

CS2204 Fundamentals of Internet Applications Development

Lecture 5 CSS – Part 2

Computer Science, City University of Hong Kong

Semester A 2023-24

Outline

- CSS Selector
 - CSS Rules Placement
 - Basic selector: ID, class, group
 - Contextual selector
 - Advanced selector: Attribute selector, pseudo class/element
- Cascading and inheritance
- Style organization

Review: HTML Form (1)

`<input>` attributes

- How to limit the length of a text input?
- How to make an input required?

Basic `<input>` types include text, radio, checkbox, password, submit, reset. What are the new input types?

Review: HTML Form (2)

Within a `<input>` tag, what is the difference between “id” and “name”?

Within a `<label>` tag, what does the attribute “for” refer to?

In a list of radio buttons (`<input type = “radio”>`), how to make sure that them are mutual exclusive (e.g., users can only check on button at a time)?

Review: HTML Form (3)

In addition to `<input>`, what are the other tag can be used to collect information in a form?

Which tag is used to group form elements?

Review: Escape characters

Oftentimes we use Escape character in a form of `&code;` to display special characters (e.g., `©` is used to display the copyright sign ©)

But what if we just want to display “©” in plain text?

Outline

- CSS Selector
 - CSS Rules Placement
 - Basic selector: ID, class, group
 - Contextual selector
 - Advanced selector: Attribute selector, pseudo class/element
- Cascading and inheritance
- Style organization

How to Set Styles for Elements?

- The basic idea
 - 1) **find/select an element**
 - 2) **apply styles on it**
- Each rule consists of:

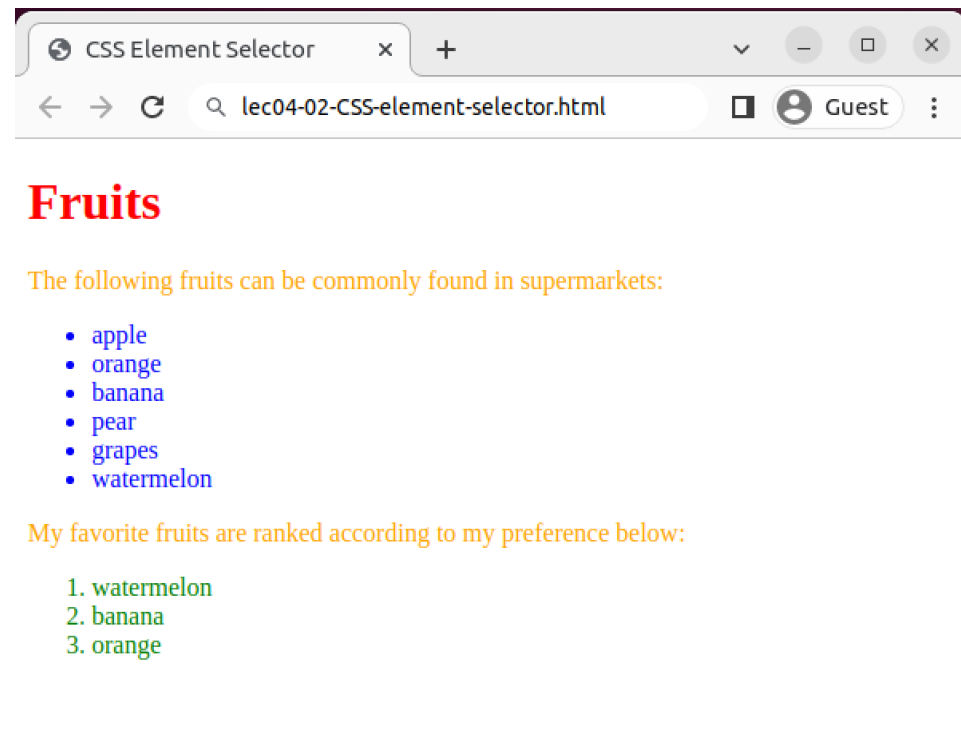
```
selector {  
    property1: value1;  
    property2: value2;  
    ...  
}
```

Element Selector

- A CSS style can be applied to **different** HTML elements by using the corresponding HTML tag as **element selector**

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>CSS Element Selector</title>
5     <style>
6       h1 {
7         color: red;
8       }
9       ul {
10        color: blue;
11      }
12       ol {
13        color: green;
14      }
15       p {
16        color: orange;
17      }
18     </style>
19   </head>
20   <body>
21     <!-- Page content begins here -->
22     <h1>Fruits</h1>
23     <p>The following fruits can be commonly found in supermarkets:</p>
24     <ul>
25       <li>apple</li>
26       <li>orange</li>
27       <li>banana</li>
28       <li>pear</li>
29       <li>grapes</li>
30       <li>watermelon</li>
31     </ul>
32     <p>My favorite fruits are ranked according to my preference below:</p>
33     <ol>
34       <li>watermelon</li>
35       <li>banana</li>
36       <li>orange</li>
37     </ol>
38     <!-- Page content ends here -->
39   </body>
40 </html>
```

Code Example: lec05-01-CSS-element-selector.html



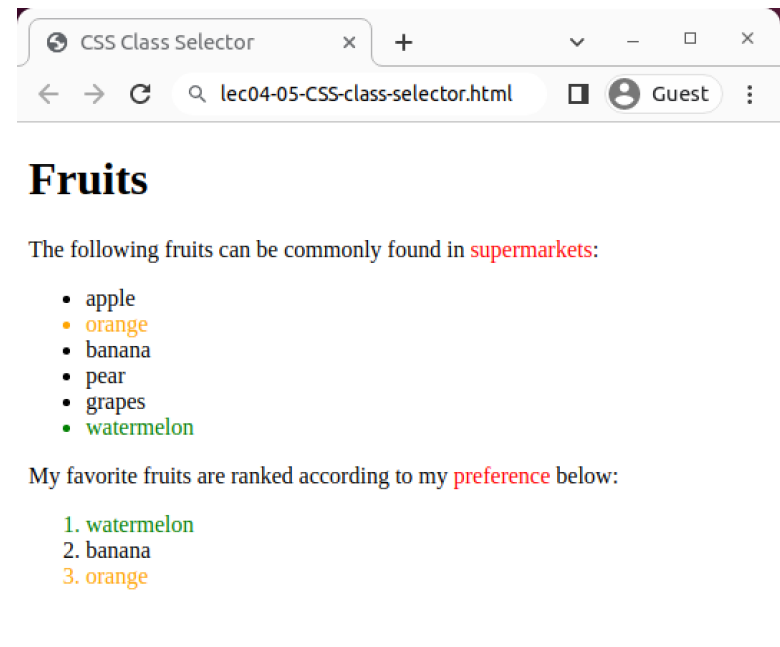
Class Selector

- A CSS style can be applied to **all** elements with the **same class name**

Code Example: lec05-02-CSS-class-selector.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>CSS Class Selector</title>
5   <style>
6     .highlight {
7       color: red;
8     }
9     .watermelon {
10      color: green;
11    }
12    .orange {
13      color: orange;
14    }
15  </style>
16 </head>
17 <body>
18   <!-- Page content begins here -->
19   <h1>Fruits</h1>
20   <p>The following fruits can be commonly found in <span class="highlight">supermarkets</span></p>
21   <ul>
22     <li>apple</li>
23     <li class="orange">orange</li>
24     <li>banana</li>
25     <li>pear</li>
26     <li>grapes</li>
27     <li class="watermelon">watermelon</li>
28   </ul>
29   <p>My favorite fruits are ranked according to my <span class="highlight">preference</span> below:</p>
30   <ol>
31     <li class="watermelon">watermelon</li>
32     <li>banana</li>
33     <li class="orange">orange</li>
34   </ol>
35   <!-- Page content ends here -->
36 </body>
37 </html>
```

The period . followed by a class name is used to specify a class selector



Different html elements can be labeled with the same class name

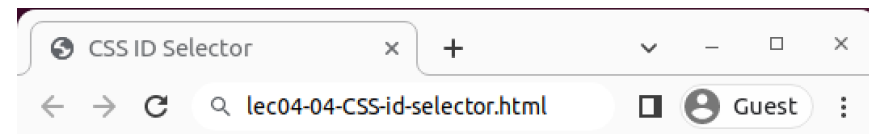
ID Selector

- Each html element can be labeled with an **ID**. A CSS style can be applied to an element by specifying its ID

Code Example: lec05-03-CSS-id-selector.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>CSS ID Selector</title>
5 <style>
6   #paragraph1 {
7     color: blue;
8   }
9   #paragraph2 {
10    color: red;
11  }
12   #favorite1 {
13    color: green;
14  }
15   #favorite2 {
16    color: yellow;
17  }
18   #favorite3 {
19    color: orange;
20  }
21 </style>
22 </head>
23 <body>
24 <!-- Page content begins here -->
25 <h1>Fruits</h1>
26 <p id="paragraph1">The following fruits can be commonly found in supermarkets:</p>
27 <ul>
28   <li>apple</li>
29   <li>orange</li>
30   <li>banana</li>
31   <li>pear</li>
32   <li>grapes</li>
33   <li>watermelon</li>
34 </ul>
35 <p id="paragraph2">My favorite fruits are ranked according to my preference below:</p>
36 <ol>
37   <li id="favorite1">watermelon</li>
38   <li id="favorite2">banana</li>
39   <li id="favorite3">orange</li>
40 </ol>
41 <!-- Page content ends here -->
42 </body>
43 </html>
```

The hash # followed by the id is used to specify an id selector



Fruits

The following fruits can be commonly found in supermarkets:

- apple
- orange
- banana
- pear
- grapes
- watermelon

My favorite fruits are ranked according to my preference below:

1. watermelon
2. banana
3. orange

Each id used to label an html element should be **unique**, i.e., it can only be used once, and no two html elements should have the same id

ID Selector (2)

- What will happen if multiple elements shared the same id?
 - Modern browsers are fault-tolerant
 - However, this practice is not encouraged
 - “ID”: identifying single element.
 - Particularly useful when applying animations/events, where shared IDs among multiple elements can cause issues

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>CSS ID Selector</title>
  <style>
    #paragraph1 {
      color: blue;
    }
    #paragraph2 {
      color: red;
    }
    #favorite1 {
      color: green;
    }
    #favorite2 {
      color: yellow;
    }
    #favorite3 {
      color: orange;
    }
  </style>
</head>
<body>
  <!-- Page content begins here -->
  <h1 id="paragraph1">Fruits</h1>
  <p id="paragraph1">The following fruits can be commonly found in supermarkets:</p>
  <ul>
    <li>apple</li>
    <li>orange</li>
    <li>banana</li>
    <li>pear</li>
    <li>grapes</li>
    <li>watermelon</li>
  </ul>
  <p id="paragraph1">My favorite fruits are ranked according to my preference below:</p>
  <ol>
    <li id="favorite1">watermelon</li>
    <li id="favorite2">banana</li>
    <li id="favorite3">orange</li>
  </ol>
  <!-- Page content ends here -->
</body>
</html>
```

Fruits

The following fruits can be commonly found in supermarkets:

- apple
- orange
- banana
- pear
- grapes
- watermelon

My favorite fruits are ranked according to my preference below:

1. watermelon
2. banana
3. orange

Why is that?

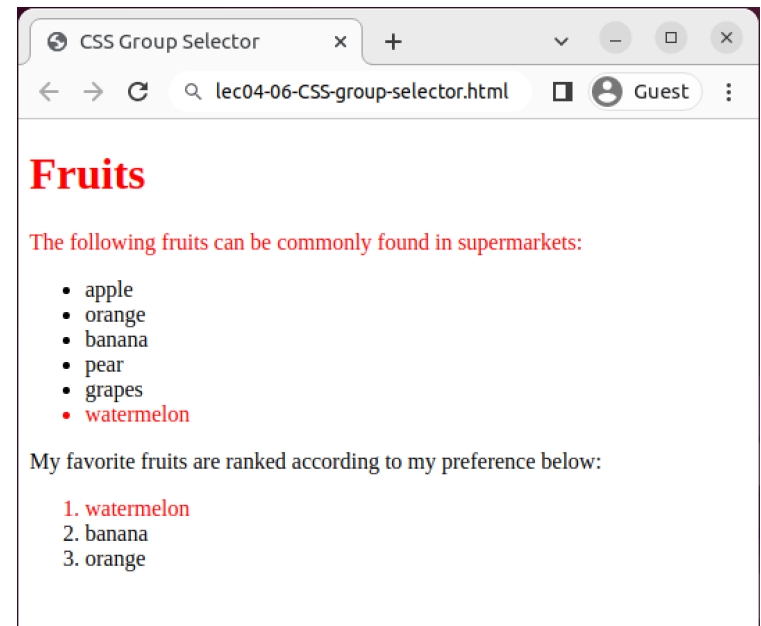
Group Selector

- A CSS style can be applied to a **group** of elements

Code Example: lec05-04-CSS-group-selector.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>CSS Group Selector</title>
5 <style>
6   h1, #paragraph1, .highlight
7     color: red;
8 </style>
9 </head>
10 <body>
11 <!-- Page content begins here -->
12 <h1>Fruits</h1>
13 <p id="paragraph1">The following fruits can be commonly found in supermarkets:</p>
14 <ul>
15   <li>apple</li>
16   <li>orange</li>
17   <li>banana</li>
18   <li>pear</li>
19   <li>grapes</li>
20   <li class="highlight">watermelon</li>
21 </ul>
22 <p>My favorite fruits are ranked according to my preference below:</p>
23 <ol>
24   <li class="highlight">watermelon</li>
25   <li>banana</li>
26   <li>orange</li>
27 </ol>
28 <!-- Page content ends here -->
29 </body>
30 </html>
```

The comma , is used to separate individual selectors to specify a group selector



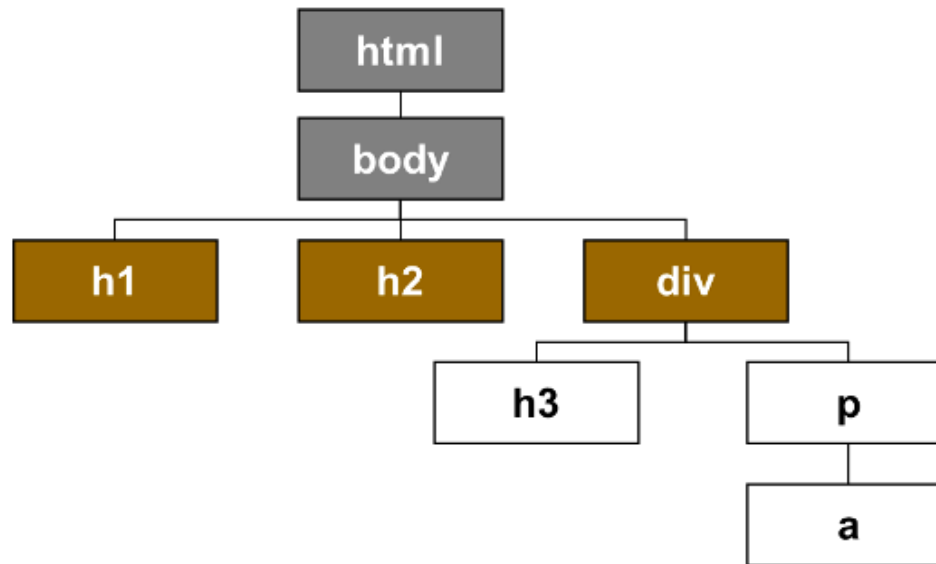
Agenda for today's lecture

- CSS Selector
 - CSS Rules Placement
 - Basic selector: ID, class, group
 - Contextual selector
 - Advanced selector: Attribute selector, pseudo class/element
- Cascading and inheritance
- Style organization

Contextual & Document Tree

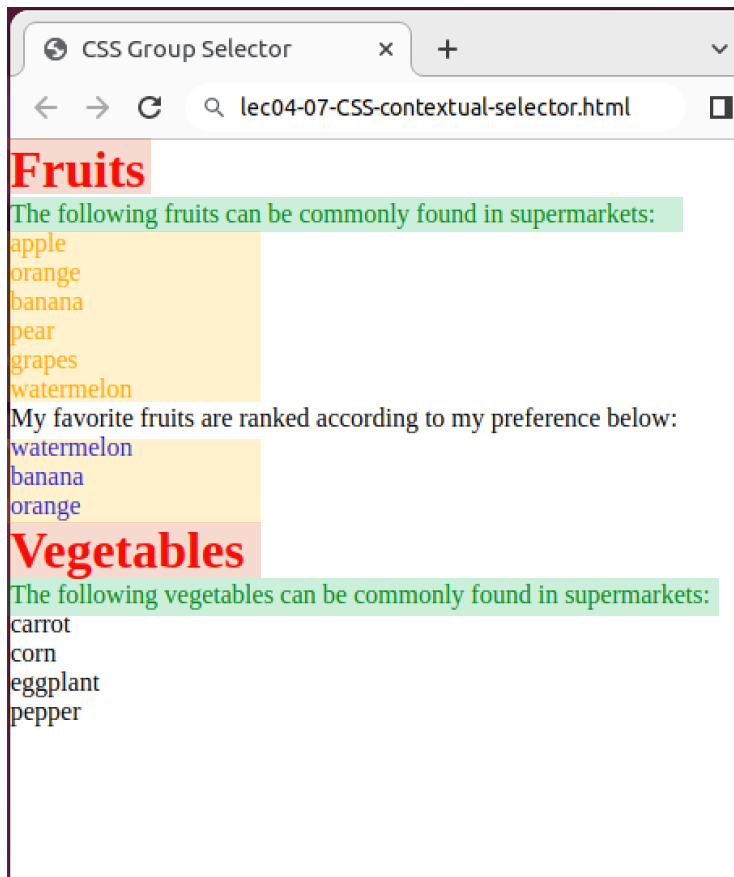
- **Contextual** means **depending on the context** - the position where you are **relative** to the surrounding
 - The **context** inside a Web page is defined by **the Document Tree**
- Different kinds of relationships are defined in a document tree, known as **Document Object Model (DOM)**:
 - **parent**: an element directly above another in the document tree
 - **child**: an element directly one level below another element within the document tree
 - **ancestor**: can be a parent, grandparent, great-grandparent, or higher within the tree
 - **descendant**: can be a child, grandchild, great-grandchild or further descendent down the line
 - **Sibling**: elements that share the same parent
- These relationships are used to form **contextual selector**

Contextual & Document Tree



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>CSS Group Selector</title>
5   </head>
6
7   <body>
8     <h1>Fruits</h1>
9     <h2>Green fruits</h2>
10    <div>
11      <h3>Sales</h3>
12      <p>The following fruits can be commonly found in <a href="">supermarkets</a>:</p>
13    </div>
14  </body>
15 </html>
```

Contextual Selectors



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>CSS Group Selector</title>
6     <style>
7       #mainContainer h1{
8         color: red;
9       }
10
11       #list1 > li{
12         color:orange;
13       }
14
15       #list2 > li{
16         color: blue;
17       }
18
19       h1 + p{
20         color: green;
21       }
22
23       * {
24         margin: 0;
25         padding: 0
26       }
27     </style>
28   </head>
29   <body>
30     <!-- Page content begins here -->
31     <div id="mainContainer">
32       <h1>Fruits</h1>
33       <p id="paragraph1">The following fruits can be commonly found in supermarkets:</p>
34       <ul id="list1">
35         <li>apple</li>
36         <li>orange</li>
37         <li>banana</li>
38         <li>pear</li>
39         <li>grapes</li>
40         <li>watermelon</li>
41       </ul>
42
43       <p>My favorite fruits are ranked according to my preference below:</p>
44       <ol id="list2">
45         <li>watermelon</li>
46         <li>banana</li>
47         <li>orange</li>
48       </ol>
49       <h1>Vegetables</h1>
50       <p id="paragraph1">The following vegetables can be commonly found in supermarkets:</p>
51       <ul id="list3">
52         <li>carrot</li>
53         <li>corn</li>
54         <li>eggplant</li>
55         <li>pepper</li>
56       </ul>
57     </div>
58     <!-- Page content ends here -->
59   </body>
60 </html>
```

Descendant Selector - to select an element that is a descendant of a defined ancestor element

Child Selector - to select an element that is a child of a defined parent element

Adjacent Sibling Selector - to select an element that appears **immediately after** another, must be at the same level in document tree

Universal Selector - represented by an asterisk * (wild card), to select any element

More on Contextual Selector

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <style>
7     li {
8       list-style: none;
9     }
10    ul li {
11      color: red;
12    }
13
14    ul li > span {
15      color: blue;
16    }
17
18    .selected li span {
19      color: green;
20    }
21  </style>
22 </head>
23 <body>
24   <ul>
25     <li>1</li>
26     <li>2</li>
27     <li>
28       <span>3</span>
29     </li>
30     <li>4</li>
31   </ul>
32   <ul class="selected">
33     <li>5</li>
34     <li>6</li>
35     <li>
36       <span>7</span>
37     </li>
38     <li>8</li>
39   </ul>
40 </body>
41 </html>
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

Code Example: lec05-06-contextual-selector-2.html

Agenda for today's lecture

- CSS Selector
 - CSS Rules Placement
 - Basic selector: ID, class, group
 - Contextual selector
 - Advanced selector: Attribute selector, pseudo class/element
- Cascading and inheritance
- Style organization

Advanced Selector (1)

Attribute Selector - select an element based on its attribute or attribute value.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <style>
7     input[type=text] {
8       width: 150px;
9       background-color: yellow;
10    }
11    input[type=button] {
12      width: 120px;
13      background-color: green;
14    }
15  </style>
16 </head>
17 <body>
18
19   <h2>Styling Forms</h2>
20
21   <form method="get" action="#">
22     <label for="ffname">First Name:</label> <br>
23     <input type="text" id="ffname" name="fname"> <br>
24     <label for="llname">Last Name:</label> <br>
25     <input type="text" id="llname" name="lname">
26     <br><br>
27     <input type="button" value="Example Button">
28   </form>
29 </body>
30 </html>
```

Styling Forms

First Name:

Last Name:

Example Button

Advanced Selector (2)

Pseudo Class: a class not defined by using class name but by the state of the element

`p:first-child {property: value;}`

select all `<p>` that must be the first child, not just child

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
  color: red;
}

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

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

/* selected link */
a:active {
  color: blue;
}
</style>
</head>
<body>

<h2>Styling a link depending on state</h2>

<p><b><a href="default.asp" target="_blank">This is a
link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited
in the CSS definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS
definition in order to be effective.</p>

</body>
</html>
```

Styling a link depending on state

[This is a link](#)

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.

Note: a:active MUST come after a:hover in the CSS definition in order to be effective.

Other pseudo class:

:focus — if the user focuses the element by clicking or using keyboard controls.

:first-child — any element that is the first child of an element

:nth-child(even) — any element that is the child of an element with an even order

Advanced Selector (3)

Pseudo Element - similar to pseudo class but depends on the position in the DOM, more detail than contextual selector

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

```
p::after {
  content: "XXXXXXX";
}
</style>
</head>
<body>
```

<p>You can use the ::first-line pseudo-element to add
a special effect to the first line of a text. Some
more text. And even more, and more, and more, and
more, and more, and more, and more, and more,
and more, and more, and more, and more.</p>

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

YOU CAN USE THE `::FIRST-LINE PSEUDO-ELEMENT` TO ADD A SPECIAL EFFECT TO THE
first line of a text. Some more text. And even more, and more, and more, and
more, and more, and more, and more, and more, and more, and more, and
more, and more.XXXXXXX

Other pseudo elements:

::first-letter – the first letter of the element

::after – insert content after an element

::before – insert content before an element

`::selection` – select the selected text

Advanced Selector (4)

More pseudo Elements - the `::selection` pseudo-element matches the portion of an element that is selected by a user.

```
<!DOCTYPE html>
<html>
<head>
<style>
::selection {
  color: red;
  background: yellow;
}
</style>
</head>
<body>

<h1>Select some text on this page:</h1>
<p>This is a paragraph.</p>
<div>This is some text in a div element.</div>

</body>
</html>
```

Select some text on this page:

This is a paragraph.

This is some text in a div element.

Critical thinking

What are the difference between pseudo class and pseudo elements?

Agenda for today's lecture

- CSS Selector
 - CSS Rules Placement
 - Basic selector: ID, class, group
 - Contextual selector
 - Advanced selector: Attribute selector, pseudo class/element
- Cascading and inheritance
- Style organization

Cascade Order

- The effect of CSS rules may **cascade**. **Properties** of an element may be set, sometimes **conflictedly**, by **more than one CSS rules**
 - This would happen even if one style sheet is used. The final result, that is which rule will win, must be determined **systematically** and **consistently**
 - **Cascade Order** is the way to determine the result that must be **standardized**, so that **all browsers will render the page in the same way**
- Three situations when more than one rules may affect an element:
 - **inheritance**
 - **overriding**
 - **same** selector
 - **different** selectors
 - **multiple origins of style**

Inheritance

- Consider the following CSS rules acting on the page in the diagram:
`body { color: blue; }`
`#pageContent { font-size: 1em; }`
`h3 { text-transform: uppercase; }`
- The `<h3>` element has 3 rules affecting it for **three properties** although **two rules** are not explicitly used for it
- CSS rules** set for **ancestors** go on **affecting descendants** is called **inheritance**

```
<body>
  <div id="pageContent">
    <h3>
      Payment
    </h3>
  </div>
</body>
```

Inheritance

- Note that **not all properties** would **have inheritance effect**
 - (default) inherited properties, e.g., color, text-, font-, line-, etc.
 - (default) non-inherited properties, e.g., border

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    body {
      color: blue;
      font: 14px/1.5 Times;
      border: 2px solid black;
    }
    .p2 {
      font: 16px/1.5 Times;
      border: 1px solid red;
    }
  </style>
</head>
<body>
  <p class="p1">The Department of Computer Science was established in 1984, and has
evolved from a teaching-oriented department into one that excels in both teaching and
research. he Department launched its first BSc in Computer Studies in 1987 (renamed and
restructured as the BSc Computer Science in 2000), followed by the MSc in Computer Science
in 1991. The Department also produced its first PhD graduate in 1994. In addition to
offering traditional courses such as foundations of computer science, computer architecture
and software engineering, our curriculum exposes students to the latest advances in cloud
computing, machine learning, and bioinformatics. Students have the opportunity to undertake
a major design and development project in areas such as electronic commerce, virtual
reality, multimedia, computer vision, cloud computing, and information security.
  </p>

  <p class="p2">The Department of Computer Science was established in 1984, and has
evolved from a teaching-oriented department into one that excels in both teaching and
research. he Department launched its first BSc in Computer Studies in 1987 (renamed and
restructured as the BSc Computer Science in 2000), followed by the MSc in Computer Science
in 1991. The Department also produced its first PhD graduate in 1994. In addition to
offering traditional courses such as foundations of computer science, computer architecture
and software engineering, our curriculum exposes students to the latest advances in cloud
computing, machine learning, and bioinformatics. Students have the opportunity to undertake
a major design and development project in areas such as electronic commerce, virtual
reality, multimedia, computer vision, cloud computing, and information security.
  </p>
</body>
</html>
```

The Department of Computer Science was established in 1984, and has evolved from a teaching-oriented department into one that excels in both teaching and research. he Department launched its first BSc in Computer Studies in 1987 (renamed and restructured as the BSc Computer Science in 2000), followed by the MSc in Computer Science in 1991. The Department also produced its first PhD graduate in 1994. In addition to offering traditional courses such as foundations of computer science, computer architecture and software engineering, our curriculum exposes students to the latest advances in cloud computing, machine learning, and bioinformatics. Students have the opportunity to undertake a major design and development project in areas such as electronic commerce, virtual reality, multimedia, computer vision, cloud computing, and information security.

The Department of Computer Science was established in 1984, and has evolved from a teaching-oriented department into one that excels in both teaching and research. he Department launched its first BSc in Computer Studies in 1987 (renamed and restructured as the BSc Computer Science in 2000), followed by the MSc in Computer Science in 1991. The Department also produced its first PhD graduate in 1994. In addition to offering traditional courses such as foundations of computer science, computer architecture and software engineering, our curriculum exposes students to the latest advances in cloud computing, machine learning, and bioinformatics. Students have the opportunity to undertake a major design and development project in areas such as electronic commerce, virtual reality, multimedia, computer vision, cloud computing, and information security.

Overriding – Same Selector

- **Properties** of an element may be set, sometimes **conflictedly**, by **more than one CSS rules**
- If they are set by the **same** selector, the **newest selector** (i.e., the one **closest** to the corresponding element) will **win**
 - non-conflicted property can be inherited

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    p {
      color: red;
      font-size: 20px;
    }
    p {
      color: blue;
    }
  </style>
</head>
<body>
  <p>this is an <span>example</span></p>
  <div>this is a div block</div>
</body>
</html>
```

this is an example

this is a div block

Overriding – Different Selectors

- **Properties** of an element may be set, sometimes **conflictedly**, by **more than one CSS rules**
- If they are selected by **different** selectors, who wins depends on the **specificity (i.e., priority)** of the selector
 - **Left:** **most significant** position
 - **Right:** **least significant** position

Selector	Specificity
!important	infinite
inline	1,0,0,0
ID selector	0,1,0,0
Class selector	0,0,1,0
Element selector	0,0,0,1
* or Inheritance	0,0,0,0

Different Selectors

- For a **single selector**, the specificity is obtained from the table directly
 - e.g., $p (0,0,0,1) < .class (0,0,1,0) < \#id (0,1,0,0)$
- For any rule, it can be declared as **important** and results in the highest priority, e.g., `p {margin-left: 5px !important}`

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <style>
7     #demo {
8       color: blue;
9     }
10    .test {
11      color: red;
12    }
13    p {
14      color: green;
15      /* color: green !important; */
16    }
17    .div {
18      color: purple;
19    }
20    span {
21      color: green;
22    }
23  </style>
24 </head>
25 <body>
26   <p id="demo" class="test">Hello world!</p>
27
28   <div class="div">
29     <span>CS2204</span>
30   </div>
31 </body>
32 </html>
```

Hello world!

CS2204

Selector	Specificity
!important	infinite
inline	1,0,0,0
ID selector	0,1,0,0
Class selector	0,0,1,0
Element selector	0,0,0,1
* or Inheritance	0,0,0,0

Code Example: lec05-12-CSS-priority-single.html

Different Selectors

- For a **compound selector**, the specificity will be **added** for **each individual selector**
 - e.g., ol li #class (0,0,0,1 + 0,0,0,1 + 0,0,1,0 = 0,0,1,2)

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <style>
7     li {
8       color: red;
9     }
10
11     ol>li {
12       color: blue;
13     }
14
15     .list>li {
16       color: green;
17     }
18
19     body .list {
20       color: skyblue;
21     }
22
23     body ol .test {
24       color: purple;
25     }
26
27   </style>
28 </head>
29 <body>
30   <ol class="list">
31     <li class="test">1</li>
32     <li>2</li>
33     <li>3</li>
34   </ol>
35 </body>
36 </html>
```

1.1
2.2
3.3

Selector	Specificity
!important	infinite
inline	1,0,0,0
ID selector	0,1,0,0
Class selector	0,0,1,0
Element selector	0,0,0,1
* or Inheritance	0,0,0,0

Code Example: lec05-13-CSS-priority-compound.html

Who Will Win In Cascading?

- To comprehensively answer this question, need to consider **four factors**:
 - **Origin of styles** - **user agent** (i.e., browser default), **author** (i.e., you) or **user** (those who are looking at your Web page)
 - **Types of style** - inline, embedded or link
 - The **order** of applying CSS rules
 - **Selector** - how the element is selected in the rule (specificity)

Origin of Styles

- First, consider the **origin** - according to the standard order of **precedence** (i.e., more important) is **author > user > user agent**
 - Author style: supplied by the author of a webpage
 - User style sheets: supplied by the user of the browser
 - Default style (user agent): supplied by the browser vendor
 - This means that the users have no way to override the Web page according to their need, not good for accessibility. Here comes the **!important declaration**
- For any rule, it can be declared as **important** and results in higher priority, e.g.,

`p {margin-left: 5px !important}`

with `!important` the precedence order of origin becomes - **user important > author important > author normal > user normal > user agent**. However, `!important` should only be used with care in author style; Otherwise, it becomes somewhat like Inline Style

- After this, move on to consider precedence **inside author styles**

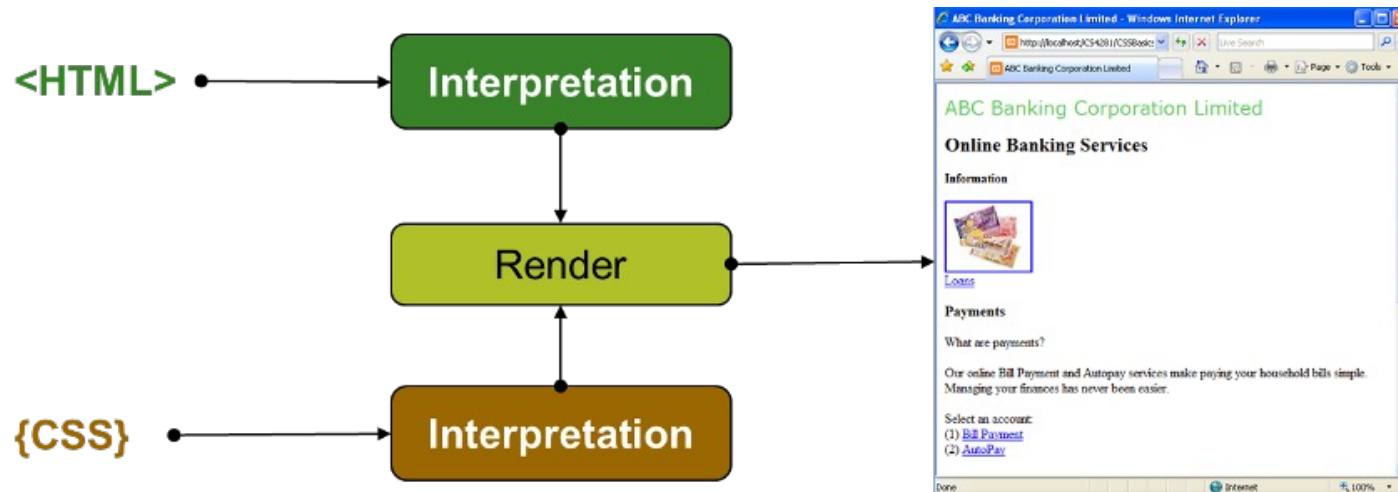
Origin of Styles

- As we have already learnt, inside Author Styles (in a Web page):
 - inline styles **ALWAYS** **override** other styles and are of **the highest priority**, except **!important**
 - **embedded** and **linked styles** have **no difference in priority** and only depend on **their specificity of each element**
 - if everything being **equal**, **the newest rule** in terms of order will win

Agenda for today's lecture

- CSS Selector
 - CSS Rules Placement
 - Basic selector: ID, class, group
 - Contextual selector
 - Advanced selector: Attribute selector, pseudo class/element
- Cascading and inheritance
- **Style organization**

How to apply CSS style?



- **Inline Style**
- **Embedded Style**
- **External Style**
 - **Link**
 - **Import**

Inline Style

- Set styles **directly** in the Web page using the **style attribute** of an element

```
<div style="color: #003366; font-size:.8em;">
```

- **Advantages:**
 - Good for **diagnostics** (testing or finding errors)
 - A **quick** way to make sure the style is properly set
- **Disadvantage:** **extremely difficult** to use or update because you need to find the elements one by one. It should not be used unless really necessary

Embedded Style

- Put CSS rules in the `<head>` section.
- The `<style>` tag is used to enclose all rules, which will be applied to **all elements selected** in the **entire page** (but not other pages)

Advantages:

- Write once, **apply to the whole page**
- Easier to change
- Good for styles for **one (this) page only**
- Can use for **overriding styles**
- Can use the **media attribute** in the `<style>` tag to **specifies what media/device type the styles are optimized for**

```
<head>
  <title>Page Title</title>

  <!-- Embedded stylesheet -->
  <style media="print">
    h2 {
      font-size: 1.5rem;
      color: #2f8d46;
      text-align: center;
    }

    p {
      font-variant: italic;
    }
  </style>
</head>
```

Intended to be viewed
on a screen in **print
preview mode**.

External Style

- Put all CSS rules in a **separate .css file (a style sheet)**
- Two ways to use an external style sheet

```
<head>
  <link rel="stylesheet" href="mystyle.css" type="text/css" media="all">
  <title>Page title</title>
</head>
<body>
```

Link style

```
<head>
  <style type="text/css" media="all">
    @import url("mystyle.css");
  </style>
  <title>Page title</title>
</head>
```

Import style

Advantages:

- Write once, **apply to all linked pages**
- Easy to update
- Good for **set up themes (consistent styles)** applying to all pages in a Web site
- Increase accessibility through the use of **consistent styles**
- Can use **media attribute**

Disadvantages: need to be careful with the effect of **multiple style sheets interaction**

External style: Link

- Linking to a style sheet means using the `<link>` tag in the **head section** to load all CSS rules and apply their effect to the page
 - Note that **more than one** style sheet can be linked
 - In the link tag, `rel` is **always** `"stylesheet"`, but `type` is **now always** `"text/css"`, because the initial design of style sheet allows other languages, not just css
 - `media` = `"all"` is used for all media type devices

```
<head>
  <link rel="stylesheet" type="text/css" href="styles1.css" media="all">
  <link rel="stylesheet" type="text/css" href="styles2.css" media="all">
  <link rel="stylesheet" type="text/css" href="styles3.css" media="all">
</head>
```

External style: Import

- Rules are in .css file but links differently with **@import**
- Syntax: **@import** *url/string list-of-mediaqueries*;

```
<head>
  <meta charset="utf-8">
  <style type="text/css" media="all">
    @import url("mystyle.css");
  </style>
</head>
```

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

Value	Description
<i>url string</i>	A url or a string representing the location of the resource to import. The url may be absolute or relative
<i>list-of-mediaqueries</i>	A comma-separated list of media queries conditioning the application of the CSS rules defined in the linked URL

- Must be at the top of the document (but after any **@charset** declaration)
- Supports media queries, so you can allow the import to be media-dependent.
- We can also import style conditionally

@import "printstyle.css" print;

Import style only if the media = "print"

Link vs. Import

- **Link**
 - Link one stylesheet to a html page
 - Asynchronously load css style
- **Import**
 - Import one stylesheet to another stylesheet – convenient to link multiple stylesheets
 - But the browser will wait for the imported file to finish downloading before it starts downloading the rest of the content.
 - Cannot be handled by old browsers
- In general, **link** is preferred over **import** for better performance

Summary: Where To Put CSS Rules?

- **Inline** - No CSS rules are required, only **properties** and **values** set in the style attribute
- **Embedded** - put **inside** the `<head>` section with `<style>`
- **External**
 - **Link**: CSS rules are put in a **.css file** linked to Web pages using `<link>`
 - **Import** - again rules are in .css file but links differently with `@import directive/command`

How to Organize Style Sheets? (1)

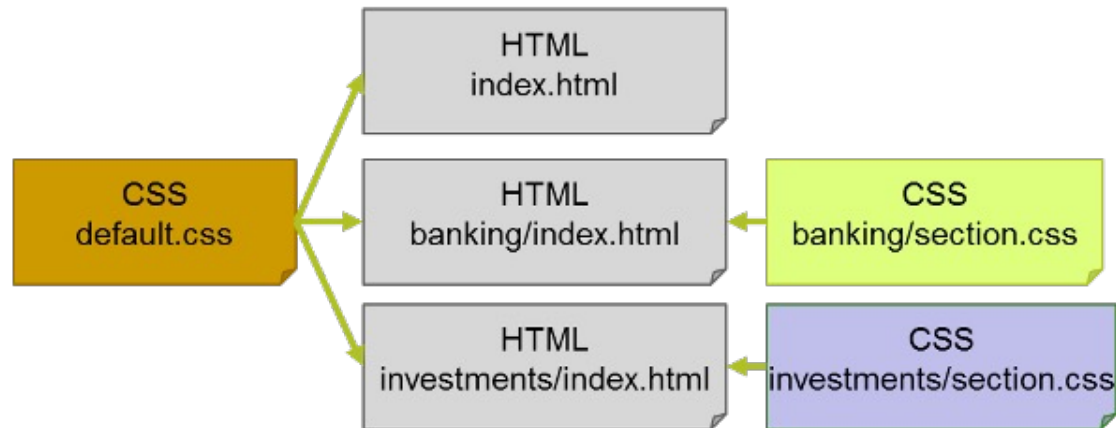
(1) By application functions

- One highest level “*default.css*” - styles used in **all** pages
- One style sheet “*section.css*” - each for sub-systems grouping styles

Folder Structure in a Web site



Linking among HTML and CSS files



How to Organize Style Sheets? (2)

(2) By content (HTML) type - to maintain consistency in interfaces throughout the Web site, can consider to

- set up style sheets to control forms, tables, lists, etc. and
- apply them to pages that have those HTML

Media Type and Media Query

- **Styles** could be used **selectively** depending on different conditions. Two ways to specify **media dependencies** for style sheet:
 - ① specify the **target media** inside a style sheet with the `@media` or `@import` at-rules **[optional]**
 - ② specify the **target medium** with **external style sheet**

```
8 <meta name="keywords" content="payment, bill payment, autopay" />
9 <title>ABC Banking Corporation Limited</title>
10 <link rel="stylesheet" href="../lec04-09-screen.css" type="text/css" media="screen" />
11 <link rel="stylesheet" href="../lec04-09-print.css" type="text/css" media="print" />
12 <!-- // not for IE
13 <style type="text/css">
14     @import url("../lec04-09-screen.css") screen;
15     @import url("../lec04-09-print.css") print;
16 </style>
17 or
18 @media screen {
19     div ul {width: 90%;}
20     div ul li {width: 100%;}
21 }
22 -->
23 </head>
```

Media Type And Media Query

- The media type has been further developed into more complicated as well as **combined** (e.g., **and**, **or**, etc.) conditions in CSS3 known as **Media Queries**

■ *common media types*

<i>Recognized Media Types</i>	<i>Description</i>
<i>all</i>	<i>Suitable for all devices</i>
<i>aural</i>	<i>Intended for speech synthesizers</i>
<i>braille</i>	<i>Intended for Braille tactile feedback devices</i>
<i>embossed</i>	<i>Intended for paged Braille printers</i>
<i>handheld</i>	<i>Intended for handheld devices (typically small screen, and monochrome)</i>
<i>print</i>	<i>Intended for paged, opaque material and for documents viewed on screen in print preview mode</i>
<i>projection</i>	<i>Intended for projected presentations, such as projectors</i>
<i>screen</i>	<i>Intended primarily for color computer screens</i>
<i>tty</i>	<i>Intended for media using a fixed-pitch character grid, such as teletypes, terminals, or portable devices with limited display capabilities</i>
<i>tv</i>	<i>Intended for television-type devices (low resolution, color, limited-scrollability screen, sound available)</i>

Personal Webp

- Add the following styles:
 - **Task 1:** Add colors for each cell in “*schedule.html*”
 - **Task 2:** Link to “*basicStyle.css*”, which uses **margin**, **padding**, **width**, **height**, **border**, **color**, **font**, etc. to customize the style for your **homepage elements**
 - **Task 3:** Link to “*navStyle.css*” and set the **tab-like** style for the **navigators**
 - **Task 4:** Link to “*print.css*” for the **print format** of the webpage
 - **Task 5:** Link to “*reset.css*” to reset the **link style** after the link is visited

