

CSS Essentials

CSS Overview



Lesson Objectives

- <Mandatory slide>
- <Brief the objectives of the course: What trainees learn after the course>
- <Remember to update the course version in Notes part>

Session 1

CSS OVERVIEW

- 1. What is CSS?**
- 2. CSS Syntax**
- 3. CSS Module**
- 4. Browser support**
- 5. Apply CSS to browser**
- 6. How CSS work?**

1. What is CSS?

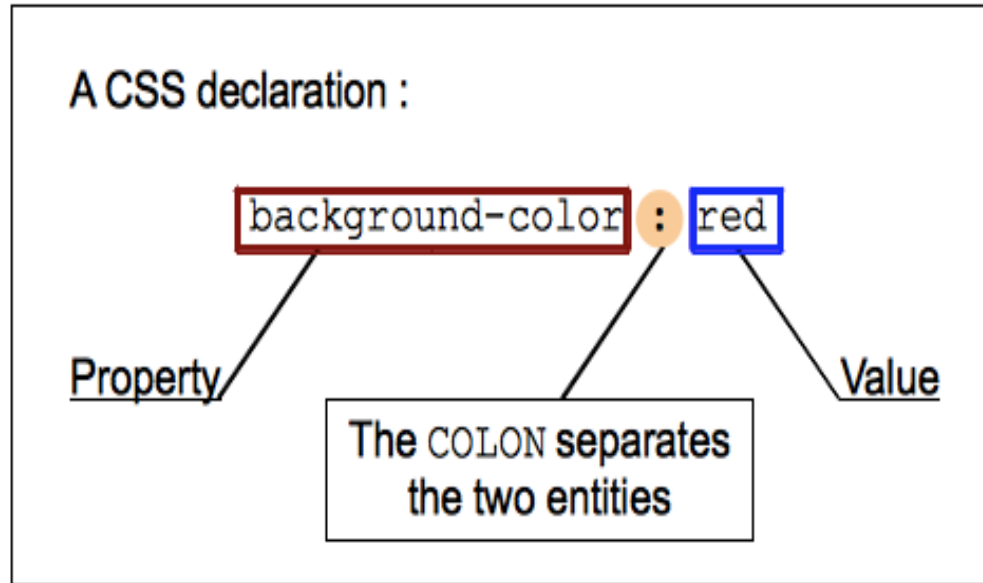
- **CSS** is a language that describes the style of an HTML document
 - CSS stands for **C**ascading **S**tyle **S**heets
 - CSS describes how HTML elements should be rendered on screen, on paper, in speech, or on other media.
 - CSS is one of the core languages of the open Web and is standardized across Web browsers according to the W3C specification

2. CSS Syntax

- The basic goal of the **CSS** language is to allow a browser engine to paint elements of the page with specific features, like colors, positioning, or decorations.
- **CSS** Syntax includes:
 - The **property** which is an identifier, that is a human-readable name, that defines which feature is considered.
 - The **value** which describe how the feature must be handled by the engine. Each property has a set of valid values, defined by a formal grammar, as well as a semantic meaning, implemented by the browser engine.

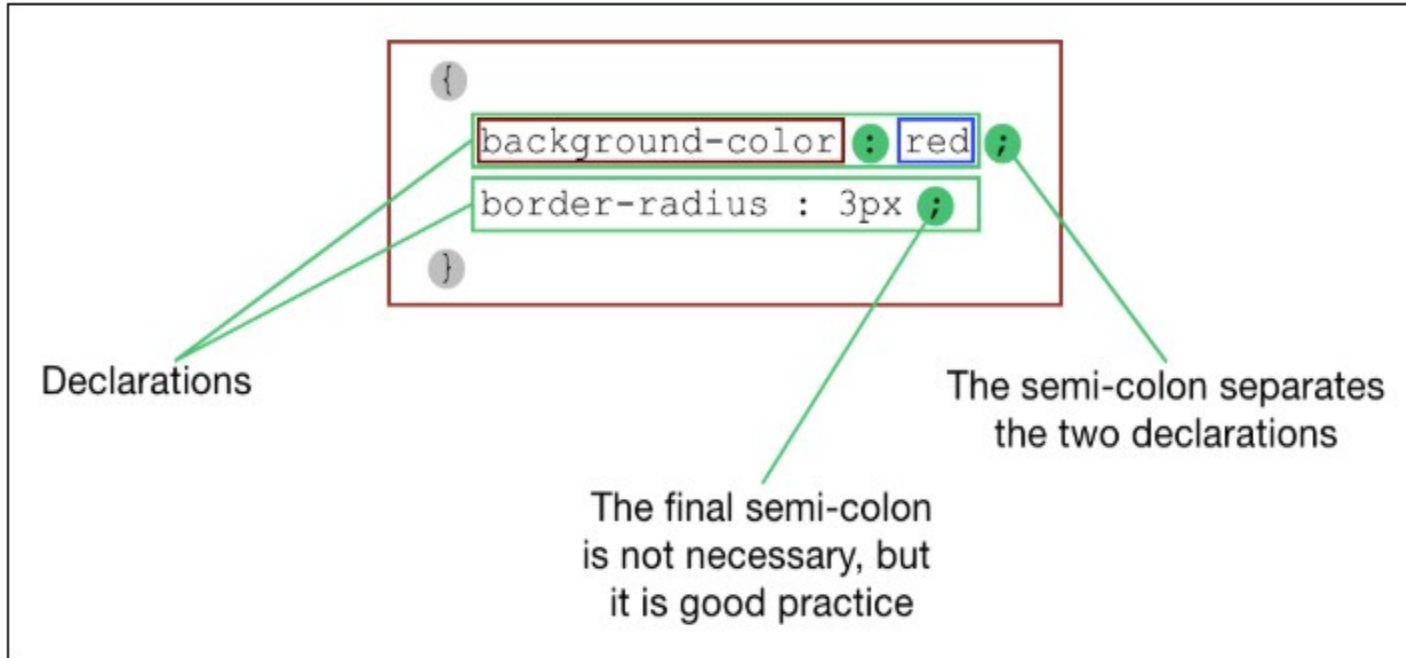
2. CSS Syntax

- **CSS Declarations:** A property and value pair is called a **declaration**



2. CSS Syntax

- **CSS Declarations block:** Declarations are grouped in **blocks**



4. Browser support

- Visit link below to see the list contains all CSS properties and how each property is supported in the different browsers

https://www.w3schools.com/cssref/css3_browsersupport.asp

5. Apply CSS to HTML

- CSS can be added to HTML elements in 3 ways:
 - **Inline** - by using the style attribute in HTML elements
 - **Internal** - by using a <style> element in the <head> section
 - **External** - by using an external CSS file

5. Apply CSS to HTML

- **Inline style:** An inline CSS is used to apply a unique style to a single HTML element

❖ *Syntax*

```
<element style="style_definitions">
```

Value	Description
<i>style_definitions</i>	One or more CSS properties and values separated by semicolons (e.g. style="color:blue;text-align:center")

- ❖ **Note:** The style attribute will override any style set globally, e.g. styles specified in the <style> tag or in an external style sheet.

5. Apply CSS to HTML

- **Internal style:** An internal CSS is used to define a style for a single HTML page. It is defined in the **<head>** section of an HTML page, within a **<style>** element

```
<head>
  <meta charset="utf-8">
  <title>My CSS experiment</title>
  <style>
    h1 {
      color: blue;
      background-color: yellow;
      border: 1px solid black;
    }

    p {
      color: red;
    }
  </style>
</head>
```

5. Apply CSS to HTML

➤ External style:

- An external style sheet is used to define the style for many HTML pages.
- With an external style sheet, you can change the look of an entire web site, by changing one file.
- Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

mystyle.css

```
body {
    background-color: lightblue;
}

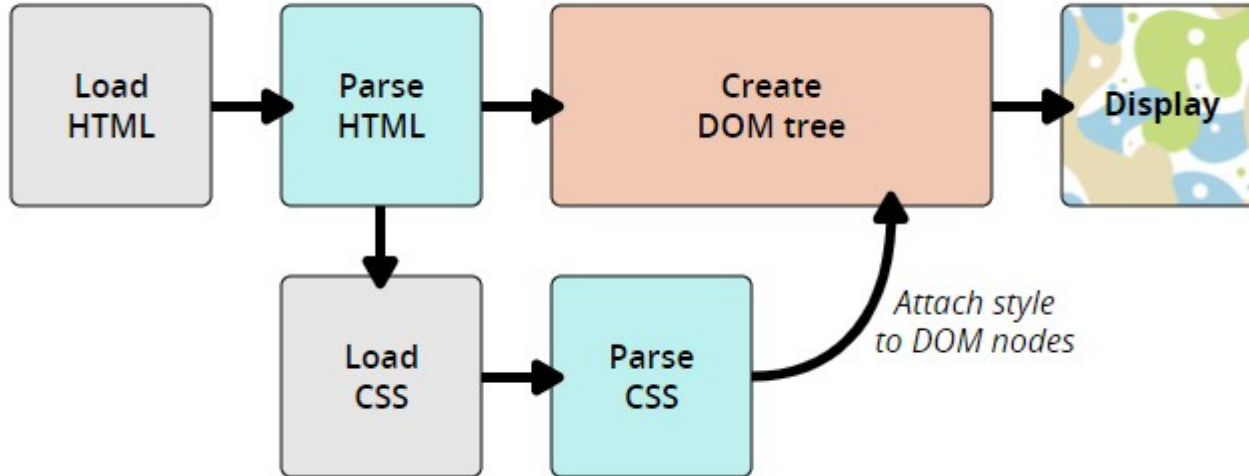
h1 {
    color: navy;
    margin-left: 20px;
}
```

5. Apply CSS to HTML

- **Cascading order:** All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:
 1. Inline style.
 2. External and internal style sheets.
 3. Browser default

6. How CSS works?

➤ Process loading a webpage



Session 2

CASCADE AND INHERITANCE

Overview

- 1. Cascade**
- 2. Inheritance**
- 3. Control inheritance**
- 4. Reset all property**

1. Cascade

- **Cascade:** Is an algorithm that defines how to combine property values originating from different sources.

```
<h1>This is my heading.</h1>
```

+

```
h1 {  
  color: red;  
}  
h1 {  
  color: blue;  
}
```



This is my heading.

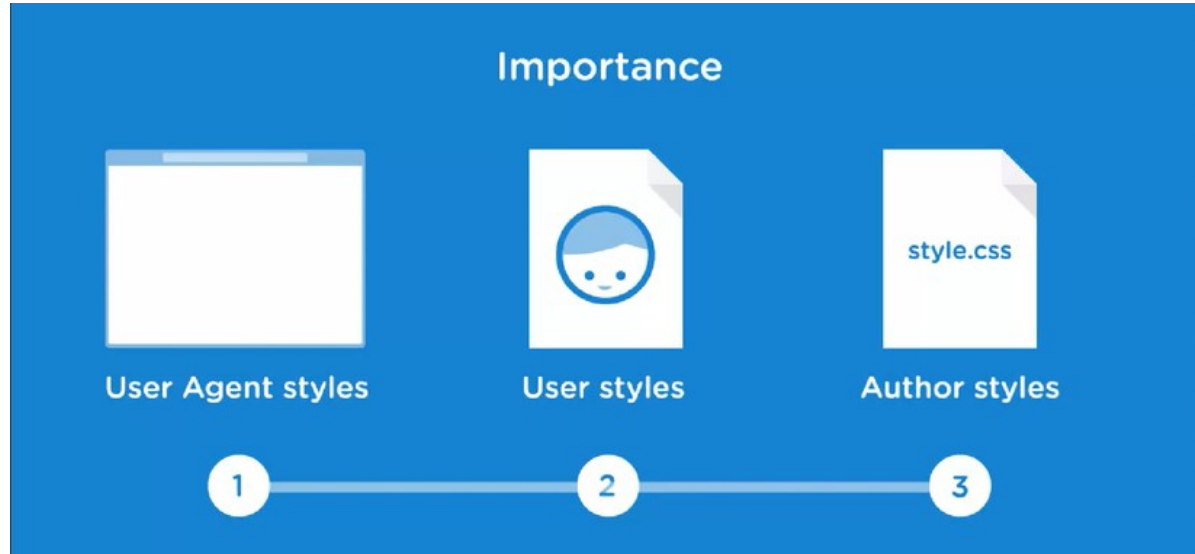
1. Cascade

- There are three main concepts that control the order in which CSS declarations are applied:
 1. *Importance*
 2. *Specificity*
 3. *Source order*

=> **Importance** is the most important. If two declarations have the same importance, the **specificity** of the rules decide which one will apply. If the rules have the same specificity, then **source order** controls the outcome.

1. Cascade

➤ Importance:



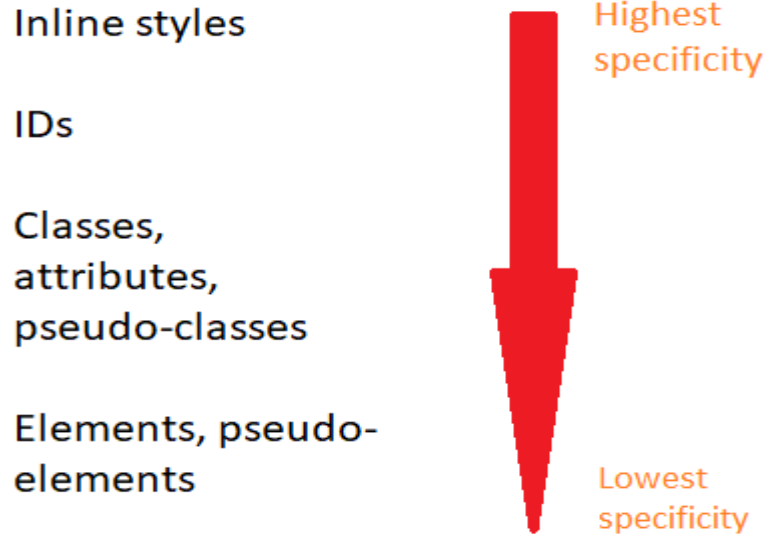
1. Cascade

➤ Importance:

1. **User-agent stylesheets:** The author of the page defines the styles for the document using one or more stylesheets, which define the look and feel of the website — its theme.
2. **Author stylesheets:** The author of the page defines the styles for the document using one or more stylesheets, which define the look and feel of the website — its theme.
3. **User stylesheets:** The user (or reader) of the web site can choose to override styles in many browsers using a custom **user stylesheet** designed to tailor the experience to the user's wishes.

1. Cascade

- **Specificity** is how the browser decides which rule applies if multiple rules have different selectors, but could still apply to the same element.



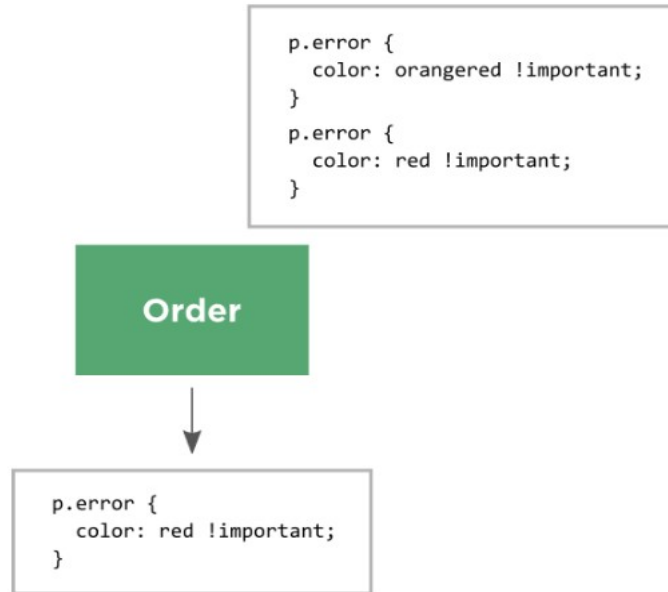
1. Cascade

➤ Specificity rules:

- ***Equal specificity: the latest rule counts*** - If the same rule is written twice into the external style sheet, then the lower rule in the style sheet is closer to the element to be styled, and therefore will be applied
- ***ID selectors have a higher specificity than attribute selectors***
- ***Contextual selectors are more specific than a single element selector***
 - The embedded style sheet is closer to the element to be styled. So in the following situation
- ***The universal selector and inherited values have a specificity of 0*** - *, body * and similar have a zero specificity. Inherited values also have a specificity of 0.

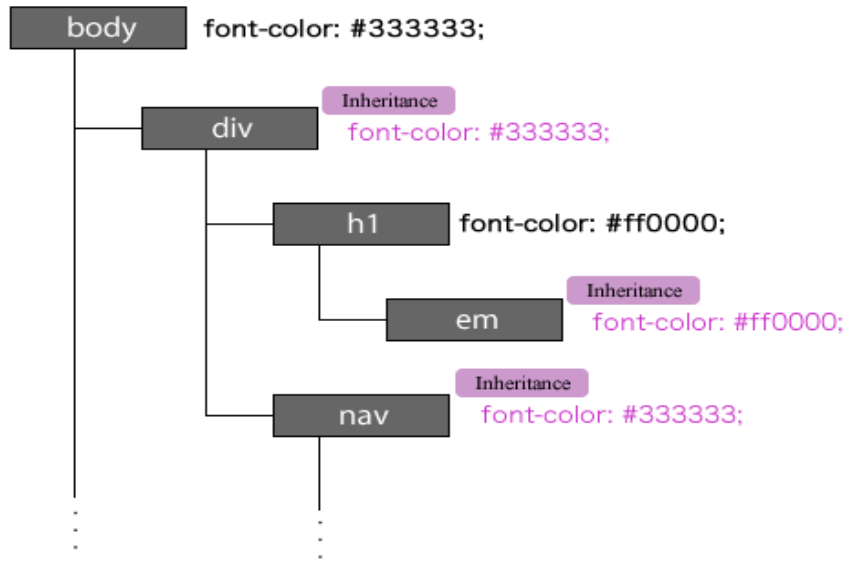
1. Cascade

- **Source order:** If the CSS rules have the same *Importance* and *Specificity*, we will consider the order of their appearance - which CSS rule that is written later will be preferred over CSS that is written first.



2. Inheritance

- **Inheritance** works on a property by property basis. When you set properties on a selector in CSS, they're either inherited by all the children of that selector or they're not



4. Reset all properties.

- The **all** property in CSS resets all of the selected element's properties.
- **Value:**
 - ***initial***: Resets all of the selected element's properties to their initial values as defined in the CSS spec.
 - ***inherit***: The selected element inherits all of its parent element's styling, including styles that are not normally inheritable.
 - ***unset***: The selected element inherits any inheritable values passed down from the parent element. If no inheritable value is available, the initial value from the CSS spec is used for each property.

Session 3

SELECTORS

- 1. CSS selector**
- 2. Type, class, and ID selectorsInheritance**
- 3. Attribute selectors**
- 4. Pseudo-classes and pseudo-elements**
- 5. Combinators**

1. CSS selector

- **CSS selectors** are used to "find" (or select) the HTML elements you want to style
- We can divide CSS selectors into five categories:
 1. Simple selector (type, id, class)
 2. Combinator selector
 3. Pseudo classes & pseudo elements
 4. Attribute selectors

2. Type, id, class selectors

- ***The element selector:*** Selects HTML elements based on the element name
- ***Id selectors:*** Uses the id attribute of an HTML element to select a specific element.
- ***Class selectors:*** Selects HTML elements with a specific class attribute

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


```
#para1 {  
    text-align: center;  
    color: red;  
}
```

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

3. Pseudo class & Pseudo element

- **Pseudo class:** Is used to define a special state of an element.

Syntax




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

Selector	Example	Example description
<u>:active</u>	a:active	Selects the active link
<u>:checked</u>	input:checked	Selects every checked <input> element
<u>:disabled</u>	input:disabled	Selects every disabled <input> element
<u>:empty</u>	p:empty	Selects every <p> element that has no children
<u>:enabled</u>	input:enabled	Selects every enabled <input> element

3. Pseudo class & Pseudo element

- **Pseudo element:** Is used to style specified parts of an element.

Syntax

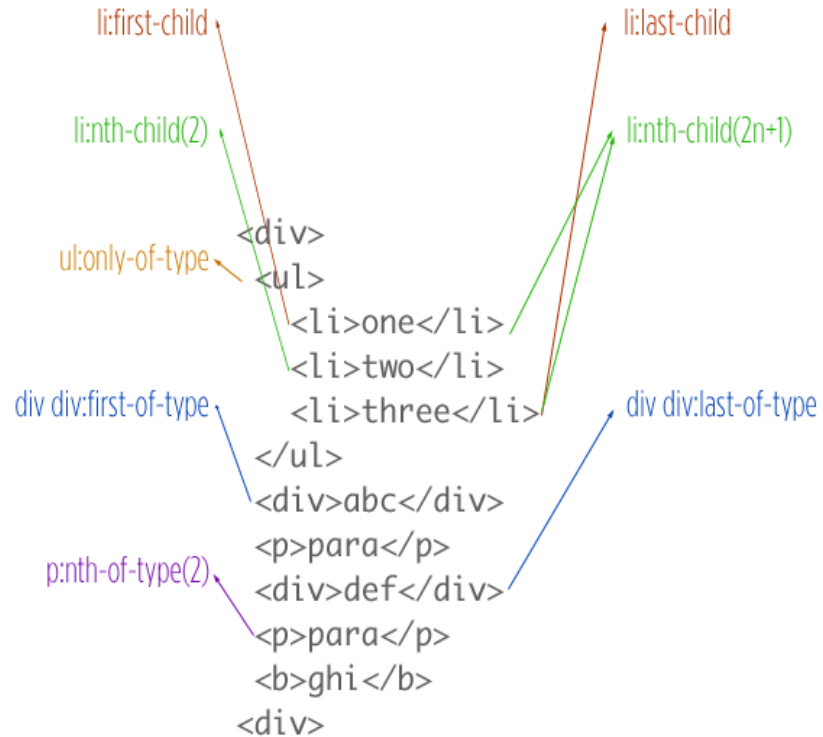


```
selector::pseudo-element {  
  property:value;  
}
```

Selector	Example	Example description
<u>::after</u>	p::after	Insert something after the content of each <p> element
<u>::before</u>	p::before	Insert something before the content of each <p> element
<u>::first-letter</u>	p::first-letter	Selects the first letter of each <p> element
<u>::first-line</u>	p::first-line	Selects the first line of each <p> element
<u>::selection</u>	p::selection	Selects the portion of an element that is selected by a user

3. Pseudo class & Pseudo element

Pseudo element Example

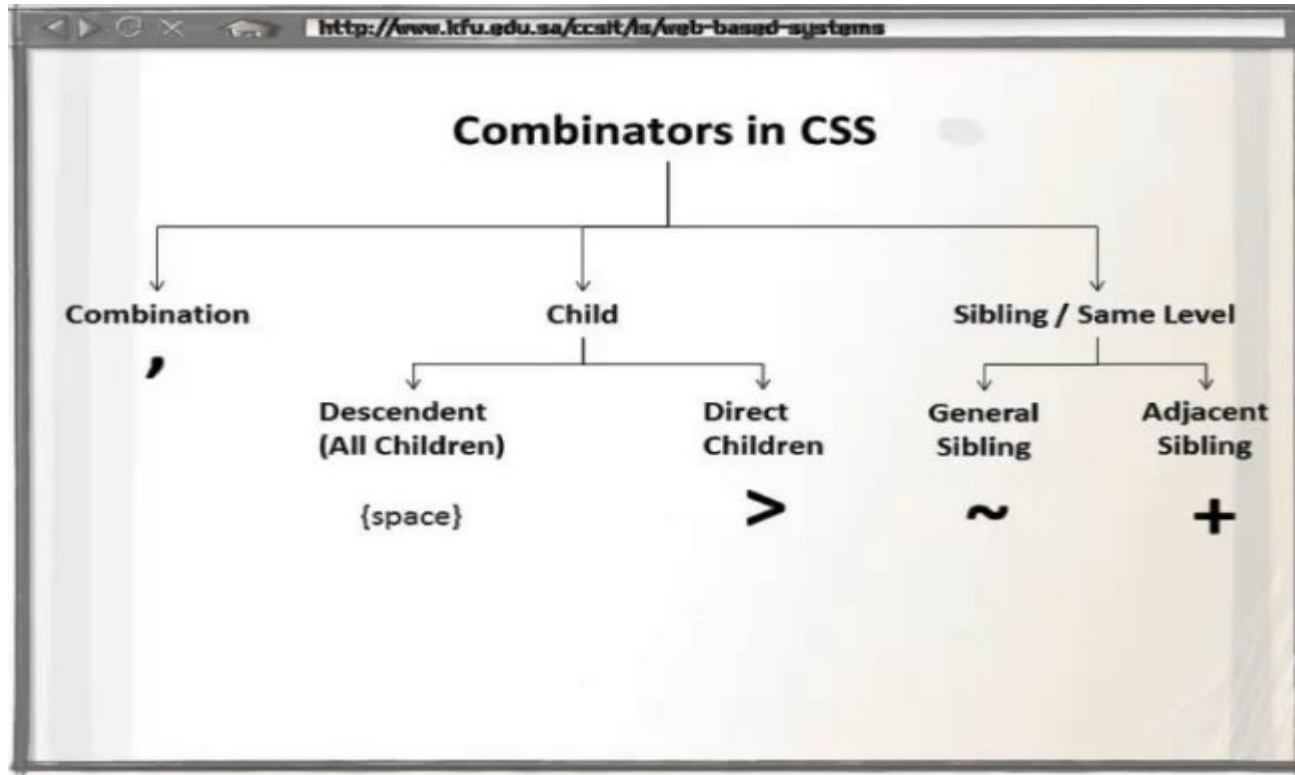


4. Combinator selectors

- **Combinator selector** is used to select element base on the relationship.
- There are four different combinators in CSS:
 - *Descendant selector (space)*
 - *Child selector (>)*
 - *Adjacent sibling selector (+)*
 - *General sibling selector (~)*

4. Combinator selectors

➤ Combinator selector



Session 4

BOX MODEL

- 1. Block and Inline boxes**
- 2. Inner and outer display types**
- 3. The CSS box model**
- 4. Use browser DevTools to view the box model**
- 5. Margins, padding, and borders**
- 6. The box model and inline boxes**
- 7. Using display: inline-block**

1. Block and Inline boxes

➤ **Block box:**

- The box will break onto a new line.
- The width and height properties are respected.
- Padding, margin and border will cause other elements to be pushed away from the box

BLOCK:



1. Block and Inline boxes

➤ Inline box:

- The box will not break onto a new line.
- The width and height properties will not apply.
- Vertical padding, margins, and borders will apply but will not cause other inline boxes to move away from the box.
- Horizontal padding, margins, and borders will apply and will cause other inline boxes to move away from the box

INLINE:



2. Inner and outer display types

➤ Inner display types:

- **Table:** These elements behave like HTML <table> elements. It defines a block-level box.
- **Flex:** The element behaves like a block element and lays out its content according to the flexbox model.
- **Grid:** The element behaves like a block element and lays out its content according to the grid model.

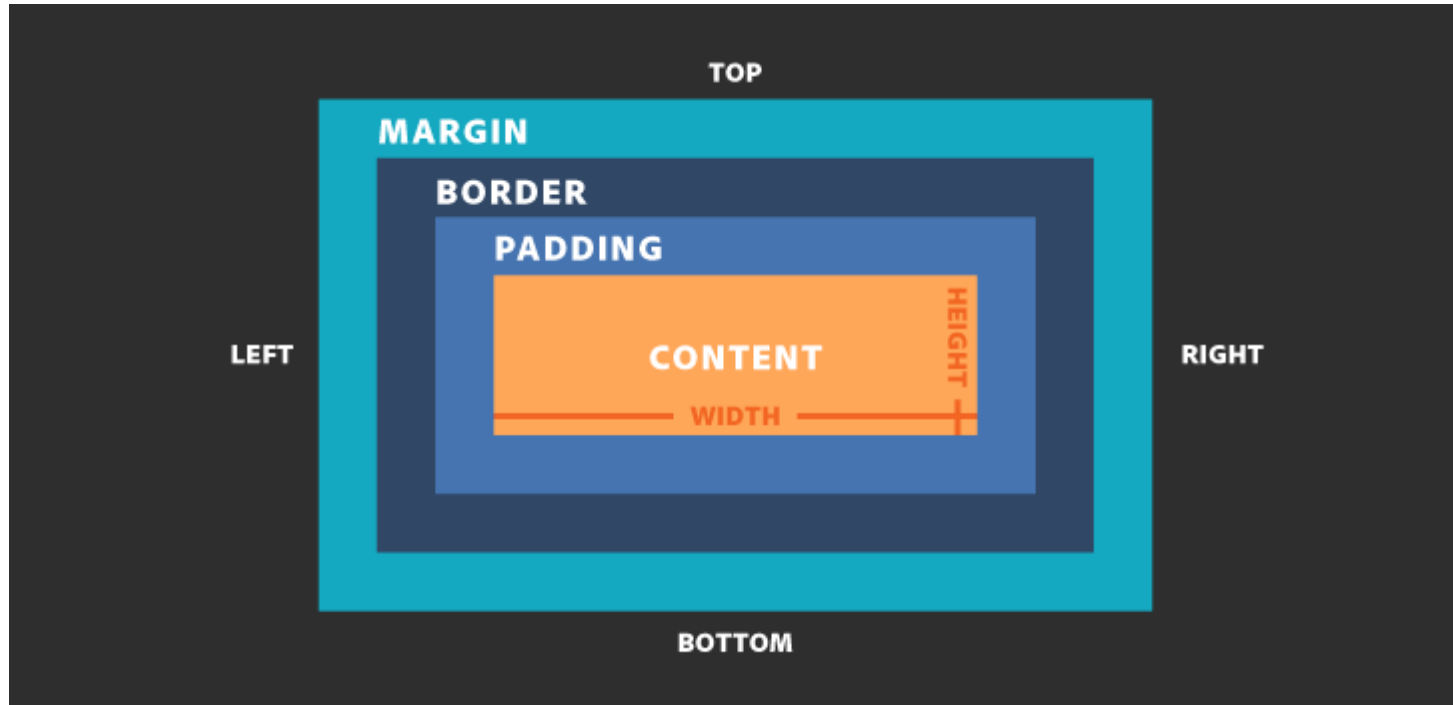
2. Inner and outer display types

➤ Outer display types:

- ***block***: The element generates a block element box, generating line breaks both before and after the element when in the normal flow.
- ***inline***: The element generates one or more inline element boxes that do not generate line breaks before or after themselves. In normal flow, the next element will be on the same line if there is space.
- ***run-in***: The element generates a run-in box. If the adjacent sibling of the element defined as display: run-in box is a block box, the run-in box becomes the first inline box of the block box that follows it.

3. The alternative CSS Box Model

➤ Box model

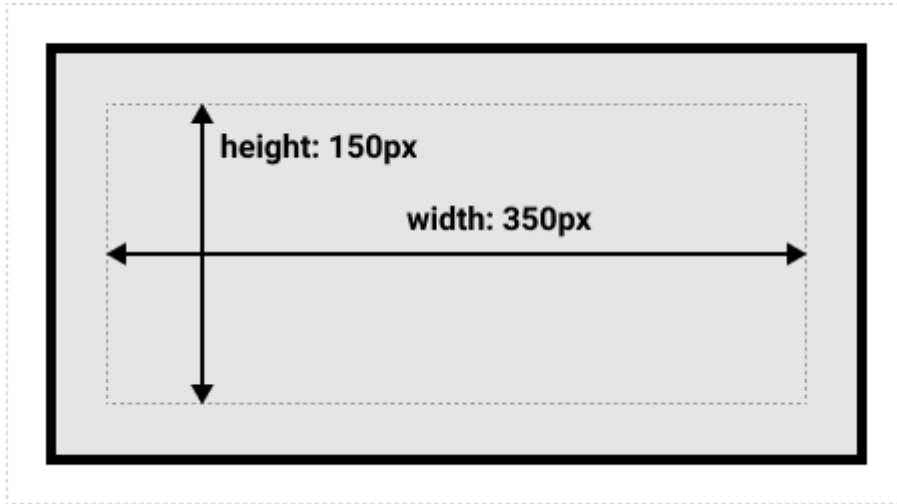


3. The CSS Box Model

- **Making up a block box in CSS we have:**
- ***Content box:*** The area where your content is displayed, which can be sized using properties like width and height.
 - ***Padding box:*** The padding sits around the content as white space; its size can be controlled using padding and related properties.
 - ***Border box:*** The border box wraps the content and any padding. Its size and style can be controlled using border and related properties.
 - ***Margin box:*** The margin is the outermost layer, wrapping the content, padding and border as whitespace between this box and other elements. Its size can be controlled using margin and related properties.

3. The CSS Box Model

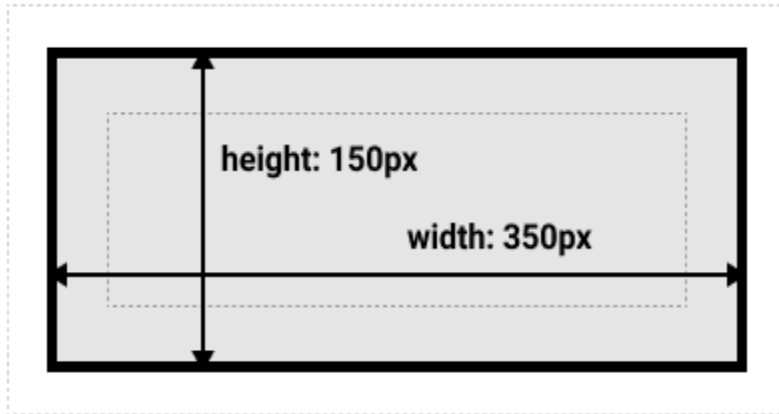
- In the **Standard box model**, if you give a box a width and a height attribute, this defines the width and height of the content box. Any padding and border is then added to that width and height to get the total size taken up



The space taken up by our box using the standard box model will actually be 410px ($350 + 25 + 25 + 5 + 5$), and the height 210px ($150 + 25 + 25 + 5 + 5$), as the padding and border are added to the width used for the content box.

3. The CSS Box Model

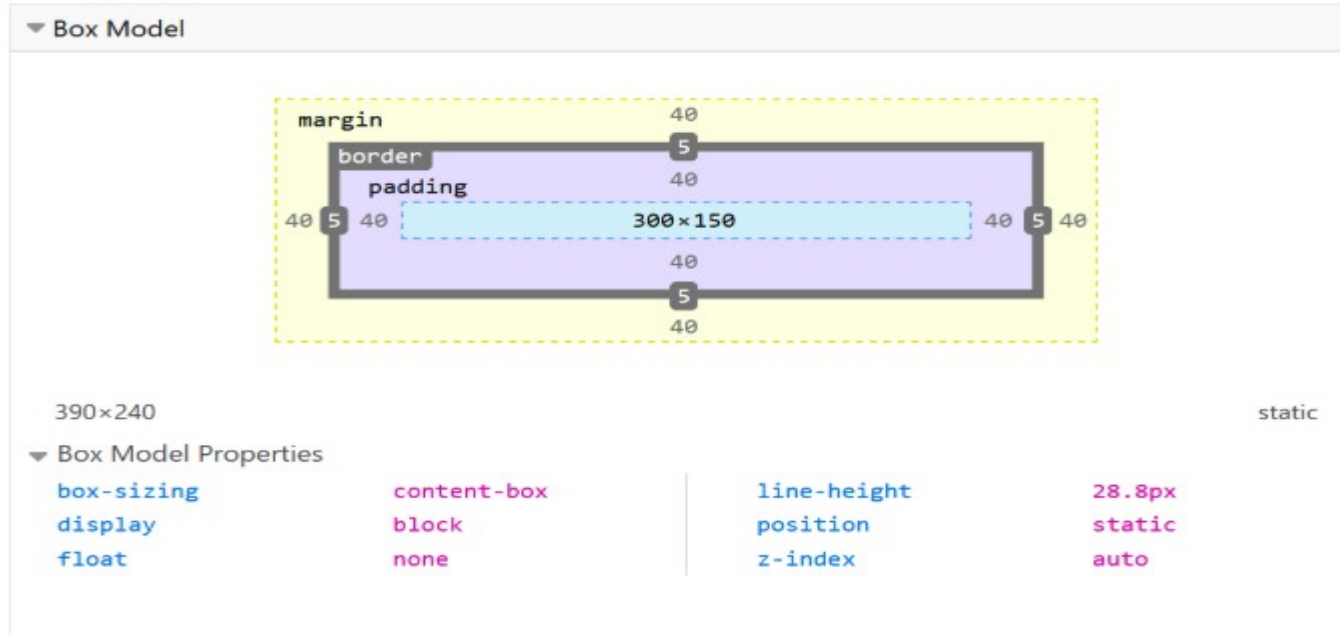
- In the **Alternative box model**, any width is the width of the visible box on the page, therefore the content area width is that width minus the width for the padding and border.



width = 350px, height = 150px

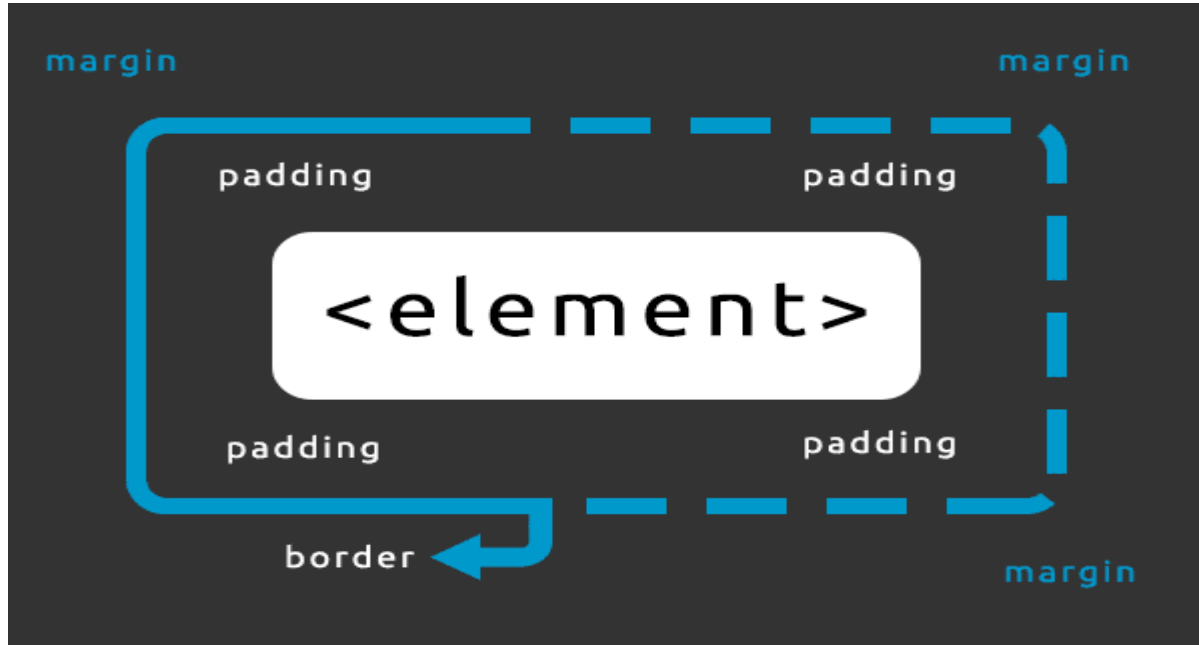
4. Use browser DevTools to view the box model

- If you inspect an element in browser DevTools, you can see the size of the element plus its margin, padding, and border. Inspecting an element in this way is a great way to find out if your box is really the size you think it is



5. Margins, paddings and borders

➤ Take a look



5. Margins, paddings and borders

- **Margin:** Is an invisible space around your box. It pushes other elements away from the box. Margins can have positive or negative values. Setting a negative margin on one side of your box can cause it to overlap other things on the page
- We can control all margins of an element at once using the margin property, or each side individually using the equivalent longhand properties:
 - ***margin-top***
 - ***margin-right***
 - ***margin-bottom***
 - ***margin-left***

5. Margins, paddings and borders

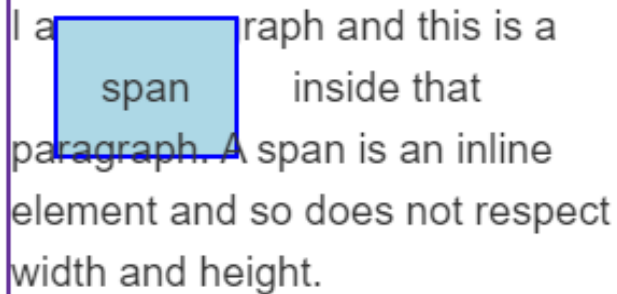
- **Padding:** Sits between the border and the content area. Unlike margins you cannot have negative amounts of padding, so the value must be 0 or a positive value, it is typically used to push the content away from the border.
- We can control all margins of an element at once using the margin property, or each side individually using the equivalent longhand properties:
 - ***padding-top***
 - ***padding-right***
 - ***padding-bottom***
 - ***padding-left***

5. Margins, paddings and borders

- **Border:** is drawn between the margin and the padding of a box.
- To set the properties of each side individually, you can use:
 - ***border-top***
 - ***border-right***
 - ***border-bottom***
 - ***border-left***
- To set the width, style, or color of all sides, use the following:
 - ***border-width***
 - ***border-style*** (*dotted, dashed, solid , double, groove, inset, outset,..*)
 - ***border-color***

6. The box model and inline boxes

- With an **Inline box**, that the width and height are ignored. The margin, padding, and border are respected but they do not change the relationship of other content to our inline box and so the padding and border overlaps other words in the paragraph.

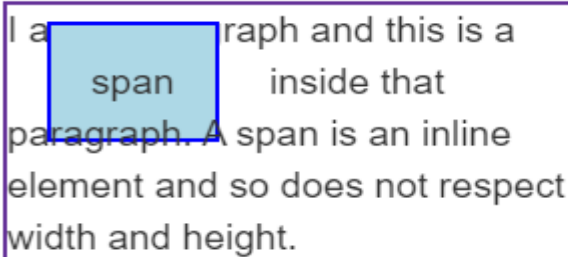


I a span raph and this is a paragraph. A span is an inline element and so does not respect width and height.

```
span {  
  margin: 20px;  
  padding: 20px;  
  width: 80px;  
  height: 50px;  
  background-color: lightblue;  
  border: 2px solid blue;  
}
```

7. Using display: inline-block

- Value “**inline-block**” provides a middle ground between inline and block.
 - The *width* and *height* properties are respected
 - *padding*, *margin*, and *border* will cause other elements to be pushed away from the box.



I a raph and this is a
span inside that
paragraph. A span is an inline
element and so does not respect
width and height.

```
span {  
  margin: 20px;  
  padding: 20px;  
  width: 80px;  
  height: 50px;  
  background-color: lightblue;  
  border: 2px solid blue;  
}
```

Session 5

OVERFLOW CONTENTS

- 1. What is overflow?**
- 2. Overflow property**

1. What is overflow?

- **Overflow:** Is a css property sets what to do when an element's content is too big to fit in its block formatting context.
- This demo below is ***default value*** of overflow property:

```
overflow: visible;
```

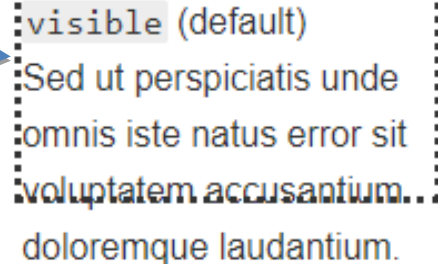
Michaelmas term lately over,
and the Lord Chancellor
sitting in Lincoln's Inn Hall.
Implacable November
weather. As much mud in the
streets as if the waters had
but newly retired from the
face of the earth.

2. Overflow property

➤ Value:

- **visible** (default): Content is not clipped and may be rendered outside the padding box.

```
1 p {  
2   width: 12em;  
3   height: 6em;  
4   border: dotted;  
5   overflow: visible; /* content is not clipped */  
6 }
```



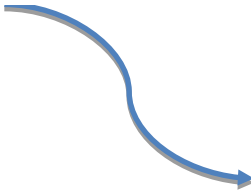
visible (default)
Sed ut perspiciatis unde
omnis iste natus error sit
voluptatem accusantium...
doloremque laudantium.

2. Overflow property

➤ Value:

- ***hidden***: Content is clipped if necessary to fit the padding box. No scrollbars are provided, and no support for allowing the user to scroll

```
1 | p { overflow: hidden; /* no scrollbars are provided */ }
```



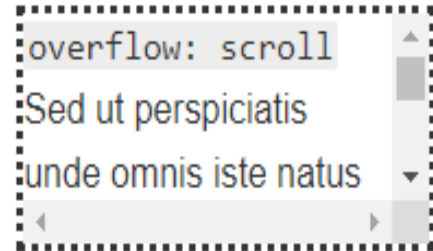
overflow: hidden
Sed ut perspiciatis unde
omnis iste natus error sit
voluntatem accusantium...

2. Overflow property

➤ Value:

- **scroll:** Content is clipped if necessary to fit the padding box. Browsers always display scrollbars whether or not any content is actually clipped, preventing scrollbars from appearing or disappearing as content changes

```
1 | p { overflow: scroll; /* always show scrollbars */ }
```

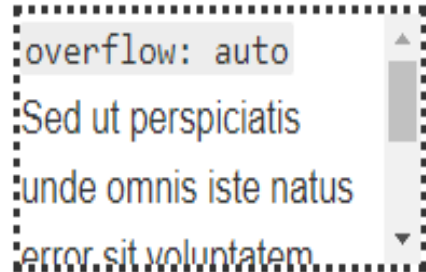


2. Overflow property

➤ Value:

- **auto:** Depends on the user agent. If content fits inside the padding box, it looks the same as visible, but still establishes a new block formatting context. Desktop browsers provide scrollbars if content overflows.

```
1 | p { overflow: auto; /* append scrollbars if necessary */ }
```



Session 6

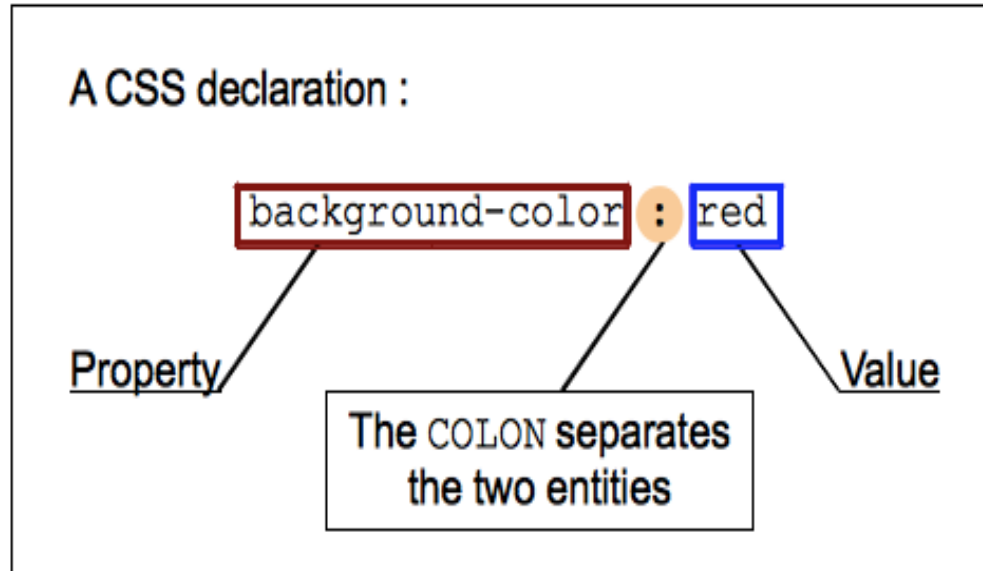
VALUES AND UNITS

Overview

- 1. CSS Value**
- 2. Distance unit**
- 3. Percentages**
- 4. Ems and rems**
- 5. Color**

1. CSS Value

- **CSS value:** Every CSS declaration includes a property / value pair. Depending on the property, the value can include a single integer or keyword, to a series of keywords and values with or without units



1. CSS Value

➤ Data type

▪ *Textual data type:*

- custom-ident

```
1  @keyframe validIdent {  
2      /* keyframes go here */  
3  }
```

- Pre-defined keywords as an ident:

```
.box {  
    float: left;  
}
```

1. CSS Value

➤ Data type

▪ *Textual data type:*

○ string:

```
1 | .item {  
2 |     grid-area: content;  
3 | }
```

○ url:

```
.box {  
    background-image: url("images/my-background.png");  
}
```


1. CSS Value

➤ Data type

▪ *Numeric data types*

- *Integer*: A whole number such as 1024 or -55
- *Number*: A decimal number
- *Dimension*: A number with a unit attached to it
- *Percentage*: A fraction of some other value

2. Distance unit

- **Distance unit (length unit):** There are 2 types of **distance** unit in CSS: Relative and Absolute
 - ❖ ***Absolute length units*** are fixed to a physical length

Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

2. Distance unit

➤ Distance unit (length unit):

❖ ***Relative length unit*** specify a length in relation to something else.

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

3. Percentage

- **Percentage** is a type that represents a fraction of some other value.
- It is always relative to another quantity.
- Each property that allows **percentages** also defines the quantity to which the percentage refers.
- This quantity can be a value of another property of the same element, the value of a property of an ancestor element, a measurement of a containing block, or something else.

3. Ems & rems

- **em** and **rem** are the two relative lengths you are likely to encounter most frequently when sizing anything from boxes to text.
 - **em** unit means *"my parent element's font-size"*: Relative to font size of the parent, in the case of typographical properties like font-size, and font size of the element itself, in the case of other properties like width.
 - **rem** unit means *"The root element's font-size"*: Relative to font size of the root element.
- If you change the `<html>` font-size in the CSS you will see that everything else changes relative to it — both rem- and em-sized text.

4. Colors

- **Colors** in CSS can be specified by the following methods:
 - Hexadecimal colors
 - RGB colors
 - RGBA colors
 - HSL colors
 - HSLA colors
 - Predefined/Cross-browser color names
 - With the **currentcolor** keyword

4. Colors

- An **RGB color** value is specified with the `rgb()` function, which has the following syntax: ***rgb(red, green, blue)***
- Each parameter (red, green, and blue) defines the intensity of the color and can be an integer between 0 and 255 or a percentage value (from 0% to 100%).

```
.one {  
  background-color: rgb(2, 121, 139);  
}  
  
.two {  
  background-color: rgb(197, 93, 161);  
}  
  
.three {  
  background-color: rgb(18, 138, 125);  
}
```



rgb(2, 121, 139)

rgb(197, 93, 161)

rgb(18, 138, 125)

4. Colors

- **HSL** stands for **Hue**, **Saturation**, and **Lightness** - and represents a cylindrical-coordinate representation of colors
- An **HSL color** value is specified with the `hsl()` function, which has the following syntax: ***hsl(hue, saturation, lightness)***
- **Hue** is a degree on the color wheel (from 0 to 360) - 0 (or 360) is red, 120 is green, 240 is blue. **Saturation** is a percentage value; 0% means a shade of gray and 100% is the full color. **Lightness** is also a percentage; 0% is black, 100% is white.

```
.one {  
  background-color: hsl(188, 97%, 28%);  
}
```

`hsl(188, 97%, 28%)`

4. Colors

- **HSLA color** values are an extension of HSL color values with an alpha channel - which specifies the opacity of the object.
- An **HSLA color** value is specified with the `hsla()` function, which has the following syntax: ***hsla(hue, saturation, lightness, alpha)***
- The **alpha** parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
.one {  
  background-color: hsla(188, 97%, 28%, .3);  
}  
  
.two {  
  background-color: hsla(321, 47%, 57%, .7);  
}
```

`hsla(188, 97%, 28%, .3)`

`hsla(321, 47%, 57%, .7)`

4. Colors


- **Predefined/Cross-browser color names:** 140 color names are predefined in the HTML and CSS color specification.



4. Colors

- With the **currentcolor** keyword: The **currentcolor** keyword refers to the value of the color property of an element.

```
#myDIV {  
  color: blue; /* Blue text color */  
  border: 10px solid currentcolor; /* Blue border color */  
}
```



This div element has a blue text color and a blue border.

Thank you

