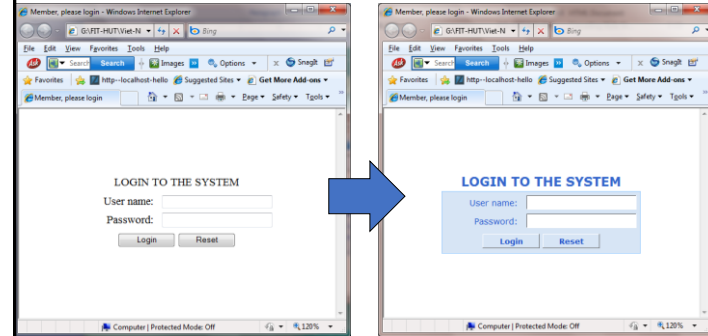


# IT4552 – Web programming

## Chapter 8. CSS

1

## Before and after using CSS



2

login.html

- ❖ ...
- ❖ `<link rel="stylesheet" type="text/css" href="style.css">`
- ❖ ...
- ❖ `<table class="forumline" width="280" border="0" cellpadding="2">`
- ❖ `<tr class="formstyle"><td>`

style.css

```
body
{
    font-family: Verdana,Tahoma,Geneva,Arial,Helvetica,sans-serif;
    font-size: 12px; ...
}

.formstyle
{
    background-color: #D7E5F5;
    font-family: Verdana,Tahoma,Geneva,Arial,Helvetica,sans-serif;
    font-size: 12px;
}

.forumline
{
    background-color: ...
}
...
```

3

## Content

- ➡ 1. Introduction to CSS
2. Specifying and applying style rules
3. Style class
4. Some useful properties
5. CSS box model

4

## 1. Introduction to CSS

- ❖ Cascading Style Sheet
- ❖ Created by Hakon Wium Lie of MIT in 1994
- ❖ Has become the W3C standard for controlling visual presentation of web pages

5

## 1.1. Benefits of CSS

- ❖ Simple syntax: easy to learn
- ❖ Powerful and flexible way to specify the formatting of HTML elements
  - Can define font, size, background color, background image, margins, etc
- ❖ Separates presentation (design elements) from content (structural logic)
  - HTML contains content and structure of a web page.
  - CSS defines a style of a web page – how the content is displayed
- ❖ Improved output flexibility (responsive design)

6

## 1.1. Benefits of CSS (2)

- ❖ Share style sheets across multiple documents or entire Web site
  - Easy to maintain consistent pages
  - Can update a common style → Reflected in all pages that use it
- ❖ Cost Savings
  - Reduced Bandwidth Costs
    - One style sheet called and cached
    - CSS require less code
  - Higher Search Engine Rankings
    - Cleaner code is easier for search engines to index
    - Greater density of indexable content

7

## 1.2. CSS Basics

- ❖ CSS defines the way that HTML elements should be presented:
  - Positioning (e.g. left, right or centered)
  - Font (size and family)
  - Text decoration (e.g. underlined)
  - Borders (solid, dashed, invisible)
  - Image usage (e.g. for backgrounds and bullets)

8

### 1.3. CSS Does Not...

- ❖ Re-order HTML
  - E.g. won't sort a table
- ❖ Perform calculations
  - Won't sum a shopping basket
- ❖ Filter
  - Won't decide what to show
  - Though JavaScript can set display or visibility of elements in order to achieve this
- ❖ These can all be done on the server
  - Or using XSLT or JavaScript on the client

### 1.4. Types of CSS Styles

- ❖ (Browser default)
- ❖ External styles
  - written in a separate document and then attached to various Web documents
  - External style sheets can affect any document they are attached to
- ❖ Internal styles (embedded styles)
  - embedded in the head of the document.
  - embedded styles affect only the tags on the page they are embedded in
- ❖ Inline Style
  - written directly in the tag on the document

### Content

1. Introduction to CSS
- ➡ 2. Specifying and applying style rules
3. Style class
4. Some useful properties
5. CSS box model

### 2.1. Specifying Style Rules

- ❖ General form of rule

```
selector { property: value }
```
- Or

```
selector { property1: value1;
           property2: value2;
           ...
           propertyN: valueN }
```
- ❖ Example

```
H1 { text-align: center;
      color: blue }
```



## 2.1. Specifying Style Rules (2)

- ❖ The *selector* is the link between the HTML document and the style. It specifies what elements are affected by the declaration.
- ❖ The *declaration* is that part of the rule that sets forth what the effect will be

selector      declaration

H1 {color: blue;}

property      value

## 2.1. Specifying Style Rules (3)

- ❖ Grouping selectors and rules

```
H1 { font-weight: bold }
H2 { font-weight: bold }
H3 { font-weight: bold }
→H1, H2, H3 { font-weight: bold }
→What is different?
b i{background-color:yellow;}
b,i{color:blue;}
```
- ❖ A selector may have more than one declaration

```
H1 { color: green }
H1 { text-align: center }
```

## 2.1. Specifying Style Rules (4)

- ❖ Values
  - The unit of any given value is dependent upon the property.
  - Some property values are from a predefined list of keywords. Others are values such as length measurements, percentages, numbers without units, color values, and URLs.

### Colors

- Name
- RGB
- Hexadecimal
- RGBa
- HSL

### Relative

- px
- em
- %
- vw,vh

### Absolute

- In
- cm
- Pt

## 2.2. Applying styles to the document

- ❖ Inline style
  - Apply a style sheet to an individual element using the **style attribute**
- ❖ Embedded style
  - Apply the basic, document-wide style sheet for the document by using the **style element**
- ❖ External style
  - Link an external style sheet to the document using the **link element** or
  - Import a style sheet using the CSS @import notation.

### 2.2.1. Inline style

- ❖ Using Style attribute
- ❖ For individual elements

```
<H1 STYLE="color: blue; font-size: 20pt;">
  A large purple Heading
</H1>
```

17

### 2.2.2. Embedded style

- ❖ Using Style element
- ❖ Putting the style sheet inside a style element at the top of your document

```
<HTML>
  <HEAD><TITLE>Bach's home page</TITLE>
    <STYLE> H1, H2 { color: green } </STYLE>
  </HEAD>
  <BODY>
    <H1>Bach's home page</H1>
    <P>Johann Sebastian Bach was a prolific composer. Among his works are:
    <UL> <LI>the Goldberg Variations
        <LI>the Brandenburg Concertos
        <LI>the Christmas Oratorio </UL>
    <H2>Historical perspective</H2>
    <P>Bach composed in what has been referred to as the Baroque period.
  </BODY>
</HTML>
```

18

### 2.2.2. Embedded style (2)

```
<STYLE type="text/css">
  <!--
    H1, H2 { color: green }
  -->
</STYLE>
```

#### Bach's home page

Johann Sebastian Bach was a prolific composer. Among his works are:

- the Goldberg Variations
- the Brandenburg Concertos
- the Christmas Oratorio

#### Historical perspective

Bach composed in what has been referred to as the Baroque period.

19

### Tree structures and inheritance

- ❖ Just as children inherit from their parents, HTML elements inherit stylistic properties.
- ❖ CSS property values set on one element will be transferred down the tree to its descendants

```
<STYLE TYPE="text/css">
  BODY { color: green }
</STYLE>
```



20

## Overriding inheritance

- ❖ Sometimes children don't look like their parents.

- ❖ E.g.

```
<STYLE TYPE="text/css">
  BODY { color: green }
  H1 { color: navy }
</STYLE>
```



21

## 2.2.3. External style

- ❖ Using Link element
- ❖ This is true "separation" of style and content.
- ❖ Keeping all your styles in an external document is simpler

```
<HEAD>
<LINK REL="stylesheet" TYPE="text/css" HREF="styles/mystyles.css">
</HEAD>
```

/\* mystyles.css - a simple style sheet \*/

```
body {
  margin-left: 10%;
  margin-right: 10%;
  color: black;
  background: white;
}
```



22

## Content

1. Introduction to CSS
2. Specifying and applying style rules
- ➡ 3. Style class
4. Some useful properties
5. CSS box model



23

## 3.1. Element Style Classes

- ❖ Proceed the HTML element by a period and a class name

```
// Define an "abstract" paragraph type
P.abstract { margin-left: 0.5in;
             margin-right: 0.5in;
             font-style: italic }
```

- ❖ To use, supply the name of the style class in the CLASS attribute of the HTML element

```
<H1>New Advances in Physics</H1>
<P CLASS="abstract">
```

This paper gives the solution to three previously unsolved problems: turning lead into gold, antigravity, and a practical perpetual motion machine.



24

### 3.2. Global Style Classes

- ❖ omit the element name

```
// Style available to all elements
.blue { color: blue; font-weight: bold }
```

- ❖ To use, simply specify the style class in the CLASS attribute of the HTML element

```
<H2 CLASS="blue">A Blue Heading</H2>
<!-- Apply to a section of text -->
This text is in the default color, but
<SPAN CLASS="blue">this text is
blue.</SPAN>
```



### 3.3. Styles through User-Defined IDs

- ❖ An ID is like a class but can be applied only once in a document

```
<HEAD>
  <TITLE>...</TITLE>
  <STYLE TYPE="text/css">
    <!--
      #foo { color: red }
    -->
  </STYLE>
</HEAD>
<BODY>
  ...
  <P ID="foo">
    ...
  </BODY>
```



### 3.4. Attribute Selectors

- ❖ An **attribute selector** provides a way to select HTML elements either by the presence of an element attribute or by the value of an attribute

- [title] { ... }



### 3.4. Attribute Selectors (2)

```
<head>
  <meta charset="utf-8">
  <title>Share Your Travels</title>
  <style>
    [title] {
      cursor: help;
      padding-bottom: 3px;
      border-bottom: 2px dotted blue;
      text-decoration: none;
    }
  </style>
</head>
<body>
  <div>
    
    <h2><a href="countries.php?id=CA" title="see posts from Canada">Canada</a></h2>
    <p>Canada is a North American country consisting of ... </p>
    <div>
      
      
      
    </div>
  </div>
```



### 3.4. Attribute Selectors (3)

#### ❖ Attribute selector

Selector	Matches
[]	A specific attribute.
[=]	A specific attribute with a specific value.
[~=]	A specific attribute whose value matches at least one of the words in a space delimited list of words.
[^=]	A specific attribute whose value <b>begins</b> with a specified value.
[*=]	A specific attribute whose value <b>contains</b> a substring.
[\$=]	A specific attribute whose value ends with a specified value.

### 3.5. Pseudo-Element and Pseudo-Class Selectors

- ❖ A **pseudo-element selector** is a way to select something that does not exist explicitly as an element in the HTML document tree but which is still a recognizable selectable object.
- ❖ A **pseudo-class selector** does apply to an HTML element, but targets either a particular state or, in CSS3, a variety of family relationships.

- a:link
- a:visited
- :focus
- :hover
- :active
- :checked
- :first-child
- :first-letter
- :first-line

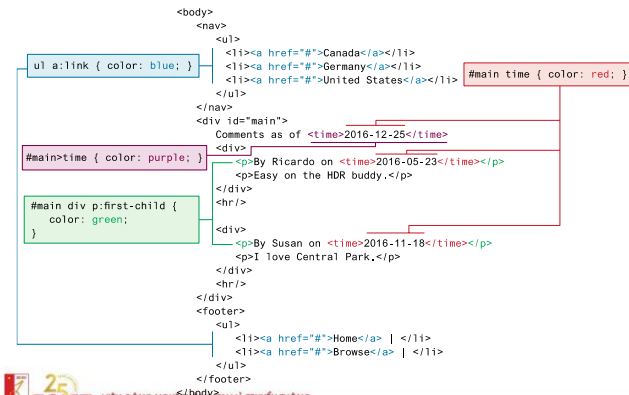
### 3.6. Contextual Selectors

#### ❖ Contextual Selectors

Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element.	<b>div p</b> Selects a <p> element that is contained somewhere within a <div> element.
Child	A specified element that is a direct child of the specified element.	<b>div&gt;h2</b> Selects an <h2> element that is a child of a <div> element.
Adjacent Sibling	A specified element that is the next sibling (i.e., comes directly after) of the specified element	<b>h3~p</b> Selects the first <p> after any <h3>.
General Sibling	A specified element that shares the same parent as the specified element.	<b>h3~p</b> Selects all the <p> elements that share the same parent as the <h3>.

### 3.6. Contextual Selectors

#### ❖ Contextual Selectors





## Content

1. Introduction to CSS
2. Specifying and applying style rules
3. Style class
- ⇒ 4. Some useful properties
5. CSS box model



## 4.1. Useful Font Properties

### ❖ font-weight

- Relative weight (boldness) of font
  - **normal** | lighter | bold | bolder | 100 | 200 | ... | 900
- ```
H1 { font-weight : 200 }
H2 { font-weight : bolder }
```

### ❖ font-style

- Font face type within a family
  - **normal** | italic | oblique
- ```
P { font-style : normal }
TH { font-style : italic }
```



## 4.1. Useful Font Properties (2)

### ❖ font-size

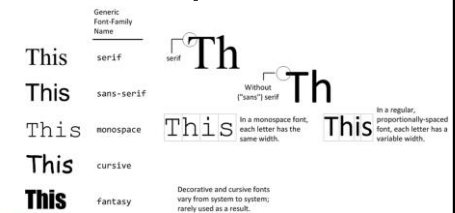
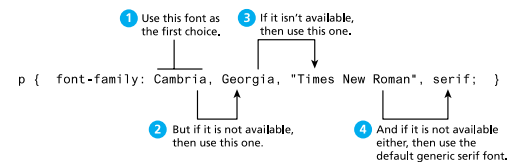
- Either relative or absolute size of font
  - pt, pc, in, cm, mm | em, ex, px, % | xx-large | x-large | large | **medium** | small | x-small | xx-small | smaller | larger
- ```
STRONG { font-size: 150% }
P { font-size: 14pt }
P { font-size: xx-large }
```

### ❖ font-family

- Typeface family for the font
- ```
H1 { font-family: Arial }
```



## 4.1. Useful Font Properties (2)



## 4.2. Useful Text Properties

### ❖ text-decoration

- Describes text additions or “decorations” that are added to the text of an element
- **none** | underline | overline | line-through | blink
- E.g. `P { text-decoration: underline }`

### ❖ vertical-align

- Determines how elements are positioned vertically
- top | bottom | **baseline** | middle | sub | super | text-top | text-bottom | %

### ❖ text-align

- Determines how paragraphs are positioned horizontally
- left | right | center | justify

## 4.2. Useful Text Properties (2)

### ❖ text-indent

- Specifies the indentation of the *first line of the paragraph*
- +/- pt, pc, in, cm, mm | +/- em, ex, px, %
- E.g. `P { text-indent: -25px } /* Hanging indent */`

### ❖ line-height

- Specifies the distance between two consecutive baselines in a paragraph
- **normal** | number | pt, pc, in, cm, mm | em, ex, px, %
  - `.double { line-height: 200% }`
  - `.triple { line-height: 3 } /* 3x the font size */`
  - `DIV { line-height: 1.5em }`

## 4.3. Useful Foreground and Background Properties

### ❖ color

- Color of the text or foreground color
- color-name | #RRGGBB | #RGB | rgb(rrr, ggg, bbb) | rgb(rrr%, ggg%, bbb%)
  - `P { color : blue }`
  - `H1 { color : #00AABB }`
  - `H3 { color : rgb(255, 0, 0 ) } /* red */`

### ❖ background-image

- Specifies an image to use as the background of region
- **none** | url(filename)
  - `H2 { background-image: url(Bluedrop.gif) ; }`

## 4.3. Useful Foreground and Background Properties (2)

### ❖ background-repeat

- Specifies how to tile the image in the region
- **repeat** | repeat-x | repeat-y | norepeat
  - `BODY {`
    - `background-image: url(Bluedot.gif) ;`
    - `background-repeat: repeat-x ;`
  - `}`

### ❖ background

- Lets you combine properties in a single entry
  - `P { background: url(wallpaper.jpg) repeat-x }`

## Content

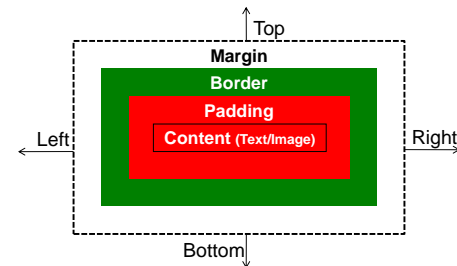
1. Introduction to CSS
2. Specifying and applying style rules
3. Style class
4. Some useful properties
- ➔ 5. CSS box model



41

## 5. CSS Box Model

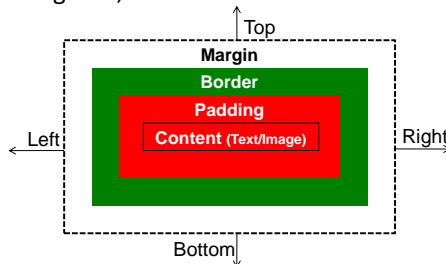
- ❖ Each HTML element have the rectangular “box”
- ❖ Each box has a content area and optional surrounding padding, border and margin area



42

## CSS Box Model - example

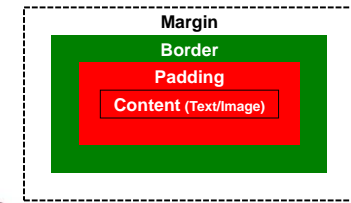
```
div#boxtest {  
  background-color: red; color: white;  
  padding: 1em;  
  border: 1em solid green;  
  margin: 1em;  
}
```



43

## CSS Box Model - color

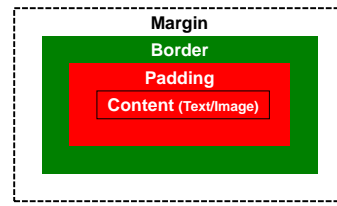
- ❖ Padding - same as the element's background-color
- ❖ Border - may have its own color (border-color property)
- ❖ Margin - always transparent (same as its ancestor's background-color)



44

## CSS Box Model - edge sizes

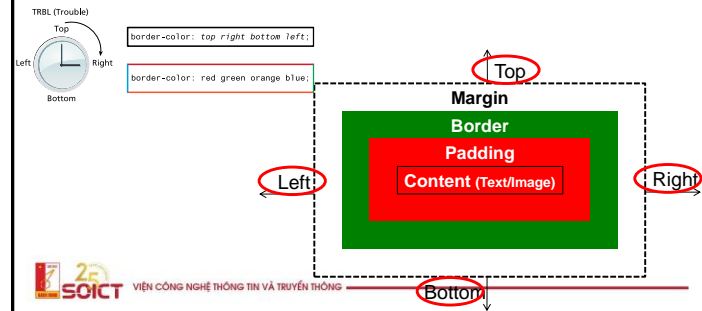
- ❖ Inner edge (Black line): Content itself or CSS width and height property may define the size
- ❖ Padding edge (Red): IE + padding width
- ❖ Border edge (Green): PE + border width
- ❖ Outer edge (Dotted black line): BE + margin width



45

## CSS Box Model – width (1)

- ❖ margin, padding, border-width
  - Define the width for all directions at once
- ❖ margin-top, padding-top, border-top-width
  - Define the width for each specific direction
  - top, right, left, bottom



46

## CSS Box Model – width (2)

- ❖ Effective values for box width
- ❖ <length> - e.g. 10pt, 3px, 1.2em
  - Effective for border, padding, margin
- ❖ <percentage> - e.g. 10%
  - Effective only for padding, margin
  - Calculated with respect to the width of the generated box's containing block
- ❖ Thin, medium, thick
  - Effective only for border

47

## Border properties

- ❖ border-width or border-top-width (top, right, left, bottom)
  - Specify the line width
- ❖ border-color or border-top-color (top, right, left, bottom)
  - Specify the line color by the color name or RGB values
- ❖ border-style or border-top-style (top, right, left, bottom)
  - Specify the line style of box's border
  - Values: solid, dotted, dashed, double, groove, ridge, inset, outset, hidden
  - Special value "none" means width 0
- ❖ border or border-top (top, right, left, bottom)
  - shorthand property for setting the width, style, and color
  - e.g. "border: 1em solid black;"

48

## TIPS: Before your experiment of box model

- ❖ Web browsers define their own default margin and padding width for some elements
- ❖ To override them, insert this CSS code at first

reset.css

```
html, body, div, span, h1, h2, h3, h4, h5, h6, p {  
  margin: 0;  
  padding: 0;  
  border: 0;  
  font-size: 100%;  
  vertical-align: baseline;  
}
```

```
* {  
  margin: 0;  
  padding: 0;  
}
```



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

49

## Content

1. Introduction to CSS
2. Specifying and applying style rules
3. Style class
4. Some useful properties
5. CSS box model

➔ 6. CSS: Layout



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

50

50

## Normal Flow

To understand CSS positioning and layout, it is essential that we understand this distinction as well as the idea of **normal flow**:

how the browser will normally display block-level elements and inline elements from left to right and from top to bottom



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

51

## Normal Flow

- **Block-level elements** such as <p>, <div>, <h2>, <ul>, and <table> are each contained on their own line.
- **Inline elements** do not form their own blocks but instead are displayed within lines.



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

52

## Normal Flow

### Block-Level Elements



Each block exists on its own line and is displayed in normal flow from the browser window's top to its bottom.

By default each block-level element fills up the entire width of its parent (in this case, it is the <body>, which is equivalent to the width of the browser window).

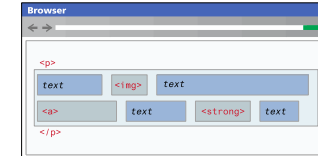
You can use CSS box model properties to customize, for instance, the width of the box and the margin space between other block-level elements.

53

## Normal Flow

### Inline Elements

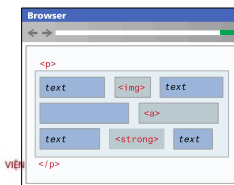
```
<p>
  This photo  of Conservatory Pond in
  <a href="http://www.centralpark.com/">Central Park</a> New York City
  was taken on October 22, 2015 with a <strong>Canon EOS 300</strong>
  camera.
</p>
```



Inline content is laid out horizontally left to right within its container.

Once a line is filled with content, the next line will receive the remaining content, and so on.

Here the content of this <p> element is displayed on two lines.



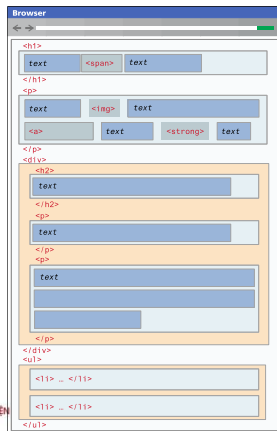
If the browser window resizes, then inline content will be "reflowed" based on the new width.

Here the content of this <p> element is now displayed on three lines.

54

## Normal Flow

### Block and Inline Elements



A document consists of block-level elements stacked from top to bottom.

Within a block, inline content is horizontally placed left to right.

Some block-level elements can contain other block-level elements (in this example, a <div> can contain other blocks).

In such a case, the block-level content inside the parent is stacked from top to bottom within the container (<div>).

55

## Positioning Elements

- **absolute** The element is removed from normal flow and positioned in relation to its nearest positioned ancestor.
- **fixed** The element is fixed in a specific position in the window even when the document is scrolled.
- **relative** The element is moved relative to where it would be in the normal flow.
- **static** The element is positioned according to the normal flow. This is the default.

56

## Positioning Elements

### Relative Positioning



```
<p>A wonderful serenity has taken possession of my ...
```

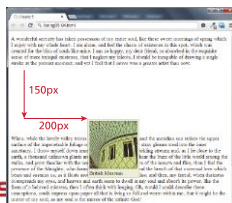
```
<figure>
```

```
  
```

```
  <figcaption>British Museum</figcaption>
```

```
</figure>
```

```
<p>When, while the lovely valley ...
```



```
figure {
```

```
  border: 1px solid #A8A8A8;
```

```
  background-color: #EDEDDE;
```

```
  padding: 5px;
```

```
  width: 150px;
```

```
  position: relative;
```

```
  top: 150px;
```

```
  left: 200px;
```

```
}
```

THÔNG

57

## Positioning Elements

### Absolute Positioning



```
<p>A wonderful serenity has taken possession of my ...
```

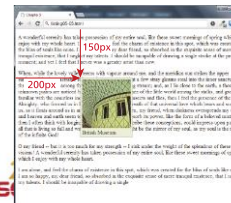
```
<figure>
```

```
  
```

```
  <figcaption>British Museum</figcaption>
```

```
</figure>
```

```
<p>When, while the lovely valley ...
```



```
figure {
```

```
  margin: 0;
```

```
  border: 1px solid #A8A8A8;
```

```
  background-color: #EDEDDE;
```

```
  padding: 5px;
```

```
  width: 150px;
```

```
  position: absolute;
```

```
  top: 150px;
```

```
  left: 200px;
```

```
}
```

THÔNG

58

## Positioning Elements

### Absolute Positioning is relative to nearest positioned ancestor



```
<p>A wonderful serenity has taken possession of my ...
```

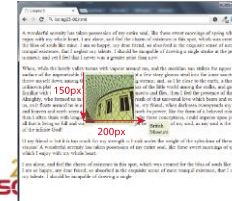
```
<figure>
```

```
  
```

```
  <figcaption>British Museum</figcaption>
```

```
</figure>
```

```
<p>When, while the lovely valley ...
```



```
figure {
```

```
  margin: 0;
```

```
  border: 1px solid #A8A8A8;
```

```
  background-color: #EDEDDE;
```

```
  padding: 5px;
```

```
  width: 150px;
```

```
  position: absolute;
```

```
  top: 150px;
```

```
  left: 200px;
```

```
}
```

```
figcaption {
```

```
  background-color: #EDEDDE;
```

```
  padding: 5px;
```

```
  position: absolute;
```

```
  top: 150px;
```

```
  left: 200px;
```

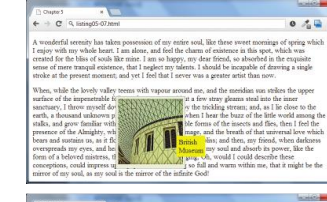
```
}
```

THÔNG

59

## Positioning Elements

### Z-Index



```
figure {
```

```
  position: absolute;
```

```
  top: 150px;
```

```
  left: 200px;
```

```
}
```

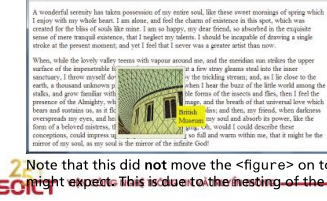
```
figcaption {
```

```
  position: absolute;
```

```
  top: 90px;
```

```
  left: 140px;
```

```
}
```



```
figure {
```

```
  ...
```

```
  z-index: 5;
```

```
}
```

```
figcaption {
```

```
  ...
```

```
  z-index: 1;
```

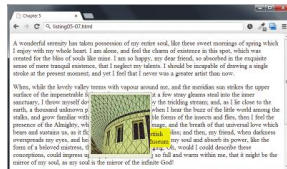
```
}
```

Note that this is not the same as the `<figure>` on top of the `<figcaption>` as one might expect. This is due to the nesting of the caption within the figure.

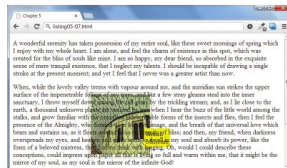
60

## Positioning Elements

### Z-Index



Instead the `<figcaption>` z-index must be set below 0. The `<figure>` z-index could be any value equal to or above 0.



If the `<figure>` z-index is given a value less than 0, then any of its positioned descendants change as well. Thus both the `<figure>` and `<figcaption>` move underneath the body text.

```
figure {
  ...
  z-index: 1;
}
figcaption {
  ...
  z-index: -1;
}
```

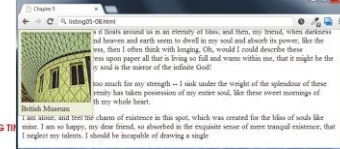
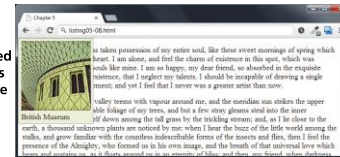
```
figure {
  ...
  z-index: -1;
}
figcaption {
  ...
  z-index: 1;
}
```

## Positioning Elements

### Fixed Position

```
figure {
  ...
  position: fixed;
  top: 0;
  left: 0;
}
```

Notice that figure is fixed in its position regardless of what part of the page is being viewed.



61

62

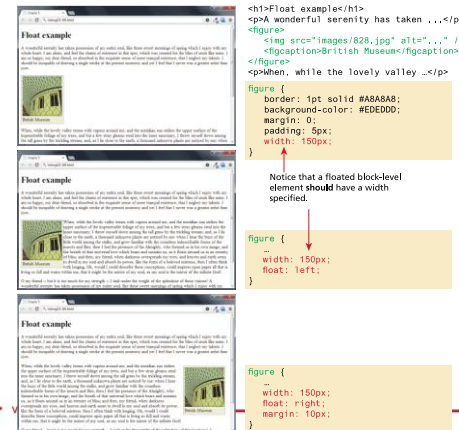
## Floating Elements

It is possible to displace an element out of its position in the normal flow via the CSS `float` property

- An element can be floated to the left or floated to the right .
- it is moved all the way to the far left or far right of its containing block and the rest of the content is “reflowed” around the floated element

63

## Floating Elements



Notice that a floated block-level element should have a width specified.

```
figure {
  width: 150px;
  float: left;
}
```

```
figure {
  width: 150px;
  float: right;
  margin: 10px;
}
```

64



## Floating Elements

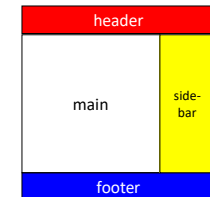
- **Clear property**
  - **left** The left-hand edge of the element cannot be adjacent to another element.
  - **right** The right-hand edge of the element cannot be adjacent to another element.
  - **both** Both the left-hand and right-hand edges of the element cannot be adjacent to another element.
  - **none** The element can be adjacent to other elements.

65

## Page layout with CSS box and div element

- ❖ Typical page layout with four regions
  - main, header, footer, sidebar
- ❖ Enclosed by div elements with id attributes

```
<div id="header">
  <p>Header content</p>
</div>
<div id="sidebar">
  <p>Sidebar content</p>
</div>
<div id="main">
  <h1>Main content</h1>
</div>
<div id="footer">
  <p>Footer content</p>
</div>
```

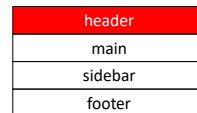


66

## Layout (1): header

- ❖ Reset default margin and padding to 0
- ❖ Specify header's property

```
* {
  margin: 0;
  padding: 0;
}
body {
  background-color: white;
  color: black;
}
div#header {
  background-color: red;
  color: white;
}
```

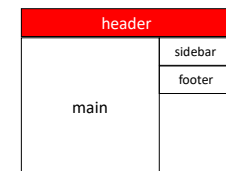


67

## Layout (2): main

- ❖ Specify main region's properties
- ❖ Set its height and shift to left side

```
div#main {
  float: left;
  height: 400px;
}
```



68

- ❖ Specify sidebar's properties
- ❖ Set its height and shift to right side
- ❖ Restrict sidebar's width to 25% of the parent

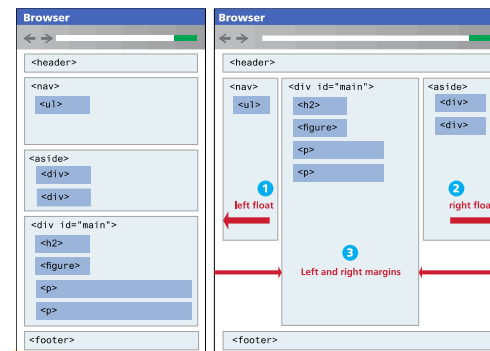
header		
main	foo- ter	side- bar

- ❖ Specify footer's properties
- ❖ Use "clear: both;" property
  - not be adjacent to an earlier floating box

The diagram shows a rectangular layout divided into four colored regions. At the top is a red horizontal bar labeled 'header'. Below it, the space is split into two vertical sections: a large white section on the left labeled 'main' and a smaller yellow section on the right labeled 'side-bar'. At the bottom is a blue horizontal bar labeled 'footer' that spans the entire width of the page.

1. Introduction to CSS
2. Specifying and applying style rules
3. Style class
4. Some useful properties
5. CSS box model
6. CSS: Layout
7. Multicolumn Layout

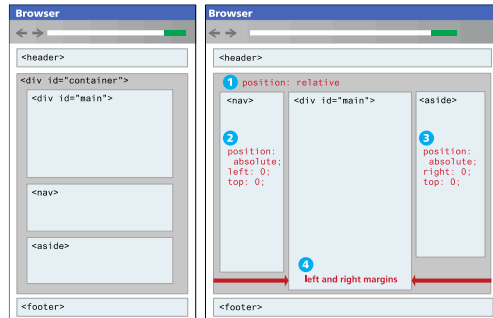
### 3 column example



18

## Constructing Multicolumn Layout

Using Positioning to Create Columns

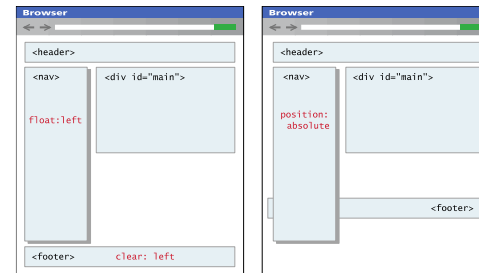


VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

73

## Constructing Multicolumn Layout

Problems with Absolute positioning



Elements that are floated leave behind space for them in the normal flow. We can also use the `clear` property to ensure later elements are below the floated element.

Absolute positioned elements are taken completely out of normal flow, meaning that the positioned element may overlap subsequent content. The `clear` property will have no effect since it only responds to floated elements.

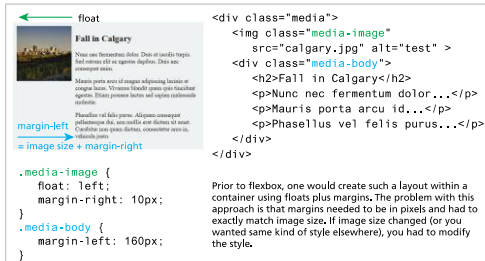


VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

74

## Constructing Multicolumn Layout

Using Flexbox to Create Columns

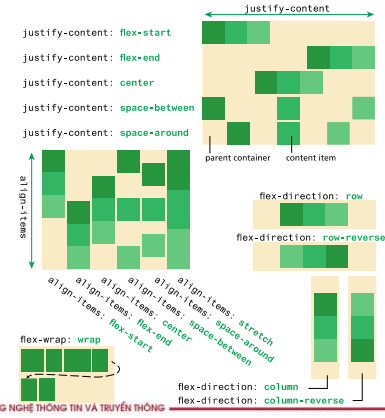


VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

75

## Constructing Multicolumn Layout

The flexbox parent (container) properties



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

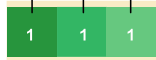
76

## Constructing Multicolumn Layout

The flexbox child (item) properties

flex-grow: 1  
flex-shrink: 1  
flex-basis: auto

flex: 1 1 auto  
These can be combined into the shorthand property instead



When the flex-grow value of each item is greater than 0, then each item will grow equally to fill the parent container.



Defines the growth factor of an element relative to the other items.

width=n  
width=n × 2

flex-basis: 200px



Defines the default size of the element before the remaining space is distributed.



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

77

## Responsive Design

Responsive Layouts

### Thiết kế đáp ứng

- Mục tiêu của thiết kế web responsive là làm cho mỗi trang web hoặc ứng dụng web xuất hiện như thể nó được thiết kế riêng cho từng thiết bị và trình duyệt mà nó được hiển thị trên đó.
- Dựa vào việc sử dụng CSS, truy vấn phương tiện (media query) và JavaScript để điều chỉnh việc trình bày nội dung cho các thiết bị.

### Media Queries của CSS3, cho phép kết hợp một style sheet với các đặc điểm hiển thị hoặc phương tiện khác nhau.

- Ví dụ: một style sheet có thể được chọn dựa trên chiều cao, chiều rộng, tỷ lệ co và độ phân giải màn hình của thiết bị.



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

78

## Responsive Design

4 elements

- Liquid layouts
- Setting viewports via the <meta> tag
- Customizing the CSS for different viewports using media queries
- Scaling images to the viewport size



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

79

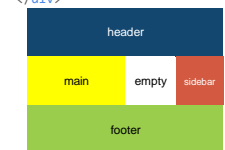
## Responsive Design

CSS Grid

### CSS Grid

- CSS Grid cho phép các nhà phát triển tạo một Grid Container xác định kích thước và hình dạng của lưới 2 chiều và cho phép đặt tên cho các khu vực trong lưới.

```
.item-a {
  grid-area: header;
}
.item-b {
  grid-area: main;
}
.item-c {
  grid-area: sidebar;
}
.item-d {
  grid-area: footer;
}
.container {
  display: grid;
  grid-template-columns: 50px 50px 50px;
  grid-template-rows: auto;
  grid-template-areas: "header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
```



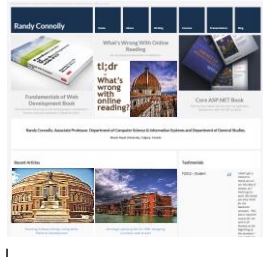
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

80

## Responsive Design

### Setting Viewports

- 1 Mobile browser renders web page on its viewport



- 2 It then scales the viewport to fit within its actual physical screen

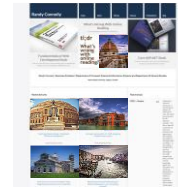


VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

81

## Responsive Design

### Setting Viewports



`<meta name="viewport" content="width=device-width" />`

- 1 Mobile browser renders web page on its viewport and because of the `<meta>` setting, makes the viewport the same size as the pixel size of screen.

- 2 It then displays it on its physical screen with no scaling.



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

82

## Responsive Design

### Setting Viewports

```
<meta name="viewport"
content="width=device-width, initial-scale=1">
```

- name=viewport:** "Trình duyệt, tôi sẽ cho biết tôi muốn khung nhìn trông như thế nào."
- width=device-width:** "Chiều rộng của khung nhìn phải luôn bắt đầu bằng chiều rộng của thiết bị."
- initial-scale=1:** "Bắt đầu ở mức thu phóng 100%."



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

83

## Responsive Design

### Media Queries

A media query is a way to apply style rules based on the medium that is displaying the file

Defines this as a media query

Device has to be a screen

CSS rules to use if device matches these conditions

```
@media only screen and (max-width:480px) { ... }
```

Only use this style if both conditions are true

Use this style if width of viewport is no wider than 480 pixels



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

84

## Responsive Design

### Media Queries

- **width** Width of the viewport
- **height** Height of the viewport
- **device-width** Width of the device
- **device-height** Height of the device
- **orientation** Whether the device is portrait or landscape
- **color** The number of bits per color

85

## Responsive Design

### Media Queries

```
/*Smart phone nhỏ*/
@media screen and (min-width: 240px) {

}

/*Iphone(480 x 640)*/
@media screen and (min-width: 320px) {

}

/*Tablet nhỏ(480 x 640)*/
@media screen and (min-width: 480px) {

}

/*Ipad dọc(768 x 1024)*/
@media screen and (min-width: 768px) {

}

/*Ipad ngang(1024 x 768)*/
@media screen and (min-width: 1024px) {

}
```

86

## Responsive Design

### Media Queries

```
/*Smart phone nhỏ*/
@media screen and (min-width: 240px) {

}

/*Iphone(480 x 640)*/
@media screen and (min-width: 320px) {

}

/*Tablet nhỏ(480 x 640)*/
@media screen and (min-width: 480px) {

}

/*Ipad dọc(768 x 1024)*/
@media screen and (min-width: 768px) {

}

/*Ipad ngang(1024 x 768)*/
@media screen and (min-width: 1024px) {

}
```

87

## Responsive Design

### Media Queries



```
styles.css

/* rules for phones */
@media only screen and (max-width:480px)
{
  #slider-image { max-width: 100%; }
  #flash-ad { display: none; }
  ...
}

/* CSS rules for tablets */
@media only screen and (min-width: 481px)
and (max-width: 768px)
{
  ...
}

/* CSS rules for desktops */
@media only screen and (min-width: 769px)
{
  ...
}
```

Instead of having all the rules in a single file, we can put them in separate files and add media queries to `<link>` elements.

```
<link rel="stylesheet" href="mobile.css" media="screen and (max-width:480px)" />
<link rel="stylesheet" href="tablet.css" media="screen and (min-width:481px)
and (max-width:768px)" />
<link rel="stylesheet" href="desktop.css" media="screen and (min-width:769px)" />
```

88

## Responsive Design


### Media Queries

#### - Responsive Grid

- Sử dụng các vùng được đặt tên, chúng ta có thể đạt được responsive layout bằng cách đặt Grid Container bên trong media query:

```
@media screen and (max-width: 699px) {
  .container {
    display: grid;
    grid-template-columns: auto;
    grid-template-rows: auto;
    grid-template-areas: "header"
                        "main"
                        "sidebar"
                        "footer";
  }
}

@media screen and (min-width: 700px) {
  .container {
    display: grid;
    grid-template-columns: 50px 50px 50px 50px;
    grid-template-rows: auto;
    grid-template-areas: "header header header header"
                        "main main . sidebar"
                        "footer footer footer footer";
  }
}
```



89

## Responsive Design

### Media Queries

#### - Ví dụ: Bootstrap Grid

- Hệ thống Bootstrap's Grid cung cấp cho người dùng một lưới hoàn toàn responsive mà không cần tự viết các media query, v.v.
- Mỗi trang Bootstrap rộng 12 cột
  - Có thể chỉ định các thành phần sẽ chiếm một số cột nhất định, tùy thuộc vào kích thước màn hình.

- Ví dụ.

```
<div class="col-xs-12 col-sm-6 col-md-4 col-lg-3"></div>
```

Div này phải rộng 12 cột trên các thiết bị cực nhỏ

Nó phải rộng 6 cột trên các thiết bị nhỏ

Rộng 4 cột trên các thiết bị trung bình

Và rộng 3 cột trên các thiết bị lớn

90

## Responsive Design

### Scaling Images

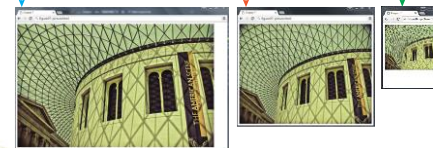
- img {
  - max-width: 100%;
- <picture>

91

## Responsive Design

### Scaling Images

```
<picture>
  <source media="(min-width: 960px)"
    srcset="images/828-large.jpg">
  <source media="(min-width: 480px)"
    srcset="images/828-medium.jpg">
  
</picture>
```

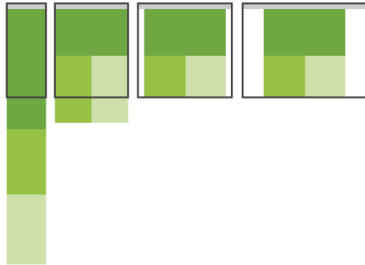


92

## Các mẫu layout đa thiết bị

Responsive Layouts

- **Mostly Fluid:** layout gồm nhiều cột với phần margin rộng hơn trên màn hình lớn.
  - Dựa vào fluid grid và hình ảnh để chia tỷ lệ từ màn hình lớn xuống kích thước màn hình nhỏ và xếp chồng các cột theo chiều dọc



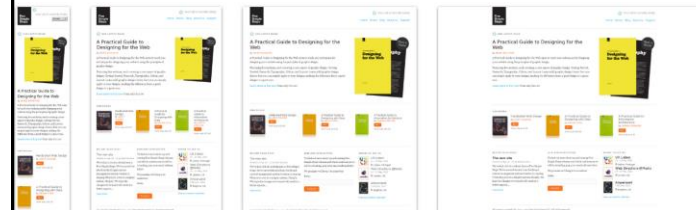
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

93

## Các mẫu layout đa thiết bị

Responsive Layouts

- **Mostly Fluid:** ví dụ



Five Simple Steps: <https://mediaqueri.es/fss/>



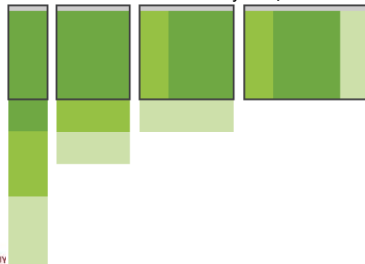
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

94

## Các mẫu layout đa thiết bị

Responsive Layouts

- **Column Drop:** một mẫu phổ biến bắt đầu với bố cục nhiều cột và kết thúc với bố cục một cột, giảm các cột dọc theo đường đi khi kích thước màn hình ngày càng hẹp.
  - Kích thước tổng thể của các phần tử trong bố cục này có xu hướng duy trì nhất quán.
  - Thích ứng với các kích thước màn hình khác nhau thay vì dựa vào các cột xếp chồng



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUY

95

## Các mẫu layout đa thiết bị

Responsive Layouts

- **Column Drop:** ví dụ



<https://mediaqueri.es/mod/>



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

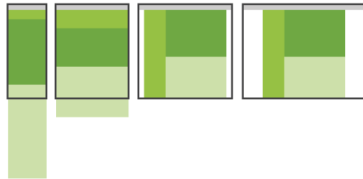
96



## Các mẫu layout đa thiết bị

Responsive Layouts

- **Layout Shifter:** mô hình này phải thay đổi nhiều nhất để thích ứng trên các kích thước màn hình khác nhau.
  - Nghĩa là, các bố cục khác nhau được sử dụng trên màn hình lớn, trung bình và nhỏ.



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

97

## Các mẫu layout đa thiết bị

Responsive Layouts

- **Layout Shifter:** ví dụ



<http://mediaqueri.es/fse/>



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

98

## Các mẫu layout đa thiết bị

Responsive Layouts

- **Tiny Tweaks:** hình thức thích nghi đơn giản nhất và cũng ít được sử dụng nhất.
  - Phù hợp cho các trang Web đơn giản chỉ bao gồm rất ít thành phần trong một bố cục cột duy nhất.
  - Thích ứng đa thiết bị có thể chỉ là một vài chỉnh sửa nhỏ đối với kích thước phông chữ và bố cục hình ảnh.



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

99

## Các mẫu layout đa thiết bị

Responsive Layouts

- **Tiny Tweaks:** ví dụ



<https://mediaqueri.es/pa/>



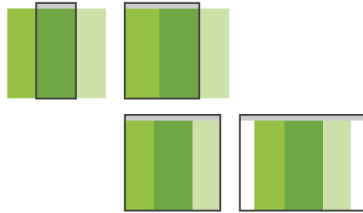
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

100

## Các mẫu layout đa thiết bị

Responsive Layouts

- **Off Canvas**: tận dụng không gian ngoài màn hình để ẩn nội dung hoặc điều hướng cho đến khi kích thước màn hình lớn hơn cho phép hiển thị hoặc người dùng thực hiện hành động để hiển thị nội dung đó.
  - Mô hình này xuất hiện trong một vài thiết kế trang web di động



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

101

## Các mẫu layout đa thiết bị

Responsive Layouts

- **Off Canvas**: Ví dụ



Tham khảo thêm: <https://www.mobile-patterns.com/> <https://pttrns.com/>



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

102

## Question?



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

103

103