden</h1>
    <div class="hidden">
        <p>The overflow property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.</p>
    </div>
    <h1>auto</h1>
    <div class="auto">
        <p>The overflow property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.</p>
    </div>
    <h1>clip</h1>
    <div class="clip">
        <p>The overflow property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.</p>
    </div>
    <h1>visible (default)</h1>
    <div class="visible">
        <p>The overflow property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.</p>
    </div>
</body>
</html>
```



Values: scroll | hidden| auto | clip | visible

Auto – allows the browser to decide whether scroll bars need to be displayed and the content is shown inside the container

Code Example: lec05-05-CSS-overflow.html

Types of Elements Regarding Positioning

Block-level element

- always starts on **a new line** and always **takes up the full width available**, e.g., `<div>`, `<p>`, `<ul>`, `<h1~6>`, etc.
- its width, height, padding and margin can be adjusted
- it usually can contain other elements
  - But `<p>` cannot contain other block-level elements, such as `<div>`

Inline-level element

- does **not** start on a new line and only takes up as much width as necessary, e.g., `<a>`, `<span>`, etc.
- **multiple** inline-level elements **can be in one line**
- set width and height is invalid

Inline-block element

- multiple can be in one line, but their width, height, padding and margin can be adjusted, e.g., `<img>`, `<input>`, `<td>`, etc.

Positioning in Webpage Layout

Static:

- this is the **default** scheme, also known as normal flow
- just lay out the content normally according to **sequence** and display property (*block* or *inline*)
- no effect for top/bottom/left/right property values
- width and height of container, the float, clear or overflow property can affect the flow
- in normal flow, there is **no overlap**

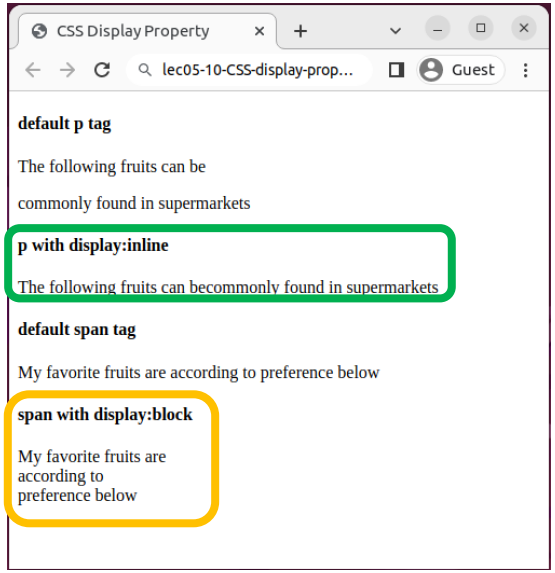
For other three schemes including fixed, absolute and relative, **overlap** may occur

Static Positioning: Display Property

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>CSS Display Property</title>
    <style>
        #exampleBlock span {
            display: block;
        }

        #exampleInline p{
            display: inline;
        }

    </style>
  </head>
  <body>
  <!-- Page content begins here -->
    <h4>default p tag</h4>
    <p>The following fruits can be</p><p>commonly found in supermarkets</p>

    <h4>p with display:inline</h4>
    <div id="exampleInline">
        <p>The following fruits can be</
    </div>

    <h4>default span tag</h4>
    <span>My favorite fruits are</span>

    <h4>span with display:block</h4>
    <div id="exampleBlock">
        <span>My favorite fruits are</sp
    </div>
  <!-- Page content ends here -->
    </body>
</html>
```

Display: block

- generates a block box
- goes to a new line
- can set dimension, i.e., width, height, etc.
- can set horizontal and vertical margins, vertical margins will be collapsed - max(margin-bottom, margin-top), i.e., take the larger one



Code Example: lec05-07-display-property.html

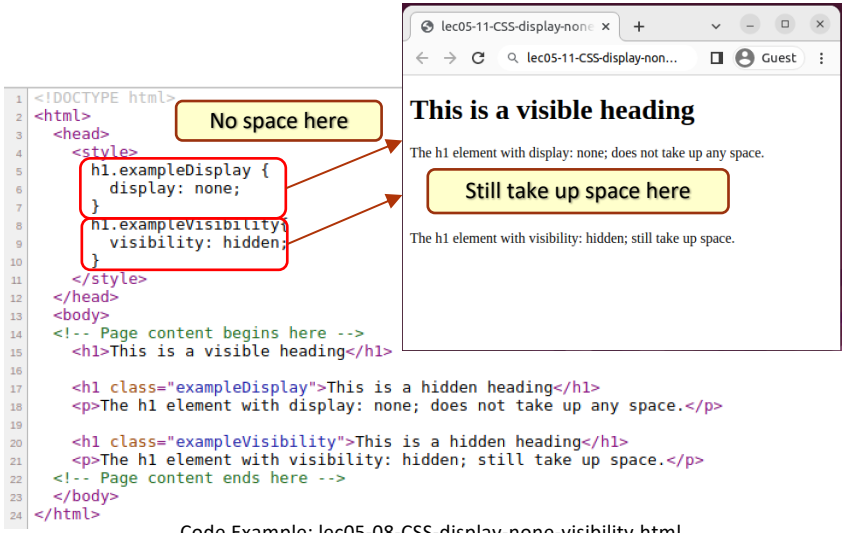
Display: inline

- generates an inline box
- layout on the same line
- cannot set dimensions
- can only set horizontal margin
- in general, don't put block element inside inline element

Static Positioning: Display Property

Display: none

- make the element disappear (but the element still exists in the DOM)
- does **not** take up any space (no box generated)
- compare with the property *visibility: hidden*




```
<!DOCTYPE html>
<html>
  <head>
    <style>
      h1.exampleDisplay {
        display: none;
      }
      h1.exampleVisibility{
        visibility: hidden;
      }
    </style>
  </head>
  <body>
  <!-- Page content begins here -->
    <h1>This is a visible heading</h1>

    <h1 class="exampleDisplay">This is a hidden heading</h1>
    <p>The h1 element with display: none; does not take up any space.</p>

    <h1 class="exampleVisibility">This is a hidden heading</h1>
    <p>The h1 element with visibility: hidden; still take up space.</p>
  <!-- Page content ends here -->
  </body>
</html>
```

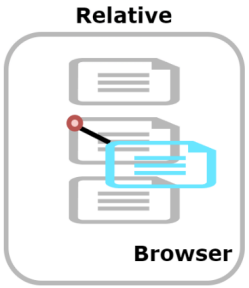
Code Example: lec05-08-CSS-display-none-visibility.html

Display: inline-block

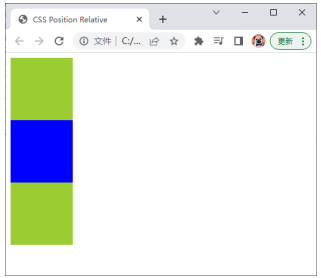
- positions like inline (do not start a new line) but behaves like block, therefore **width** and **height** can be set
- useful to style label and input elements in forms

Relative Position

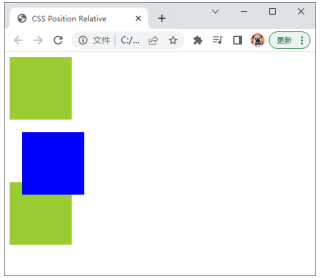
- Position **first** follows the normal flow and is then **adjusted by top/bottom/left/right values**
- Can overlap with other elements, controlled by the z-index




```
    <style>
        div {
            height: 100px;
            width: 100px;
            background-color: yellowgreen;
        }
        .relative {
            background-color: blue;
        }
    </style>
</head>
<body>
    <div></div>
    <div class="relative">

    </div>
    <div></div>
</body>
</html>
```




```
    <style>
        div {
            height: 100px;
            width: 100px;
            background-color: yellowgreen;
        }
        .relative {
            background-color: blue;
            position: relative;
            top: 20px;
            left: 20px;
        }
    </style>
</head>
<body>
    <div></div>
    <div class="relative">

    </div>
    <div></div>
</body>
</html>
```

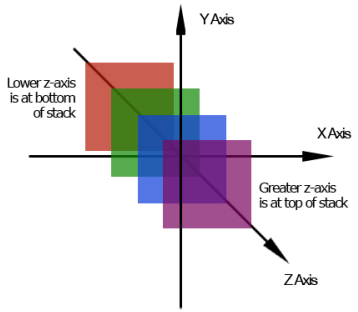


Code Example: lec05-09-CSS-relative.html

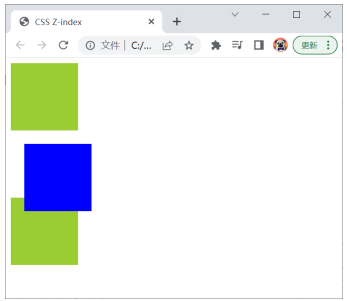
Z-Index

Z-index property specifies the stack order of an element

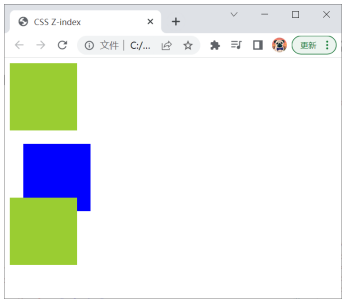
- Only works for relative, fixed and absolute positions




```
8     <style>
9         div {
10            height: 100px;
11            width: 100px;
12            background-color: yellowgreen;
13        }
14        .relative {
15            background-color: blue;
16            position: relative;
17            top: 20px;
18            left: 20px;
19            z-index: 1;
20        }
21    </style>
22 </head>
23 <body>
24    <div></div>
25    <div class="relative">
26
27    </div>
28    <div></div>
29 </body>
30 </html>
```



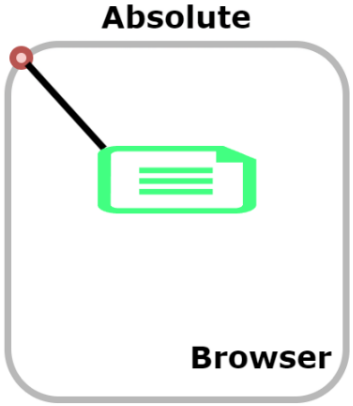

```
8     <style>
9         div {
10            height: 100px;
11            width: 100px;
12            background-color: yellowgreen;
13        }
14        .relative {
15            background-color: blue;
16            position: relative;
17            top: 20px;
18            left: 20px;
19            z-index: -1;
20        }
21    </style>
22 </head>
23 <body>
24    <div></div>
25    <div class="relative">
26
27    </div>
28    <div></div>
29 </body>
30 </html>
```



Code Example: lec05-10-CSS-Z-index.html

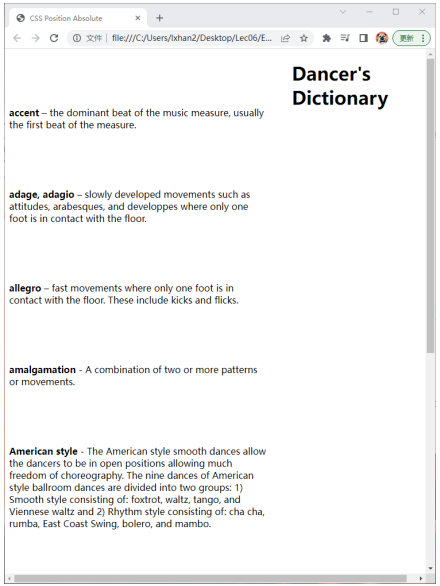
Absolute Position

- The box is taken out of the normal flow
- The **offset property** of the box is relation to **nearest ancestor element**
- If the positioned element has no positioned ancestors, it uses the document body and moves along with page scrolling



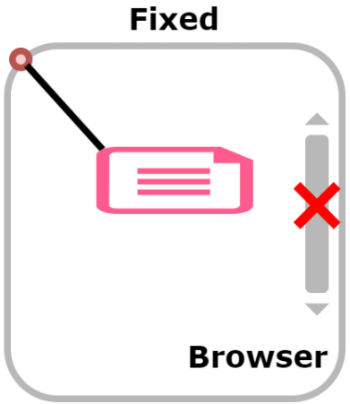
Code Example: lec05-11-CSS-absolute.html

```
8     <style>
9         h1 {
10            position: absolute;
11            top: 0px;
12            left: 500px;
13            width: 250px;
14        }
15        p {
16            margin-top: 100px;
17            width: 450px;
18        }
19    </style>
20</head>
21<body>
22    <h1>
23        Dancer's Dictionary
24    </h1>
25    <p>
26        <b>accent</b> – the dominant beat of the
27    </p>
28    <p>
29        <b>adage, adagio</b> – slowly developed m
30    </p>
31    <p>
32        <b>allegro</b> – fast movements where onl
33    </p>
```

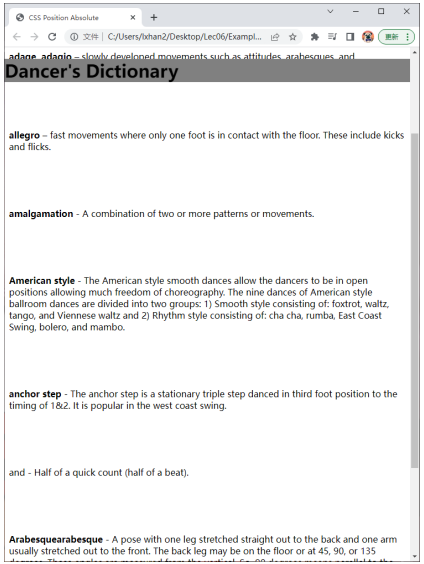


Fixed Position

The **offset property** of the box is relation to the viewpoint – always stays in the same place even during page scroll




```
8     <style>
9         h1 {
10            position: fixed;
11            top: 0px;
12            left: 0px;
13            width: 100%;
14            background-color: gray;
15        }
16        p {
17            margin-top: 100px;
18        }
19    </style>
20</head>
21<body>
22    <h1>
23        Dancer's Dictionary
24    </h1>
25    <p>
26        <b>accent</b> – the dominant beat of the
27    </p>
28    <p>
29        <b>adage, adagio</b> – slowly developed
30    </p>
31    <p>
32        <b>allegro</b> – fast movements where on
33    </p>
```



Code Example: lec05-13-CSS-fixed.html

Summary

Absolute:

- controlled by top/bottom/left/right values relative to the nearest ancestor element with positioning, if not existing, go up, until up to browser window
- absolute elements are removed from the normal flow
- used to control position within a container, which is not static in positioning
- can overlap with other elements, then controlled by z-index

Fixed:

- controlled by top/bottom/left/right values relative to the initial containing block (browser window)
- fixed elements are removed from normal flow (as if they are not present in laying out other elements)
- can overlap with other elements, controlled by the z-index

Float Property

The basic technique to set multi-column designs in normal flow:

```
{float: value;}
```

- where the value can be `left`, `right` or `none`.
- The above are values of the **float property**, not to be confused with the ***properties*** *left*, *right*, *top* and *bottom* used for ***positioning***

Definition in HTML standard : "*A float is a box that is shifted to the left or right on the current line*"

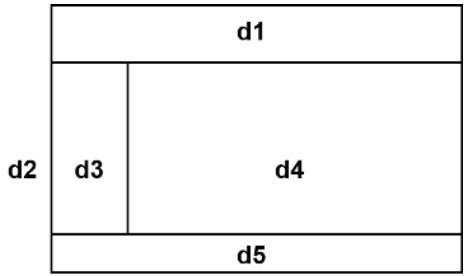
- The box is taken out of the normal flow

Float – 2 columns design

```
<div id="d1"></div>
<div id="d2">
    <div id="d3"></div>
    <div id="d4"></div>
</div>
<div id="d5"></div>
```

Float **left** for menu

- `#d3 {float: left;}`
- may need to set margin-left of #d4



Float **right** for menu

- `#d3 {float: right;}`
- d3 float right relative to the following element (i.e., d4)

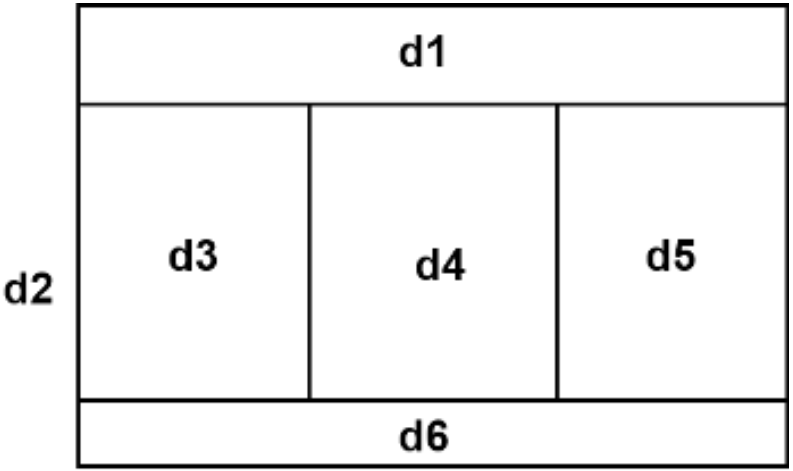
Code Example: lec05-14-CSS-float.html

Float – 3 columns design

Float left:

- #d3, #d4, #d5 {float: left;}

```
<div id="d1"></div>
<div id="d2">
    <div id="d3"></div>
    <div id="d4"></div>
    <div id="d5"></div>
</div>
<div id="d6">
</div>
```



Code Example: lec06-15-CSS-float2.html

Clear

To stop floating sequence at some point, `{float: none:}` can be used. It is also possible to do this by the clear property:

```
{float: left|right|both}
```

Depending on the width of the container, the text may or may not be always starting at a new row. To make sure it starts always on a new row, can stop the float of text by

```
{clear: both;}
```

- i.e., **no** element should appear on its left and right
- Another common use of clear is to make sure an element starts on a new line, e.g., a footer to appear always in new row regardless how other blocks are floating (second diagram below)

The clear Property

This is some text. This is some text. This is some text. This is some text. This is some text. This is some text.

Remove the "clear" class to see the effect.

```
<style>
img {
  float: left;
}

p.clear {
  clear: left;
}
</style>
```

```
<h1>The clear Property</h1>

<img src="pineapple.png" width="100" height="132">

<p class="clear">This is some text. This is some text.
This is some text. This is some text. This is some
text. This is some text.</p>
<p><strong>Remove the "clear" class to see the effect.
</strong></p>
```

What is kind of CSS selector is `p.clear{}`? Why cannot it be `.clear p{}` ?

Code Example: lec05-16-CSS-clear.html

Column Related Property

Use of **float property** to design multi-column layout is fairly complicated although flexible. There are other choices available in the fast-developing CSS3 techniques (e.g., flex)

One of these features is the column-count, a property to be set in the container and its content will be split into the assigned columns

```
<div id="container">
    <p>content....
        .....
    </p>
</div>
#container {
  column-count: 3;
}
```

There are column related properties to further manage the columns

Fixed Layout

Not to be confused with **Fixed Positioning.**

For normal flow (static positioning), layout will usually change if window (container) size changes, e.g., elements are forced into new line when the width is reduced. To prevent this layout change, fixed layout can be used

Use a big container (body or div) to hold the page and set its width (height) to a **fixed**, **absolute** value (e.g., 1024px)

Advantages:

- the pages look identical, no matter what window size (device) is used
- fixed width/height elements will not overpower text on smaller monitors

Limitations:

- cause horizontal/vertical scrolling in smaller browser windows
- result in large area of white space in larger screen
- cannot fit in small screen device

Liquid Layout

Also known as **fluid layout**. The size of elements will enlarge or shrink when the window (container) size changes.

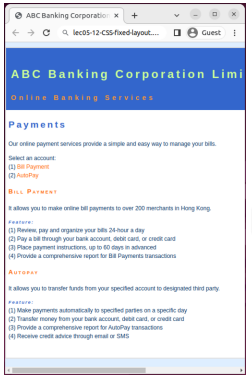
At all dimensions in relative units, e.g., **percentages**. thus, dimensions enlarge or shrink according to the container dimension

- **Advantages**:
  - pages expand and contract to fill the available space; all available screen space can be used
  - consistency in relative width/height e.g., handle requirement like larger font sizes compared to other content
  - Important concept in Responsive Design
- **Limitations**:
  - little precise control over the width/height of the various elements of the page
  - Issues with wide screen – making content arranged too wide to be read comfortably
  - when a fixed width element is placed inside a liquid column, e.g., the column is rendered without enough space for an image, browsers will either increase the column width, or cause overlap in text and image

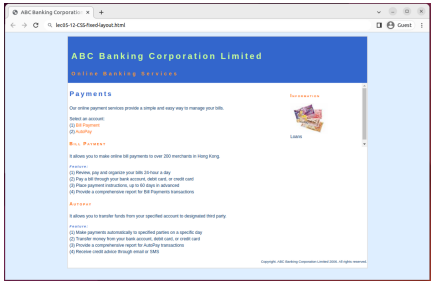
Fixed vs Liquid Layout

Fixed Layout

```
#mainContainer {
        color: inherit;
        background-color: white;
        margin: 10px auto;
        border: 2px solid #E0E0E0;
        width: 840px;}

#header {
        color: inherit;
        background-color: #3366CC;
        margin: 0px;
        padding: 15px 10px 5px;}

#information {
        padding: 15px 15px;
        width: 200px;
        height: 150px;
        overflow: auto;
        float: right; }
```



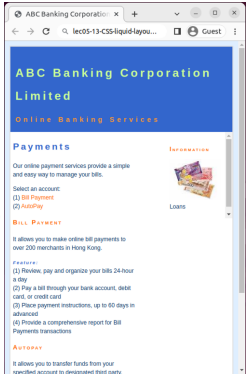
Narrow Window



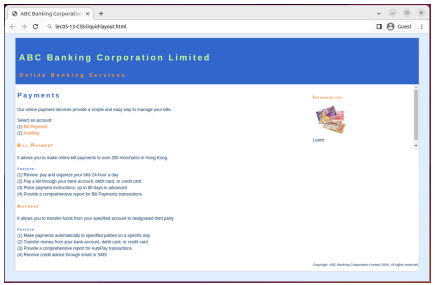
Wide Window

Code Example: lec06-17-CSS-fixed-layout.html

Liquid Layout




```
#mainContainer {
        color: inherit;
        background-color: white;
        margin: 10px auto;
        border: 2px solid #E0E0E0;
        width: 95%;}

#header {
        color: inherit;
        background-color: #3366CC;
        margin: 0px;
        padding: 15px 10px 5px;}

#information {
        padding: 15px 15px;
        width: 25%;
        height: 150px;
        overflow: auto;
        float: right; }
```



Code Example: lec06-18-CSS-liquid-layout.html

Advanced layout module: flex (flexible box layout)

Flexible Boxes

Try to resize the browser window.

A container with "flex-wrap: nowrap;" will never wrap its items.

Note: Flexbox is not supported in Internet Explorer 10 or earlier versions.

Flexible Boxes

Try to resize the browser window.

A container with "flex-wrap: nowrap;" will never wrap its items.

Note: Flexbox is not supported in Internet Explorer 10 or earlier versions.

```
<head>
<style>
.flex-container {
  display: flex;
  flex-wrap: nowrap;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
</head>
<body>
<h1>Flexible Boxes</h1>

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

<p>Try to resize the browser window.</p>
<p>A container with "flex-wrap: nowrap;" will never
wrap its items.</p>
<p><strong>Note:</strong> Flexbox is not supported in
Internet Explorer 10 or earlier versions.</p>

</body>
</html>
```

`display: flex;` turn its child elements into flex items.

The `flex-wrap` property specifies whether the flexible items should wrap or not:

- `nowrap`: the flexible items will be wrap
- `Wrap`: will wrap if necessary
- `wrap-reverse`: will wrap in reverse order if necessary

Code Example: lec05-19-CSS-flex.html

More flex properties (1)

The `nth-of-type()` is a CSS pseudo-class selector that selects elements based on their position within a parent container

The flex-grow Property

Note: Internet Explorer 10 and earlier versions do not support the flex-grow property.

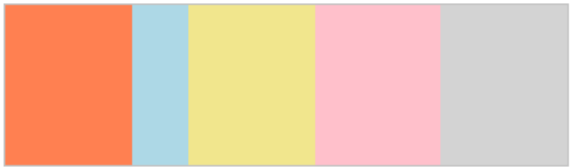
The `flex-grow` property specifies how much the item will **grow relative to the rest of the flexible items** inside the same container.

```
<head>
<style>
#main {
  width: 350px;
  height: 100px;
  border: 1px solid #c3c3c3;
  display: flex;
}

#main div:nth-of-type(1) {flex-grow: 1;}
#main div:nth-of-type(2) {flex-grow: 3;}
#main div:nth-of-type(3) {flex-grow: 1;}
#main div:nth-of-type(4) {flex-grow: 1;}
#main div:nth-of-type(5) {flex-grow: 1;}
</style>
</head>
<body>

<h1>The flex-grow Property</h1>

<div id="main">
  <div style="background-color:coral;"></div>
  <div style="background-color:lightblue;"></div>
  <div style="background-color:khaki;"></div>
  <div style="background-color:pink;"></div>
  <div style="background-color:lightgrey;"></div>
</div>

<p><b>Note:</b> Internet Explorer 10 and earlier
versions do not support the flex-grow property.</p>

</body>
```

Code Example: lec05-20-CSS-flex-property.html

More flex properties (2)

The flex-shrink Property



Note: Internet Explorer 10 and earlier versions do not support the flex-shrink property.

The `flex-shrink` property specifies how the item will **shrink relative to the rest of the flexible items** inside the same container.

```
<head>
<style>
#main {
  width: 350px;
  height: 100px;
  border: 1px solid #c3c3c3;
  display: flex;
}

#main div {
  flex-grow: 1;
  flex-shrink: 1;
  flex-basis: 100px;
}

#main div:nth-of-type(2) {
  flex-shrink: 3;
}
</style>
</head>
<body>

<h1>The flex-shrink Property</h1>

<div id="main">
  <div style="background-color:coral;"></div>
  <div style="background-color:lightblue;"></div>
  <div style="background-color:khaki;"></div>
  <div style="background-color:pink;"></div>
  <div style="background-color:lightgrey;"></div>
</div>

<p><b>Note:</b> Internet Explorer 10 and earlier
versions do not support the flex-shrink property.</p>
```

Code Example: lec05-21-CSS-flex-property2.html

Agenda

CSS Layout

- Review: The box model
- Positioning
- Viewing pattern/typical layout

CSS3

- Effects
- Transform
- Transition
- Animation

CSS3 overview

Introduce rgba(): specifying a color using **r**ed, **g**reen, **b**lue, and **a**lpha (transparency) values.
e.g., `rgba(255, 0, 0, 0.5)` represents read color with 50% transparency

Specify element effects such as `box-shadow`, `border-radius`

Used in media queries to apply styles based on different media types

Specifies the transition effect for a CSS property over a specified duration.

Allows applying transformations to elements, such as scaling, rotating, skewing, or translating them.

`@keyframes`: Used to define animations by specifying keyframes at different percentages of the animation's duration.

Effect: Round Border (1)

border-radius: *value*

the size of the circle radius, or the semi-major and semi-minor axes of the ellipse

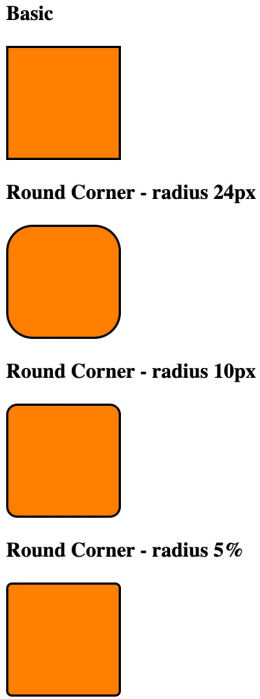
The value can be in px or percentage

Can be specified for each corner

```
border-radius: 24px 0 24px 0;
```

```
#rb24 {
    border-radius: 24px;
}

#rb10 {
    border-radius: 10px;
}


#percent {
    border-radius: 5%;
}
```

```
<h3>Basic</h3>
<div class="com" id="b">
</div>

<h3>Round Corner - radius 24px </h3>
<div class="com" id="rb24">
</div>

<h3>Round Corner - radius 10px </h3>
<div class="com" id="rb10">
</div>

<h3>Round Corner - radius 5% </h3>
<div class="com" id="percent">
</div>
```



Question time: What if we change the *border-radius* to 50%?

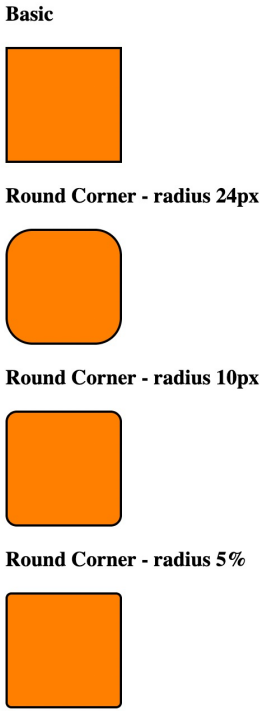
Code Example: lec05-22-CSS-CSS3-radius-border.html

Effect: Round Border (2)

In some old webpages, CSS3 properties can be specified as

- `border-radius: 10px;` (official standard)
- `-webkit-border-radius: 10px;` (Safari, Chrome)
- `-moz-border-radius: 10px;` (Firefox)

known as vendor-specific prefixes a particular browser when the property is not supported by all as a standard property



Code Example: lec05-23-CSS-CSS3-radius-border.html

Effect: Box Shadow

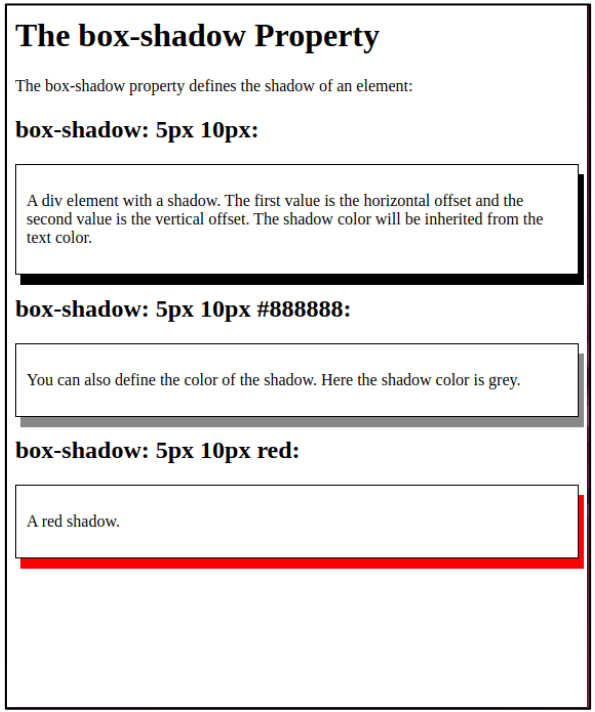
Create a shadow for a box

box-shadow: x, y, blur, spread, color

- x, y - position of the box relative to element
- the blur radius

```
<!DOCTYPE html>
<html>
<head>
<style>
#example1 {
  border: 1px solid;
  padding: 10px;
  box-shadow: 5px 10px;
}

#example2 {
  border: 1px solid;
  padding: 10px;
  box-shadow: 5px 10px #888888;
}

#example3 {
  border: 1px solid;
  padding: 10px;
  box-shadow: 5px 10px red;
}
</style>
</head>
<body>

<h1>The box-shadow Property</h1>
<p>The box-shadow property defines the shadow of an element:</p>

<h2>box-shadow: 5px 10px:</h2>
<div id="example1">
  <p>A div element with a shadow. The first value is the horizontal offset and the second va
</div>

<h2>box-shadow: 5px 10px #888888:</h2>
<div id="example2">
  <p>You can also define the color of the shadow. Here the shadow color is grey.</p>
</div>

<h2>box-shadow: 5px 10px red:</h2>
<div id="example3">
  <p>A red shadow.</p>
</div>

</body>
</html>
```

Code Example: lec05-24-CSS-CSS3-box-shadow.html

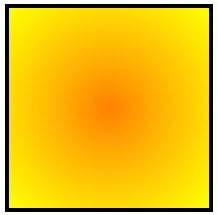


Effect: Gradient (1)

Gradient: a color composed of gradual change of two (or more) colors e.g.,

```
background: -webkit-linear-gradient(circle, #FF8000, #FFFF00);
```

Gradient



`#FF8000` is the hexadecimal representation of colors, where the six digits represents red (FF), green (80), blue (00), respectively. In hexadecimal system, FF = 255 in decimal, which is the largest value; 80 = 128 (2^8 + 0) in decimal; 00 = 0 in decimal, which is the smallest value

Note that the "interpolated" color (gradient result) is a special type of image and therefore cannot be used as a value for any color related property

Effect: Gradient (2)

Linear gradient

- top to bottom: `linear-gradient(yellow, green);`
- left to right: `linear-gradient(left, yellow, green);`
- angle: `linear-gradient(top left, yellow, green);`

Radial gradient

- circle: `radial-gradient(circle, yellow, green);`
- eclipse: `radial-gradient(ellipse, yellow, green);`

Effect: Gradient (3)

What if we want to create a rainbow background?

```
<head>
    <meta charset="utf-8">
    <title>CSS Techniques</title>
    <style>
        div.com {
            width: 100px;
            height: 100px;
            border: 2px black solid;
        }

        #left {
            width: 50%;
            float: left;
            padding: 25px;
        }

        #linear-g-top {
            background: linear-gradient(top, yellow, green);
            background: -webkit-linear-gradient(top, yellow, green);
            background: -moz-linear-gradient(top, yellow, green);
        }

        #linear-g-left {
            background: linear-gradient(left, yellow, green);
            background: -webkit-linear-gradient(left, yellow, green);
            background: -moz-linear-gradient(left, yellow, green);
        }

        #linear-g-top-left {
            background: linear-gradient(top left, yellow, green);
            background: -webkit-linear-gradient(top left, yellow, green);
            background: -moz-linear-gradient(top left, yellow, green);
        }

        #radial-g-cricle {
            background: radial-gradient(circle, yellow, green);
            background: -webkit-radial-gradient(circle, yellow, green);
            background: -moz-radial-gradient(circle, yellow, green);
        }

        #radial-g-ellipse {
            background: radial-gradient(ellipse, yellow, green);
            background: -webkit-radial-gradient(ellipse, yellow, green);
            background: -webkit-radial-gradient(ellipse, yellow, green);
        }
    </style>
</head>

<body>
    <div id="left">
        <h3>Linear gradient top to bottom</h3>
        <div class="com" id="linear-g-top">
        </div>

        <h3>Linear gradient left to right</h3>
        <div class="com" id="linear-g-left">
        </div>

        <h3>Linear gradient angle</h3>
        <div class="com" id="linear-g-top-left">
        </div>

        <h3>Radial gradient circle</h3>
        <div class="com" id="radial-g-cricle">
        </div>

        <h3>Radial gradient ellipse</h3>
        <div class="com" id="radial-g-ellipse">
        </div>
    </div>
</body>
```

Linear gradient top to bottom

Linear gradient left to right

Linear gradient angle

Radial gradient circle

Radial gradient ellipse

Code Example: lec05-25-CSS-CSS3-gradient-basic.html

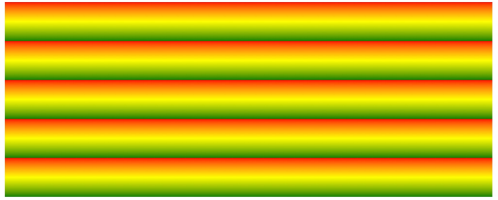
Effect: Gradient (4)

```
<!DOCTYPE html>
<html>
<head>
<style>
div.com {
    width: 500px;
    height: 200px;
  }

#g-rainbow{
  background-color: red; /* For browsers that do not support gradients */
  background-image: linear-gradient(to right, red, orange, yellow, green, blue, indigo, violet);
}

#g-transparent {
  background-image: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));
}

#g-repeat-linear {
  background-color: red; /* For browsers that do not support gradients */
  background-image: repeating-linear-gradient(red, yellow 10%, green 20%);
}

#g-repeat-linear-angle {
  background-color: red; /* For browsers that do not support gradients */
  background-image: repeating-linear-gradient(45deg,red,yellow 7%,green 10%);
}

</style>
</head>
<body>

<h3>Rainbow Background</h3>
<div class="com" id="g-rainbow">
</div>

<h3>Transparent Gradient Background</h3>
<div class="com" id="g-transparent">
</div>

<h3>Repeated Gradient Background</h3>
<div class="com" id="g-repeat-linear">
</div>

<h3>Repeated Gradient Background With Angle</h3>
<div class="com" id="g-repeat-linear-angle">
</div>
</body>
</html>
```

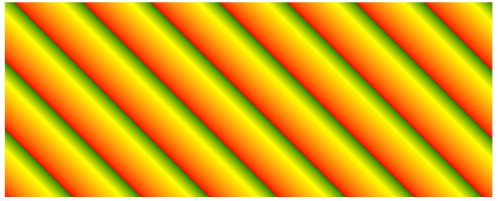
Rainbow Background

Transparent Gradient Background

Repeated Gradient Background



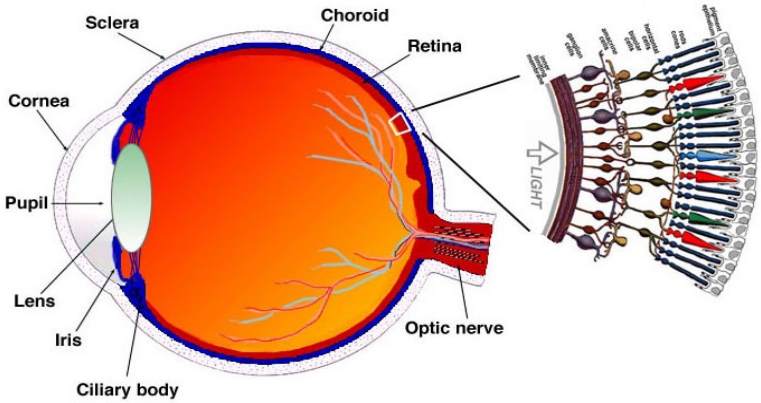
Repeated Gradient Background With Angle



Code Example: lec05-26-CSS-CSS3-gradient-advanced.html

Human Vision

How we see color - Colm Kelleher
https://www.youtube.com/watch?v=l8_fZPHasdo



. A drawing of a section through the human eye with a schematic enlargement of the retina.

Source: https://webvision.med.utah.edu/book/part-i-foundations/simple-anatomy-of-the-retina/

Retina consists of an array of rods and three kinds of cones

- Rods are for night vision
- Cones are for color vision

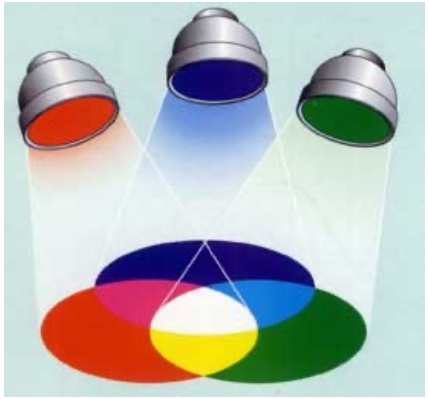
Three kinds of cones:

- L-cone: most sensitive to **red** light
- M-cone: most sensitive to **green** light
- S-cone: most sensitive to **blue** light

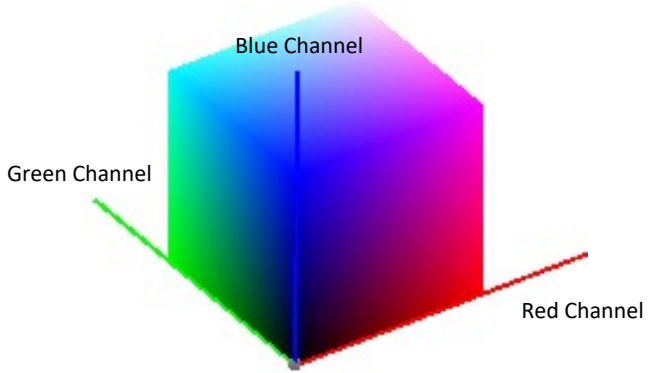
Rods take over cones in low-light environment, therefore we often find it difficult detect colors when the light dims

RGB Color Space

Red, green and blue are 3 primary colours for lights. Other colours are combinations of red, green and blue by different proportions.



http://www.booksmartstudio.com/color_tutorial/colortheory4.html



RGB Color Cube

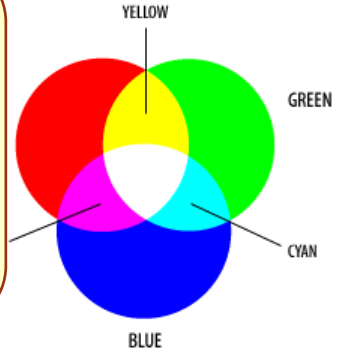
RGB Color Model

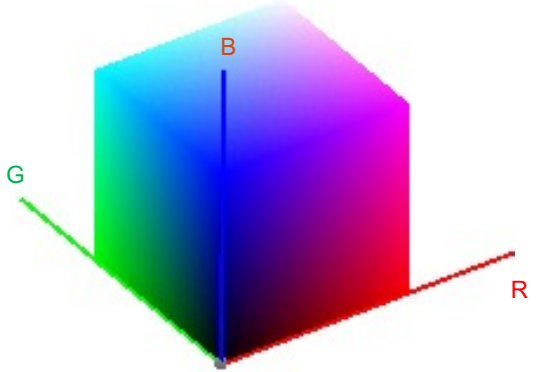
Different colors can be obtained when different values of R, G, B are specified

- e.g., R = 100, G = 100, B = 0 will result in a yellow color
  R = 255, G = 255, B = 0 will result in a brighter yellow color
  R = 255, G = 255, B = 255 will result in a white color
  R = 0, G = 0, B = 0 will result in a black color

What about rgba?

"a" refers to alpha, which represents the transparency of the color from 0 to 1 (when a = 0, there is not color)





RGB Color Cube

What about the hexadecimal color model

For RGB color model: each color (Red, green and blue) can have a value between 0 and 255.

For Hexadecimal color model: a 6-digit representation of color in web design, with each pair of digits representing Red, green and blue values from 00 to FF.

Hexadecimal ("hex") is a base-16 numeral system, which count number in a way like:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Thus, F in hex is 15 in decimal, which is 15×16^0,
so FF in hex is $15 \times 16^1 + 15 \times 16^0 = 255$

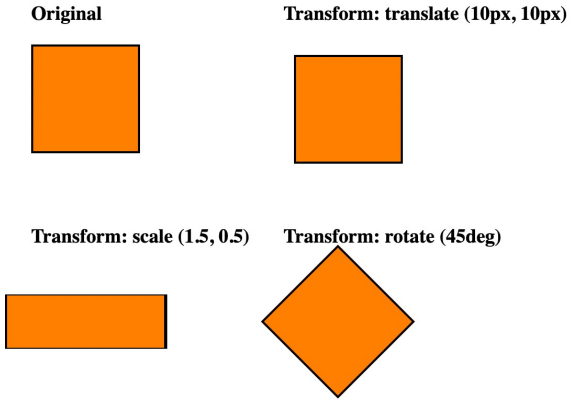
Html color name	HEX	RGB
Aqua	#00FFFF	0, 255, 255
Blue	#0000FF	0, 0, 255
Navy	#000080	0, 0, 128
Yellow	#FFFF00	255, 255, 0
Gold	#FFD700	225, 215, 0
Lime	#00FF00	0, 255, 0
Green	#008000	0, 128, 0
Red	#FF0000	255, 0, 0
maroon	#800000	128, 0, 0
Fuchsia	#FF00FF	255, 0, 255
Purple	#800080	128, 0, 128
White	#FFFFFF	255, 255, 255
Silver	#C0C0C0	192, 192, 192
Gray	#808080	128, 128, 128

Transform

Transform performs transformation on the original style (position) of an element to the target style (position)

The value of transform property is different value functions:

- transform: translate(10px, 10px);
- transform: scale(1.5, 0.5);
- transform: rotate(45deg);
- the default transform origin is at center



Code Example: lec05-26-CSS-CSS3-transform.html

Transition (1)

Two states of a property changing gradually over a period of time

Usually associate with an event to cause the property change

```
#trans {
        transition: background-color 3s linear;
        -webkit-transition: background-color 3s linear;
}
#trans:hover {
        background-color: #004080;
}
```

Transition (2)

```
#trans {
        transition: background-color 3s linear;
        -webkit-transition: background-color 3s linear;
}
#trans:hover {
        background-color: #004080;
}
```

e.g., property is background-color:

initial state: the original background color

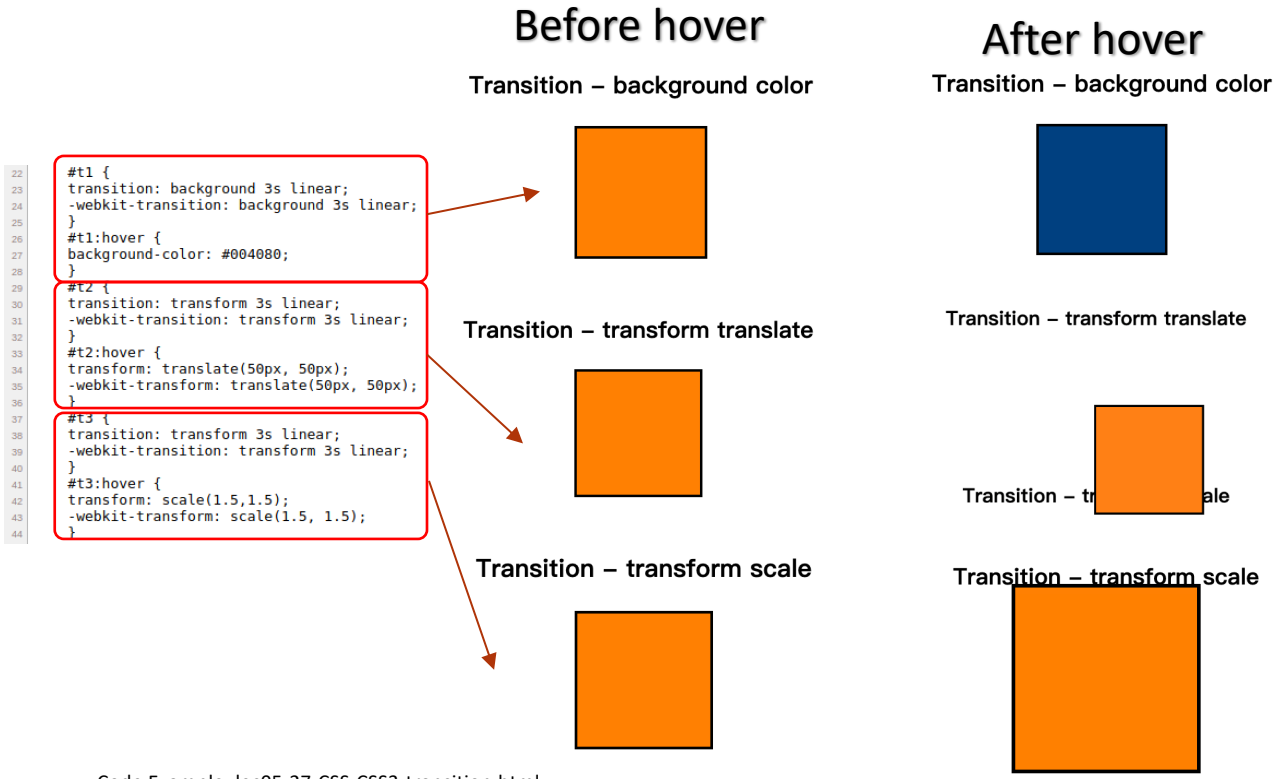
final state: background color `#004080`

triggering event: mouse over (e.g., `:hover`)

different from normal behavior: the gradual constant change (linear) over a period of 3 seconds rather than an instant change

Transition (3): Example

```
#t1 {
transition: background 3s linear;
-webkit-transition: background 3s linear;
}
#t1:hover {
background-color: #004080;
}
#t2 {
transition: transform 3s linear;
-webkit-transition: transform 3s linear;
}
#t2:hover {
transform: translate(50px, 50px);
-webkit-transform: translate(50px, 50px);
}
#t3 {
transition: transform 3s linear;
-webkit-transition: transform 3s linear;
}
#t3:hover {
transform: scale(1.5,1.5);
-webkit-transform: scale(1.5, 1.5);
}
```



Code Example: lec05-27-CSS-CSS3-transition.html

Transition (3)

Speed curve: the speed of change over the duration

- **`ease`** - specifies a transition effect with a slow start, then fast, then end slowly (default)
- **`linear`** - specifies a transition effect with the same speed from start to end
- **`ease-in`** - specifies a transition effect with a slow start
- **`ease-out`** - specifies a transition effect with a slow end
- **`ease-in-out`** - specifies a transition effect with a slow start and end
- **`cubic-bezier(n,n,n,n)`** - define your own values in a cubic-bezier function

Transition (4): Example


```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: yellowgreen;
  transition: width 2s;
}

#div1 {transition-timing-function: linear;}
#div2 {transition-timing-function: ease;}
#div3 {transition-timing-function: ease-in;}
#div4 {transition-timing-function: ease-out;}
#div5 {transition-timing-function: ease-in-out;}

div:hover {
  width: 300px;
}
</style>
</head>
<body>

<h1>The transition-timing-function Property</h1>

<p>Hover over the div elements below, to see the different speed curves:</p>

<div id="div1">linear</div><br>
<div id="div2">ease</div><br>
<div id="div3">ease-in</div><br>
<div id="div4">ease-out</div><br>
<div id="div5">ease-in-out</div><br>

</body>
</html>
```

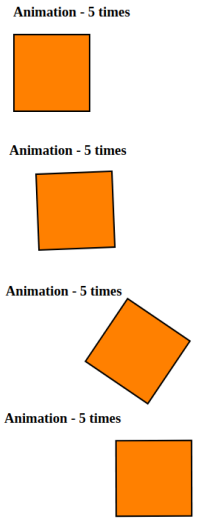
Code Example: lec05-28-CSS-CSS3-transition-speed.html

Animation (1)

An extension of transition by having more complicated properties and styles

- It can have an iteration count (to perform the animation for a number of times) and more than 2 states. Introduce the key frame concept (**`@keyframes`** specifies the animation code)

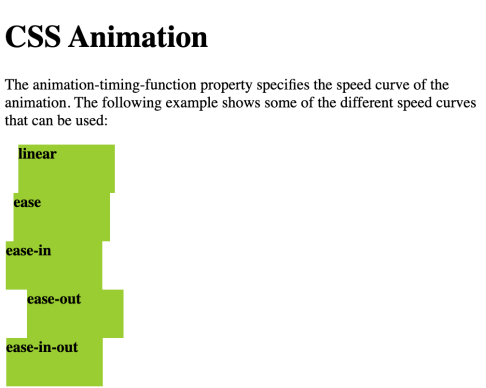
```
#rt {
animation-name: divRotate;
animation-duration: 2s;
animation-iteration-count: 5;
-webkit-animation-name: divRotate;
-webkit-animation-duration: 2s;
-webkit-animation-iteration-count: 5;
}
@keyframes divRotate {
  from {
    transform: rotate(0deg);
    margin-left:0px;
  }
  to {
    transform: rotate(360deg);
    margin-left: 150px;
  }
}
@-webkit-keyframes divRotate {
  from {
    -webkit-transform: rotate(0deg);
    margin-left: 0px;
  }
  to {
    -webkit-transform: rotate(360deg);
    margin-left: 150px;
  }
}
```



Code Example: lec05-29-CSS-CSS3-advance-tech.html

Animation (2)

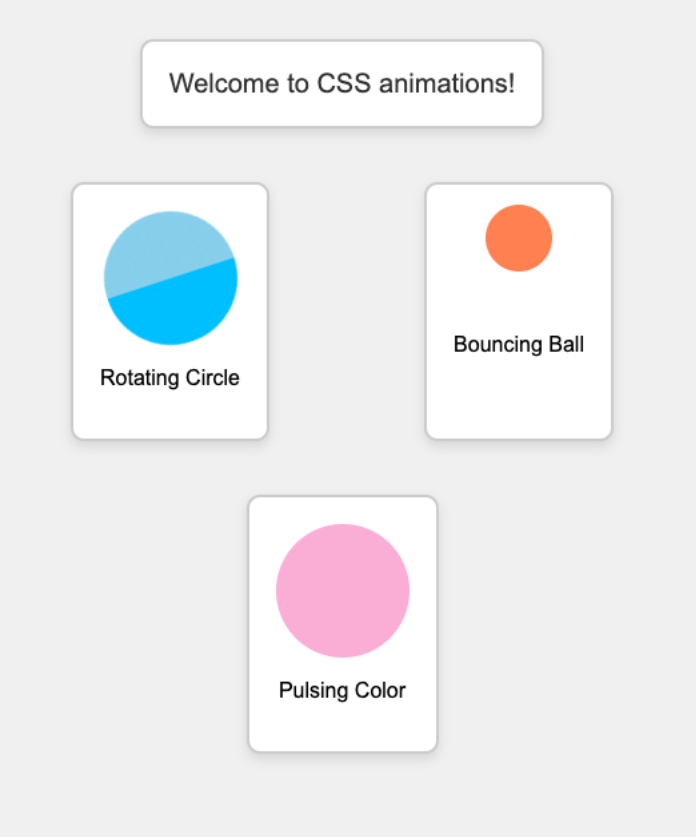
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 50px;
  background-color: yellowgreen;
  font-weight: bold;
  position: relative;
  animation: mymove 5s infinite;
}

#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}

@keyframes mymove {
  from {left: 0px;}
  to {left: 300px;}
}
</style>
</head>
<body>

<h1>CSS Animation</h1>

<p>The animation-timing-function property specifies the
speed curve of the animation. The following example
shows some of the different speed curves that can be
used:</p>

<div id="div1">linear</div>
<div id="div2">ease</div>
<div id="div3">ease-in</div>
<div id="div4">ease-out</div>
<div id="div5">ease-in-out</div>

</body>
</html>
```



Code Example: lec05-30-CSS-CSS3-animation-speed.html

Animation (3)

```css
.fade-in {
    opacity: 0;
    animation: fadeInAnimation ease-in-out 3s;
    animation-fill-mode: forwards;
    font-size: 20px;
    color: #333;
}

@keyframes fadeInAnimation {
    from { opacity: 0; }
    to { opacity: 1; }
}

.rotate {
    width: 100px;
    height: 100px;
    background: linear-gradient(to right, skyblue 50%, deepskyblue 50%);
    border-radius: 50%;
    animation: rotation 4s infinite linear;
}

@keyframes rotation {
    from { transform: rotate(0deg); }
    to { transform: rotate(360deg); }
}

.bounce {
    width: 50px;
    height: 50px;
    background-color: coral;
    border-radius: 50%;
    animation: bounceAnimation 2s infinite;
}

@keyframes bounceAnimation {
    0%, 20%, 50%, 80%, 100% { transform: translateY(0); }
    40% { transform: translateY(-30px); }
    60% { transform: translateY(-15px); }
}

.pulse {
    width: 100px;
    height: 100px;
    background-color: violet;
    border-radius: 50%;
    animation: pulseAnimation 3s infinite;
}

@keyframes pulseAnimation {
    0%, 100% { background-color: violet; }
    50% { background-color: pink; }
}
```



Code Example: lec05-31-CSS-CSS3-more-animation-speed.html

Lecture summary

All the HTML elements or a group of elements can be seen as a box (or block) with margin, padding, border, content, width and height

Positioning is how the browser puts elements into containers. This determines the layout of a website. There are three types of HTML elements regarding positioning: `block, inline, inline-block`

Four basic layout scheme: static, relative, absolute, fixed, where overflow may or may not occur

Floating property allows developers to adjust element position and create multi-column design in normal flow

Two commonly used layout: fixed and liquid layouts. Both have their advantages and disadvantages

Flex – flexible box layout is an advance layout module that allows flexible setting of element positions

CSS3 introduces a lot features that made it possible to create advance effects and animation without JavaScript