



Research & Development Team

Introduction to ANGULAR 2

Web site : www.pnpsw.com

E-mail : sommai.k@pnpsw.com

การติดตั้ง (Installation)

NodeJs สำหรับ Windows

- <https://nodejs.org/en/download/current/>

NodeJs สำหรับ Linux / Mac

- <https://nodejs.org/en/download/package-manager/>

Visual Studio Code

- <https://code.visualstudio.com/>

การติดตั้ง (Installation) #2

Check node version

- `node -v`

Angular CLI

- `npm install -g @angular/cli`

Check cli version

- `ng help`

การติดตั้ง (Installation) #3

TypeScript

- `npm install -g typescript`

Test TypeScript

- `tsc -v`

ความรู้เบื้องต้นเกี่ยวกับ Node.js

- ประวัติ ความเป็นมา สถานการณ์ที่เหมาะสมแก่การใช้งาน

Node.js เป็นภาษาที่ทำงานอยู่ในฝั่ง Server ซึ่ง Syntax ที่ใช้ก็คือ JavaScript โดยจะออกแบบมาให้ทำงานแบบ Event-Driven ก็คือ จะทำงานเมื่อเกิดเหตุการณ์ตามที่กำหนดไว้ โดยสามารถกำหนดให้ทำงานแบบ Asynchronous (ทำงานในลำดับต่อไปโดยไม่ต้องรอให้งานก่อนหน้าเสร็จ) หรือ Synchronous (ทำงานต่อไป เมื่องานแรกเสร็จแล้ว) ก็ได้

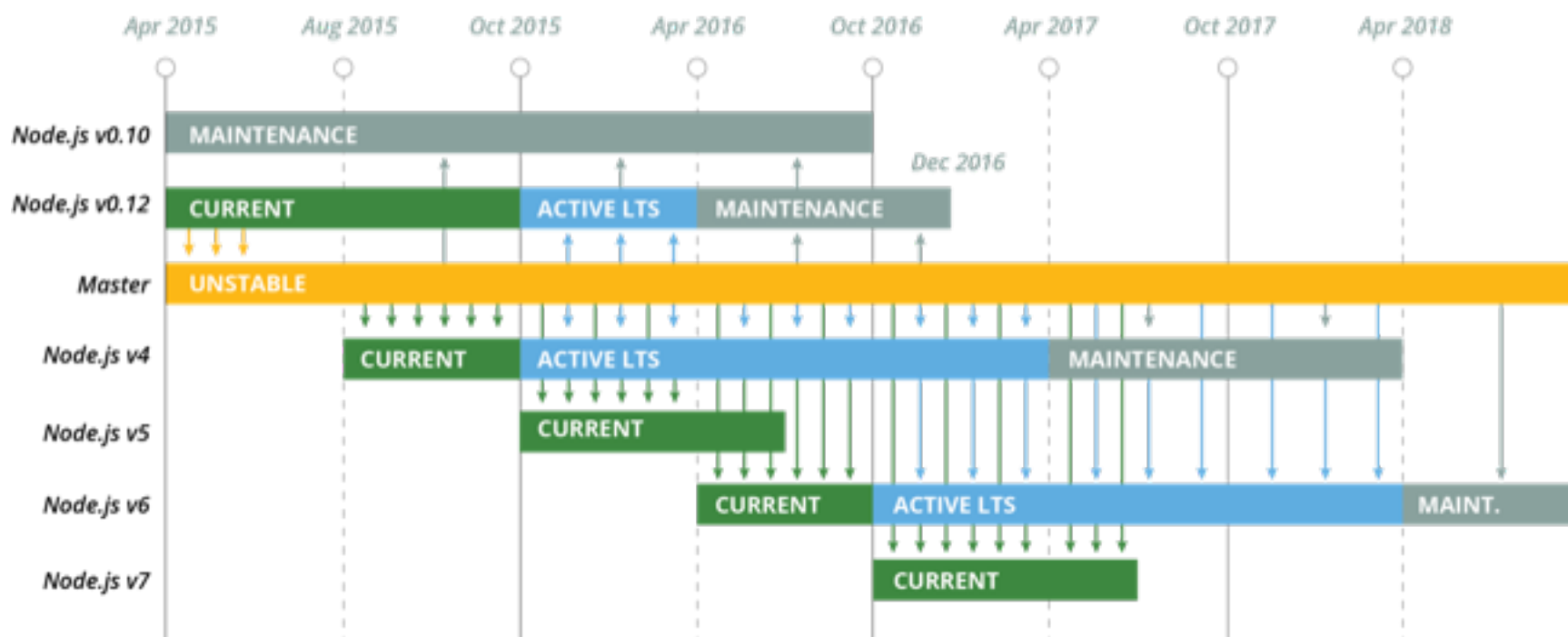
Node.js จะใช้ Compiler จาก Google JavaScript Engine V8 ซึ่งทำให้การประมวลผลรวดเร็วมาก คล้ายกับ parallel execution แต่ความจริงมันคือ single thread คือ แต่ละงานจะถูกนำมาเข้าคิวไว้ แล้วค่อยประมวลผลตามคิว

ความรู้เบื้องต้นเกี่ยวกับ Node.js

node.js เหมาะสำหรับ งานที่ประมวลผลส่ง server ซึ่งเป็นงานที่อาจจะต้อง interface หรือไม่ต้องก็ได้ เช่น การทำตัวเองเป็น http server ในการดึงหน้าเว็บมาแสดงผลให้กับ user หรือการเปิด socket เพื่อรับส่งข้อมูลกันระหว่าง server กับ user ที่อาจจะเอาไปทำเป็นห้อง chat , ทำเกม, ทำระบบที่ป้อนข้อมูลเพื่อคำนวณเอาผลลัพธ์ เป็นต้น หรือ ตัวอย่างงานที่ไม่ต้อง interface เช่นเอาไปทำ spider crawler เว็บ หรือ การรวบรวมค่าจาก streaming เหล่านี้ไม่จำเป็นต้อง interface ต่างก็ใช้ node.js ทำงานด้วยกันทั้งนั้น

เลือก version ก่อนการติดตั้ง

Node.js Long Term Support Release Schedule



COPYRIGHT © 2015 NODESOURCE, LICENSED UNDER CC-BY 4.0

JavaScript

JavaScript

ภาษาจาวาสคริปต์ คือ ภาษาโปรแกรมคล้ายภาษา C ถูกใช้ร่วมกับภาษาเอชทีเอ็มแอลในการพัฒนาเว็บเพจ ประมวลผลในเครื่องของผู้ใช้ ช่วยให้การนำเสนอเป็นแบบโต้ตอบกับผู้ใช้ได้ในระดับหนึ่ง ภาษานี้มีชื่อเดิมว่า LiveScript ถูกพัฒนาโดย Netscape Navigator เพื่อช่วยให้เว็บเพจสามารถแสดงเนื้อหาที่มีการเปลี่ยนแปลงได้ ตามเงื่อนไข หรือสภาพแวดล้อมที่แตกต่างกัน หรือโต้ตอบกับผู้ใช้ได้มากขึ้น เพราะภาษา HTML ที่เป็นภาษาพื้นฐานของเว็บเพจทำได้เพียงแสดงข้อมูลแบบคงที่ (Static Display)

การแทรก JavaScript ใน HTML

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

```
<script language="javascript">
```

```
.....  
</script>
```

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

~>

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

```
<script type="text/javascript">
```

```
.....  
</script>
```

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

แสดงข้อความ Hello World! ที่หน้าเว็บ

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

```
<script type="text/javascript">  
document.write("Hello World!");  
</script>
```

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



Hello World!

document.write เป็นคำสั่งที่ใช้เขียนผลลัพธ์บนหน้าเว็บ

การเรียกใช้ไฟล์ JavaScript ที่อยู่นอก

- บันทึกไฟล์ JavaScript ให้มีนามสกุล .js
- เขียนส่วนของ tag script ให้ src อ้างอิงไปที่ไฟล์ ที่บันทึกไว้
ดังนี้

```
<html>
<head>
<script src="xxx.js"></script>
</head>
<body>
</body>
</html>
```

JavaScript Comments

- การใส่หมายเหตุบรรทัดเดียว ให้ใส่เครื่องหมาย // ไว้หน้าบรรทัดนั้น
- การใส่หมายเหตุหลายบรรทัด เริ่มต้นด้วย /* และปิดท้ายด้วย */

```
<script type="text/javascript">
// This will write a header:
document.write("<hl>This is a header</hl>");
// This will write two paragraphs:
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

```
<script type="text/javascript">
/*
The code below will write
one header and two paragraphs
*/
document.write("<hl>This is a header</hl>");
document.write("<p>This is a paragraph</p>");
document.write("<p>This is another paragraph</p>");
</script>
```

Variable

ชื่อของตัวแปรใน JavaScript สามารถขึ้นต้นด้วยตัวอักษรใหญ่ (A-Z) ตัวอักษรเล็ก (a-z) และ เครื่องหมาย _ ตามด้วย ตัวอักษร ตัวเลข หรือ เครื่องหมาย _ ก็ได้

ชื่อตัวแปรใน JavaScript จะเข้มงวดในการใช้ตัวอักษรใหญ่เล็กด้วย เช่น Sum SUM sum จะถือว่าไม่เป็นตัวแปรเดียวกัน

Variable

ประกาศตัวแปร x และ carname

```
var x;  
var carname;
```

ประกาศตัวแปร x และ carname พร้อมกำหนดค่าเริ่มต้น

```
var x=5;  
var carname="Volvo";
```

ตัวแปรจะถูกประกาศอัตโนมัติ เมื่อมีการกำหนดค่าโดยไม่ต้องประกาศ var

```
x=5;  
carname="Volvo";
```

Data Types

- Integer
- Number
- Boolean
- String
- Array
- Object

Special Characters

Code	Outputs
\'	single quote
\"	double quote
\&	ampersand
\\	backslash
\n	new line
\r	carriage return
\t	tab
\b	backspace
\f	form feed

Array

Array ก็คือข้อมูลหลายๆตัวมาเรียงกันเป็นลำดับ เช่น

```
var employee = new Array(5)
employee[0] = "Bill"
employee[1] = "Bob"
employee[2] = "Ted"
employee[3] = "Alice"
employee[4] = "Sue"
```

ในการใช้จริงเราไม่จำเป็นต้องกำหนด length ก็ได้โดย length จะยืดหยุ่นได้ตามตัวแปรลำดับสุดท้ายT

```
var employee = new Array("Bill", "Bob", "Ted", "Alice",
    "Sue");
```

Arithmetic Operator

กำหนดให้ $y = 5$

Operator	Description	Example	Result
+	Addition	$x = y + 2$	$x = 7$
-	Subtraction	$x = y - 2$	$x = 3$
*	Multiplication	$x = y * 2$	$x = 10$
/	Division	$x = y / 2$	$x = 2.5$
%	Modulus (division remainder)	$x = y \% 2$	$x = 1$
++	Increment	$x = ++y$	$x = 6$
--	Decrement	$x = --y$	$x = 4$

Logical Operator

กำหนดให้ $x = 6$ และ $y = 3$

Operator	Description	Example
&&	and	$(x < 10 \ \&\& \ y > 1)$ is true
	or	$(x == 5 \ \ y == 5)$ is false
!	not	$!(x == y)$ is true

Comparison Operator

กำหนดให้ $x = 5$

Operator	Description	Example
==	is equal to	$x == 8$ is false
===	is exactly equal to (value and type)	$x === 5$ is true $x === "5"$ is false
!=	is not equal	$x != 8$ is true
>	is greater than	$x > 8$ is false
<	is less than	$x < 8$ is true
>=	is greater than or equal to	$x >= 8$ is false
<=	is less than or equal to	$x <= 8$ is true

Assignment Operators

กำหนดให้ **x = 10** และ **y = 5**

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
=	x=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

if Statement

```
if (condition)
{
  code to be executed if condition is true
}
```

รูปแบบ

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10

var d=new Date();
var time=d.getHours();

if (time<10)
{
  document.write("<b>Good morning</b>");
}
</script>
```

ตัวอย่าง

if...else if...else Statement

```
if (condition1)
{
code to be executed if condition1 is true
}
else if (condition2)
{
code to be executed if condition2 is true
}
else
{
code to be executed if condition1 and
condition2 are not true
}
```

รูปแบบ

switch Statement

```
switch(n)
{
case 1:
    execute code block 1
    break;
case 2:
    execute code block 2
    break;
default:
    code to be executed if n is
    different from case 1 and 2
}
```

รูปแบบ

for Loop

```
for (ประกาศตัวแปรใหม่พร้อมกำหนดค่าเริ่มต้น;เงื่อนไขการหยุด;เพิ่มค่าให้ตัวแปร) {  
คำสั่งต่างๆ ที่จะให้ทำซ้ำ  
}
```

```
<html>  
<body>  
<script type="text/javascript">  
var i=0;  
for (i=0;i<=10;i++)  
{  
document.write("The number is " + i);  
document.write("<br />");  
}  
</script>  
</body>  
</html>
```

while loop

```
while (condition) {  
    ขุดคำสั่ง  
}
```

```
<html>  
<body>  
<script type="text/javascript">  
var i=0;  
while (i<=10)  
{  
document.write("The number is " + i);  
document.write("<br />");  
i=i+1;  
}  
</script>  
</body>  
</html>
```

Do..While

Do while จะเป็นการวน loop ชนิดที่ทำจนกว่า เงื่อนไขจะเป็นจริง concept จะคล้ายๆกับ while แต่อย่าสับสนนะ while จะทำงานจนเงื่อนไขเป็นเท็จ แต่ do..while จะทำจนเงื่อนไขเป็นจริง

- `do {ชุดคำสั่ง} while (condition);`

Break Statement

จะทำหน้าที่หยุดการทำงานของ loop แบบทันทีทันใด ดังนั้นเมื่อ break ทำงาน loop จะหยุดการทำงานอย่างกะทันหัน

```
a = new Array(5,4,3,2,1)
sum = 0
for (i=0;i<a.length;i++)
{
    if (i==3) {break;}
    sum += a[i];
}
document.write(sum);
```

Continue Statement

Continue ก็ทำงานคล้ายๆกับ Break คือเข้ามาขัดขวางการทำงานแต่ก็มีข้อแตกต่างตรงที่ Break ใช้หยุด loop ทั้งหมด แต่ Continue ใช้เพื่อหยุดแค่ loop ปัจจุบันเพียง loop เดียว หรือจะเรียกการทำงานของมันเป็นว่า Skip ก็ได้

```
i = 1 ;  
sum = 0;  
while (i<10)  
{  
    i*=2;  
    if (i==4) {continue;}  
    sum += i+1;  
}
```

Function

function คือโปรแกรมย่อยที่ทำงานอย่างใดอย่างหนึ่ง ถูกสร้างขึ้นแยกออกจากโปรแกรมหลักเพื่อให้สามารถเรียกใช้ได้อย่างสะดวก
การสร้าง function ของ JavaScript มีรูปแบบดังนี้

```
function ชื่อฟังก์ชัน(พารามิเตอร์1,พารามิเตอร์2,..){  
    คำสั่งต่าง ๆ  
}
```

Return Statement

เป็น Function ที่คืนค่าได้ เรามองว่ามันเป็นตัวแปรตัวหนึ่ง ที่เก็บค่าๆหนึ่งอยู่ได้เลย

```
function ชื่อฟังก์ชัน(พารามิเตอร์1,พารามิเตอร์2,..){  
    คำสั่งต่าง ๆ  
    return ค่าที่ส่งออกไป  
}
```

Event คืออะไร

Event ก็คือ Action ต่างๆที่เกิดขึ้นกับส่วนต่างๆในเว็บเพจ เช่น เมื่อเราเอา mouse ไปทับตัว link ก็จะทำให้เกิด event onmouseover ที่ตัว link พอเอา mouse ออก ก็จะทำให้เกิด event onmouseout พอเรา click ก็จะทำให้เกิด event onclick เป็นต้น การทำงานของ event ก็จะมีอยู่ 2 ขั้นตอน คือ

1. ตรวจสอบการเกิด event ที่เรากำหนดไว้
2. เมื่อเกิด event ขึ้น ก็จะไปเรียก function หรือคำสั่งต่างๆมาทำงาน

Event

Event	ความหมาย
onAbort	เกิดเมื่อผู้ใช้ยกเลิกการ load ภาพ
onBlur	เกิดเมื่ออ็อบเจกต์นั้นถูกย้าย focus ออกไป
onChange	เกิดเมื่อผู้ใช้เปลี่ยนแปลงค่าในฟอร์มรับข้อมูล
onClick	เกิดเมื่ออ็อบเจกต์นั้นถูก click
onError	เกิดเมื่อการ load เอกสารหรือภาพเกิดข้อผิดพลาด
onFocus	เกิดเมื่ออ็อบเจกต์นั้นถูก focus
onLoad	เกิดเมื่อโหลดเอกสารเสร็จ
onMouseover	เกิดเมื่ออ็อบเจกต์นั้นถูกเลื่อน mouse pointer ไปทับ
onMouseout	เกิดเมื่ออ็อบเจกต์นั้นถูกเลื่อน mouse pointer ที่ทับอยู่ออกไป
onSelect	เกิดเมื่อผู้ใช้เลือกข้อความ(ใช้ mouse ลาก)ในช่องรับข้อความ
onSubmit	เกิดเมื่อผู้ใช้ submit แบบฟอร์ม
onUnload	เกิดเมื่อผู้ใช้ออกจากเว็บเพจ

วิธีการใช้ Event

เราจะใส่ event ลงไปใน tag ของ html เลย เช่น เวลาจะทำตัว link เราใช้ tag <A> ถ้าจะทำให้มันมีข้อความ Alert ขึ้นเวลาเอา mouse ไป over เขียนโค้ดได้ดังนี้

```
<a href=""  
onmouseover="window.alert('Onmouseover ทำงาน')">  
ทดสอบ onmouseover  
</a>
```

การใช้ JavaScript ร่วมกับ Form

การอ้างอิงข้อมูลที่อยู่บนฟอร์ม ทำได้โดยใช้ Object `document` จากนั้นตามด้วย `.` และชื่อ form `.` ชื่อของ Object ต่าง ๆ บนฟอร์ม เช่น

`document.ชื่อฟอร์ม.ชื่อTextField.value`

ถ้าต้องการเปลี่ยนข้อความที่รับมาจาก TextField ให้เป็นตัวเลข จะใช้ function `eval()` เช่น

`num = eval(document.ชื่อฟอร์ม.txtName.value)`

Try...Catch Statement

```
try
{
    //Run some code here
}
catch(err)
{
    //Handle errors here
}
```

CSS

CSS

ย่อมาจาก Cascading Style Sheets

- เป็นภาษาที่มีรูปแบบการเขียน Syntax ที่เฉพาะ และถูกกำหนดมาตรฐานโดย W3C (World Wide Web Consortium) เช่นเดียวกับ HTML และ XHTML
- ใช้สำหรับตกแต่งเอกสาร HTML/ XHTML ให้มีหน้าตา สี สัน ตัวอักษร เส้น ขอบ พื้นหลัง ระยะห่าง ฯลฯ อย่างที่เราต้องการ ด้วยการกำหนดคุณสมบัติให้กับ Element ต่างๆ ของ HTML เช่น `<body>`, `<p>`, `<h1>` เป็นต้น

โครงสร้างคำสั่ง CSS

- คำสั่งของ CSS ประกอบด้วย selector, property และ value

selector { property:value }

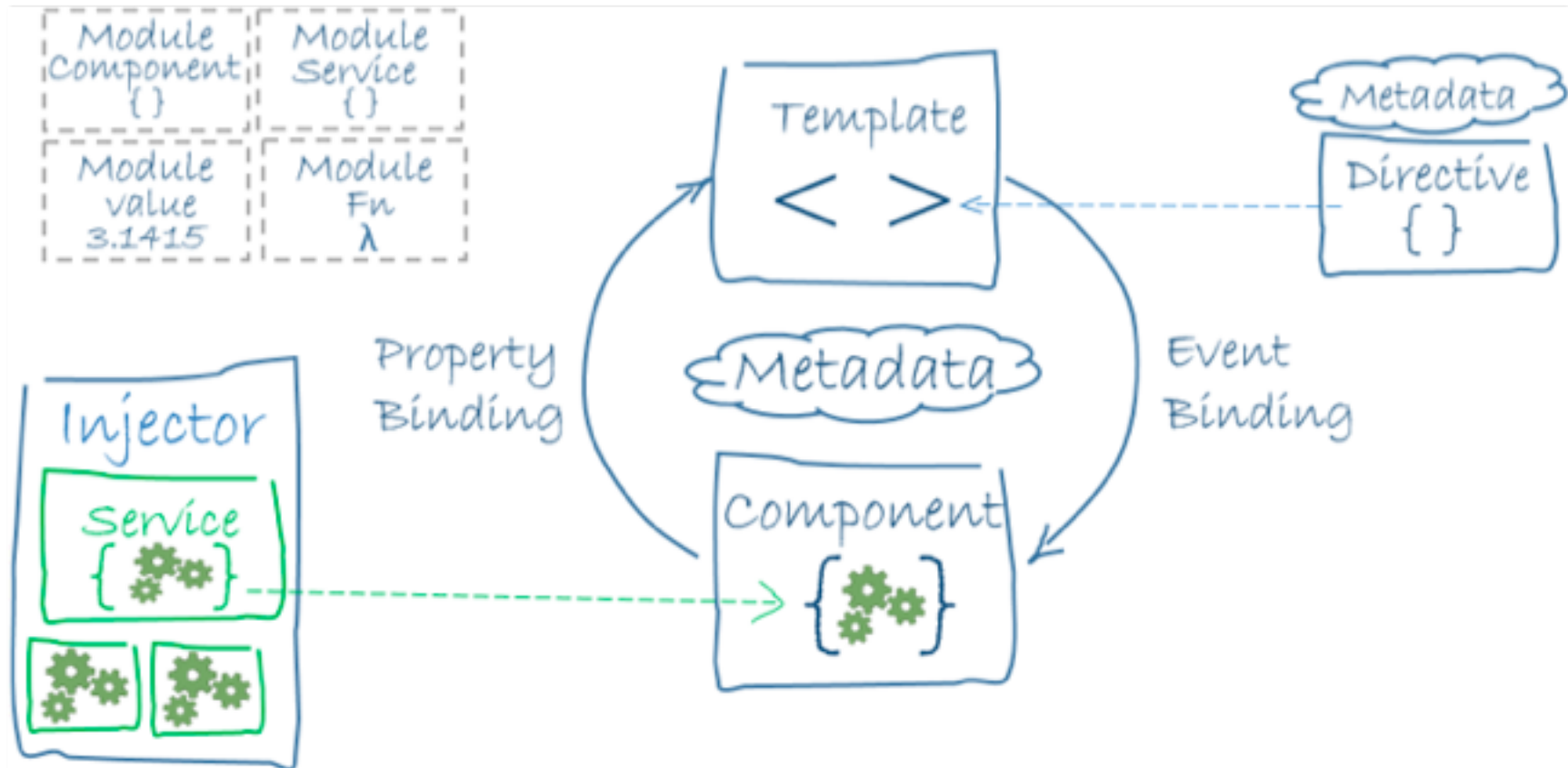
selector { property1:value1; property2:value2 }

- **selector** สามารถเป็น HTML Tag ต่างๆ เช่น <body>, <p> หรือเป็น Class name หรือ ID ที่เรากำลังตั้งชื่อ
- **property** คือ คุณสมบัติในการจัดรูปแบบการแสดงผล เช่น color สำหรับกำหนดสี, font-size สำหรับกำหนดขนาดตัวอักษร
- **value** เป็น ค่า ที่เรากำหนดให้กับ property ต่างๆ เช่น color:white, font-size:14px

วิธีใช้งาน Style Sheet

- เราสามารถใช้ CSS ได้ 3 แบบคือ
 1. Inline Styles
 2. Internal Style Sheet
 3. External Style Sheet

Angular 2 Architecture Overview



Architecture Overview #2

- Basics of Typescript
- Components, Bootstrap, and the DOM
- Directives and pipes
- Data binding
- Dependency Injection
- Services and other business logic
- Data Persistence
- Routing

Basics of Typescript

- JavaScript that scale
- Starts and ends with JavaScript
- Strong tools for large apps
- State of the art JavaScript

Basics of Typescript #2

Create file app.ts

```
var message:string = "Hello World"  
console.log(message)
```

Compile

```
tsc app.ts
```

Run

```
node app.js
```

Output

```
Hello World
```

TypeScript – Keywords

break	as	any	switch
case	if	throw	else
var	number	string	get
module	type	instanceof	typeof
public	private	enum	export
finally	for	while	void
null	super	this	new
in	return	true	false
any	extends	static	let
package	implements	interface	function
new	try	yield	const
continue	do	catch	

TypeScript and OOP

```
class Greeting {  
    greet():void {  
        console.log("Hello World!!!")  
    }  
}  
  
var obj = new Greeting();  
obj.greet();
```

Built-in types

- number
- string
- Boolean
- void
- null
- undefined
- any

Variable Declaration

```
var [identifier] : [type] = value ;
```

Ex: var name:string = 'my name is angular.io';

```
var [identifier] : [type];
```

Ex: var name:string;

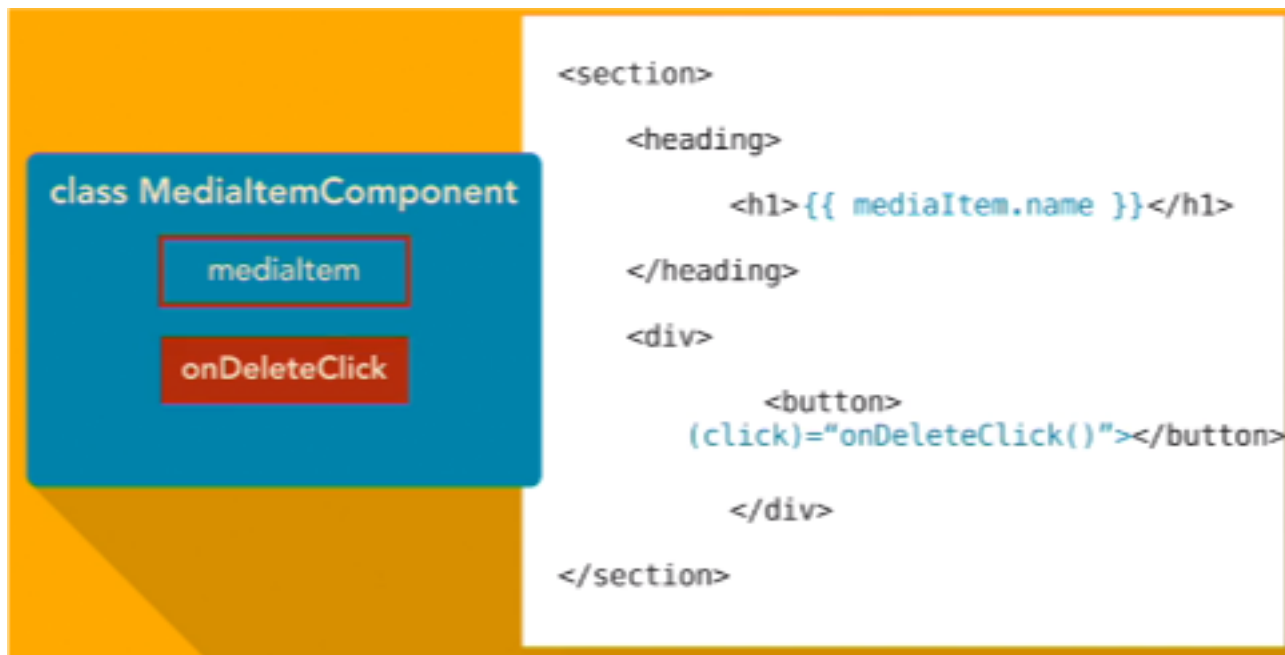
```
var [identifier] = value ;
```

Ex: var name = 'my name is angular.io';

```
var [identifier] ;
```

Ex: var name;

Components, Bootstrap, and the DOM



The diagram illustrates the structure of a `MediaItemComponent` class and its corresponding HTML template. On the left, a blue box represents the class, containing two properties: `mediaItem` (in a light blue box) and `onDeleteClick` (in a red box). On the right, a white box with an orange border shows the HTML template for the component, which uses the `mediaItem` and `onDeleteClick` props to render a heading and a button.

```
<section>
  <heading>
    <h1>{{ mediaItem.name }}</h1>
  </heading>
  <div>
    <button
      (click)="onDeleteClick()"></button>
  </div>
</section>
```

Components, Bootstrap, and the DOM #2

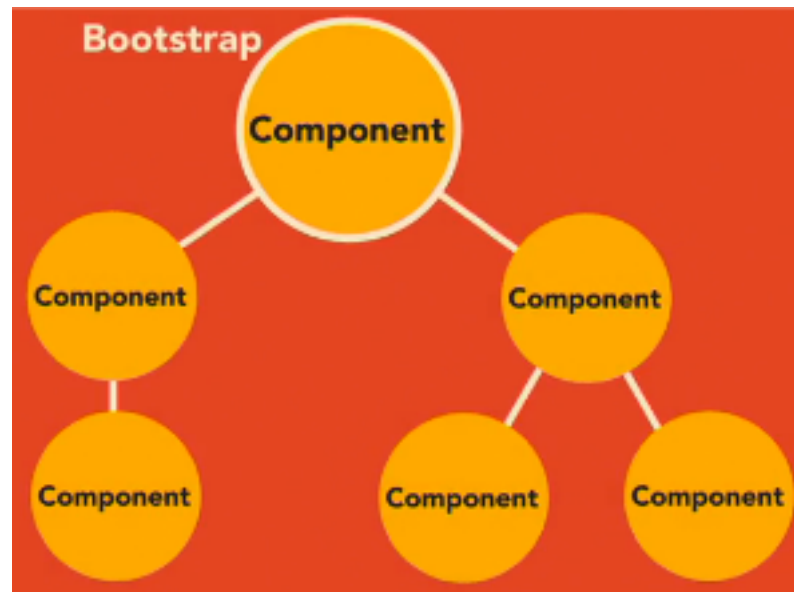
```
class MediaItemComponent
```

```
class ViewRatingComponent
```

```
selector: 'view-rating'
```

```
<section>
  <heading>
    <h1>{{ mediaItem.name }}</h1>
  </heading>
  <div>
    <button>
      (click)="onDeleteClick()"</button>
    </div>
    <view-rating></view-rating>
  </section>
```

Components, Bootstrap, and the DOM #3



New Project

- `ng new projectname --skip-git --routing`

Test

- `cd projectname`
- `ng serve`
- Open web browser (Chrome)
- Goto url <http://localhost:4200>



app works!

CSS Framework

- Materialize CSS (<http://materializecss.com/>)

Installation

- Use command line
- Goto projectname folder
- Type : `npm install materialize-css`
- Config css, js (`angular-cli.json`)

```
"styles": [  
  "../node_modules/materialize-css/dist/css/materialize.min.css",  
  "styles.css"  
],  
"scripts": [  
  "../node_modules/jquery/dist/jquery.min.js",  
  "../node_modules/materialize-css/dist/js/materialize.min.js"  
],
```

Icon Fonts

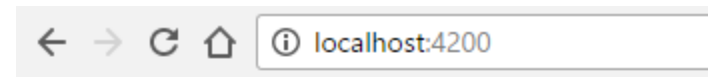
1. Goto url (
<https://fonts.googleapis.com/icon?family=Material+Icons>)
2. คัดลอก source มาใส่ใน file style.css

```
@font-face {
  font-family: 'Material Icons';
  font-style: normal;
  font-weight: 400;
  src: local('Material Icons'), local('MaterialIcons-Regular'),
    url(https://fonts.gstatic.com/s/materialicons/v21/2fcrYFNaTjcS6g4U3t-Y5ZjZjT5FdEJ140U2DjYJC3mY.woff2) format('woff2');
}

.material-icons {
  font-family: 'Material Icons';
  font-weight: normal;
  font-style: normal;
  font-size: 24px;
  line-height: 1;
  letter-spacing: normal;
  text-transform: none;
  display: inline-block;
  white-space: nowrap;
  word-wrap: normal;
  direction: ltr;
  -webkit-font-feature-settings: 'liga';
  -webkit-font-smoothing: antialiased;
}
```

app.component.html

```
<h1>
  {{title}}
</h1>
<a class="waves-effect waves-light btn">
  <i class="material-icons left">cloud</i>button
</a>
```



app works!



New component

Run command

- ng g component home

```
installing component
create src\app\home\home.component.css
create src\app\home\home.component.html
create src\app\home\home.component.spec.ts
create src\app\home\home.component.ts
```

```
└─ src
  └─ app
    └─ home
      ├── home.component.css
      ├── home.component.html
      ├── home.component.spec.ts
      └── home.component.ts
    ├── app.browser.module.ts
    ├── app.component.css
    ├── app.component.html
    ├── app.component.spec.ts
    ├── app.component.ts
    ├── app.node.module.ts
    └── index.ts
```


Component

- HTML
- CSS
- Javascript / TypeScript (Script)

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {
  constructor() { }
  ngOnInit() {
  }
}
```

Directives

- `*ngIf`
- `*ngFor`
- `[ngSwitch]`, `[ngSwitchCase]`, `[ngSwitchDefault]`
- `[ngClass]`

Built-in directives

ngIf

```
<section *ngIf="showSection">
```

ngFor

```
<li *ngFor="let item of list">
```

ngClass

```
<div [ngClass]="{'active': isActive,  
'disabled': isDisabled}">
```

Built-in directives #2

```
<div [ngSwitch]="conditionExpression">  
  <ng-template [ngSwitchCase]="case1Exp">One</ng-template>  
  <ng-template ngSwitchCase="case2LiteralString">Two</ng-  
  template>  
  <ng-template ngSwitchDefault>Three</ng-template>  
</div>
```

```
//  
export class AppComponent {  
  title = 'app';  
  conditionExpression = "A";  
  case1Exp = "A";  
}
```

PIPES

- Pipes transform displayed values within a template.

Built-In

```
DecimalPipe = number[:format]
```

Script

```
price = 12300.5;
```

Html

```
{{price | number:'1.2-3'}}
```

PIPES

Built-In

```
LowerCasePipe = lowercase
```

Html

```
{{item | lowercase}}
```

Built-In

```
UpperCasePipe = uppercase
```

Html

```
{{item | uppercase}}
```

PIPES

Built-In

```
DatePipe = date[:format]
```

Script

```
currentDate = new Date();
```

Html

```
{{currentDate | date:'dd/MM/y H:mm:ss'}}
```

Routing

create file app-routing.module.ts

```
import { HomeComponent } from './home/home.component';  
export const appRoutes = [  
  { path: '', component: HomeComponent }  
];
```

app.component.html

```
<h1>  
  {{title}}  
</h1>  
<router-outlet></router-outlet>
```


Workshop #1

- ng g component product
- app-routing.module.ts

```
{ path: 'product', component: ProductComponent }
```
- app.component.html

```
<nav>
  <div class="nav-wrapper">
    <a routerLink="" class="brand-logo">{{title}}</a>
    <ul id="nav-mobile" class="right hide-on-med-and-down">
      <li><a routerLink="/product">Product</a></li>
    </ul>
  </div>
</nav>
<router-outlet></router-outlet>
```

- ใส่ค่า 'product' ที่ server.routes.ts

Grid System

- มี 12 Column
- มี 3 ขนาด คือ S, M, L
- S = ขนาดหน้าจอมือถือ ($\leq 600\text{px}$)
- M = ขนาดหน้าจอ Tablet ($\leq 992\text{px}$)
- L = ขนาดหน้าจอ Computer ($> 992\text{px}$)

Example

```
<div class="row">
```

```
  <div class="col s12"> <div class="card-panel">I am always full-width (col s12)</div> </div>
```

```
  <div class="col s12 m6"><div class="card-panel">I am full-width on mobile half on tablet desktop (col s12 m6)</div>
```

```
</div>
```

```
</div>
```

Navigate to other page

Import

```
import { Router } from '@angular/router';
```

Inject

```
constructor(private router : Router) { }
```

Navigate

```
this.router.navigate(['']);
```

@Input

- ใช้สำหรับเชื่อมโยงค่าตัวแปรมาจากอีก component

Script #ต้นทาง

```
@Input() name;
```

Html#ต้นทาง

```
<p>  
  user works! {{name}}  
</p>
```

Script#ปลายทาง

```
appName = "Home App";
```

Html#ปลายทาง

```
<app-user [name]="appName"></app-  
user>
```

TEMPLATE

- `{{ }}` : RENDERING
- `[]` : BINDING PROPERTIES
- `()` : HANDLING EVENTS
- `[()]` : TWO-WAY DATA BINDING
- `*` : THE ASTERISK

{{ }} : RENDERING

- ใช้สำหรับนำค่าจาก script มาแสดงผลใน html

Script

```
title = 'app works!';
```

Html

```
<a routerLink="" class="brand-logo">{{title}}</a>
```

[] : BINDING PROPERTIES

- ใช้สำหรับเชื่อมโยงค่าตัวแปรมาจากอีก component
- ต้องใช้ร่วมกับ @Input

Script

```
appName = "Home App";
```

Html

```
<app-user [name]="appName"></app-user>
```

() : HANDLING EVENTS

- ใช้สำหรับจัดการกับ Event ของ user

Script

```
onBtnClick(){  
    console.log("Hello World");  
}
```

Html

```
<button class="btn" (click)="onBtnClick();">Click</button>
```


[()] : TWO-WAY DATA BINDING

- ใช้สำหรับเชื่อมโยงค่าตัวแปรกับ user input

Script

```
appName = "Home App";
```

Html

```
<input type="text" [(ngModel)]="appName"/>
```

* : THE ASTERISK

- ใช้สำหรับเรียกใช้ directive

Script

```
items = ["One", "Two", "Three"];
```

Html

```
<ul class="collection">
```

```
  <li class="collection-item" *ngFor="let item of items">{{item}}</li>
```

```
</ul>
```

SERVICE

- `ng g service servicename`

```
installing service
create src\app\home.service.spec.ts
create src\app\home.service.ts
WARNING Service is generated but not provided, it must be provided to be used
```

import library

```
import { Http, Response, RequestOptions, Headers } from '@angular/http';
import { Observable } from 'rxjs/Rx';
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/catch';
```

Inject http object

```
constructor(private http : Http) { }
```

GET data from server

Service Script

```
loadItem(): Observable<any[]> {  
    return this.http.get('change to your url')  
        .map((res: Response) => {  
            res.json()  
        })  
        .catch((error: any) => Observable.throw(error.json().error || 'Server  
error'));  
}
```

GET data from server

Component Script

import

```
import { HomeService } from '../home.service'
```

set providers

```
@Component({
```

```
...
```

```
  providers: [HomeService]
```

```
...
```

```
})
```

inject

```
constructor(private homeService : HomeService) { }
```

GET data from server

```
onBtnClick() {  
    this.homeService.loadItem()  
        .subscribe(  
            datas => {  
                this.items = datas;  
            },  
            err => {  
                console.log(err);  
            })  
        );  
}
```

POST

```
addData(body: Home): Observable<HomeResp> {  
    let bodyString = JSON.stringify(body); // Stringify payload  
    let headers = new Headers({ 'Content-Type':  
'application/json' }); // ... Set content type to JSON  
    let options = new RequestOptions({ headers: headers }); //  
Create a request option  
    return this.http.post(this.dataUrl, body, options) //  
...using post request  
    .map((res: Response) => res.json()) // ...and calling  
.json() on the response to return data  
    .catch((error: any) => Observable.throw(error.json().error  
|| 'Server error')); //...errors if any  
}
```

Any questions?

