

CT214H – Web Programming Fundamentals

# Chapter 2

# Cascading Style Sheet (CSS)

Tran Cong An  
([tcan@cit.ctu.edu.vn](mailto:tcan@cit.ctu.edu.vn))

# Content

1. Introduction to CSS
2. CSS levels and syntax
3. CSS selectors
4. CSS declaration
5. CSS properties
6. Inheritance
7. The layout
8. The Box model
9. CSS convention

# Introduction to CSS

# Why CSS?

- HTML markup: used to represent
  - **Semantic:** e.g. <h1> **means** top-level heading
  - **Presentation:** <h1> elements **look** a certain way (*bold and big*)
- It is advisable to **separate** semantics from presentation:
  - It's easier to present documents on **multiple platforms** (browser, cellphone, ...)
  - It's easier to generate documents with **consistent look**
  - Semantic and presentation changes can be made **independently** of one another

**HTML was NEVER intended to contain tags  
for formatting a web page!**

# What is CSS?

- A language that **describes the style** of an HTML document *describes how HTML elements should be displayed*
- CSS **removed the style** formatting from the HTML page
- CSS benefits:
  - Easier to **maintain** and update
  - Greater **consistency** in design
  - More **formatting options**
  - **Lightweight** code
  - **Faster** download times
  - Ease of presenting different styles to **different viewers**, etc.



**Cons:** the compatibility with the browsers

# What is CSS?

Web page = HTML (structure) + CSS: presentation)

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h1>My First CSS Example</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

## My First CSS Example

This is a paragraph.



No CSS

```
<style>
  body {
    background-color: lightblue;
  }
  h1 {
    color: white;
    text-align: center;
  }
  p {
    font-family: verdana;
    font-size: 20px;
  }
</style>
```

## My First CSS Example

This is a paragraph.



With CSS

# CSS Levels and Syntax

# Levels of CSS

## 1. Inline:

- specified for a **specific occurrence of a tag** and apply only to that tag
- appear **in the tag** itself

## 2. Internal (document):

- apply to the **whole document** in which they appear
- appear in the **head of the document**

## 3. External:

- can be applied to **any number of documents** (*entire website*)
- defined in **separate files**



# Levels of CSS

## 1. **Inline:** uses the **style attribute** of the HTML element

- General form:

`style="property1: value1; property_2: value2;..."`

- Example:

```
<!DOCTYPE html>
<html>
  <body>
    <h1 style="color:blue;">This is a Blue Heading</h1>
  </body>
</html>
```

# Levels of CSS

## 2. Internal: defined in the **<head> section** of an HTML page, within a **<style>** element

- General form:

```
<style>  
    rule list  
</style>
```

- Example:



```
<html>  
  <head>  
    <style>  
      body {background-color: powderblue;}  
      h1   {color: blue;}  
      p    {color: red;}  
    </style>  
  </head>  
  <body>  
    <h1>This is a heading</h1>  
    <p>This is a paragraph.</p>  
  </body>  
</html>
```

# Levels of CSS

## 3. External:

- CSS rules are defined in a separate file (**.css file**)
- To use an external style sheet, add a link to it in the **<head>** section of the HTML page
- Example:

```
body {  
    background-color: powderblue;  
}  
h1 {  
    color: blue;  
}  
p {  
    color: red;  
}
```

CSS file

```
<!DOCTYPE html>  
<html>  
  <head>  
    <link rel="stylesheet"  
          href="styles.css">  
  </head>  
  <body>  
    <h1>This is a heading</h1>  
    <p>This is a paragraph.</p>  
  </body>  
</html>
```

HTML file

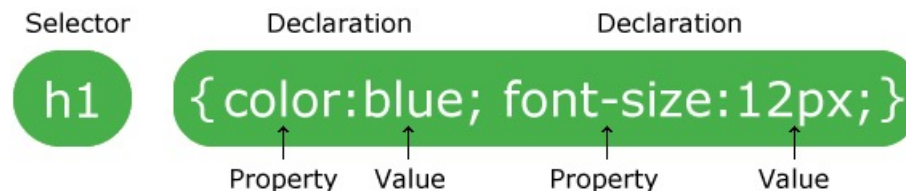
# Levels of CSS

## - CSS **conflict resolution**:

- When more than one style sheet applies to a specific tag in a document, the lowest level style sheet has precedence
  1. Inline CSS
  2. Internal CSS
  3. External CSS
  4. Browser default format
- In the same level, the **last CSS rule** will be applied

# CSS Syntax

- A **CSS block** (internal) or **CSS file** (external) consists of **CSS rules**
- A **CSS rule** consists of a **selector** and a **declaration block** surrounded by curly braces
- The **selector** “selects” HTML elements in the webpage
- A **declaration block** contains one or more **declarations**, separated by a colon
- Each **declaration** includes a **CSS property name** and a **value**



**Select and style**

# CSS Selectors

*Used to select element(s) to style*

# HTML-Element Selector

- Selects **all HTML elements** based on **element name** (HTML tag)
- Syntax: **HTML-tag** { **property**: value; ... }
- Example:

```
p {  
    text-align: center;  
    color: red;  
}
```

# Class Selector

- Selects elements with a specific **class attribute**
- Syntax: **.classname** { **property**: value; ... }

- Example:

```
.header {  
    font-weight: bold;  
}
```

- **Combination** of element and class selector:
  - Syntax: **HTML-tag.class-selector** (**no space**)
  - Selection: all HTML-tag elements that belong to the class specified by the class selector



# ID Selector

- Selects **one unique element** with a specific `id` attribute
- Syntax: `#id { property: value; ... }`
- Example:

```
#header {  
    font-weight: bold;  
}
```

# Attribute Selector

- Selects elements that **have specific attributes** or **attribute values**
- Syntax:
  - `[attribute]`: selects elements with specified attribute
  - `[attribute="value"]`: selects elements with a specified attribute and value
  - `[attribute~="word"]`: selects elements with an attribute value **containing** a specified **word** (*space-separated*)
  - `[attribute*="value"]`: select elements whose attribute value **contains** a specified value

# Attribute Selector

- Syntax (*cont.*):
  - `[attribute]="word"`: selects elements with the specified attribute **starting** with the specified **word**
  - `[attribute^="value"]`: selects elements whose attribute value **begins** with a specified **value**
  - `[attribute$="value"]`: selects elements whose attribute value **ends** with a specified **value**
- **Note:** this selector can be **combine** with the element selector
  - Syntax: `HTML-tag[attribute-selector]`

# Special Selectors

- **Descendant:** select elements that are **descendants** of (**inside**) a specified element (*ascendant element*)

- Syntax: ascendant-selector      descendant-selector



*white space character: space, tab,  
line feed, form feed, new line*

- Example:

```
ul li {  
  color: blue;  
}
```

- **Note:** use symbol **>** to select only the “direct” children of a specified element (*also called child selector*)

# Special Selectors

## - Pseudo-classes:

- Selects special elements that are not in the document tree
- Prefixed with “:”

Pseudo	Selection	Example
:link	all unvisited links	a:link
:visited	all visited links	a:visited
:hover	element with the mouse pointer hovering	a:hover (hovered link)
:focus	element which has the focus	input:focus (focusing input)
:active	the active element	a:active (active link)
:first-child	the first child of its parent	p:first-child (first child <p>)

- Some others: [https://www.w3schools.com/cssref/css\\_selectors.asp](https://www.w3schools.com/cssref/css_selectors.asp)

# Special Selector

- Universal selector:
  - Selects all elements in the webpage
  - Syntax: \*
  - Example: apply red color to all elements in the webpage

```
* {  
  color: red;  
}
```

# Combine Selectors

- A CSS selector can contain more than one simple selector
- Four CSS selector combinators:
  1. Descendant selector (space)
  2. Child selector (>)
  3. Adjacent sibling selector (+)
  4. General sibling selector (~)
- Example:

```
div#header p em.required {  
  color: white;  
}
```



# Group Selectors

- Group selectors:
  - To share the same declaration to multiple selectors (to save space)
  - Syntax: use comma (,) to separate the grouped selector
  - Example:

```
<style>
  a:hover, a:focus {
    background-color: green;
  }
  a:hover { color: red; }
  a:focus { color: yellow; }
</style>
```



# Cascade Rule

- Used to **solve the conflicts** occurring when there are two or more values for the same property on the same element
- **Sources** of conflict:
  - Conflicting values between **levels of style sheets**
  - Within one style sheet (overlapped selectors)
  - Inheritance can cause conflicts
  - Property values can come from style sheets written by the document author, the browser user, and the browser defaults

# Cascade Rule

- Cascade (specificity precedence) rules:
  - **ids** are more specific than **classes**
  - **classes** are more specific than **element names**
  - Style rules that **directly** target elements are more specific than style rules that are **inherited**
  - If elements have the same specificity, the **later rule** wins

```
div strong { color: red; }  
strong { color: blue; }  
  
<div>  
  <strong>What color am I?</strong>  
</div>
```

```
strong { color: red; }  
strong { color: blue; }  
  
<div>  
  <strong>What color am I?</strong>  
</div>
```

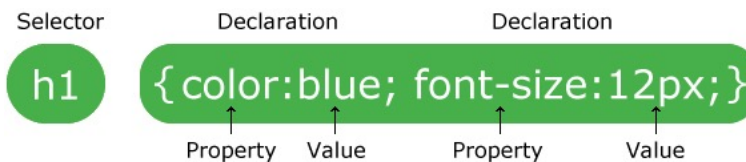
# CSS Declaration

*Used to apply styles to selected elements*

# CSS Declaration

- **CSS declaration:** `{ property: value; ...}`
- **CSS properties:**
  - CSS3: more than 70 properties
  - CSS4: 129 properties
  - Categories (vary):
    - Fonts, lists, alignment, margins, colors, backgrounds, borders, positioning, etc.
    - Animations, backgrounds, box model, flexbox, grid, positioning, transitions, typography, etc.

## CSS rule



# CSS Declaration

- **CSS property values:** vary in forms
  - Keywords: left, right, small, large, etc.
  - Number (*with no space*)
  - Unit: px (*pixel*), in (*inches*), cm, mm, pt (*points*), pc (*picas, 12 pts*), em (*height of the letter 'm'*), ex (*height of the letter 'x'*), % (*percentage*), vw (*view port width*)
  - Color:
    - Name (140 color names)
    - Hex values, RGB (*red, green, blue*), RGBA (*A: alpha channel*), HLS (*hue, saturation, lightness*), HSLA
  - Special values: normal (*default value of the property*), inherit (*from parent element*), auto (*browser default value*)

# CSS Basic Properties

*Font, background, text, list, table*

# Font

Property	Description
font	Setting all properties for font in one declaration (font-style, font-variant, font-weight, font-size/line-height, font-family, etc.)
font-family	A prioritized list of [generic] font family names for an element
font-size	Sets the size of a font (absolute or relative), default: 16px/1em
font-style	Sets the style (normal   italic   oblique) for a text
font-variant	Specifies whether a text should be displayed in small-cap (normal   <b>small-caps</b> )
font-weight	Sets the thickness of a font (normal   bold   bolder   lighter   <u>number</u> )

Some other font properties: font-size-adjust (*Firefox only*), font-stretch

# Background

Property	Value
background	bg-color bg-image position/bg-size bg-repeat bg-origin bg-clip bg-attachment (e.g. background:url(smiley.gif) 10px 20px/50px 50px;)
background-color	color-value (color name   rgb   rgba   hsl   hsla)
background-image	image-url
background-position	position-name (e.g. left top   left center   etc.)   x% y% (relative)   xpos ypos (absolute)
background-size	<b>auto</b> (origin size)   <u>length</u> (width height)   cover   contain
background-repeat	<b>repeat</b>   repeat-x   repeat-y   no-repeat
background-origin	<b>padding-box</b>   border-box   content-box
background-clip	Painting area: <b>border-box</b>   padding-box   content-box
background-attachment	<b>scroll</b> (with the page), fixed, local (scroll with the element content)



# Text

Property	Value
color	color-value (color name   rgb   rgba   hsl   hsla)
text-align	left   right   center   justify
text-decoration	line (none   underline   overline   line-through) color style (solid   double   dotted   dashed   wavy)
text-indent	length (fixed or relative)
text-justify	auto   inter-word   inter-character   none
text-overflow	clip   ellipsis   <u>string</u> (to represent the clipped text)
text-shadow	h-shadow v-shadow [blur-radius] color (e.g. 2px 2px 1px black)
text-transform	none   capitalize   uppercase   lowercase

- Other properties: direction, line-height, letter-spacing, word spacing, etc.

# List

Property	Values
list-style	list-style-type list-style-position list-style-image
list-style-type	none   disc   circle   square   decimal   decimal-leading-zero   lower-roman   upper-roman   lower-alpha   upper-alpha   <i>etc.</i>
list-style-position	inside   outside
list-style-image	none   url(...)
list-style-type	auto   length

# Table

Property	Values
border	border-width border-style border-color
border-width	top right bottom left (name or value, e.g. thin medium thick 10px;)
border-style	top right bottom left (none   hidden   dotted   dashed   solid   double   groove   ridge   inset   outset)
border-color	top right bottom left
border-collapse	separate   collapse
border-spacing	<i>length</i>   <i>h-length v-length</i>
caption-side	top   bottom
empty-cell	show   hide
table-layout	auto (cells' width belongs to the content)   fixed (equal column width if the column width is not set; otherwise, uses the width of the 1 <sup>st</sup> row)

# Inheritance

# Inheritance

- CSS styles are inherited from parent to child

Instead of selecting all elements individually

```
a, h1, p, strong {  
    font-family: Helvetica;  
}
```

You can style the parent and the children will inherit the styles

```
body {  
    font-family: Helvetica;  
}
```

You can style the parent and the children will inherit the styles

```
h2, h3 {  
    font-family: Consolas;  
}
```

# Exception in Inheritance

- While many CSS styles are inherited from parent to child, not all CSS properties are inherited

```

<a href="/home">
  Back to <em>Home</em>
</a>
+
a {
  display: block;
  font-family: Arial;
}

```

<em> inherits the **font-family** property, but not the **display**



Back to Home

- There's **no rule** for what properties are inherited or not: the inheritance behavior defined in the **CSS specification**

Initial value	inline
Applies to	all elements
Inherited	no

Default value:	?
Inherited:	no

<https://developer.mozilla.org/en-US/docs/Web/CSS/display>

# User Agent Styles

```
<h1>Chocolate</h1>
<p>
  <a href="...">CSS</a> is wonderful
</p>
```

+

```
body {
  color: red;
  font-family: Helvetica;
}
```

**Chocolate**CSS is wonderful**Chocolate**CSS is wonderful

?  
<a>  
inherits  
font-family,  
but not the  
color!

- This is because the browser has its own default styles
  - Browser loads **its own default stylesheet** on every webpage

```
<html>
  <head>
    <title>User Agent Style</title>
    <link rel="stylesheet" href="user-agent-style.css"/>
    ...
  </head>
```

# User Agent Styles

- So, to style the `<a>` links, we need to **override** the browser default style by **explicitly setting** a color:

```
<h1>Chocolate</h1>
<p>
  <a href="...">CSS</a> is wonderful
</p>
```



**Chocolate**

CSS is wonderful

+

```
body {
  color: red;
  font-family: Helvetica;
}

a {
  color: red;
}
```



**Chocolate**

CSS is wonderful



# The Layout

*Block vs. inline elements, element grouping, overflow, display modes, visibility, floating and positioning*

# Block and Inline Elements

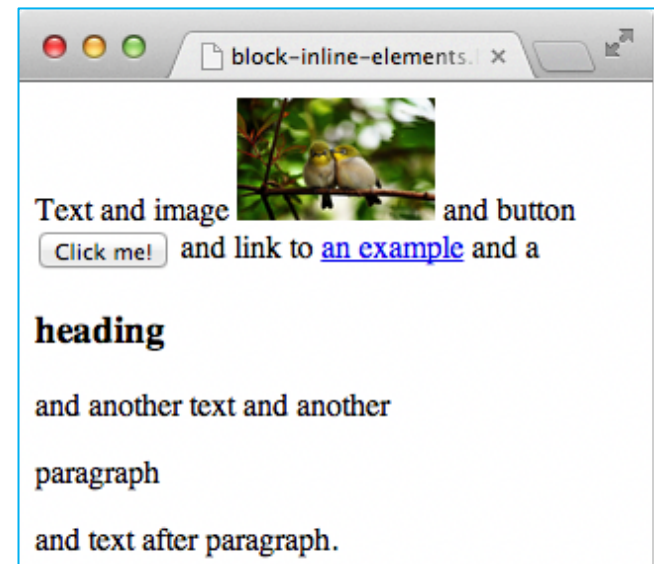
- The browser treats every element as a box
- There are 2 types of boxes: **block** and **inline**
- **Block elements:**
  - Always start on a **new line**
  - Fill up the vertically from top to bottom on the page
  - Example: `<p>`, `<h1>` to `<h6>`, `<div>`, `<ul>`, `<ol>`, `<pre>`, etc.
- **Inline elements:**
  - Doesn't start on a new line, i.e. one can **sit next to others**
  - Doesn't respect to the top and bottom margins/paddings
  - Example: `<img>`, `<a>`, `<input>`, `<span>`, etc.

# Inline and Block Elements

```

<html>
  <body>
    <p>Text and image 
      and a button <input type="button" value="Click me!">
      and a link to <a href="list.html">
        an example</a>
      and a
    <h3>heading</h3>
    and another text
    and another
    <p>paragraph</p>
    and text after paragraph.
  <p>
</body>
</html>

```



# Inline and Block Elements

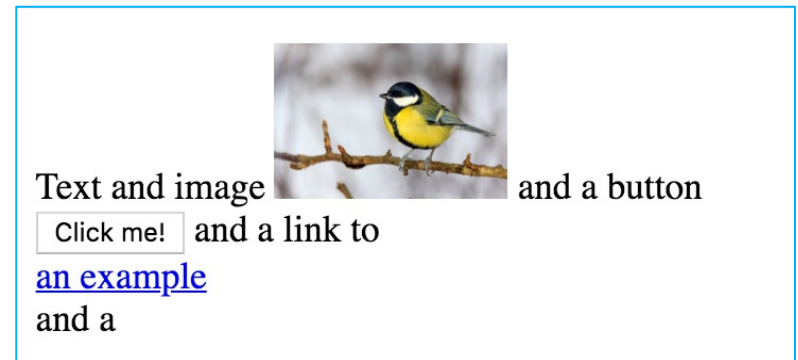
- To change the display mode of an element:

- Use the 'display' property:

`display: block | inline`

- Example:

```
a {  
    display: block;  
}
```



```
<p>Text and image   
and a button <input type="button" value="Click me!">  
and a link to <a href="list.html">an example</a>  
<p>
```

# Grouping Elements

- Grouping elements lets us **style multiple elements** as a whole
- **Container elements** are used to group elements:
  - **<span>**: **inline** container

```
<span>Hello <a href= "... ">World</a>!</span>
```
  - **<div>**: **block** container

```
<div>Hello <a href= "... ">World</a>!</div>
```
- **Width** and **height** properties can be used to resize the block and `<img>` elements

# Overflow

- The content of an element may **overflows** (bigger) its container
- The **overflow**, **overflow-x**, **overflow-y** properties are used to specify what happens
- Possible values for these properties:
  - **visible**: the overflow is not clipped and rendered outside the element's box
  - **hidden**: the overflow is clipped, the rest content is visible
  - **scroll**: the overflow is clipped but a scrollbar is added
  - **auto**: if the overflow is clipped, a scrollbar will be added

# Overflow

```
<div style="height:10px; background-color:yellow;">Hello</div>
```

```
<br>
```

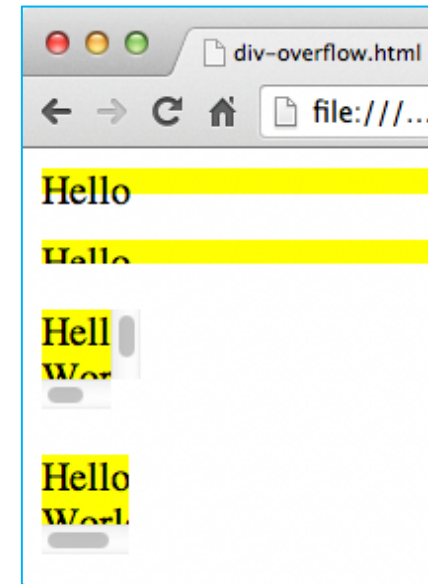
```
<div style="width:40px; height:10px;  
    background-color:yellow;  
    overflow:hidden;"> Hello</div>
```

```
<br>
```

```
<div style="width:40px; height:40px;  
    background-color:yellow;  
    overflow:auto;">Hello World</div>
```

```
<br>
```

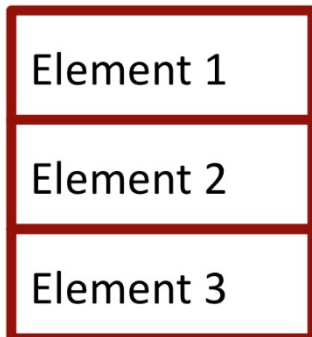
```
<div style="width:40px; height:40px;  
    background-color:yellow;  
    overflow-x:auto;  
    overflow-y:hidden;">Hello World</div>
```



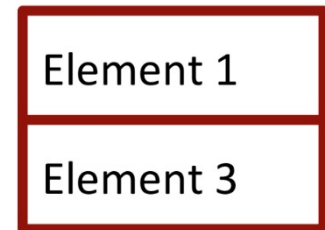
# Display

- The 'display' property specifies how an element is displayed
- Possible value: inline, block, list-item (same as block), etc.

```
<span id="s1">Element 1</span>  
<span id="s2">Element 2</span>  
<span id="s3">Element 3</span>
```



```
#s1, #s2, #s3 {  
  display: block;  
}
```



```
#s1, #s3 {display: block;}  
#s2 {display: none;}
```



```
#s1, #s2, #s3 {display: inline;}
```

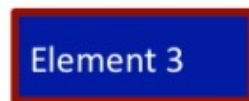


# Visibility

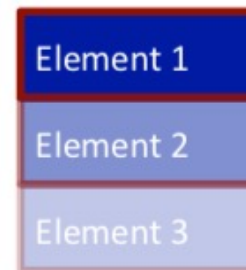
- The '**visibility**' property specifies whether an element is visible
  - Value: visible, hidden, collapse (used only with table row/column)
- The '**opacity**' property sets the opacity level for an element
  - Value: from 0.0 (fully transparent) to 1.0 (fully opaque)
- Note that, hidden elements (set with display property) still take up space on the page



```
#s1, #s3 {display: block;}  
#s2 {display: none;}
```



```
#s1, #s2, #s3 {display: block;}  
#s2 {visibility: hidden;}
```



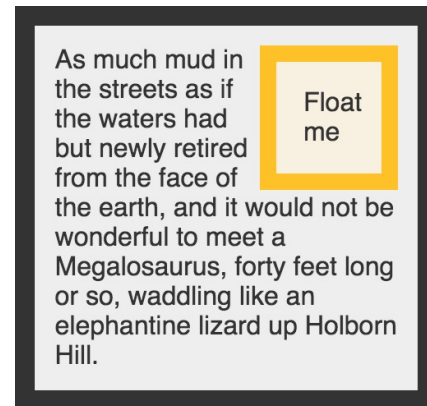
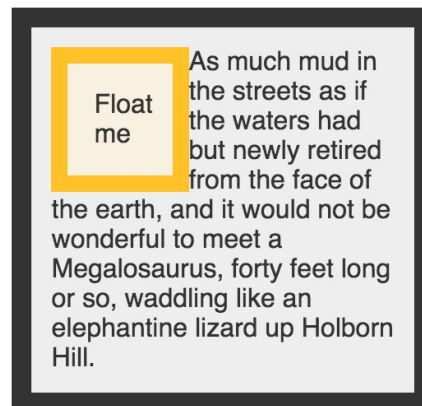
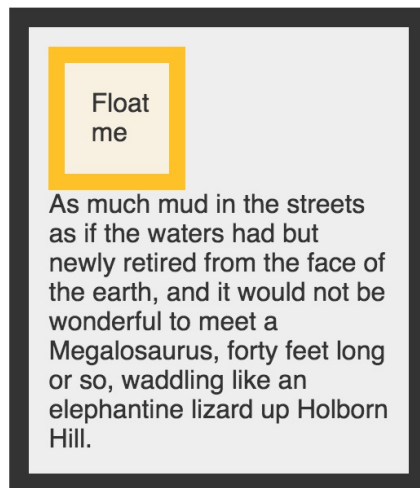
```
#s1 {opacity: 1;}
```

```
#s1 {opacity: 0.5;}
```

```
#s1 {opacity: 0.25;}
```

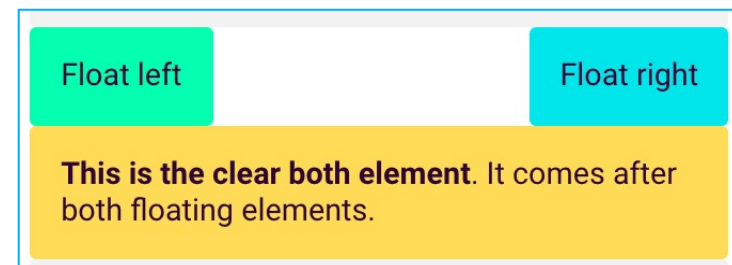
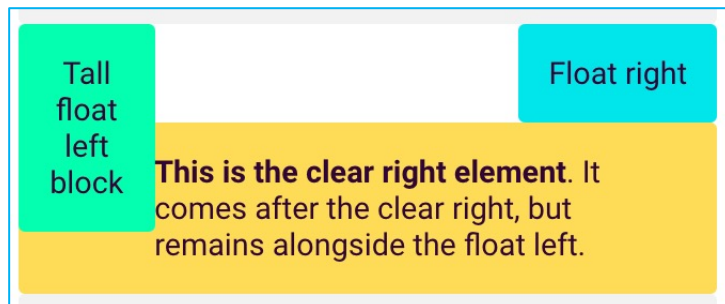
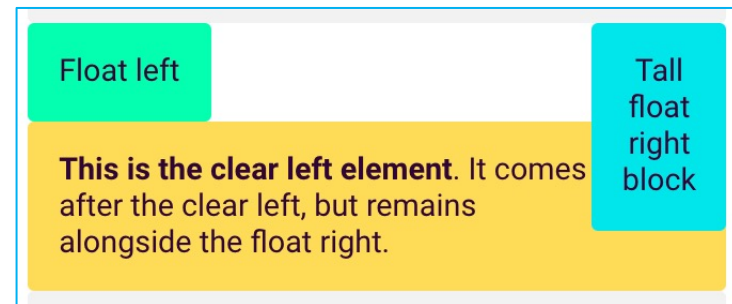
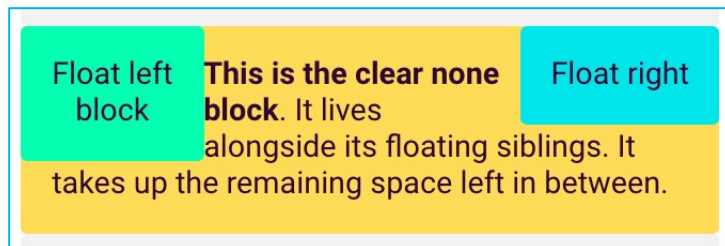
# Floating

- The 'float' property is used to **float** boxes on the sides of other boxes and the siblings will **wrap around** the floating element
- Values:
  - none (default): the element doesn't float
  - left: the element floats to the left of its container
  - right: the element floats to the right of its container



# Floating

- The 'clear' property is used to specify that an element should **not wrap around** above floated elements
- **Value: none** (wrap floating elements on both sides), **left** (doesn't wrap the left element), **right** (doesn't wrap the right element), **both** (doesn't wrap both left and right)




# Floating

The screenshot shows a web browser window with the title 'floating.html'. The address bar shows the file path: `file:///localhost/Users/tcan/Dropbox/B...`. The page content is as follows:


## Phong Lan

**float: left;**



Họ Phong lan là một họ thực vật có hoa, thuộc bộ Măng tây, lớp thực vật một lá mầm.

Họ Phong lan là một trong những họ lớn nhất của thực vật và có các thành viên mọc trên toàn thế giới.



**float: right;**

**clear: right;**

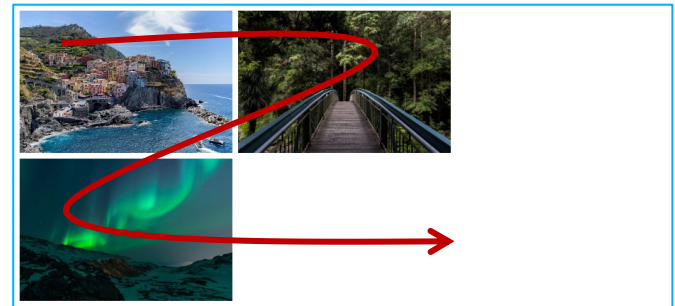
Các loài lan chủ yếu mọc trên cây cao, sống biểu sinh lâu năm. Chúng được gọi chung là Phong lan. Bên cạnh đó cũng có các loài mọc trong đất, tức là Địa lan và có một số loài mọc trên đá tức Thạch lan.

**clear: left;**

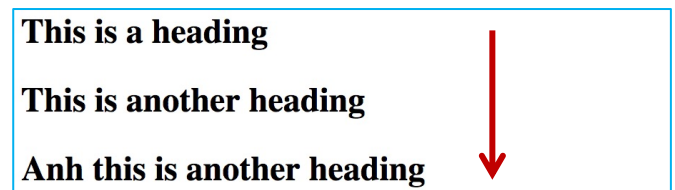
# Positioning

- In normal flow (default),
  - **inline elements** flow from **left to right**, wrapping to the next line when needed
  - **block elements** flow from **top to bottom** (new line after every element)

```
<body>  
    
    
    
</body>
```



```
<body>  
  <h2>This is a heading</h2>  
  <h2>This is another heading</h2>  
  <h2>Anh this is another heading</h2>  
</body>
```



# Positioning

- The '**position**' property can be used to specify the type of positioning method applied for an element
  - **static**: the element is positioned according to the normal flow (element's position isn't affected by the top, bottom, left, and right properties)
  - **relative**: the element is positioned relative to its normal position (using the top, bottom, left and right properties)
  - **fixed**: the element is positioned relative to the browser window
  - **absolute**: The element is positioned relative to its first positioned (not static) ancestor element (if there is no such element, <html> is used)
  - **sticky**: the element will be “stuck” when its position reaches a threshold value (top, bottom, left, or right)

# Position

```
div.relative {  
  position: relative;  
  left: 30px;  
  width: calc(100% - 30px);  
  border: 3px solid #73AD21;  
}
```

## position: relative;

An element with position: relative; is positioned relative to its normal position:

This element has position: relative;

```
div.relative {  
  position: relative;  
  width: 300px;  
  height: 100px;  
  border: 3px solid #73AD21;  
}  
  
div.absolute {  
  position: absolute;  
  top: 30px;  
  right: 0;  
  width: 150px;  
  height: 50px;  
  border: 3px solid #73AD21; }
```

## position: absolute;

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

This div element has position: relative;

This div element has position: absolute;

# The Box Model



# The Box Model

- Each element can be considered as a **box** in the page layout
- The box model is essentially a box that wraps around an element consisting of margins, borders, paddings, and content



# The Box Model – The Margin

- The clears an area outside the border
- The margin is transparent
- The 'margin' property is used to set the size of the margin
  - `margin: <top> [<right> [<bottom> [<left>]]]>`
  - `margin-top/right/bottom/left: <value> | auto`
- If the margin is `auto` and the box's width is fixed, the margin will be set as large as possible

# The Box Model – The Border

- A border goes around the padding and content
- A border has a color, style, and the thickness
- To set the border properties:
  - `border`: `[thickness]` `<style>` `[color]`
  - `border-width`: `<value>` | `<top>` `[right]` `[bottom]` `[left]`
  - `border-style`: `none` | `hidden` | `dotted` | `dashed` | `solid` | `double`  
| `groove` | `ridge` | `inset` | `outset`
  - `border-color`: `<color>` | `<top>` `[right]` `[bottom]` `[left]`

# The Box Model – The Border

`width:340px; margin:auto; border-style: solid`

`width:340px; margin-left:auto; margin-right: 5px;`

`border-style:dashed; border-color:cyan yellow green blue;`

`border-style:groove;`

`border-style:outset;`

`border-style:inset;`

`border-style:ridge;`

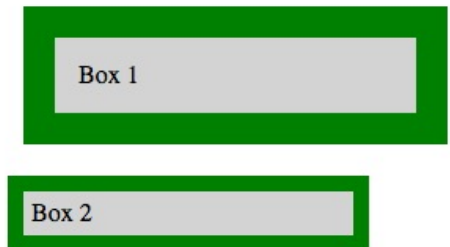
# The Box Model – The Padding

- Padding clears an area around the content
- The padding is transparent (affected by the content background color)
- To set the padding property:
  - `padding: size | <top> [right] [bottom] [left]`
  - `padding-top/right/bottom/left: value`

```
.box1 {  
  background-color: lightgrey;  
  width: 200px;  
  border: 20px solid green;  
  padding: 15px;  
  margin: 20px;  
}
```

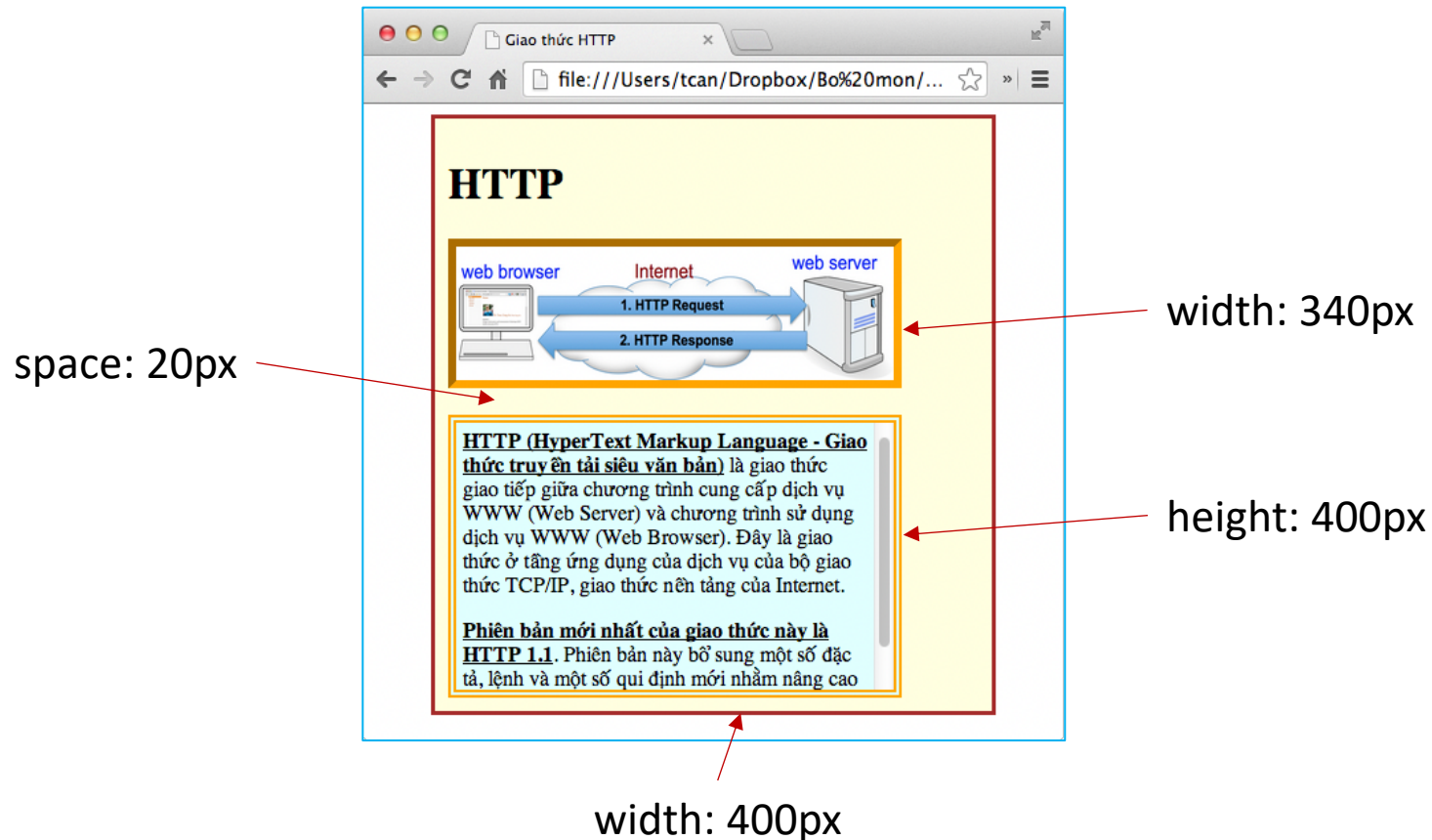
```
.box2 {  
  background-color: lightgrey;  
  width: 200px;  
  border: 10px solid green;  
  padding: 5px;  
  margin: 10px;  
}
```

## Demonstrating the Box Model



# The Box Model – Exercise

- Design the following webpage:



# A Typical Page Layout



# A Typical Page Layout

```
<div class="header">
  <h1>Header</h1>
  <p>Resize...</p>
</div>
```

## Header

```
<div class="topnav">
  <a href="#">Link</a>
  <a href="#">Link</a>
  <a href="#">Link</a>
  <a href="#" class="navright">Link</a>
</div>
```

Resize the browser window to see the responsive effect.

Link Link Link

Link

## Side

Lorem ipsum dolor sit amet, consectetur adipiscing elit..

## Main Content

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique. Quisque vehicula, risus eget aliquam placerat, purus leo tincidunt eros, eget luctus quam orci in velit. Praesent scelerisque tortor sed accumsan convallis.

## Side

Lorem ipsum dolor sit amet, consectetur adipiscing elit..

```
<div class="column middle">
  <h2>Main Content</h2>
  <p>Lorem ipsum ....</p>
  <p>Lorem ipsum ...</p>
</div>
```

```
<div class="column side">
  <h2>Side</h2>
  <p>Lorem ipsum ...</p>
</div>
```

## Footer

```
<div class="footer">
  <h2>Footer</h2>
</div>
```



# A Typical Page Layout

```
/* Style the header */
```

```
.header {  
  background-color: #f1f1f1;  
  padding: 20px;  
  text-align: center;  
}
```

```
/* Style the right top nav links */
```

```
a.rightnav{  
  float: right;  
}
```

```
/* Style the top navigation bar */
```

```
.topnav {  
  overflow: hidden;  
  background-color: #333;  
}
```

```
/* Style the topnav links */
```

```
.topnav a {  
  float: left;  
  display: block;  
  text-align: center;  
  padding: 14px 16px;  
  text-decoration: none;  
}
```

```
/* Change color on hover */
```

```
.topnav a:hover {  
  background-color: #ddd;  
  color: black;  
}
```

# A Typical Page Layout

```
/* Create three unequal columns */
```

```
.column {  
    float: left;  
    padding: 10px;  
}
```

```
/* Left and right column */
```

```
.column.side {  
    width: 25%;  
}
```

```
/* Middle column */
```

```
.column.middle {  
    width: 50%;  
}
```

```
/* Clear floats after the columns */
```

```
.row:after {  
    content: "";  
    display: table;  
    clear: both;  
}
```

```
/* Footer */
```

```
.footer {  
    padding: 20px;  
    text-align: center;  
    background: #ddd;  
    margin-top: 20px;  
}
```

# CSS Convention

# Coding Convention

- A CSS rule can be written as follow:

```
selector { property: value; ... }
```

- However, it is recommended to use the following style:

```
selector {  
    property: value;  
    ...  
}
```

- Using comments in CSS  
is also recommended:

```
/* comment (multiple lines) */
```

```
/* Style the header */  
.header {  
    background-color: #f1f1f1;  
    padding: 20px;  
    text-align: center;  
}
```

# Naming Convention

- Class name or ID should describe the semantic, not format
  - Example: use `warning` instead of `redbox`
- Names are written in lowercase
- Words are separated by a hyphen (-)
  - Example: `header-info` (instead of `headerInfo`)
- Use prefix for subclass:
  - Example: `footer footer-logo, footer-copyright`

# Appendix

# Further Readings

- SCSS (*CSS extension language: <https://sass-lang.com>*) = CSS + ...
  - ... variables
  - ... nesting
  - ... partial files + import
  - ... Operators
  - ... etc.
- CSS properties (more advanced properties)
- CSS Animation
- Flexbox (\* CSS4)

# Resources

- CSS tutorial:
  - <https://www.w3schools.com/css/default.asp>
  - <https://www.tutorialspoint.com/css/>
  - <https://sass-lang.com/guide>: SCSS basics
- CSS references:
  - <https://www.w3schools.com/cssref/default.asp>
  - <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
- CSS tools:
  - <https://html-css-js.com/css/generator/>: CSS code generator
  - <https://webcode.tools/css-generator/>



# Question?

*Chapter 2 – Cascading Style Sheet*