NAZARBAYEV
UNIVERSITY

Web Programming and Problem Solving

CSS (part 2)

Date: 12.09.2022

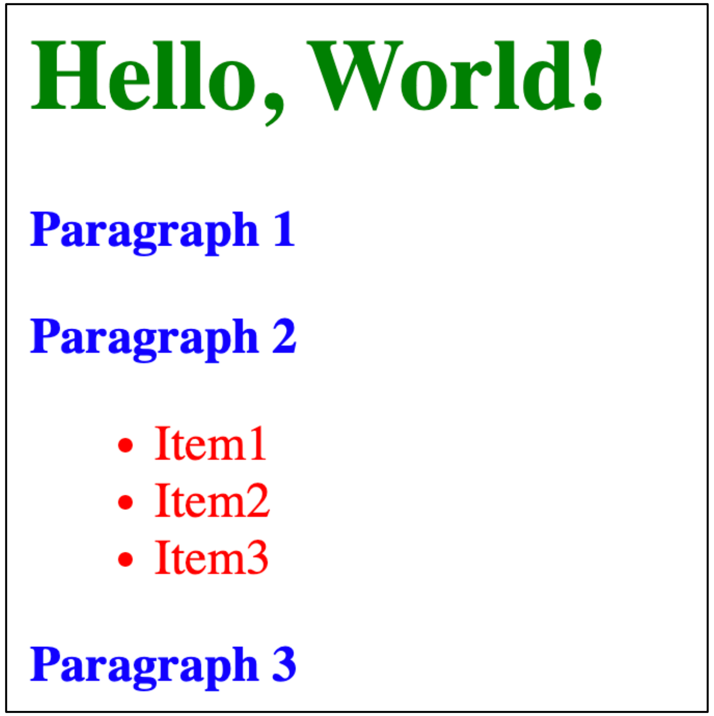
Instructor: Zhandos Yessenbayev

Content

- CSS selectors
  - DOM-based
  - Pseudo-class
  - Pseudo-elements
- Conflict Resolution

CSS Selectors

- What if we want to change a particular paragraph?
  - How do we select a specific element?
- What if one element is changed in several places?
  - How to resolve the conflicts?



Hello, World!

Paragraph 1

Paragraph 2

- Item1
- Item2
- Item3

Paragraph 3

CSS Selectors

- To distinguish between elements, we use selectors:
  - Element Types (Tags)
  - Element Classes
  - Element Attributes
  - Element IDs
  - DOM-based
  - Pseudo-class
  - Pseudo-elements

Element Selectors

Selection of one or more **elements**:

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

h1, p {
  color: blue;
  font-size: 12pt;
}
```

Selects and changes the properties of body element

Selects and changes the properties of h1 and p elements (*Note the comma in between*)

Class Selectors

Class is an identifier that can group together multiple elements

```
<p class="second"> … </p>

<li class="second item"> … </li>
```

Definition of the class **second** for two elements. Elements can belong to several classes.

```
.second {
  color: red;
}

li.item {
  color: purple;
}
```

Selects the elements with the specified class (second or item)

Note a dot before class name

Attribute Selectors

Selection of the elements by their attributes

```
h1[style] {
  text-align: center;
}

li[name] {
  color: gray;
}
li[name="item1"] {
  color: orange;
}
```

Selects all h1 tags with their style attribute defined

Selects all li elements with their name attribute defined as well as those which have specific value for name attribute

ID Selectors

ID is an identifier of an element unique within the document

```
<p  id=”last"> … </p>
```

Assignment of **ID** to the element

```
#last {
  color: red;
  font-size: 15pt;
}
```

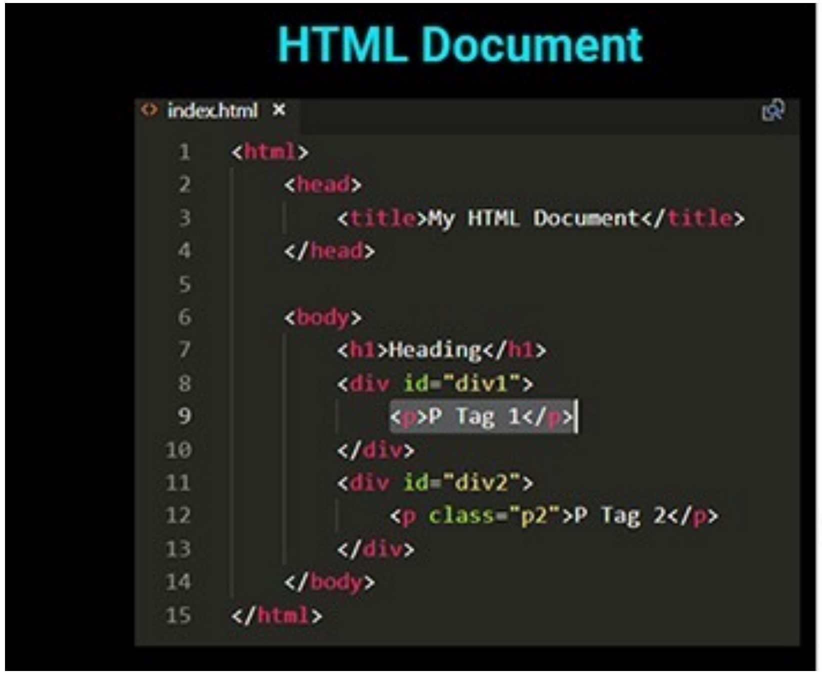
Selects the elements by its **ID**

Note a hash before the ID

DOM

- HTML document can be viewed as a **tree**-like structure.
- This structure is represented as **Document Object Model (DOM)** in memory
- Elements are called nodes, such as:
  - Root (Document)
  - Parent
  - Child/Children
  - Siblings (head, body)

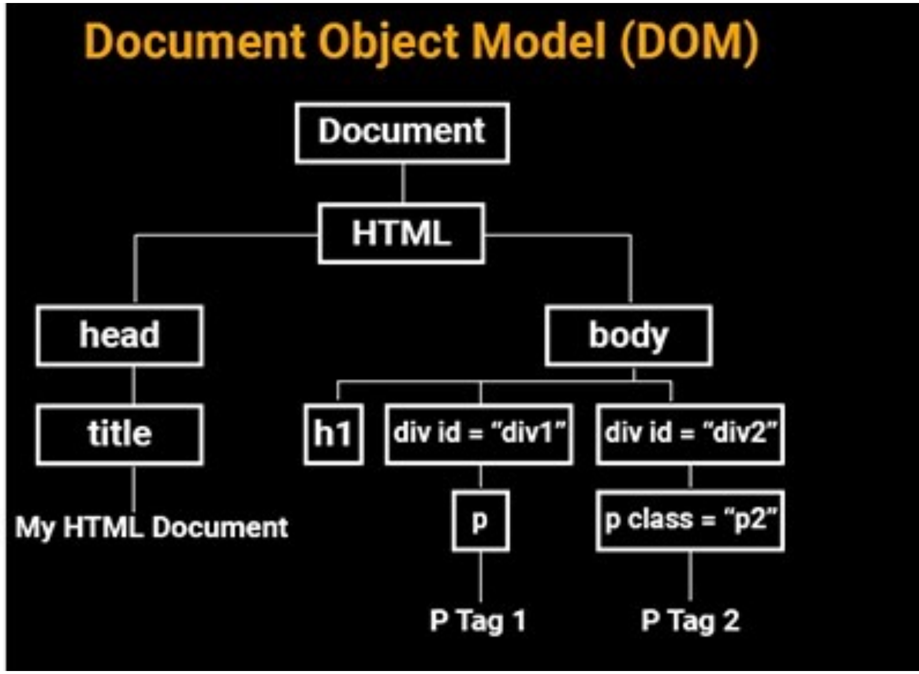
HTML Document



```
<html>
    <head>
        <title>My HTML Document</title>
    </head>

    <body>
        <h1>Heading</h1>
        <div id="div1">
            <p>P Tag 1</p>
        </div>
        <div id="div2">
            <p class="p2">P Tag 2</p>
        </div>
    </body>
</html>
```

Document Object Model (DOM)



Children Selectors

To select direct children of some element, use **>** :

```
ul > li {
  color: red;
}
```

Selects only the direct **li** elements of **ul**

To select any other direct child of some element:

```
body li {
  color: green;
}
```

Selects all **li** elements of **body**

NAZARBAYEV UNIVERSITY

Pseudo-Class Selectors

A pseudo-class is used to define a special **state** of an element.

The syntax:

```
selector:pseudo-class {
  property: value;
}
```

```
/* unvisited link */
a:link {
  color: red;
}
/* visited link */
a:visited {
  color: green;
}
/* mouse over link */
a:hover {
  color: hotpink;
}
/* selected link */
a:active {
  color: blue;
}
```

a tag’s states

Pseudo-Class Selectors

To select the **first child** of an element:

```
ul li:first-child {
  color: blue;
}
```

To select the **n-th child** of an element:

```
ul li:nth-child(3) {
  color: green;
}
```

Pseudo-Element Selectors

A pseudo-element is used to style specified **parts** of an element.

The syntax:

```
selector::pseudo-element {
  property: value;
}
```

Note **double colon** in syntax

```
p::first-letter {
  color: red;
}
p::first-line {
  color: black;
}
p::selection {
  color: red;
  background: yellow;
}
::marker {
  color: orange;
}
```

Conflict Resolution

Application of CSS rules depends on three main concepts:

- **Cascade**
- **Inheritance**
- **Specificity**

To resolve the conflicts, we need to understand them well.

Cascade

Cascade is a concept which means that the origin and the order of CSS rules matter, i.e. the latest rules is applied

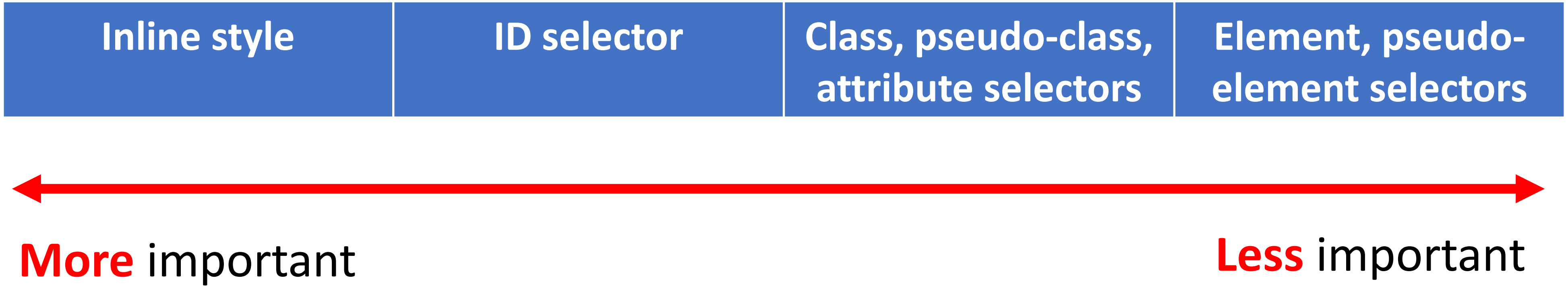
```
h1 {
  color: blue;
}
/* this rule overrides
   the previous rule */
h1 {
  color: green;
}
```

Source order only matters when the specificity weights of the rules are the same!

Specificity

Specificity is the <u>weight</u> that the browser uses to decide which property value is applied to an element.

Weight is composed of 4 numbers based on the <u>location</u> of a rule and the <u>number</u> of appearance of the selectors


Inline style
ID selector
Class, pseudo-class, attribute selectors
Element, pseudo-element selectors
More important
Less important

Specificity

Selector	Inline	ID	Class	Element	Specificity
h1	1	0	0	1	[1,0,0,1]
p	0	0	0	1	[0,0,0,1]
p.second	0	0	1	1	[0,0,1,1]
ul li:first-child	0	0	1	2	[0,0,1,2]
#last	0	1	0	0	[0,1,0,0]

```
#last{
  color: blue;
}
```

vs

```
body p:last-child{
  color: green;
}
```

Inheritance

Inheritance means that elements can inherit the properties defined in their parents or ancestors.

- Some properties can’t be inherited like *weight* or *margin*.
- CSS provides five special property values for elements:
  - inherit – turn on inheritance
  - initial – property’s default
  - revert – browser's default
  - revert-layer – previous layer
  - unset – set to inherit or initial

```
body {
  color: blue;
}
/* revert to browser’s
   default value*/
h1 {
  color: revert;
}
```

!important

Importance is the mechanism to apply a rule no matter what the order, specificity or inheritance of other rules

```
h1 {
  color: blue !important;
}
/* this rule is not applied */
h1 {
  color: green;
}
```

However, it is **not** recommended to use it unless really necessary

Summary

- **Key takeaways**:
  - The selection can be done:
    - using element’s type, class, attributes and ID
    - based on DOM (structure of HTML)
    - Using psedo-classes and pseudo-elements
  - Three concepts are important in conflct resolution
    - Cascade
    - Specificity
    - Inheritance
  - Use **important** keyword only when really necessary

Thanks for Attention!