Software Studio
軟體設計與實驗

CSS

Hung-Kuo Chu
Department of Computer Science
National Tsing Hua University

CS2410

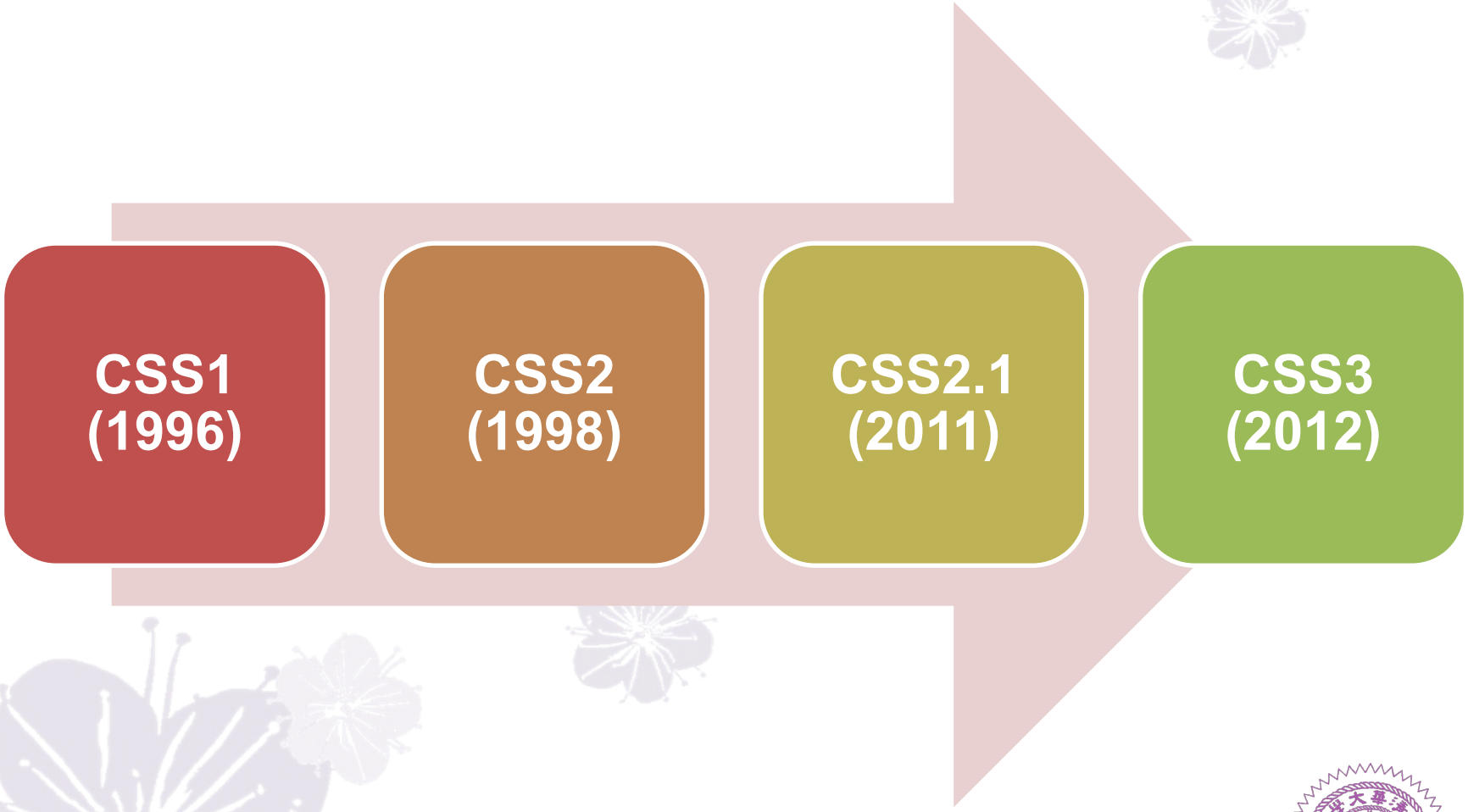
CSS

What is CSS?

- **C**ascading **S**tyle **S**heets by W3C.
- A style sheet language used for describing the appearance of a document written in a markup language (e.g., HTML, XML).
- CSS is designed primarily to enable the separation of **appearance** and **content.**
  - layout, colors, and fonts.

CSS demo

CSS History



CSS Style Rule

selector {property1 : value1 [; property2 : value2 [; …]]}

- Selector
  - Points to the HTML element you want to style.
- Declaration
  - The declaration block contains one or more declarations separated by semicolons.
  - Each declaration includes a CSS property name and a value, separated by a colon.
  - A declaration **always ends with a semicolon.**
- Example
  - body {color : white; background : red;}

CSS Usage#1

- In html
  - Define within the **<head>** block
  - Using **<style>** element

```
<!DOCTYPE html>
<html>
  <head>
    <style>
    h1 { color: white; text-align: center;}
    </style>
  </head>
</html>
```

CSS Usage#2

- Inline Styles
  - Use **style attribute**

```
<!DOCTYPE html>
<html>
  <body>
  <h1 style="color:blue;margin-left:30px;">Hello</h1>
  </body>
</html>
```

CSS Usage#3

- External Style Sheet
  - Use **<link>** element

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

html file

```
h1 {
        color: white;
        text-align: center;
}
```

mystyle.css

Color Property

- Support property:
  - background-color
  - color (text color)
- Support value:
  - Blue -> predefined color name (reference)
  - rgb(255, 99, 71) -> (range 0~255)
  - #ff6347 -> (hex format)
  - hsl(0, 100%, 50%)
  - rgba(255, 99, 71, 0.5) -> (alpha range 0~1)
  - hsla(0, 100%, 50%, 0.5)
- **Wrong syntax**
  - Blank space is NOT allowed! rgb (255, 0, 0)

x

Color Example

```
<!DOCTYPE html>
<html>
<body>
  <h1 style="background-color:rgb(160, 120, 130);">Hello1</h1>
  <h1 style="color:rgba(160, 120, 130, 0.7);">Hello2</h1>
  <h1 style="border:2px solid Tomato;">Hello3</h1>
  <h1 style="background-color:#ff6347;">Hello4</h1>
</body>
</html>
```

Hello1

Hello2

Hello3

Hello4

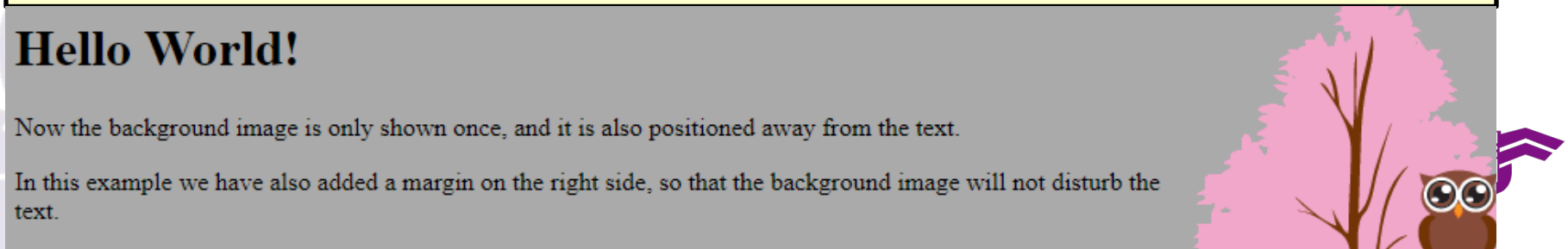
Background Property

- Support property
  - background-color
  - background-image
  - background-repeat
  - background-position
  - etc.
- Support value
  - Reference

Background Example

- The property order of **Shorthand** usage:
  - **{ background: color, image, repeat, attachment, position }**
  - It does not matter if one of the property values is missing

```
body {
    background: #aaaaaa url("img_tree.png") no-repeat fixed right
top;
}
```



Text Property

- Support property
  - color
  - align
  - decoration
  - indent
  - etc.
- Support value
  - Reference

Text Example

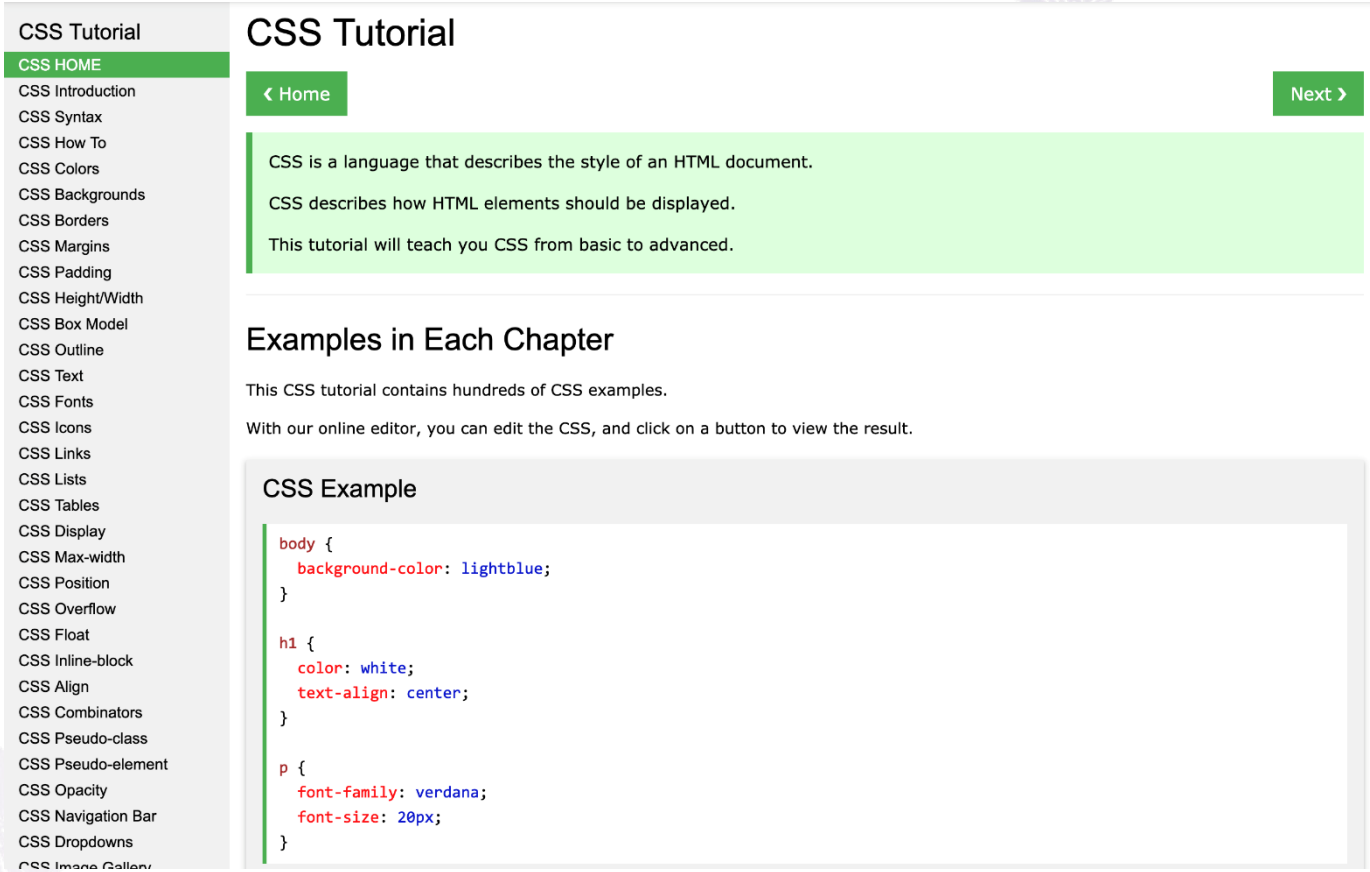
```
h1 {
    text-align: center;
    text-transform: uppercase;}
p {
    text-indent: 50px;
    text-align: justify;}
a {
    text-decoration: none;
    color: #008CBA;}
```

TEXT FORMATTING

This text is styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties. The paragraph is indented, aligned, and the space between characters is specified. The underline is removed from this colored "Try it Yourself" link.

Other Useful Property

- Font
- Icons
- Links
- Lists
- Tables
- …etc



Selectors

- In CSS, **selectors** are patterns used to select the element(s) you want to style.
- Basic Usage
  - element
    - select all the html elements you specify
  - #id
    - select elements with specified id
  - .class
    - select elements with specified class name

Selectors

CSS

```
h1{color: red;}

#df{color: green;}

.gf{color: blue;}
```

HTML

```
<h1>Hello</h1>
<h2 id="df">Hello</h2>
<h3 class="gf">Hello</h3>
```

Hello

Hello

Hello

Selectors

- CSS Combinators
  - element, element
    - div, p: all elements and all <p> elements
  - element element
    - div p: all <p> elements inside elements
  - element > element
    - div > p: all <p> elements where the parent is a element
- Advanced usage of **attribute selectors**
  - [attribute] -> all elements with target attribute
  - [attribute = target] -> all element with target value
- Reference

Example

Which selector will select <p> element?

```
<div>
  <h2>My name is Donald</h2>
  <p>I live in Duckburg.</p>
</div>
```

div p ?

div > p ?

```
<div>
  <span>
  <p>I will not be styled.</p>
  </span>
</div>
```

div p ?

div > p ?

Example

Which selector will select <p> element?

```
<div>
  <h2>My name is Donald</h2>
  <p>I live in Duckburg.</p>
</div>
```

div p => O

div > p => O

```
<div>
  <span>
  <p>I will not be styled.</p>
  </span>
</div>
```

div p => O

div > p => X

Pseudo-Class

- A **pseudo-class** is used to define a special state of an element.
  - Style an element when mouse hovers over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus
- Syntax

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

```
a:hover {
    color: #FF00FF;
}
```

- Reference

Pseudo-Element

- A CSS **pseudo-element** is used to style specified parts of an element.
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element
- Syntax

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

```
p::first-line {
    color: #ff0000;
}
```

- Reference

Cascade

- CSS is an acronym for ***Cascading*** *Style Sheets*, which indicates that the notion of the cascade is important.

```
<p class="better" id="winning">One selector to rule them all!</p>
```

```
p{color: red;}

.better{color: green;}

#winning{color: blue;}
```

- Who wins? blue!

Cascade - Order

- Three major rules to define the order:
  1. **Importance**
  2. **Specificity**
  3. **Source order**
- Earlier ones will overrule later ones
  - (1 > 2 > 3)

Cascade - Importance

- Declaration will always win over all others: **!important**.

```
selector {
    property:value !important;
}
```

```
<p class="better" id="winning">One selector to rule them all!</p>
```

```
p{color: red;}

.better{color: green !important;}

#winning{color: blue;}
```

Who wins?

green!

Cascade - Specificity

- Define the weight of the selectors
- **Thousands**: all declaration in style attribute
  - <h1 style="color: blue;">
- **Hundreds**: id selector
- **Tens**: class selector, attribute selector or pseudo-class
- **Ones**: element selector or pseudo-element

Cascade - Specificity

Selector	Thousands	Hundreds	Tens	Ones	Total specificity	Priority
h1	0	0	0	1	0001	#5
#id	0	1	0	0	0100	#2
h1 + p::first-letter	0	0	0	3	0003	#4
li > a[href*="en-US"] > .inline-warning	0	0	2	2	0022	#3
In style attribute	1	0	0	0	1000	#1

Cascade - Specificity

- Who win? Ans: blue
- #id win

```
<p class="better" id="winning">One selector to rule them all!</p>
```

```
p{color: red;} → 0001 #3

.better{color: green;} → 0010 #2

#winning{color: blue;} → 0100 #1
```

Cascade – Source Order

- later rules will win over earlier rules
- Who win? Ans: blue
- #winning win → later rules

```
<p id="better" id="winning">One selector to rule them all!</p>
```

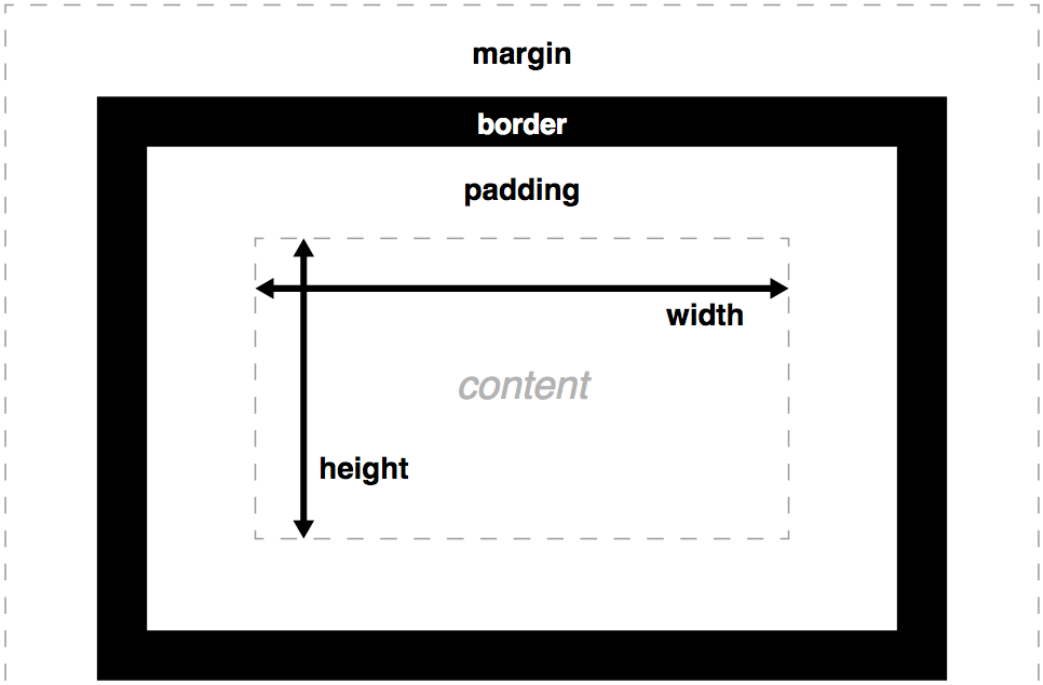
```
p{color: red;} → 0001 #3

#better{color: green;} → 0100 #2

#winning{color: blue;} → 0100 #1
```

Inheritance

- Child elements will inherit some common properties, such as font or color, from parent element
- Inherited properties by default
  - font, color, etc
- Not inherited properties by default
  - margin, padding, border, and background-image
- Check the reference for property inheritance.

Box Model in HTML

- Each element in HTML is represented as a rectangular box, with the box's **content**, **padding**, **border**, **margin**, **width** and **height**



Box Model Example

- What is unit px? Is there any other units?
- Reference

```
body {
  margin: 0;
}

#wrapper * {
  padding: 20px;
  margin: 40px;
  font-size: 20px;
  border: 20px solid rgba(0,0,0,0.5);
}
```

Box Model - Unit

- Frequently used units:
  - px → pixel
  - em → width of a capital letter M in current font-size
  - vw, vh → $^{1}/_{100}$ of the width and height of the viewport
- Unitless values
  - margin: 0
  - line-height: 1.5 → 1.5 times height to the font-size

Overflow Property

- Controls what happens to content that is too big to fit into an area.
- **auto**
  - add scroll bar
- **scroll**
  - add scroll bar
- **hidden**
  - hide the content out of the box
- **visible**
  - show the content out of the box
- Reference

Examples

```
div.ex1 {
  overflow: auto;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris tempus turpis id ante mollis dignissim. Nam sed dolor non tortor lacinia lobortis id dapibus nunc. Praesent

```
div.ex1 {
  overflow: hidden;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris tempus turpis id ante mollis dignissim. Nam sed dolor non tortor lacinia lobortis id dapibus nunc. Praesent iaculis

```
div.ex1 {
  overflow: visible;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris tempus turpis id ante mollis dignissim. Nam sed dolor non tortor lacinia lobortis id dapibus nunc. Praesent iaculis tincidunt augue. Integer efficitur sem eget risus cursus, ornare venenatis augue hendrerit. Praesent non elit metus. Morbi vel sodales ligula.

Normal Flow of Content Rendering

- The content are rendered in the following **normal flow** by default:
  - Block elements are laid out vertically
  - Inline elements are laid out horizontally
- Three ways to change normal flow
  - **display** property
  - **float** property
  - **position** property

Display Property

- The values of **display** property:
- **block**
  - content before and after the box appears on a separate line
- **inline**
  - content flows with document's text (surrounding text effect)
  - can not set width and height
- **inline-block**
  - as inline type but can change width/height without adding new line before and after block
- Reference

Examples

- display: inline; width: 60px; → no effect

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris tempus turpis id ante mollis dignissim. Nam sed dolor non tortor lacinia lobortis id dapibus nunc.

- display: block;

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Mauris tempus turpis id ante mollis dignissim.
Nam sed dolor non tortor lacinia lobortis id dapibus nunc.

- display: inline-block; width: 60px

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Mauris
tempus
turpis id
ante
mollis
dignissim.Nam sed dolor non tortor lacinia lobortis id
dapibus nunc.

Float Property

- The values of **float** property:
- **left**
  - The element floats to the left of its container
- **right**
  - The element floats to the right of its container
- **none** (default)
  - The element does not float (will be displayed just where it occurs in the text).
- **inherit**
  - The element inherits the float value of its parent
- Assign Reading: Float and Clear

Example

Before: Normal flow

```
<h1>2 column layout example</h1>
<div>
  <h2>First column</h2>
  <p> Lorem ipsum ...</p>
</div>
<div>
  <h2>Second column</h2>
  <p>Nam ...</p>
</div>
```

2 column layout example

First column

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate.

Second column

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut.

Example

After

```
div:nth-of-type(1) {
  width: 48%;
  float: left;
}

div:nth-of-type(2) {
  width: 48%;
  float: right;
}
```

div:nth-of-type(n):
Selects every element that is the n-th element of its parent

2 column layout example

First column

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate.

Second column

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut.

Position Property

- Specifies how an element is positioned in a document.
- **Static** (default)
  - Normal Flow
- **Relative**
- **Fixed**
- **Absolute**
- Reference

position: Normal flow;

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

This div element has position: relative;

Position - relative

- The element is positioned **relative** to its normal (Normal flow) position
- **left**: value
  - the contents shift on the right direction
- **right**: value
  - the contents shift on the left direction
- **bottom**: value
  - the contents shift on the up direction
- **top**: value
  - the contents shift on the down direction
- Empty region will not be filled

Position – relative

```
div.relative {
    position: relative;
    left: 30px;
    bottom: 50px;
    border: 3px solid #73AD21;
}
```

position: Normal flow;

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

This div element has position: relative;

position: relative;

This div element has position: relative;

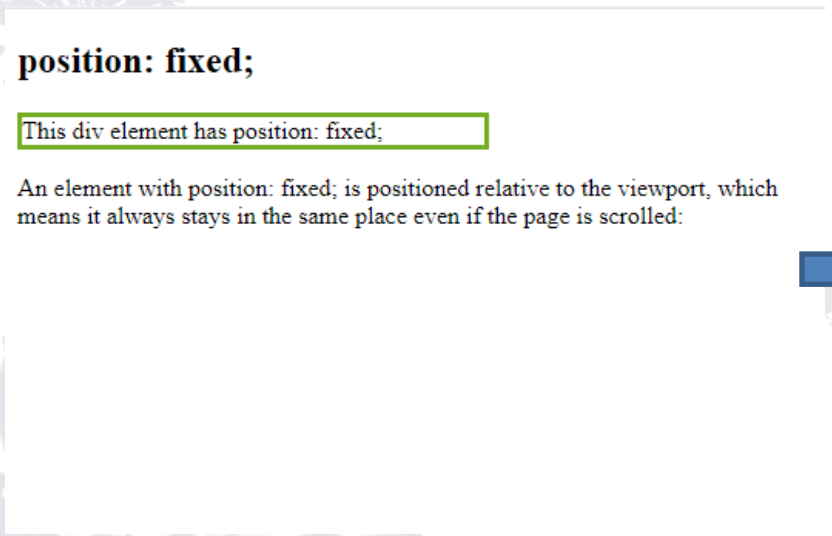
An element with position: relative; is positioned relative to its normal position:

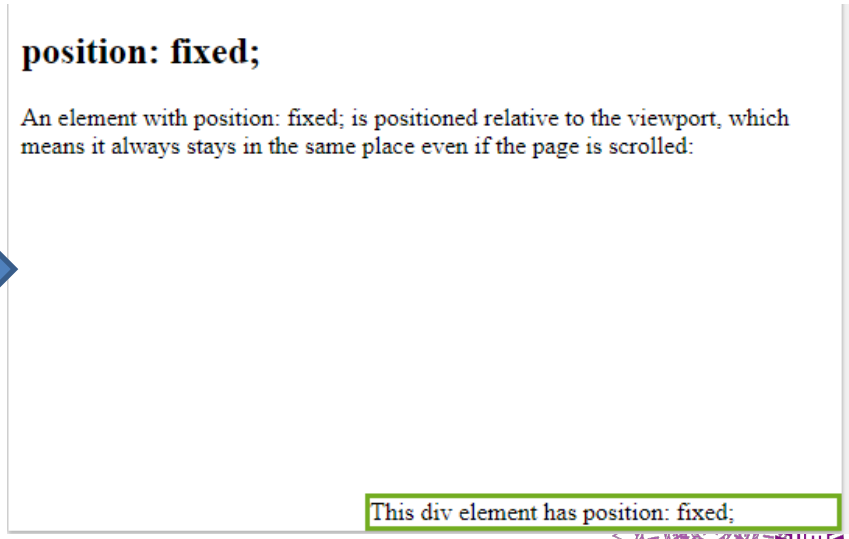
Position – fixed

- The element is positioned relative to the **viewport** (i.e., **browser window**), which means it **always stays in the same place even if the page is scrolled**.
- A fixed element does not leave a gap in the page where it would normally have been located.
- The following element will fill the gap.

Position - fixed

```
div.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 3px solid #73AD21;
}
```





Position - absolute

- An element with **position: absolute**; is positioned relative to the nearest positioned ancestor (relative to body if no ancestor)
- Positioned ancestor cannot be static

Position - absolute

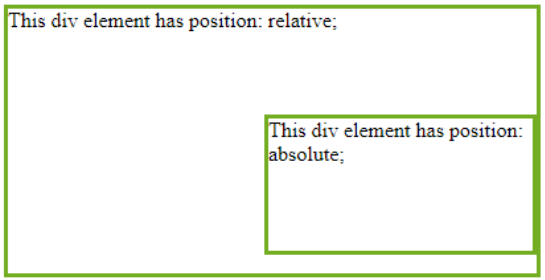
```
<div class="relative">This div element has position: relative;
  <div class="absolute">This div element has position:
absolute;</div>
</div>
```

```
div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #73AD21;}
```

```
div.relative {
    position: relative;
    width: 400px;
    height: 200px;}
```

position: absolute;

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



RWD

What is RWD?

- **R**esponsive **W**eb **D**esign is a dynamic mechanism that makes your web page look good on all devices.
- Responsive web design uses only HTML and CSS.
- **Responsive web design does not involve JavaScript.**

What is RWD?

- Same web page (content) adapts to different display layouts.

Web Page with RWD

width: 1200px

width: 800px

width: 500px

Link: https://www.nthu.edu.tw/

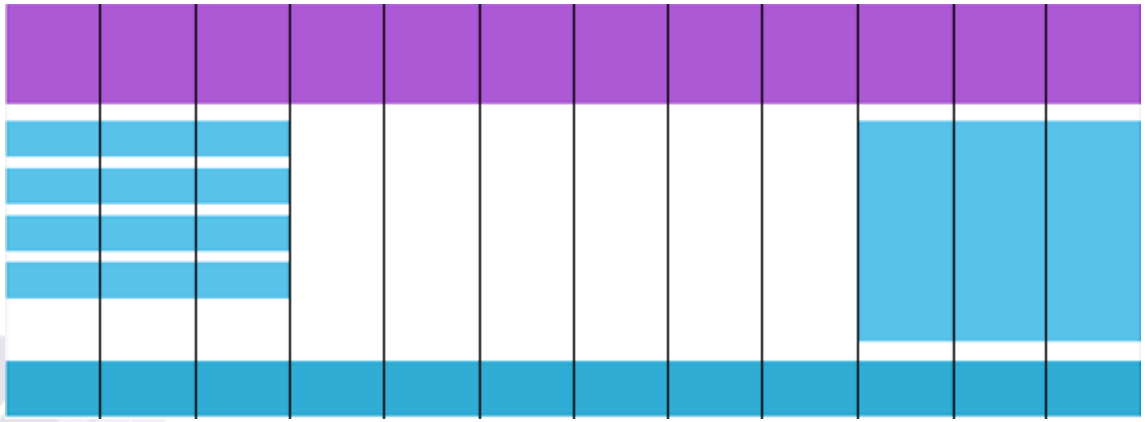
Common Way to Do RWD

- Grid
- Flexbox
- Bootstrap
  - A powerful front-end library and HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.
  - You will meet bootstrap in the 3rd LAB!

Bootstrap

Grid

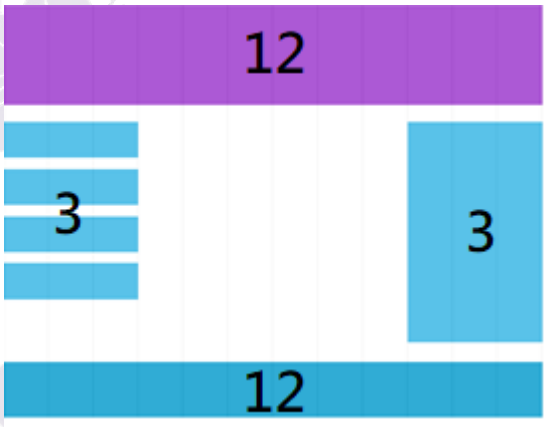
- A responsive grid-view often has 12 columns.
- Total width of 100%

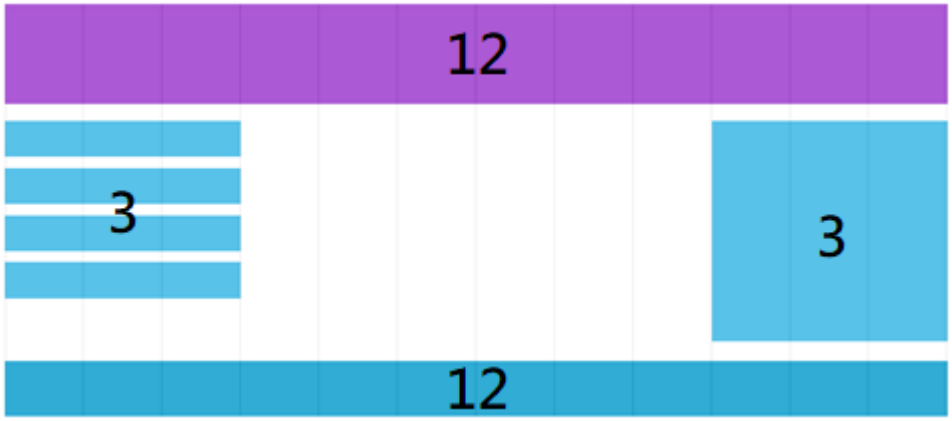


Reference : RWD Grid

Grid

- Shrink and expand as you resize the browser window.

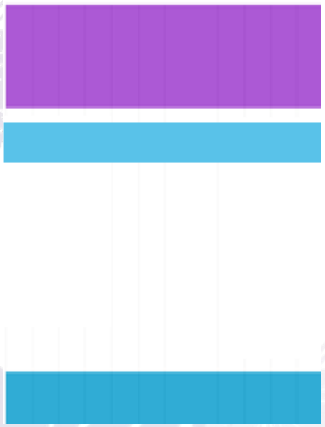


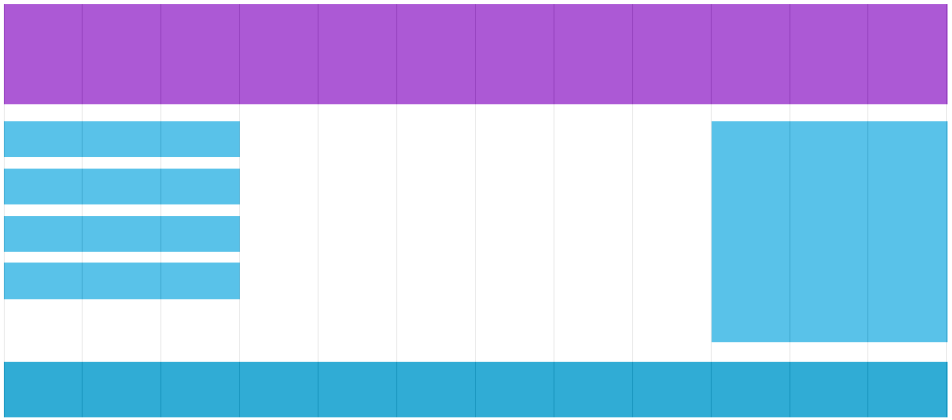


Reference : RWD Grid

Grid

- Or define different layouts for different window size.

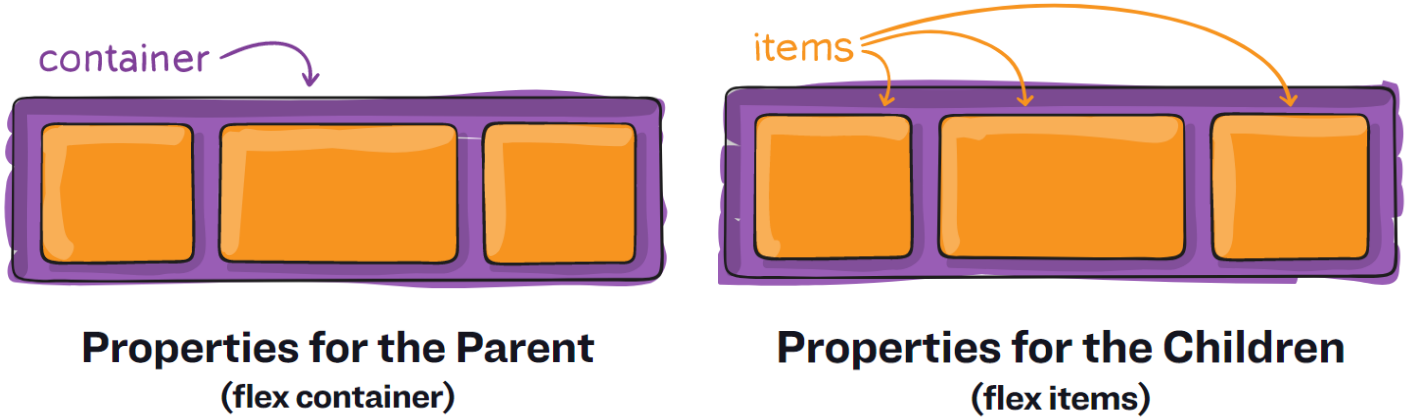




Reference : RWD Grid

Flexbox

- The basic element of flexbox system: **flex container & flex item**

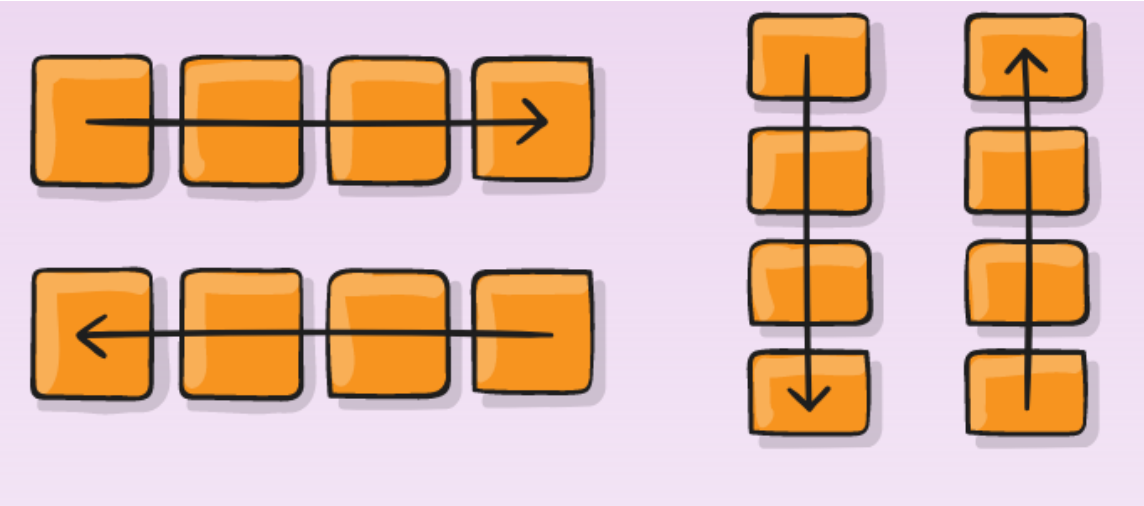



```
.container {
  display: flex; /* or inline-flex */
}
```

Reference: RWD Flexbox

Flexbox

- Define the **flex-direction** of a container.

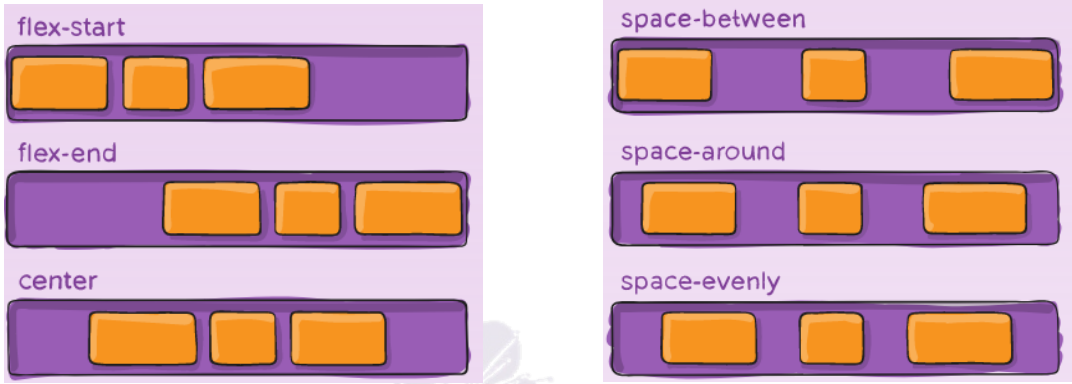


```
.container {
  flex-direction: row | row-reverse | column | column-revers
}
```

Reference: RWD Flexbox

Flexbox

- Define the way a container **justifies content** along the main axis (may be either row or column-wise).

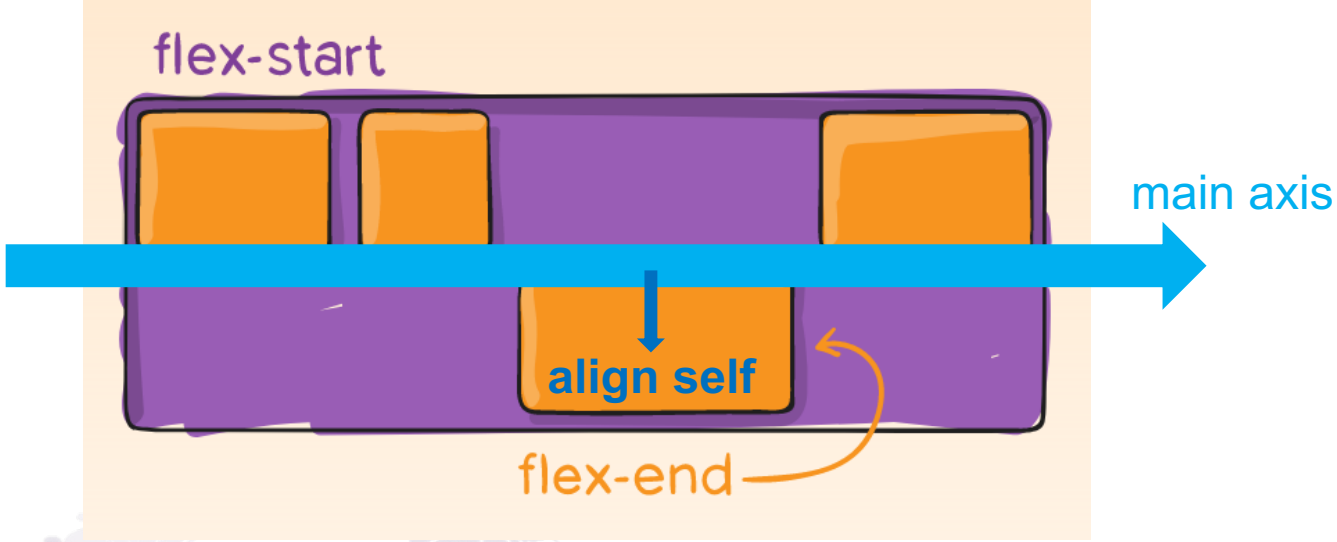



```
.container {
  justify-content: flex-start | flex-end | center | ...
}
```

Reference: RWD Flexbox

Flexbox

- This allows the default alignment to be overridden for individual flex items.




```
.item {
  align-self: auto | flex-start | flex-end | center | ...
}
```

Reference: RWD Flexbox

Reference

- CSS document
- Bootstrap document
- MDN
- RWD Grid
- RWD Flexbox


thank you!
Question?