

CS2204 Fundamentals of Internet Applications Development

Lecture 6 CSS – Part 4

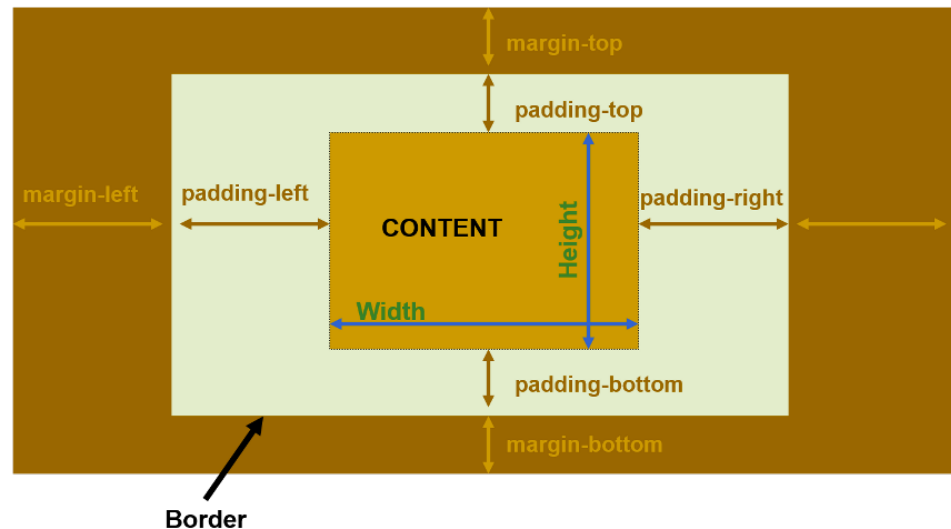
Computer Science, City University of Hong Kong

Semester A 2023-24

Last Week

Each tag can be treated as a discrete element box on the screen and controlled by CSS, with the **properties**

The Box Model



Review: the box model

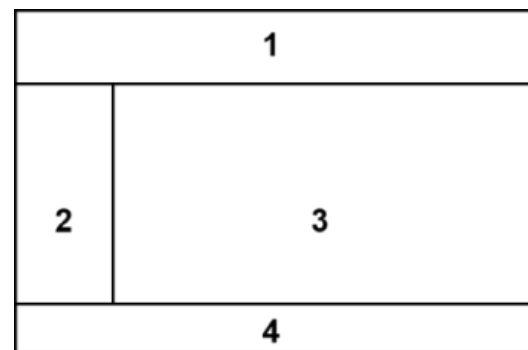
1. What is the purpose of specifying the overflow property?
2. What are the possible values of overflow property?

Outline

- Positioning
 - Four types of positioning
 - Review of float and clear
- Responsive Design
- CSS3 advanced techniques

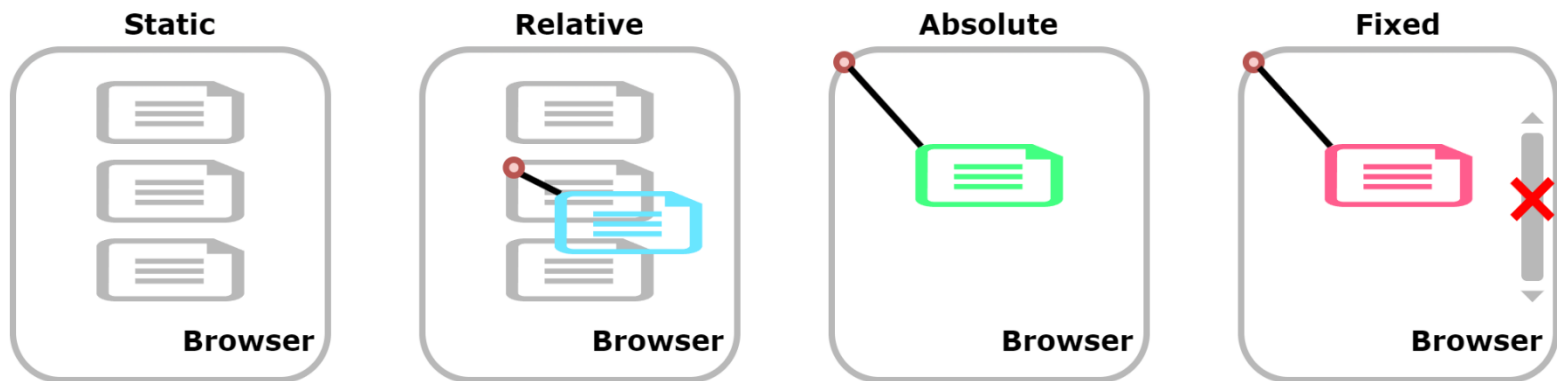
Structuring Your Page For Layout

- Consider a typical layout with 4 sections:
 - Header (<header>)
 - Content (<section>)
 - Navigation (<nav>)
 - Footer (<footer>)
- Three typical layouts
 - normal flow
 - floating
 - positioning



Positioning

- **Layout** depends on how the browser puts elements into containers. This is called **positioning**
- Position: **static** | **relative** | **absolute** | **fixed**



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

Positioning

- **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

Types of Elements (1)

Block-level element

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

```
<!DOCTYPE html>
<html>
<body>

<p style="border: 1px solid black">Hello
World</p>
<div style="border: 1px solid black">Hello
World</div>

<p>The P and the DIV elements are both block
elements, and they will always start on a new
line and take up the full width available
(stretches out to the left and right as far
as it can).</p>

</body>
</html>
```

Hello World

Hello World

The P and the DIV elements are both block elements, and they will always start on a new line and take up the full width available (stretches out to the left and right as far as it can).

Types of Elements (2)

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**
- setting width and height is **invalid**

```
<!DOCTYPE html>
<html>
<body>

<p>This is an inline span <span style="border: 1px
solid black">Hello World</span> element inside a
paragraph.</p>

<p>The SPAN element is an inline element, and will
not start on a new line and only takes up as much
width as necessary.</p>

</body>
</html>
```

Code Example: lec07-02-CSS-inline.html

This is an inline span Hello World element inside a paragraph.

The SPAN element is an inline element, and will not start on a new line and only takes up as much width as necessary.

Types of Elements (3)

Inline-block element

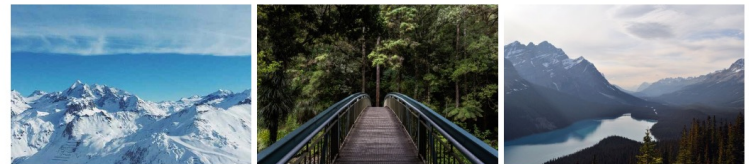
- multiple elements can be in **one line**, but their **width**, **height**, **padding** and **margin** can be adjusted, e.g., `<img>`, `<input>`, `<td>`, etc.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<h2>Images Side by Side</h2>
<p>img are inline-block elements by default</p>
  
  
  
</body>
</html>
```

Images Side by Side

img are inline-block elements by default

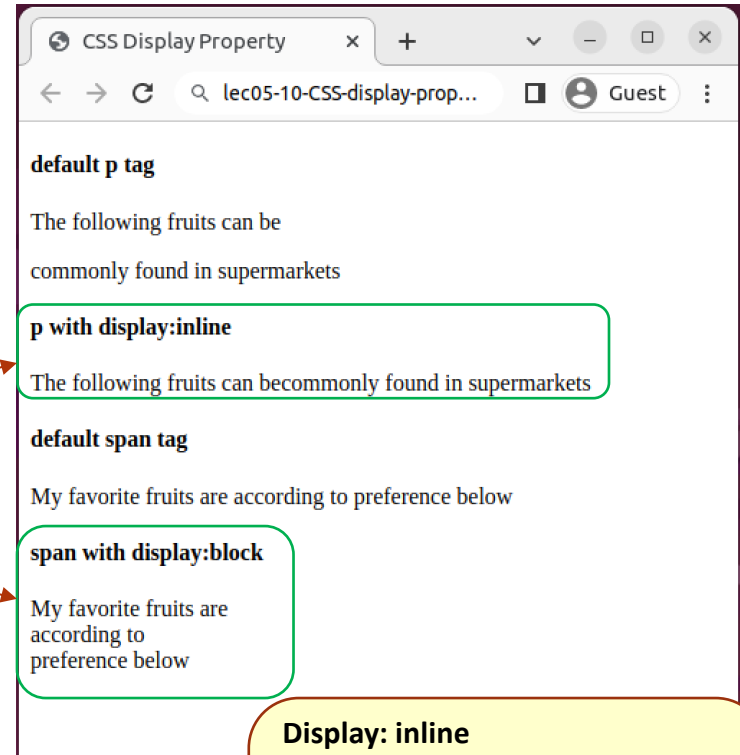


Critical thinking: What if we change the width property of the first image to 100%?

Change Display Property (1)

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



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

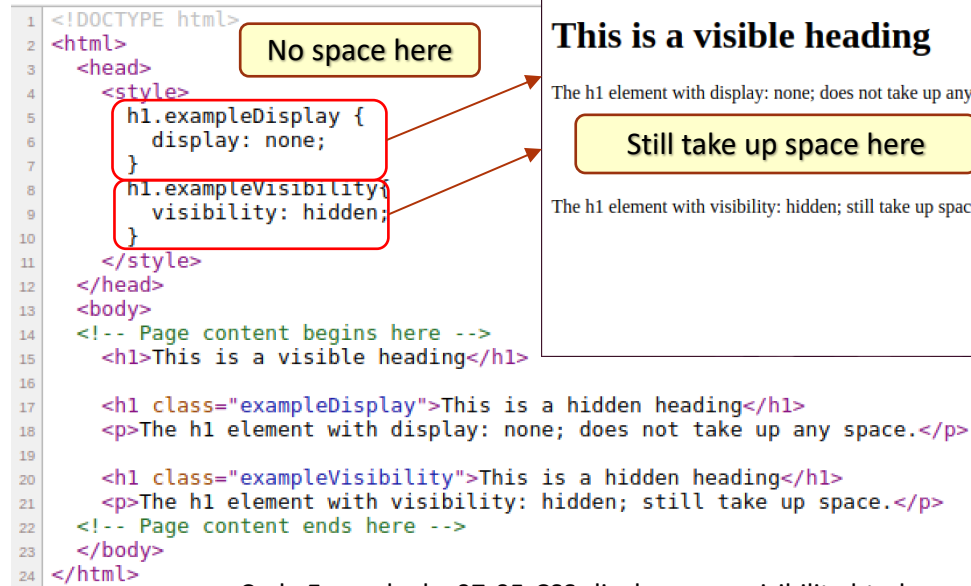
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>CSS Display Property</title>
6   <style>
7     #exampleBlock span {
8       display: block;
9     }
10
11     #exampleInline p{
12       display: inline;
13     }
14
15   </style>
16 </head>
17 <body>
18   <!-- Page content begins here -->
19   <h4>default p tag</h4>
20   <p>The following fruits can be</p><p>commonly found in supermarkets</p>
21
22   <h4>p with display:inline</h4>
23   <div id="exampleInline">
24     <p>The following fruits can be</p><p>commonly found in supermarkets</p>
25   </div>
26
27   <h4>default span tag</h4>
28   <span>My favorite fruits are</span> <span>according to</span> <span>preference below</span>
29
30   <h4>span with display:block</h4>
31   <div id="exampleBlock">
32     <span>My favorite fruits are</span> <span>according to</span> <span>preference below</span>
33   </div>
34   <!-- Page content ends here -->
35 </body>
36 </html>
```

Code Example: lec07-04-CSS-display-property.html

Change Display Property (2)

- **Display: none**

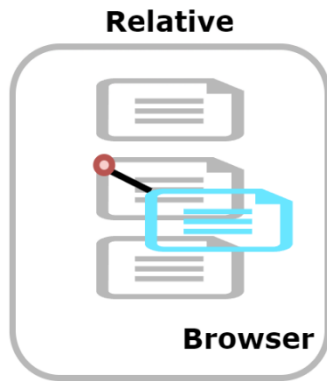
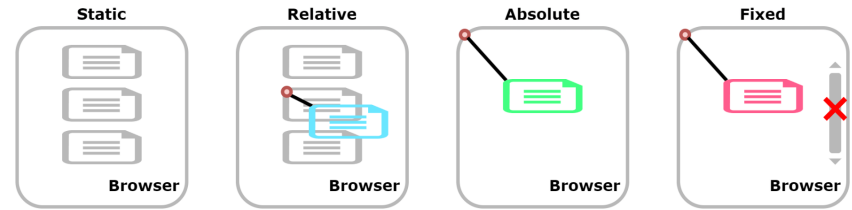
- make the element disappear (but the **element still exists** in the DOM)
- **does not take up any space** (no box generated)
- compare with the property *visibility: hidden*



Code Example: lec07-05-CSS-display-none-visibility.html

Relative Position

- Relative:
 - 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**



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



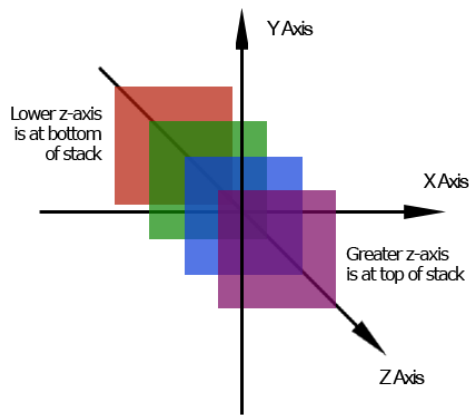
```
8   <style>
9     div {
10      height: 100px;
11      width: 100px;
12      background-color: yellowgreen;
13    }
14    .relative {
15      background-color: blue;
16      position: relative;
17      top: 20px;
18      left: 20px;
19    }
20  </style>
21  </head>
22  <body>
23    <div></div>
24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



Code Example: lec07-06-CSS-relative.html

Z-Index

- **Z-index** property specifies the **stack order** of an element
- **Only** works for relative, absolute and fixed 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>
```

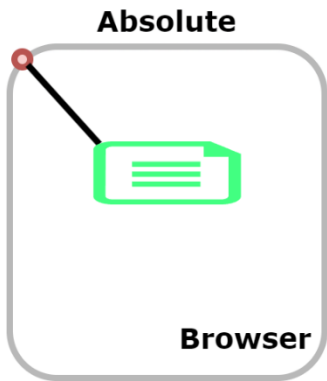
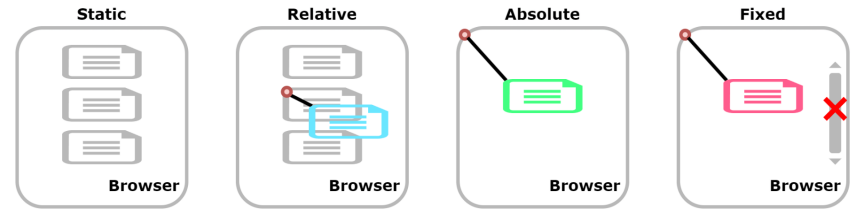


Code Example: lec07-07-CSS-Z-index.html

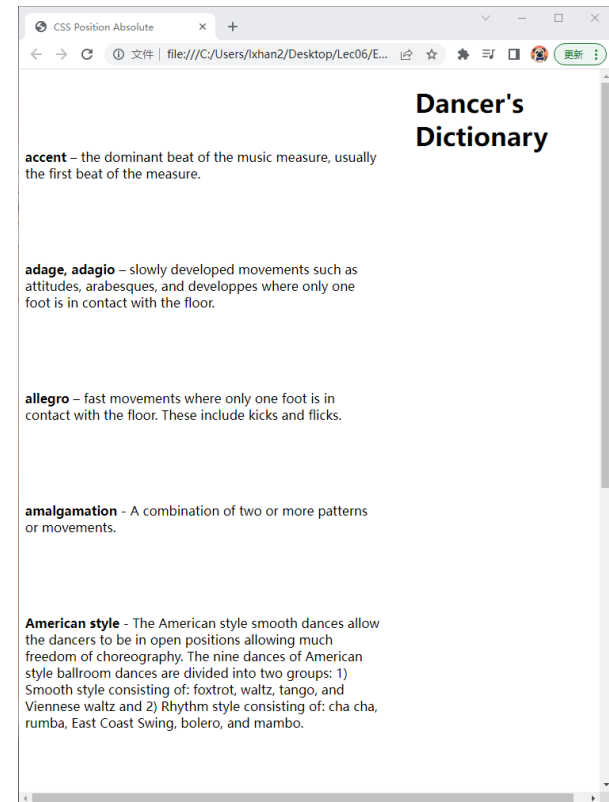
Absolute Position

- Absolute:

- The box is **taken out of** the **normal flow**
- The **offset property** of the box is relation to **its containing element**



```
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>
```



Code Example: lec07-08-CSS-absolute.html

Absolute Position (2)

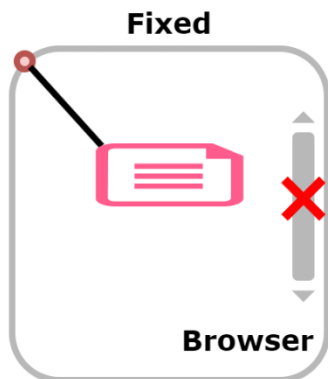
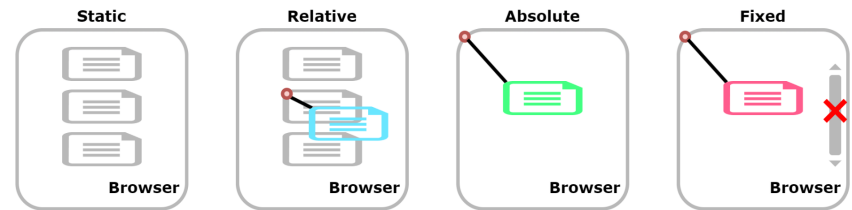
The **offset property** of the box is relative to **the nearest positioned ancestor** that are **not static**

If an absolute positioned element has no positioned ancestors or the ancestors are static, it uses the `document<body>`, and moves along with page scrolling.

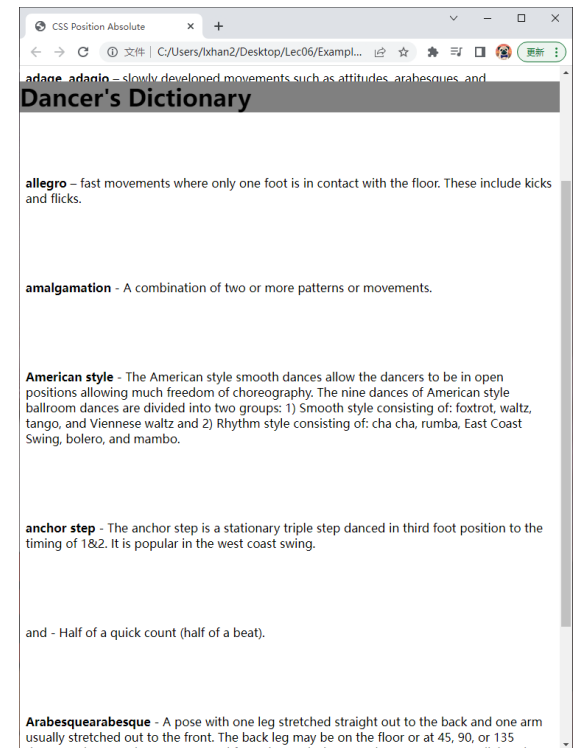
If **none** of **top**, **bottom**, **left** and **right** is set, the starting position is in the normal flow

Fixed Position

- Fixed:
 - The **offset property** of the box is **relation to the browser window**



```
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: lec06-13-CSS-fixed.html

absolute and fixed

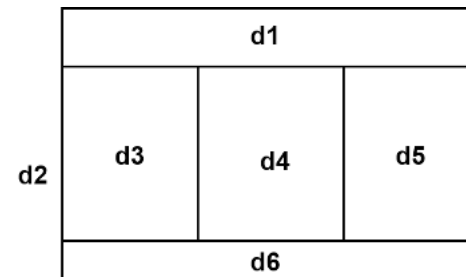
Absolute elements

- Removed from the **normal flow**
- Controlled by top/bottom/left/right values relative to **ancestor elements**; if not existing, go up until up the browser window
- Used to control position within a **container that is not static**
- Can overlap with other elements, then controlled by z-index

Fixed elements

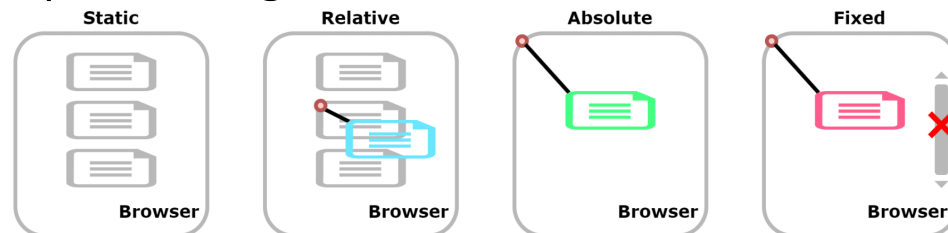
- Removed from **normal flow**
- Controlled by top/bottom/left/right values relative to the **initial containing block (browser window)**
- Can overlap with other elements, controlled by the z-index

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



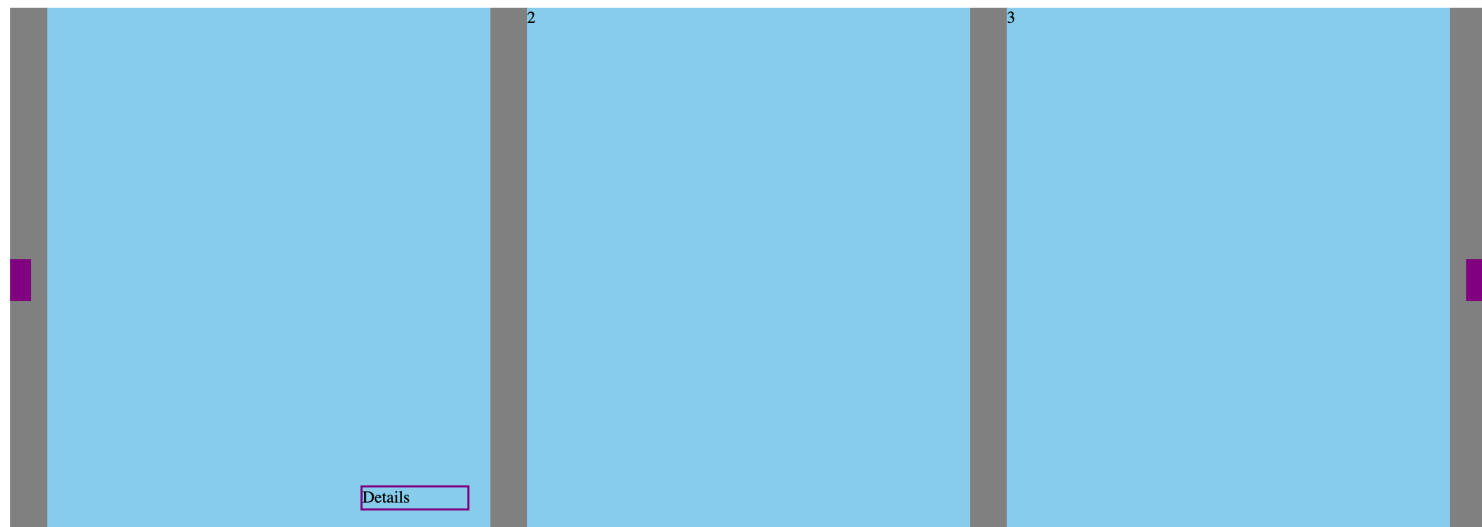
Review of Float Properties

- For each **floating** element
 - it will **leave** the **normal** flow
 - it does **not** occupy the **original position** in the normal flow
 - its **width** and **height** can be set (inline-block property)
- For **multiple** floating elements
 - they will be displayed in the **same** line and **aligned on top**
 - if parent box cannot hold all boxes, they will occupy multiple lines
- Element placement
 - **Vertical** direction: normal flow
 - **Horizontal** direction: floating
 - **Precise** placement: positioning



Example

- main container (w: 80%, h: 500px)
- three boxes (w: 30%, h: 500px)
 - one: margin-left: 2.5%; two: left/right: 2.5%; three: margin-right: 2.5%
- blocks (w: 20px, h: 40px)
 - lb: left: 0px; top: 240px; rb: right: 0px; top: 240px
- details (w: 100px, h: 20px, border: 2px solid purple)



Clear

- To **stop** floating sequence at some point, we can **clear** the float property using:
`{clear: left | right | both; }`
- **No** element should appear on its **left**, **right** or **both** sides

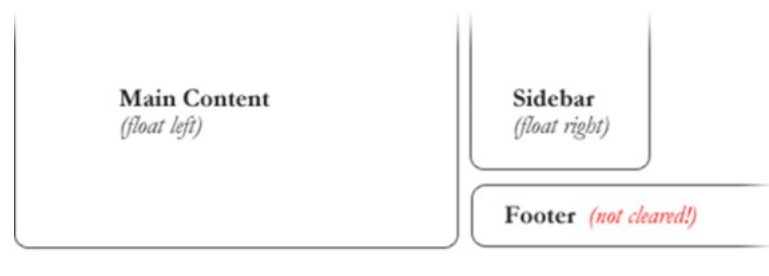
Image Gallery

Try resizing the window to see what happens when the images does not have enough room.



Code Example: lec06-14-CSS-clear.html

- One 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)



Source: <https://css-tricks.com/all-about-floats/>

Clear

- Another common use of clear property: if a parent box does not have height specified, all its child boxes are floating

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <style>
7     .box {
8       width: 300px;
9       background-color: red;
10      border: 1px solid black;
11      margin: 0 auto;
12    }
13    .one {
14      width: 100px;
15      height: 100px;
16      background-color: yellow;
17      float: left;
18    }
19    .two {
20      width: 100px;
21      height: 100px;
22      background-color: green;
23      float: left;
24    }
25    .three {
26      clear: both;
27    }
28  </style>
29 </head>
30 <body>
31   <div class="box">
32     <div class="one">one</div>
33     <div class="two">two</div>
34     <div class="three"></div>
35   </div>
36 </body>
37 </html>
```



Code Example: lec06-15-CSS-clear2.html

calc()

- it is function to be used as the property value
 - for example, box-1 contains box-2, and we want to set the width of box-2 to be always less than that of box-1 by 100px
 - `width: calc(100% - 100px);`

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <style>
7     .one {
8       width: 80%;
9       height: 100px;
10      background-color: skyblue;
11      margin: 0 auto;
12    }
13    .two {
14      width: calc(100% - 100px);
15      height: 50px;
16      background-color: red;
17    }
18  </style>
19 </head>
20 <body>
21   <div class="one">
22     <div class="two"></div>
23   </div>
24 </body>
25 </html>
```

Code Example: lec06-16-CSS-calc.html

Outline

- Positioning
 - Four types of positioning
 - Review of float and clear
- Responsive Design
- CSS3 advanced techniques

Fixed Layout (1)

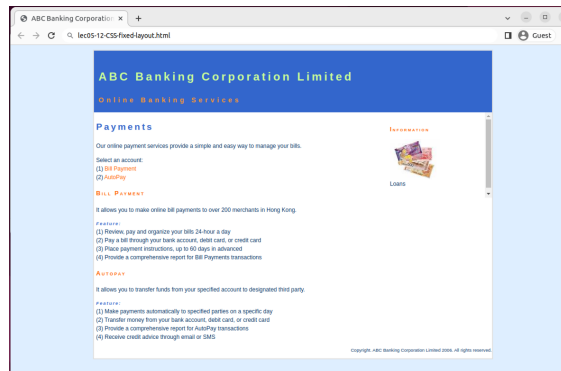
- To prevent this layout change with the change of window (container) size
- **Not** to be confused with **Fixed Positioning**.
- Use a **big container** (body or div) to hold the page and set its **width** (height) to a **fixed, absolute** value (e.g., 1024px)

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

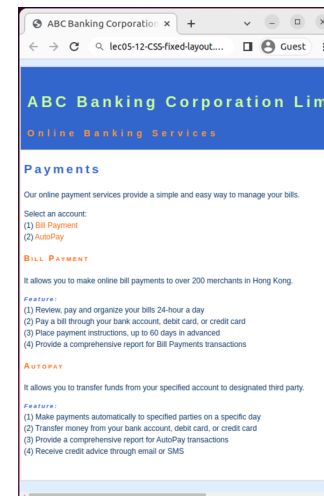
#header {
  color: inherit;
  background-color: #3366CC;
  margin: 0px;
  padding: 15px 10px 5px;}

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

Wide Window



Narrow Window



Fixed Layout (2)

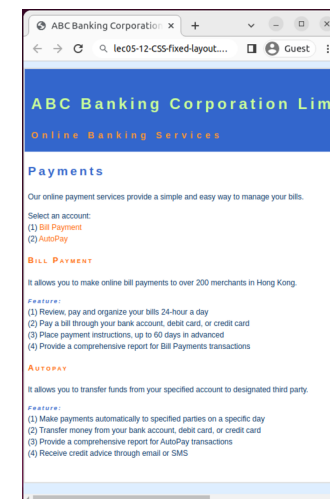
Advantages:

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

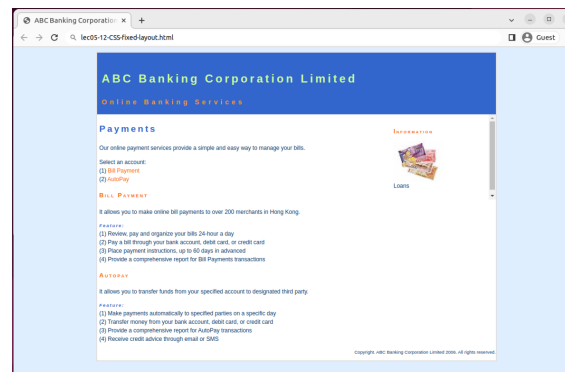
Limitations:

- Cause horizontal/vertical scrolling in smaller browser windows
- Result in large area of white space in larger screen
- Cannot fit in small screen device

Narrow Window



Wide Window

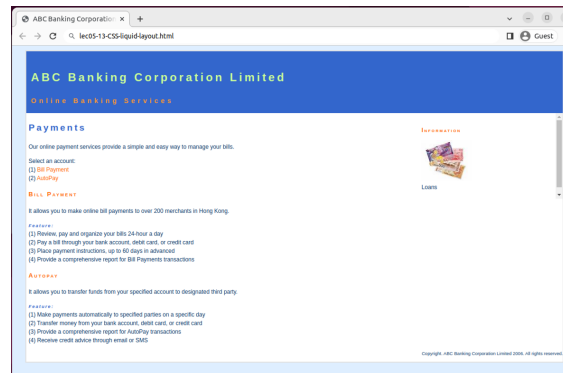


Liquid Layout (1)

- Also known as **fluid layout**.
- The element sizes will enlarge/shrink with the window (container) size.
- Set all dimensions in **relative units**, e.g., **percentages**.

```
#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;  
}
```

Wide Window



Narrow Window



Code Example: lec07-12-CSS-liquid-layout.html

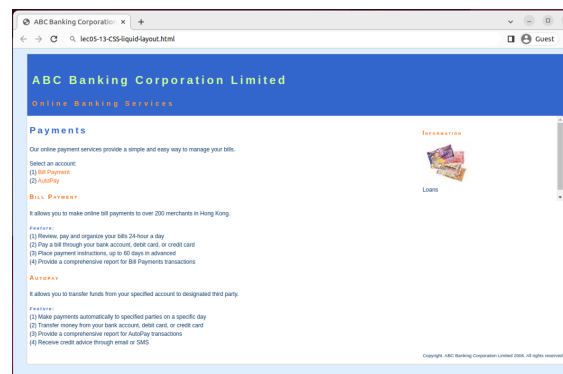
Liquid Layout (2)

Advantages:

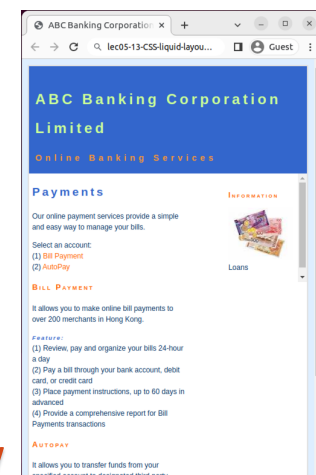
- Pages expand/contract to fill the available space; all available space can be used
- Consistency in relative width/height
- An important concept in Responsive Design

Limitations:

- Little precise control over the width/height
- Possible display issues: text are too wide to be read comfortably, or too small for words to show up clearly on small screen
- Problems exist when a fixed width element is placed inside a liquid column



Wide Window



Narrow Window

Mobile Friendly Layout: why?

Different devices (desktop, tablet, phones) can lead to different display/control of webpages.

Display, Navigation, Scroll & Tab:

- Different screen sizes
- Orientation: portrait vs. landscape
- No mouse and no mouseover (hover)
- Swipe to scroll - horizontal and vertical
- Click / tab

Viewport in responsive web design

Viewport - The user's **visible area** of a web page

- Why do we need to set the viewport?
 - **Fixed-size** pages designed for computer screens were too large to fit the viewport of phones / tablets
- Set through <meta> tag: `<meta name="viewport" content="value">`
- The value can be
 - Fixed size: `content="width=320"`
 - Designated device: `content="width=device-width"`
 - Designated device & scale (most commonly used):
`content="width=device-width, initial-scale=1.0"`

Viewport example

`<meta name="viewport" content="width=device-width, initial-scale=1.0">`



Without the viewport meta tag



With the viewport meta tag

Source: https://www.w3schools.com/css/css_rwd_viewport.asp

Responsive Layout Design Principles

Responsive layout can respond to different **environments** (devices)

Key principles:

- **Liquid/fluid** layout: always use **relative sizes**

Responsive Layout Design (1)

Use **media query** to make it fully **responsive**

- Select different CSS files according to the media type

`<link rel="stylesheet" type="text/css" href="screen.css" media="screen">`

`<link rel="stylesheet" type="text/css" href="print.css" media="print">`

- CSS3 introduces more media query

- If the browser window is 600px or smaller

`@media only screen and (max-width: 600px) {...}`

- If the browser window is 768px or larger (mobile-first design)

`@media only screen and (min-width: 768px) {...}`

Responsive Layout Design (2)

Write up different CSS for different media and use media query to select suitable CSS

- If the browser window is 600px or smaller

```
<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 600px)" href="mobile.css">
```

- If the browser window is 768px or larger (mobile-first design)

```
<link rel="stylesheet" type="text/css" media="screen and (min-device-width: 768px)" href="desktop.css">
```

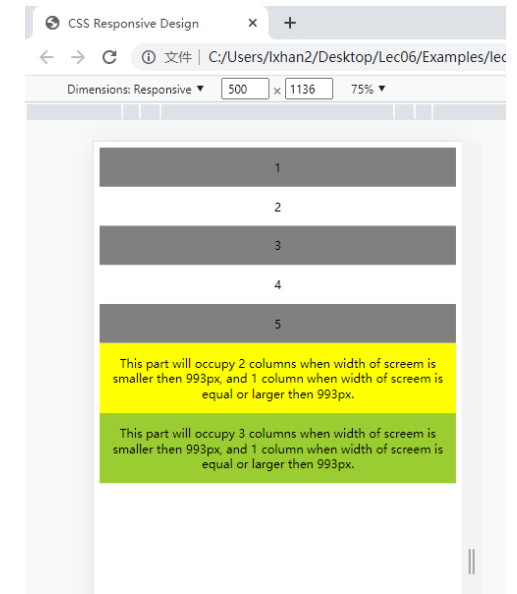
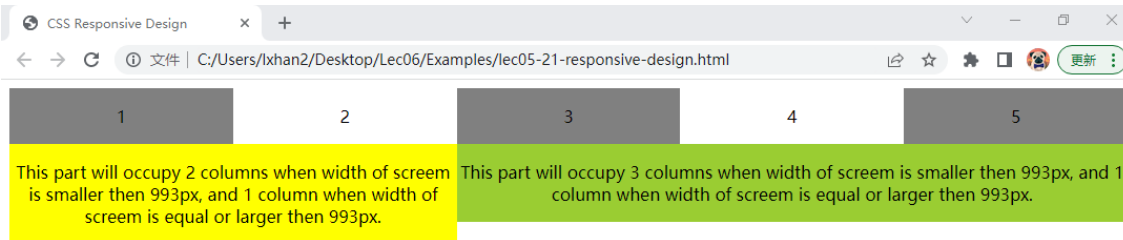
Responsive Layout Example

```
40 <body>
41   <div class="col m1 col-gray">
42     <p>1</p>
43   </div>
44   <div class="col m1">
45     <p>2</p>
46   </div>
47   <div class="col m1 col-gray">
48     <p>3</p>
49   </div>
50   <div class="col m1">
51     <p>4</p>
52   </div>
53   <div class="col m1 col-gray">
54     <p>5</p>
55   </div>
56
57   <div class="col m2 col-yellow">
58     <p>This part will occupy 2 columns when
59   </div>
60   <div class="col m3 col-blue">
61     <p>This part will occupy 3 columns when
62   </div>
63 </body>
```

```
13
14 .col {
15   width: 100%;
16   float: left;
17 }
18 @media (min-width: 993px) {
19   .col.m1 {
20     width: 20%;
21   }
22   .col.m2 {
23     width: 40%;
24   }
25   .col.m3 {
26     width: 60%;
27 }
```

If the width of screen is smaller than 993px

Use **media query** to select different CSS: if the width of screen is equal of larger than 993px



Outline

- Positioning
 - Four types of positioning
 - Review of float and clear
- Responsive Design
- CSS3 advanced techniques

CSS3 [Optional]

The latest version of CSS, only compatible with IE9 or above browsers.

Additional effects

- Round Corners
- Box Shadow
- Gradient
- Transform
- Transition
- Animation

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;

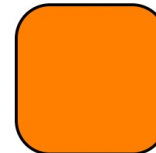
```
#rb24 {  
    border-radius: 24px;  
}  
  
#rb10 {  
    border-radius: 10px;  
}  
  
#percent {  
    border-radius: 5%;  
}
```

```
<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>
```

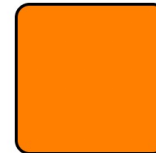
Basic



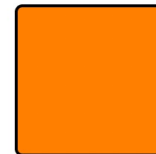
Round Corner - radius 24px



Round Corner - radius 10px



Round Corner - radius 5%



Critically thinking: What if we change the *border-radius* to 50%?

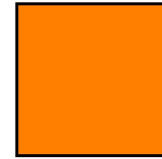
Round Border (2) [Optional]

In some old webpages, CSS3 properties can be specified as

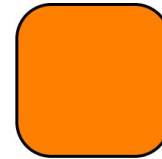
- *border-radius: 10px;* (official standard)
- *-webkit-border-radius: 10px;* (Safari, Chrome)
- *-moz-border-radius: 10px;* (Firefox)

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

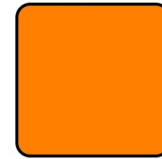
Basic



Round Corner - radius 24px



Round Corner - radius 10px



Round Corner - radius 5%



Box Shadow [Optional]

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5   #example1 {
6     border: 1px solid;
7     padding: 10px;
8     box-shadow: 5px 10px;
9   }
10
11   #example2 {
12     border: 1px solid;
13     padding: 10px;
14     box-shadow: 5px 10px #888888;
15   }
16
17   #example3 {
18     border: 1px solid;
19     padding: 10px;
20     box-shadow: 5px 10px red;
21   }
22 </style>
23 </head>
24 <body>
25
26 <h1>The box-shadow Property</h1>
27 <p>The box-shadow property defines the shadow of an element:</p>
28
29 <h2>box-shadow: 5px 10px:</h2>
30 <div id="example1">
31   <p>A div element with a shadow. The first value is the horizontal offset and the second v
32 </div>
33
34 <h2>box-shadow: 5px 10px #888888:</h2>
35 <div id="example2">
36   <p>You can also define the color of the shadow. Here the shadow color is grey.</p>
37 </div>
38
39 <h2>box-shadow: 5px 10px red:</h2>
40 <div id="example3">
41   <p>A red shadow.</p>
42 </div>
43
44 </body>
45 </html>
```

Code Example: lec07-15-CSS-box-shadow.html

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

The box-shadow Property

The box-shadow property defines the shadow of an element:

box-shadow: 5px 10px:

A div element with a shadow. The first value is the horizontal offset and the second value is the vertical offset. The shadow color will be inherited from the text color.

box-shadow: 5px 10px #888888:

You can also define the color of the shadow. Here the shadow color is grey.

box-shadow: 5px 10px red:

A red shadow.

Gradient [Optional]

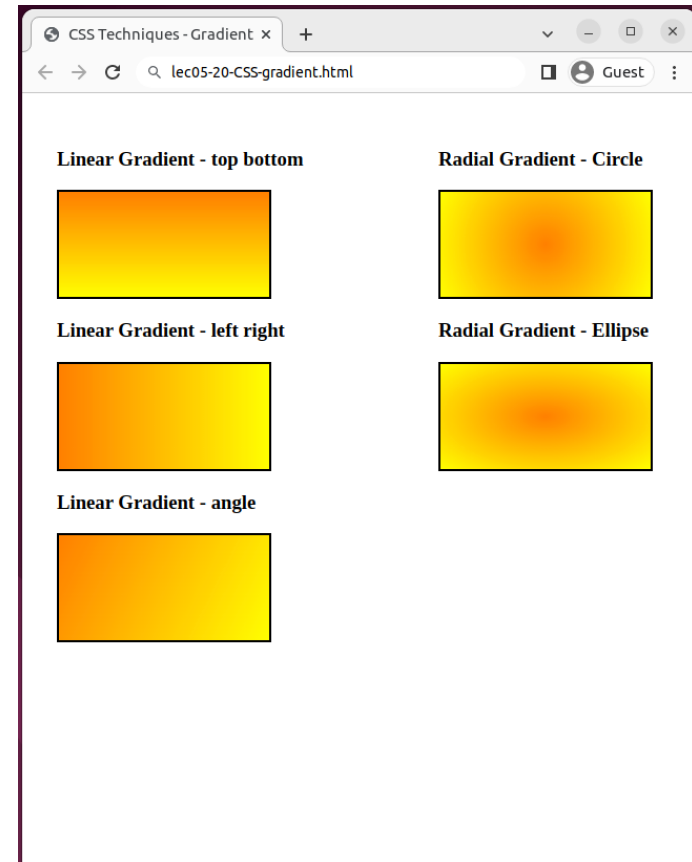
- In CSS2, it is not easy to create a color composed of **gradual change** of **two colors** which is called **gradient**, e.g.

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

- Specify 2 colors (start & end) and the browser would "**interpolate**" the colors in between. Note the result is a special type of image and therefore cannot be used as a value for any color related property
- **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);`

Gradient Example [Optional]

```
27 #g1 {
28 background: linear-gradient(top, #FF8000, #FFFF00);
29 background: -webkit-linear-gradient(top, #FF8000, #FFFF00);
30 background: -moz-linear-gradient(top, #FF8000, #FFFF00);
31 }
32 #g2 {
33 background: -webkit-linear-gradient(left, #FF8000, #FFFF00);
34 background: -moz-linear-gradient(left, #FF8000, #FFFF00);
35 }
36 #g3{
37 background: -webkit-linear-gradient(top left, #FF8000, #FFFF00);
38 background: -moz-linear-gradient(top left, #FF8000, #FFFF00);
39 }
40 #g4{
41 background: radial-gradient(circle, #FF8000, #FFFF00);
42 background: -webkit-radial-gradient(circle, #FF8000, #FFFF00);
43 background: -moz-radial-gradient(circle, #FF8000, #FFFF00);
44 }
45 #g5{
46 background: -webkit-radial-gradient(ellipse, #FF8000, #FFFF00);
47 background: -moz-radial-gradient(ellipse, #FF8000, #FFFF00);
48 }
49 </style>
50 </head>
51 <body>
52 <div id="left">
53 <h3>Linear Gradient - top bottom</h3>
54 <div class="com" id="g1">
55 </div>
56 <h3>Linear Gradient - left right</h3>
57 <div class="com" id="g2">
58 </div>
59 <h3>Linear Gradient - angle</h3>
60 <div class="com" id="g3">
61 </div>
62 </div>
63 <div id="right">
64 <h3>Radial Gradient - Circle</h3>
65 <div class="com" id="g4">
66 </div>
67 <h3>Radial Gradient - Ellipse</h3>
68 <div class="com" id="g5">
69 </div>
70 </div>
```



Transform [Optional]

- Transform and/or animation can create visually interesting effect for Web pages. As the property name implies, transform performs **transformation** on the **original style** (position) of an element to the **target style** (position)
- Transform is the property, and the value is the result of different **value functions**:
 - transform: translate(10px, 10px);
 - transform: scale(2, 1.5);
 - transform: rotate(45deg);
 - the default transform origin is at centre

```
40  
41  
42  
43  
44
```

```
#t {  
  transform: rotate(10deg);  
  -webkit-transform: rotate(10deg);  
  -moz-transform: rotate(10deg);  
}
```

Transform



Transition [Optional]

- Best understood as 2 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;  
}
```

- The property is background-color:
 - **initial state** is the original background color
 - **final state** is background color #004080
 - triggering event is **mouse over** (:hover)
 - **different from normal** :hover (i.e., without transition) is the gradual constant change (linear) over a period of 3 seconds rather than an instant change

Transition [Optional]

- The **linear value** is called **the speed curve** used to define the **speed of change** over the duration, can be:
 - **ease** - specifies a transition effect with a slow start, then fast, then end slowly (this is 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)** - lets you define your own values in a cubic-bezier function

Transition Example [Optional]

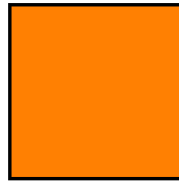
Before hover

Transition – background color

After hover

Transition – background color

```
#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);  
}
```

Transition – transform translate

Transition – transform translate

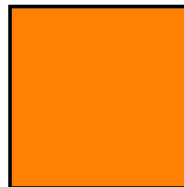


```
#t3 {  
  transition: transform 3s linear;  
  -webkit-transition: transform 3s linear;  
}  
#t3:hover {  
  transform: scale(1.5,1.5);  
  -webkit-transform: scale(1.5, 1.5);  
}
```

Transition – transform scale

Transition – transform scale

Transition – transform scale



Animation [Optional]

- 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)

```
52 #rt {  
53   animation-name: divRotate;  
54   animation-duration: 2s;  
55   animation-iteration-count: 5;  
56   -webkit-animation-name: divRotate;  
57   -webkit-animation-duration: 2s;  
58   -webkit-animation-iteration-count: 5;  
59 }  
60 @keyframes divRotate {  
61   from {  
62     transform: rotate(0deg);  
63     margin-left: 0px;  
64   }  
65   to {  
66     transform: rotate(360deg);  
67     margin-left: 150px;  
68   }  
69 }  
70 @-webkit-keyframes divRotate {  
71   from {  
72     -webkit-transform: rotate(0deg);  
73     margin-left: 0px;  
74   }  
75   to {  
76     -webkit-transform: rotate(360deg);  
77     margin-left: 150px;  
78   }  
79 }
```

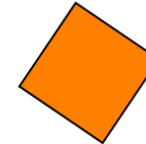
Animation - 5 times



Animation - 5 times



Animation - 5 times



Animation - 5 times



Personal Webpage (Version-08)

- Add a new **blog** page
 - Create “*blog.html*” and adopt a **similar layout** to the homepage
 - Add a **link** under Personal info to navigate back to the home page
 - Use an **unordered list** as a quick link to the blog
 - Add the **title**, **description**, **image** and **posted time** for each blog. Remember to add some proper styles to them respectively

