

# **Web Technologies**

## **Final**

**Edited 2  
11/05/2018**

## Content

### Word of Web technology

- Web Application Architecture
- Content Management System
- Everything as a Service
- REST
- CSRF

### PHP

- Basic Syntax
- PDO
- OOP
- Composer
  - Dependencies
  - Namespace

### HTML

### CSS

- Intro to CSS
- CSS Specificity
- CSS3 Media Queries
- CSS Flexbox
- CSS Font-face and Icon-font
- CSS3 Transformation, Transition and Animation

### Laravel

- Installation
- Project Structure
- Artisan Command
- Route
- Blade Template
- Migration
- Factory
- Seeder
- Model
- Controller
- Authentication
- Authorization

### Javascript

### JSON

### jQuery

<b>Content</b>	<b>2</b>
<b>Word of Web technology</b>	<b>4</b>
<b>HTML</b>	<b>5</b>
โครงสร้างของหน้า	5
โครงสร้างของคำสั่ง	5
<b>CSS</b>	<b>6</b>
Intro to CSS	6
CSS Specificity	7
CSS3 Media Queries	9
CSS Flexbox	10
CSS Font-face and Icon-font	11
CSS3 Transformation, Transition and Animation	14
<b>JavaScript</b>	<b>15</b>
JavaScript คืออะไร	15
วิธีการใช้งาน	15
ลำดับการทำงาน	16
การประมวลผลแบบ	16
Function	16
Operators	17
Conditions	17
Loops	18
<b>JSON</b>	<b>19</b>
<b>JQuery</b>	<b>20</b>
<b>PHP</b>	<b>21</b>
Basic Syntax	21
PDO	21
OOP	21
Composer	21
Dependencies	21
Namespace	21
<b>Laravel</b>	<b>22</b>
Laravel - Installation	22
Laravel - Project Structure	23
Laravel - Artisan Command	24
Laravel - Route	25
Laravel - Blade Template	27



Web Application Architecture

Content Management System

Everything as a Service

REST

CSRF ( Cross-site Request Forgery )

- การโจมตีแบบ CSRF จะใช้ “ตัวตน (Identity)” และ “สิทธิ์ (Privilege)” ของเหยื่อที่มีบนเว็บไซต์ ในการปลอมตัวเป็นเหยื่อและกระทำการหรือธุกรรมไม่พึงประสงค์ แฮกเกอร์จะพยายามใช้ประโยชน์จากเหยื่อที่มี Login Cookies เก็บไว้ในเบราว์เซอร์ ส่งผลให้เว็บไซต์ E-commerce ที่ส่ง Cookie ไปเก็บข้อมูลการพิสูจน์ตัวตนของผู้ใช้มักตกเป็นเป้าหมายของการโจมตีนี้

[>>อ่านเพิ่มเติม bit.ly/Web-CSRF<<](http://bit.ly/Web-CSRF)



## HTML

>>อ่านเพิ่มเติม [<<](http://bit.ly/Web-HTML)

คือ ภาษาหลักที่ใช้ในการเขียนเว็บเพจ โดยใช้ Tag ในการกำหนดการแสดงผล HTML

โครงสร้างของหน้า

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>

</body>
</html>
```

### ★ <!DOCTYPE html>

- บอกให้ web browser ทราบว่า เราใช้คำสั่ง HTML รุ่นใด และบอกชนิดของเอกสาร (Document Type Definition : DTD) ที่ใช้ ซึ่งจะช่วยให้ web browser แปลเอกสารได้อย่างถูกต้อง

### ★ <html></html>

- ในการใช้งาน HTML เราจะต้องเริ่มด้วย <html> และปิดด้วย </html> เสมอ

### ★ <head></head>

- คำสั่งที่อยู่ในนี้จะแสดงไม่ผลบน web browser

### ★ <title></title>

- ในส่วนตัวอักษรที่อยู่ในคำสั่งนี้จะอยู่ใน title bar ของ web page

### ★ <body></body>

- คำสั่งที่อยู่ในนี้จะแสดงผลบน web browser

โครงสร้างของคำสั่ง

## 1. Tag

คำสั่งที่ใช้ในภาษา HTML อยู่ในเครื่องหมาย < และ > ใช้สำหรับจัดรูปแบบข้อความ ภาพหรือ วัตถุอื่นๆ

### ★ Tag เปิด และ tag ปิด

- <h1></h1> ใช้เน้นหัวข้อเรื่อง
- <p></p> ใช้จัดพารากรاف
- <b></b> ใช้กำหนดให้ตัวอักษรเป็นตัวหนา

### ★ Tag เปิดไม่มี tag ปิด

- <hr> ใช้สร้างเส้นคั่น
- <br> ใช้สำหรับการขึ้นบรรทัดใหม่

## 2. Attribute

เป็นส่วนขยายใน tag ใช้สำหรับจัดรูปแบบเพิ่มเติมหรืออธิบายของ tag เช่น ขนาด สี ระยะห่าง เป็นต้น เช่น

```
<p align="center"></p>

```

## CSS

### 1. Intro to CSS

Cascading Style Sheet หรือ Style sheet ที่ใช้กำหนดรูปแบบหน้าตาของไฟล์ HTML นั้นเอง โดยสมบัติของ CSS จะมีสมบัติ Cascading คือ คำสั่งที่อยู่บนสุดจะมีลำดับสำคัญสูงกว่าคำสั่งด้านล่างเสมอ CSS สามารถใช้กำหนดรูปแบบ Font สี จากหลังและอื่นๆที่แสดงบนหน้าเว็บไซต์ทั้งหมด การใช้ CSS มีทั้งแบบภายใน และภายนอก กล่าวคือสามารถเขียน CSS ไว้ในไฟล์ HTML เลยหรือแยกเป็นไฟล์ Style Sheet ต่างหากแล้วเรียกใช้ภายหลังก็ได้

EX.

แบบเขียนไว้ในไฟล์ HTML

```
@extends('layouts.app')

@section('content')
<style type="text/css">

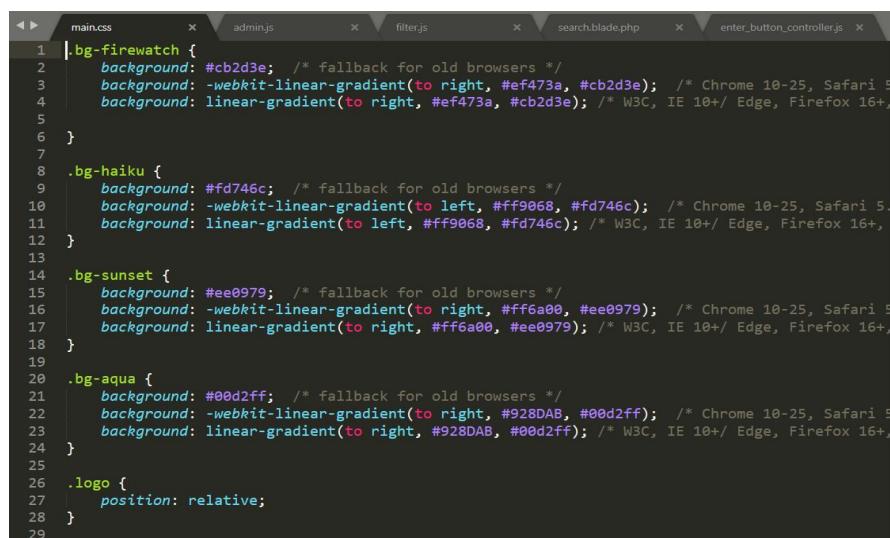
    /* Style the tab content */
    .table {
        padding: 0px 12px;
        border: 1px solid #ccc;
        width: 80%;
    }

    #map {
        height: 300px;
        width: 500px;
        margin: auto;
        clear: both;
    }

    #summit {
        width: 30%;
        height: 20%;
    }

</style>
```

แบบแยกเป็นไฟล์ .css



```
.bg-firewatch {
    background: #cb2d3e; /* fallback for old browsers */
    background: -webkit-linear-gradient(to right, #ef473a, #cb2d3e); /* Chrome 10-25, Safari 5 */
    background: linear-gradient(to right, #ef473a, #cb2d3e); /* W3C, IE 10+/ Edge, Firefox 16+, Opera */
}

.bg-haiku {
    background: #fd746c; /* fallback for old browsers */
    background: -webkit-linear-gradient(to left, #ff9068, #fd746c); /* Chrome 10-25, Safari 5 */
    background: linear-gradient(to left, #ff9068, #fd746c); /* W3C, IE 10+/ Edge, Firefox 16+, Opera */
}

.bg-sunset {
    background: #ee0979; /* fallback for old browsers */
    background: -webkit-linear-gradient(to right, #ff6a00, #ee0979); /* Chrome 10-25, Safari 5 */
    background: linear-gradient(to right, #ff6a00, #ee0979); /* W3C, IE 10+/ Edge, Firefox 16+, Opera */
}

.bg-aqua {
    background: #00d2ff; /* fallback for old browsers */
    background: -webkit-linear-gradient(to right, #928DAB, #00d2ff); /* Chrome 10-25, Safari 5 */
    background: linear-gradient(to right, #928DAB, #00d2ff); /* W3C, IE 10+/ Edge, Firefox 16+, Opera */
}

.logo {
    position: relative;
```

การเรียกใช้

```
<link rel="stylesheet" href="css/main.css">
```

## 2. CSS Specificity

>>อ่านเพิ่มเติม [<<](http://bit.ly/CSS-Specificity)

CSS Specificity คือ ค่าบางอย่างที่จะบอก Browser ว่าให้เลือกใช้ Style ตัวไหนมาแสดงผลบนหน้าเว็บ และจะต้องมีการคำนวนค่าออกมาเพื่อให้ Browser รู้

### การคำนวนค่า CSS Specificity

- การสร้าง Style ให้กับ Element สามารถทำได้หลายแบบ ทั้ง **การเขียนแบบ Inline** หรือ **การเขียนแบบใช้ Selector** ซึ่งเป็นที่นิยมใช้กันมากโดยการเขียนแบบ Selector นี้ก็มีหลายรูปแบบ เช่น

○ Type Selectors คือการใช้ Element เป็น selector เช่น h1, input, image

○ Class Selectors เช่น .container, .wrapper (**จะขึ้นต้นด้วยจุด**)

○ ID Selectors ใช้ id เป็น Selector เช่น #name, #profile (**จะขึ้นต้นด้วย#**)

○ Attributes Selectors

- element[attribute] ใช้กับ element ที่มี attribute นี้
- element[attribute=value] ใช้กับ element ที่ attribute=value เป็นๆ
- element[attribute~=value] ใช้กับ element ที่ attribute มี value นี้อยู่ด้วย
- element[attribute|=value] ใช้กับ element ที่ attribute มี value- นี้ขึ้นต้นของทั้งหมด
- element[attribute^=value] ใช้กับ element ที่ attribute มี value นี้ขึ้นต้นของทั้งหมด
- element[attribute\$=value] ใช้กับ element ที่ attribute มี value นี้ลงท้ายของทั้งหมด
- element[attribute\*=value] ใช้กับ element ที่ attribute มี value เป็นsubstring

หมายเหตุ กรณีไม่มี element ให้ถือว่ามีค่าเป็น \* หรือ all หรือ ทุก element เลย

และการใช้ ^ กับ \$ จะคล้ายกับของ UNIX ที่ ^ขึ้นต้นประโยค, \$ลงท้ายประโยค

○ Pseudo-Classes เช่น :hover, :active

○ และอื่นๆ เช่น

- Universal Selector ( \*) ใช้กับทุก element
- Combinators ( +, >, ~, '' )
  - element1+element2 ใช้กับ element2 อันเดียว ที่อยู่หลัง(ถัดจาก) element1
  - element1>element2 ใช้กับ element2 ทุกอัน ที่อยู่หลัง(ถัดจาก) element1
  - element1~element2 ใช้กับ element2 ทุกอัน ที่มี element1 เป็น parent
  - element element ใช้กับ element2 ทุกอัน ที่อยู่ใน element1
- Negation Pseudo-Class ( :not())
- Pseudo-Elements ( ::before )

ซึ่งแต่ละแบบก็จะมีค่า Specificity ที่ต่างกันออกไป และสามารถจัดกลุ่มการเขียน Style แต่ละแบบที่มีค่า Specificity เท่ากันได้ดังนี้

- (1) Inline Style
- (2) ID Selectors
- (3) Class Selectors, Attributes Selectors, Pseudo-Classes
- (4) Type Selectors, Pseudo-Elements
- (5) Universal Selector, Combinators, Negation Pseudo-Class

โดยจะแบ่ง Selector 4 ประเภทนี้ออกเป็น 4 หลัก ซึ่งแต่ละหลักก็จะแยกกันโดยสิ้นเชิงโดยหลักช้ายสุดจะมีค่า Specificity สูงสุดแล้วค่อยๆ เรียงมาทางขวาเป็นน้อยสุด 1, 2, 3, 4

หรือมองง่ายๆ ให้เป็นเลขหลักพัน ร้อย สิบ หน่วยแบบ

แบบ (1) มีค่า specificity = 1000

แบบ (2) มีค่า specificity = 100

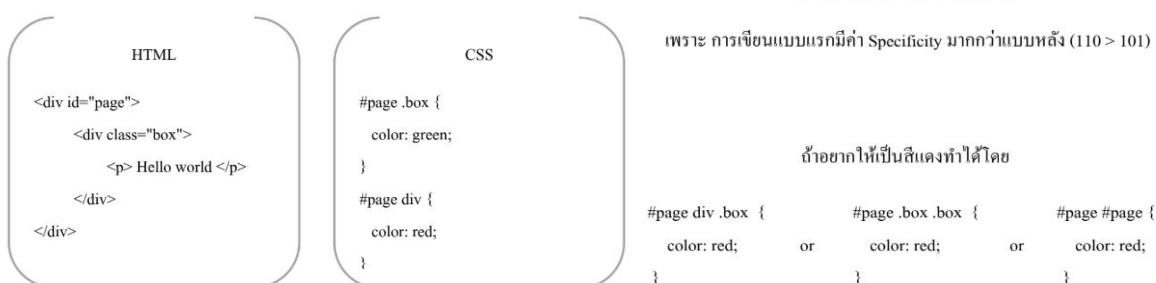
แบบ (3) มีค่า specificity = 10

แบบ (4) มีค่า specificity = 1

แบบ (5) มีค่า specificity = 0 (ไม่มีค่า)

EX.	div a {...}	= 1+1	= 2
	.box a {...}	= 10+1	= 11
	div h1 a apan {...}	= 1+1+1+1	= 4
	.page h1 #title {...}	= 10+1+100	= 111
	#nav .selected > a:hover	= 100+10+1+10	= 121
	li:first-child h2 .title	= 1+10+1+10	= 22
	สมมติโค้ดขึ้นมา		ผลลัพธ์

## Hello world



### 3. CSS3 Media Queries

>>อ่านเพิ่มเติม [<<](http://bit.ly/CSS3MediaQueries)

Media queries เป็นองค์ประกอบของการออกแบบที่มีการเปลี่ยนแปลงตามอุปกรณ์ที่ใช้หรือขนาดมุมมองของเบราว์เซอร์โดยใช้ได้กับหลาย media type เช่น print, screen, speech, TV เป็นต้น  
หลักการใช้งาน

CSS ที่ได้กำหนดไว้ จะทำงานก็ต่อเมื่อคุณสมบัติของอุปกรณ์ตรงกับเงื่อนไขของ Media Queries

#### วิธีใช้งาน

รูปแบบการเขียน @media (\_conditions\_) { ... }

#### ★ ขั้นต้นด้วย @media ตามด้วยชื่ออุปกรณ์

@media print { ... } - คำสั่งนี้จะทำงานเมื่อเป็นการพิมพ์

และเราสามารถกำหนดได้หลายอุปกรณ์ ดังนี้

@media screen, print { ... } - คำสั่งนี้จะทำงานเมื่อเป็นหน้าจอเบราว์เซอร์ หรือ การพิมพ์ต

#### ★ การกำหนดเงื่อนไข Expression เพิ่มเติม

@media media-types and media-feature { ... }

#### Media Features ที่ใช้บ่อย

width ความกว้างของ viewport

height ความสูงของ viewport

device-width ความกว้างของสวนแสดงผลของอุปกรณ์

device-height ความสูงของสวนแสดงผลของอุปกรณ์

orientation แนวการแสดงผลของอุปกรณ์ โดยมี portrait เป็นแนวตั้ง, landscape เป็นแนวนอน

#### ตัวอย่าง

สามารถเพิ่ม prefix “min-” เพื่อแสดงเงื่อนไขขั้นต่ำและ “max-” เพื่อแสดงเงื่อนไขสูงสุดได้

@media (max-width: 700px) { ... }

- คำสั่งนี้จะทำงานเมื่อความกว้างของ viewport มีค่าเท่ากับ หรือน้อยกว่า 700px

@media (min-width: 600px) { ... }

- คำสั่งนี้จะทำงานเมื่อความกว้างของ viewport มีค่าเท่ากับ หรือมากกว่า 600px

@media screen and (min-width: 320px) { ... }

- คำสั่งนี้จะทำงานเมื่อหน้าจอของเบราว์เซอร์มีความกว้างของ viewport เท่ากับหรือมากกว่า 320px

```
@media screen and (max-width: 500px) {  
    body {  
        background-color: blue;  
    }  
}
```

- คำสั่งนี้ทำให้พื้นหลังของ body เป็นสีน้ำเงิน เมื่อหน้าจอของเบราว์เซอร์มีความกว้างของ viewport เท่ากับ หรือน้อยกว่า 500px

## 4. CSS Flexbox

[>>อ่านเพิ่มเติม bit.ly/CSS-Flexbox<<](http://bit.ly/CSS-Flexbox)

### วิธีใช้ CSS Flexbox

อย่างที่กล่าวมา CSS Flexbox จะเป็นการจัดการ block ที่เก็บ box มี 2 มิติ ตัวอย่าง

```
<div>  
  <div>a</div>  
  <div>b</div>  
</div>
```

### Style ที่ใช้ใน block ที่เก็บ box กำหนดโดย CSS

1. display คือการระบุประเภทของ box ที่เราจะใช้ โดยในที่นี้คือ flex กำหนดโดย

```
.flexbox-container {  
  display : flex;  
}
```

2. flex-direction คือการระบุว่าจะให้ box จัดที่แบบไหน กำหนดโดย

```
.flexbox-container {  
  display : flex;  
  flex-direction : column;  
}
```

### Style ที่ใช้ในตัว box ที่อยู่ใน block กำหนดโดย HTML

1. order คือการกำหนดลำดับของ box กำหนดโดย

```
<div class="flexbox-container">  
  <div style="order : 2;">a</div>  
  <div style="order : 1;">b</div>  
</div>
```

value ที่ใช้ได้ของ order คือ integer (default = 0)

2. flex-grow คือการกำหนดขนาดของ box เทียบกับพื้นที่ว่างที่เหลืออยู่ กำหนดโดย

```
<div class="flexbox-container">  
  <div style="flex-grow : 3;">a</div>  
  <div style="flex-grow : 2;">b</div>  
</div>
```

## 5. CSS Font-face and Icon-font

>>อ่านเพิ่มเติม [<<](http://bit.ly/FontFace-IconFont)

Font face: เป็นกลไกที่ถูกพัฒนาขึ้นมาเพื่อช่วยให้เราสามารถเรียกใช้ font ที่ไม่ได้มีอยู่หรือถูกติดตั้งไว้ที่เครื่องของ users ได้โดยมีหลักการง่ายๆ คือ ทำหน้าที่เป็น ตัวระบุที่อยู่ของ font file ที่เราต้องการจะใช้

Format: รูปแบบการเขียน @font-face rule

1. Font อยู่บน web server ( ซึ่งไม่ได้อยู่ที่เดียวกับ file html ของเรา, ต้องการใช้บน codepen.io )

หลักการทำงาน: ให้เครื่องของ users ไปอ่าน font file ที่

<https://www.yourdomain.com/fonts/yourFont.ttf> แทนการอ่าน font file ในเครื่องของ user

```
@font-face {  
    font-family: 'yourFont';  
    src: url('https://www.yourdomain.com/fonts/yourFont.ttf');  
}
```

2. Font อยู่บน web server ( ซึ่งอยู่ที่เดียวกับหรือใกล้เคียงกับ file html ของเรา )

หลักการทำงาน: ให้เครื่องของ users ไปอ่าน font file ที่ไฟล์ yourFont.ttf แทนการอ่าน font file ในเครื่องของ user

```
@font-face {  
    font-family: 'yourFont';  
    src: url('yourFont.ttf');  
}
```

3. ดู font ที่อยู่บนเครื่องเรา ก่อน ถ้าไม่มีให้ไปอ่าน จาก url

หลักการทำงาน: อ่าน font file ที่อยู่บนเครื่องของ users ก่อน ถ้าไม่มีค่อยไปอ่าน font file ที่

<https://www.yourdomain.com/fonts/yourFont.ttf>

```
@font-face {  
    font-family: 'yourFont';  
    src: local('yourFont.ttf'),  
        url('https://www.yourdomain.com/fonts/yourFont.ttf');  
}
```

สามารถใส่ font-weight, font-style ลงไปเพิ่มได้การนำไปใช้: รูปแบบการนำ @font-face ไปใช้

```
p {  
    font-family: 'yourFont';  
}
```

Icon-font: นอกจากจะสามารถย่อขยาย ได้โดยที่ไม่เกิดปัญหาแล้ว ยังแก้ปัญหา 1 icon / 1png ได้ด้วยโดย icon font จะทำให้เรารวมทุก icon ไว้ในไฟล์ font ไฟล์เดียวได้ ดังนั้นเร็ว ก็จะโหลดไฟล์มาแค่ไฟล์เดียวเท่านั้น โดยใช้ CSS3 Font face ช่วยในการโหลด font เข้ามาใช้

EX.

### ການ Setting

```
@font-face {  
    font-family: 'Footer Icon';  
    src:  
        url('..../icon-css-zen-garden/fonts/icon-css-zen-garden.eot');  
        src: url('..../icon-css-zen-garden/fonts/icon-css-zen-garden.woff')  
            format('woff'),  
        url('..../icon-css-zen-garden/fonts/icon-css-zen-garden.ttf')  
            format('truetype');  
    font-weight: 300;  
    font-style: normal;  
}
```

### ການໃຊ້ Icon font

```
footer [class*="zen-"]::before {  
    display: block;  
    position: absolute;  
    color: black;  
    top: 0;  
    left: 0;  
    overflow: visible;  
    font-size: 40px;  
    text-indent: 0;  
    font-family: "Footer Icon";  
    font-weight: normal;  
    font-variant: normal;  
    text-transform: none;  
    line-height: 1;  
    -webkit-font-smoothing: antialiased;  
}  
.zen-validate-html::before {  
    content: "a";  
}
```

```

.zen-validate-css::before {
    content: "b";
}

.zen-license::before {
    content: "c";
}

.zen-accessibility::before {
    content: "d";
}

.zen-github::before {
    content: "e";
}

```

ตารางแสดงการ mapping ระหว่าง Icon และ character

Icon	Character
	"a"
	"b"
	"c"
	"d"
	"e"

## 6. CSS3 Transformation, Transition and Animation

>>อ่านเพิ่มเติม [bit.ly/Transformation-Transition-Animation](http://bit.ly/Transformation-Transition-Animation)<<

- Transformation ให้ความสามารถในการเลื่อน (Translate), หมุน (Rotate), และขยาย (Scale) Coordinate Space ของ Element
  - Transition ให้ความสามารถที่จะเปลี่ยน CSS properties จากค่าหนึ่งไปยังอีกค่าหนึ่งได้อย่างลื่นไหล ภายในระยะเวลาที่ระบุ
  - Animation ให้ความสามารถที่จะเปลี่ยน CSS properties จากค่าหนึ่งไปยังอีกค่าหนึ่งโดยใช้ต่อต่อระยะเวลาที่กำหนด โดยใช้ Keyframes ซึ่ง developer สามารถระบุและควบคุมได้ มันเยี่ยมมากกกก แนะนำ

## JavaScript

>>อ่านเพิ่มเติม [<<](http://bit.ly/Web-JavaScript)

### 1. JavaScript คืออะไร

คือ ภาษาคอมพิวเตอร์สำหรับการเขียนโปรแกรมบนระบบอินเทอร์เน็ต ที่กำลังได้รับความนิยมอย่างสูง JavaScript เป็นภาษาสคริปต์เชิงวัตถุ (ที่เรียกว่า "สคริปต์" (script) ซึ่งในการสร้างและพัฒนาเว็บไซต์ (ใช้ร่วมกับ HTML) เพื่อให้เว็บไซต์ของเราดูมีการเคลื่อนไหว สามารถตอบสนองต่อผู้ใช้งานได้มากขึ้น)

JavaScript เป็นภาษาสคริปต์ที่มีลักษณะดังนี้

- เป็นภาษาสคริปต์แบบ lightweight programming language (ภาษาสคริปต์แบบสัน្តิ)
- JavaScript สามารถใช้ร่วมกับ HTML และ CSS เพื่อสร้างและพัฒนาเว็บไซต์ ที่ใช้ได้กับ web browser รุ่นใหม่ๆ และใช้ได้ทั้ง PCs, laptops, tablets, smart phones, ด้วย

ความสามารถของ Javascript

- ทำให้สามารถเขียนโปรแกรมแบบง่ายๆ ได้ โดยไม่ต้องพึ่งภาษาอื่น
- มีคำสั่งที่ตอบสนองต่อผู้ใช้งาน
- ใช้ตรวจสอบข้อมูลได้ เช่น เตือนเมื่อกรอกรูปแบบข้อมูลผิดพลาด
- ใช้ตรวจสอบผู้ใช้ได้ เช่น ตรวจสอบว่าผู้ใช้งานใช้ Browser อะไร
- สร้าง Cookies ได้ 
- สร้างลูกเล่นให้เว็บไซต์ได้

### 2. วิธีการใช้งาน

เขียน Code ใน tag `<script></script>`

```
<head>
  <title></title>
  <script>
    document.write("<p>Learn programming Javascript</p>");
    document.write("<h1>www.mindphp.com</h1>");
  </script>
</head>
```

ใช้คำสั่ง `document.getElementById(id)` ซึ่งคือคำสั่งสำหรับการเข้าถึง Element Id ที่ต้องการใน HTML เช่น

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
</head>
<body>
  <h1 id="demo">My First JavaScript</h1>
  <script>
    document.getElementById("demo").innerHTML="Hello Nerd.";
  </script>
</body>
</html>
```

ผลลัพธ์ คือ

# Hello Nerd.

### 3. ลำดับการทำงาน

คำสั่งที่สั่งให้ JavaScrīpt ดำเนินงานตามลำดับที่ได้สั่งไว้ โดยการเขียนคำสั่งจะสิ้นสุดหรือจบคำสั่งด้วยเครื่องหมาย เชมิโคลอน (;)

ข้อควรระวัง การใช้อักษรตัวใหญ่กับอักษรตัวเล็กมีความแตกต่าง เพราะมีการแสดงผลที่แตกต่างกันไป เช่น

- การใช้คำสั่ง `getElementById` กับ `getElementbyID` มีผลลัพธ์ต่างกัน

```
<body>
  <p id="demo">Learning</p>
  <div id="name">Dream</div>

  <script>
    document.getElementById("demo").innerHTML="Self-Learning";
    document.getElementById("name").innerHTML="My Dream";
  </script>
</body>
```

ผลลัพธ์ คือ

Self-Learning

Dream

### 4. การประกาศตัวแปร

การเก็บค่า (value) ต่างๆเอาไว้ ซึ่งในบางครั้งอาจจะนำมาใช้ในการคำนวณค่าต่างๆ หรือ การเก็บค่าใดๆ เพื่อเอาไว้สำหรับอ้างอิงหรือตรวจสอบ

โดยขั้นตอนการประกาศตัวแปรในภาษา JavaScript นั้น จะขึ้นต้นด้วยคำว่า var และตามด้วยชื่อตัวแปร

ตัวอย่างของการประกาศตัวแปร

`var x;`

สำหรับในกรณีนี้เป็นการประกาศตัวแปรแบบกำหนดค่าด้วยเครื่องหมายเท่ากับแล้วตามด้วยค่าตัวแปร

ตัวอย่างของการกำหนดค่าตัวแปร

`var x = 10;`      `var x = true;`      `var x = "JavaScript";`

### 5. Function

คือ ชุดคำสั่งที่ใช้ในการทำงานอย่างโดยย่างหนึง ซึ่งจะทำงานเมื่อถูกเรียกใช้งาน  
ลักษณะของฟังก์ชันเป็นดังนี้

```
function function_name(argument) {
  // body...
}
```

?

`function_name` การตั้งชื่อฟังก์ชันมีหลักการคล้ายกับการตั้งชื่อตัวแปร โดยนิยม Camel Case

?

`argument` หรือพารามิเตอร์ คือข้อมูลบางอย่างที่ฟังก์ชันต้องใช้ในการประมวลผล ซึ่งอาจกิมเมนต์จะมีหรือไม่มีก็ได้

?

`// body...` อาจมีการ `return` ค่าได้

การเรียกใช้ Function

สำหรับการเรียกใช้ Function นั้นให้ระบุชื่อฟังก์ชันพร้อมอาร์กิวเมนต์ (ถ้ามี) ไว้ ณ จุดที่ต้องการใช้ฟังก์ชัน

```
function_name(argument);
```

## 6. Operators

เครื่องหมายในการดำเนินการกับจำนวน 2 จำนวน หรือมากกว่า

### ★ Arithmetic Operators

- + Addition
- - Subtraction
- \* Multiplication
- / Division
- % Remainder
- ++ Increment
- -- Decrement

### ★ Assignment Operators

- = Assignment
- += Addition assignment
- -= Subtraction assignment
- \*= Multiplication assignment
- /= Division assignment
- %= Remainder assignment

### ★ Comparison Operators

- < Less than
- <= Less than or equal
- > Greater than
- >= Greater than or equal
- == Equality
- != Inequality

### ★ Logical Operators

- && And
- || Or
- ! Not

## 7. Conditions

รูปแบบการเขียน

```
if (true) {  
    // code...  
} else if (true) {  
    // code...  
} else {  
    // code...  
}
```

## 8. Loops

○ For Loop

รูปแบบการเขียน

```
for (var i = 0; i < Things.length; i++) {  
    Things[i];  
};  
  
for (x in array) {  
    array[x];  
};
```

○ While Loop

รูปแบบการเขียน

```
while (condition) {  
    // code...  
};  
  
do {  
    // code...  
}  
while (condition);
```

## JSON

(JavaScript Object Notation)

ชื่อกีบอกในตัวเล่าย่าว่ามันคือ javascript ที่อยู่ในรูปแบบของ object ซึ่งมันช่วยให้เราสามารถทำงานได้ง่ายขึ้นซึ่งข้อดีมีดังนี้

1. ถูกออกแบบมาให้ง่ายสำหรับการแลกเปลี่ยนข้อมูล
2. ลักษณะข้อมูลสามารถอ่านหรือแก้ไขได้ง่าย
3. สามารถใช้ Javascript ช่วยในการเข้าถึงข้อมูลภายใน JSON ได้ง่าย
4. มีภาษาหลาย ๆ ภาษาที่รองรับการใช้งาน JSON

JSON เป็นข้อมูลในรูปแบบของ Object เราจะเรียกว่า JSON Object เริ่มจากเครื่องหมายปีกกาเปิด { และสิ้นสุดที่ปีกกาปิด } ภายใน object จะมีข้อมูลที่เรียกว่า member วิธีการเขียน json มีดังนี้

```
{"name": "value", "name": "value", ...}
```

## JQuery

jQuery คือ JavaScript Library ซึ่งถูกออกแบบมาเพื่อให้การเขียน JavaScript นั้นมีความสะดวกและง่ายขึ้น เพราะว่าการนำ JavaScript เอาไปประยุกต์กับงานจำพวกเว็บ (Client-side JavaScript) นั้นเป็นสิ่งที่ยุ่งยาก ไม่ว่าจะเป็นเรื่องความไม่เข้ากันของ Web Browser แต่ละค่าย, DOM หรือ API เป็นต้น

ดังนั้น jQuery จึงรวมเอา Object และ Function ต่างๆ ที่จำเป็นมาไว้ในรูปแบบของ Library พอกเป็นเช่นนี้แล้ว ไม่ว่าโค้ดที่คุณเขียนจะใช้ JavaScript หลายบรรทัดขนาดไหน ก็สามารถทำให้สั้นลงได้ อาจทำให้เหลือสั้นเพียงแค่บรรทัดเดียวเท่านั้น

jQuery ประกอบด้วยฟีเจอร์ต่างๆ ดังนี้

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

ตัวอย่างการใช้

JavaScript

```
document.getElementById("txt").value = "computer";
```

JQuery

```
$("#txt").val("computer");
```

# PHP

(PHP Hypertext Preprocessor)

PHP ได้รับการพัฒนาและออกแบบมา เพื่อใช้งานในการสร้างเอกสารแบบ HTML โดยสามารถสอดแทรกหรือแก้ไขเนื้อหาได้โดยอัตโนมัติ ดังนั้นจึงกล่าวว่า PHP เป็นภาษาที่เรียกว่า server-side หรือ HTML-embedded scripting language นั่นคือในทุกๆ ครั้งก่อนที่เครื่องคอมพิวเตอร์ซึ่งให้บริการเป็น Web server จะส่งหน้าเว็บเพจที่เขียนด้วย PHP ให้เรา มันจะทำการประมวลผลตามคำสั่งที่มีอยู่ให้เสร็จสิ้นก่อน และจึงค่อยส่งผลลัพธ์ที่ได้ให้เรา ผลลัพธ์ที่ได้นั้นคือเว็บเพจที่เราเห็นนั่นเอง ถือได้ว่า PHP เป็นเครื่องมือที่สำคัญชนิดหนึ่งที่ช่วยให้เราสามารถสร้าง Dynamic Web pages (เว็บเพจที่มีการโต้ตอบกับผู้ใช้) ได้อย่างมีประสิทธิภาพและมีลูกเล่นมากขึ้น

## 1. Basic Syntax

เพื่อเป็นการบ่งบอกให้รู้ว่า ส่วนใดเป็นคำสั่ง PHP ที่อยู่ภายในเอกสาร HTML จึงได้มีการกำหนดสัญลักษณ์ไว้ดังนี้ <?php // code... ?>

### 1.1. Variables

```
$mystring = "Hello World!";
$myinteger = 10;
$myfloat = 3.14;
```

### 1.2. Echo / Print

```
echo $mystring . "computer";
echo $myinteger + $myfloat;
print $mystring . "computer";
print $myinteger + 20;
```

## 2. PDO

## 3. OOP

## 4. Composer

### 4.1. Dependencies

### 4.2. Namespace

## Laravel

### Laravel - Installation

1. ติดตั้ง composer

2. Update composer

```
composer self-update
```

3. สร้าง Laravel Project

```
composer create-project --prefer-dist laravel/laravel project_name
```

## Laravel - Project Structure

 app	folder สำหรับเก็บ model ต่างๆ
 Http	
 Controllers	folder สำหรับเก็บ controller ต่างๆ
 Middleware	folder สำหรับเก็บ middleware ต่างๆ
 Policies	folder สำหรับเก็บ policy ต่างๆ
 Provider	folder สำหรับเก็บ provider ต่างๆ
 bootstrap	
 config	folder สำหรับเก็บการตั้งค่าต่างๆ
 app.php	file สำหรับตั้งค่า app
 mail.php	file สำหรับตั้งค่า mail
 database	
 factories	folder สำหรับเก็บ factory ต่างๆ
 migrations	folder สำหรับเก็บ migration ต่างๆ
 seeds	folder สำหรับเก็บ seeder ต่างๆ
 node_modules	folder สำหรับเก็บ javascript library
 public	folder สำหรับเก็บรูปภาพ/js/css ที่ client สามารถเข้าถึงได้
 css	folder สำหรับเก็บ css ที่ทำการ compile แล้ว
 js	folder สำหรับเก็บ js ที่ทำการ compile แล้ว
 resources	
 assets	
 js	folder สำหรับเก็บ javascript ที่เปลี่ยนไป
 sass	folder สำหรับเก็บ css ที่เปลี่ยนไป
 lang	folder สำหรับเก็บข้อความภาษาต่างๆ
 views	folder สำหรับเก็บ blade template
 routes	folder สำหรับเก็บ route file ต่างๆ
 api.php	file สำหรับกำหนด route แบบ api
 web.php	file สำหรับกำหนด route แบบ web
 storage	
 app	
 public	folder สำหรับเก็บไฟล์ต่างๆ
 framework	
 views	folder สำหรับ blade template ที่ compile แล้ว
 tests	
 vendor	folder สำหรับเก็บ PHP library
 .env	file สำหรับตั้งค่า database และ connection ต่างๆ
 .env.example	
 .gitignore	file สำหรับระบุ file/folder ที่จะไม่อัพโหลด ของ git
 artisan	
 composer.json	file สำหรับระบุ PHP library ที่ใช้
 package.json	file สำหรับระบุ Javascript library ที่ใช้
 webpack.mix.js	file สำหรับระบุ javascript/sass ที่จะถูก compile

## Laravel - Artisan Command

### Available Command

tinker เปิด tinker

### Config

config:clear ล้างค่า cache ควรใช้หลังแก้ไข .env

### db

db:seed สร้าง DatabaseSeeder ทำงาน

db:seed --class=FooSeeder สร้าง seeder แบบระบุ class

### key

key:generate สร้าง app key ควรทำหลังจากการ clone project

### make

make:auth สร้าง controller, view สำหรับระบบ authentication

make:controller BarsController สร้าง controller

make:controller BarsController --model=Bar สร้าง controller แบบระบุ model ที่จะทำงานด้วย

make:controller BarsController --model=Bar --resource สร้าง resource controller

make:controller BarsController --model=Bar --api สร้าง api resource controller

make:factory BarsFactory สร้าง factory

make:factory BarsFactory --model=Bar สร้าง factory แบบระบุ model ที่จะสร้าง

make:migration CreateBarTableMigration สร้าง migrate file

make:model Bar สร้าง model

make:model Bar -m สร้าง model พร้อม migrate file

make:policy BarPolicy สร้าง policy

make:seeder BarsTableSeeder สร้าง seeder

### migrate

migrate สร้าง up 파일ที่ยังไม่ได้ถูก migrate

migrate:rollback สร้าง down 파일 migrate ครั้งล่าสุดทั้งหมด

migrate:reset สร้าง down 파일 migrate ทั้งหมด

migrate:refresh สร้าง down 파일 migrate ทั้งหมด และ up ใหม่ทั้งหมด

### route

route:list แสดง route ทั้งหมดของ project

## Laravel - Route

### What is route?

route คือการกำหนดว่า url ต่างๆที่เข้ามาจะแสดง page ไหน

### Basic Route

```
2 Route::get("/bars", function(){
3     return "this is Bar page";
4 });
```

หากเข้าไปที่ path *localhost:8000/bars* ก็จะได้ this is Bar page

```
6 Route::get("/bars", function(){
7     return view("bars.index");
8 });
```

แสดง page ที่อยู่ใน *resources/views/bars/index.blade.php*

### Route with Parameter

```
10 Route::get("/bars/{id}", function($id){
11     return "id = " . $id;
12 });
```

หากเข้าไปที่ *localhost:8000/bars/1* ก็จะได้ id = 1

### Route with Optional Parameter

```
13
14 Route::get("/bars/{id?}", function($id){
15     if(is_null($id))
16         return "not have id";
17     else
18         return "id = " . $id;
19 });
```

หากเข้าไปที่ *localhost:8000/bars* ก็จะได้ not have id

### Pass Parameter to Blade Template

```
21 Route::get("/bars/{id}/{name}", function($id, $name){
22     return view("bars.show", ["id" => $id, "name" => $name]);
23     return view("bars.show", compact("id", "name"));
24     return view("bars.show")->with("id", $id)->with("name", $name);
25 });
```

วิธีส่ง ค่าไปหา blade template ทำได้ 3 แบบ (อย่า return ซ้อนกัน 3 อันนะ ปัญญาอ่อน แค่กูซึ่งเกียจเขียนหลายรอบ)

### Route to Controller

```
27 Route::get("/bars", "BarsController@index");
```

### Naming Route

```
27 Route::get("/bars", "BarsController@index")->name("bar.index");
```

## Checking Parameter

```
27 Route::get("/bars/{id}", "BarsController@showId")->where("id", "[0-9]+");  
28 Route::get("/bars/{name}", "BarsController@showName")->where("name", "[a-zA-Z]+")
```

ถ้าไปที่ `localhost:8000/bars/1` จะไปที่ BarsController@showId

ถ้าไปที่ `localhost:8000/bars/john` จะไปที่ BarsController@showName

## Route Method

```
30 Route::get("/bars/{id}", "BarsController@show");  
31 Route::post("/bars", "BarsController@create");  
32 Route::put("/bars/{id}", "BarsController@update");  
33 Route::delete("/bars/{id}", "BarsController@destroy");
```

ไม่ได้มีแค่ get นะ มี 4 method ตามคอนเซปต์ของ REST

### Display Variable

สามารถใช้ {{ }} ภายในส่วนใดของ HTML ก็ได้ เพื่อแสดงค่าของตัวแปร

```
<p>Hi, {{ $username }}</p>
<p> {{ $user->id }} </p>
```

เรียก function ก็ได้นะ

```
<a href="{{ url('/users') }}"> List</a>
```

### If Statement

```
@if($user->role === "admin")
    Hi Admin
@elseif($user->role === "customer")
    Welcome
@else
    Hi
@endif
```

### Loop

```
@for ($i = 0; $i < 10; $i++)
    The current value is {{ $i }}
@endfor

@foreach ($users as $user)
    <p>This is user {{ $user->id }}</p>
@endforeach

@while (true)
    <p>Im looping forever.</p>
@endwhile
```

## Loop Variable

ตัวแปรที่สามารถใช้ได้ใน loop

```
@for($users as $user)
    // แสดงหมายเลขของลูปว่ารอบที่เท่าไหร (เริ่มที่ 0)
    Index. {{ $loop->index }}
    // แสดงหมายเลขของลูปว่ารอบที่เท่าไหร (เริ่มที่ 1)
    No. {{ $loop->iteration }}
@if($loop->first)
    this is first round
@elseif($loop->last)
    this is last round
@endif
@endfor
```

## Comment

สำหรับ blade template ห้ามใช้ // ในการ comment ให้ใช้ {{-- --}}

```
{{-- this is a comment --}}
```

## PHP

สามารถใช้คำสั่ง php ได้ภายในคำสั่ง @php

```
<p>
    @php
        $name = 'natthapach';
    @endphp
    my name is {{ $name }}
</p>
```

Layout file

```
# layouts/app.blade.php
<div> ----- Header ----- </div>
<div>
    @yield('content')
</div>
<script src='main.js' />
@stack('js')
```

```
# home.blade.php
@extends('layouts.app')

@section('content')
    <div> this is content </div>
@endsection

@push('js')
    <script src='home.js' />
@endpush
```