JavaScript

JavaScript

JavaScript เป็นภาษาโปรแกรมเชิงวัตถุแบบไดนามิกไทป์ (Dynamic Types) ไวยากรณ์ได้มาจากโครงสร้างของภาษา Java กับภาษา C ต้องทำงานอยู่บน JavaScript Engine ที่อยู่บนเว็บบราวเซอร์

JavaScript

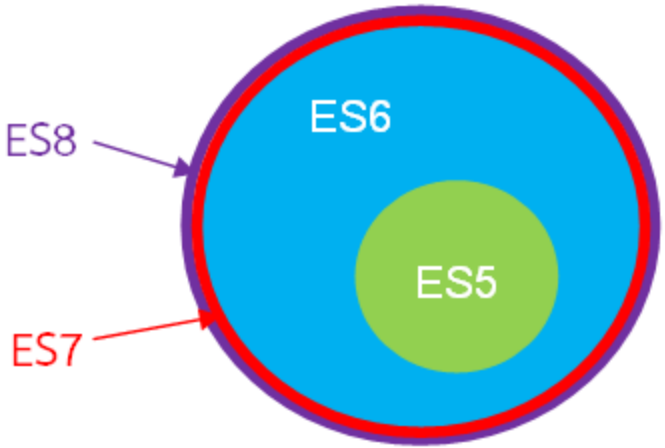
ปัจจุบัน JavaScript สามารถทำงานได้หลากหลายรูปแบบขึ้น

- **Client (Frontend)** :: jQuery, Vue, Angular, React and etc.
- **Server (Backend)** :: Node.js

JavaScript :: ECMAScript

องค์กร ECMA International (องค์กรจัดการมาตรฐานแห่งยุโรป) เป็นผู้กำหนดมาตรฐาน JavaScript เรียกมาตรฐานนี้ว่า ECMA-262 ส่วนตัวภาษา JavaScript นั้นมีชื่อเรียกเต็มยศว่า ภาษา ECMAScript

ES6 (ECMAScript 2015) เป็นมาตรฐานใหม่ล่าสุดของ JavaScript ประกาศใช้เมื่อ มิถุนายน 2558 หลังจากไม่ได้เปลี่ยนแปลงมานานเกือบ 6 ปี ใน ES5



ECMAScript :: ES5

Variable

- ต้องไม่ซ้ำกับ Reserved Word
- ขึ้นต้นด้วยตัวอักษร หรือเครื่องหมาย “_” หรือเครื่องหมาย $
- สามารถใช้ตัวเลขร่วมได้แต่ห้ามใช้ตัวเลขขึ้นต้น
- รูปแบบตัวพิมพ์เล็กพิมพ์ใหญ่ถือเป็นคนละตัว

```
var a = 10
var $code = 1, _defaule2 = 2, _ref = 3, $Code = 4
```

ECMAScript :: ES5

String

การเชื่อมสตริงใช้เครื่องหมาย “ + ”

```
var firstName = "George R."
var lastName = "R. Martin"
var name = firstName + lastName // George R.R. Martin
```

Comments

```
//var firstName = "George R."

/*var lastName = "R. Martin"
var name = firstName + lastName // George R.R. Martin */
```

Note: conditions, Loop, Operator, Function เหมือนกับภาษา PHP

ECMAScript :: ES6

Variables Var, Let and Const

- **`var`** เป็น Function Scoped จะประกาศจุดไหนก็ตาม จะขยับไปอยู่ใกล้จุดประกาศฟังก์ชั่นที่ใกล้ที่สุด (hoisting)

- **`let`** เป็น block-scoped ประกาศอยู่ใน block ไหน ก็จะทำงานอยู่ภายในนั้น สามารถเปลี่ยนค่าได้

- **`const`** เป็น block-scoped ประกาศอยู่ใน block ไหน ก็จะทำงานอยู่ภายในนั้น เป็นค่าคงที่ไม่สามารถเปลี่ยนค่าได้

ECMAScript :: ES6

```
// พิมพ์ 2 ออกมาทั้งสองครั้ง
for(var i = 0; i < 2; i++) {
  // เนื่องจากในจังหวะที่ console.log ทำงาน loop ได้วนไปจนครบแล้ว
  // ทำให้ในขณะนั้นค่า i มีค่าเป็น 2
  // อย่าลืมว่า i ประกาศโดยใช้ var มันจึงผลักตัวเองออกจาก for
  // หรือพูดง่ายๆคือ i นั้นเป็นตัวแปรตัวเดียวทุกครั้งที่วนลูปก็เพิ่มค่าใส่ตัวแปรเดิม
  setTimeout(function() { console.log(i) }, 100) // 2 2
}

// ผลลัพธ์ได้ 0 และ 1
// ใช้ let ประกาศตัวแปรทำให้ตัวแปรเป็น block-scoped
// พูดง่ายๆคือ i จะเกิดขึ้นทุกครั้งที่วนลูป
// และไม่ซ้ำกับ i ก่อนหน้า
// เนื่องจากเป็น block-scoped มันจึงมีช่วงชีวิตอยู่แค่ใน {}
for(let i = 0; i < 2; i++) {
  setTimeout(function() { console.log(i) }, 100) // 0 1
}
```

ECMAScript :: ES6

Import / Export

```
1  // dog.js
2  export const DEFAULT_COLOR = 'white'
3  export function walk() {
4    console.log('Walking...')
5  }
6
7  // main.js
8  // เลือกนำเข้าเฉพาะ DEFAULT_COLOR
9  import { DEFAULT_COLOR } from './dog.js'
10
11 // main.js
12 // นำเข้าทุกสรรพสิ่งที่ export จาก dog
13 // แล้วตั้งชื่อใหม่ให้ว่า lib
14 import * as lib from './dog.js'
15
```

ECMAScript :: ES6

Destructuring

```
let person = {
  age: 24,
  gender: 'male',
  name: {
    firstName: 'firstName',
    lastName: 'lastName'
  }
}

// ถ้าเราต้องการค่าเหล่านี้ออกจากอ็อบเจ็กต์ ต้องมาประกาศตัวแปรแบบนี้
let age = person.age
let genger = person.gender
let name = person.name
let firstName = name.firstName
let lastName = name.lastName

// หากใช้ Destructuring จะเหลือแค่นี้
let { age, gender, name } = person
let { firstName, lastName } = name

// แต่ในความเป็นจริงแล้ว name เป็นเพียงแค่ทางผ่าน
// เราไม่ต้องการมัน เราต้องการแค่ firstName และ lastName
// จึงใช้ Destructuring ซ้อนเข้าไปอีกครั้ง
// เพียงเท่านี้ตัวแปร name ก็จะไม่เกิดขึ้นมาให้รำคาญใจ
let { age, gender, name: { firstName, lastName } } = person

```

ECMAScript :: ES6

Spread Operator

```
let obj1 = { a: 1, b: 2 }
let obj2 = { c: 3, d: 4 }
console.log({ ...obj1, ...obj2 }) // {"a":1,"b":2,"c":3,"d":4}

let arr1 = [1, 2, 3]
let arr2 = [4, 5, 6]
console.log([...arr1, ...arr2]) // [1,2,3,4,5,6]

```

ECMAScript :: ES6

Arrow Function

```
function Dog() {
  this.color = 'white'

  setTimeout(function() {
    // this ตัวนี้หมายถึง this ใน context ของฟังก์ชันนี้
    // จึงไม่มีการพิมพ์อะไรออกไป เพราะในฟังก์ชันนี้ this ไม่มีค่าของ color
    console.log(this.color)
  }, 100)
}

new Dog()

// ถ้าต้องการให้พิมพ์ค่า color ออกมาต้องแก้ไขใหม่เป็น
function Dog() {
  this.color = 'white'
  let self = this

  setTimeout(function() {
    // เรียกผ่านตัวแปร self แทน
    console.log(self.color)
  }, 100)
}

// หรือใช้ arrow function ดังนี้
function Dog() {
  this.color = 'white'

  setTimeout(() => {
    // this ของ arrow function นี้จะหมายถึง
    // this ตัวบน
    console.log(this.color)
  }, 100)
}

```

ECMAScript :: ES6

Default Values

```
1  // ES5
2  function foo(genger) {
3    gender = gender || 'male;
4  }
5
6  // ES2015
7  function foo(gender = 'male') {
8
9  }
10
```

ECMAScript :: ES6

Named Parameters

```
1  // ES5
2  function request(options) {
3    var method = options.method || 'GET'
4    var ssl = options.ssl || false
5    console.log(method)
6  }
7  request({})
8
9  // ES2015
10 function request({ method='GET', ssl=false }) {
11   console.log(method)
12 }
13 request({})
14
```

ECMAScript :: ES6

Rest Parameters

```
1  // ไม่ว่าจะส่งตัวเลขเข้ามากี่ตัว ตัวแรกจะเป็น initial
2  // ส่วนตัวอื่นจะเก็บอยู่ในอาร์เรย์ชื่อ rest
3  function sum(initial, ...rest) {
4    return rest.reduce((prev, cur) => prev + cur, initial)
5  }
6
7  console.log(sum(10, 1, 2, 3)) // 16
8
```

ECMAScript :: ES6

ลดความซ้ำซ้อน ลดรูปแบบการประกาศฟังก์ชั่นในอ็อบเจกต์ โดยละคีย์เวิร์ด function

```
1  // ES5
2  const obj = {
3    foo: function() {
4
5    }
6  }
7
8  // ES2015
9  const obj = {
10   foo() {
11
12   }
13 }
14
```

ECMAScript :: ES6

Template String

```
1  const name = 'Nuttavut Thongjor'
2  console.log(`สวัสดีชาวโลก ผมชื่อ${name}`)
3
4  // นอกจากนี้ template string ยังเอื้อต่อการทำ multiline string ด้วย
5  const longString = `
6    Lorem Ipsum is simply dummy text of the printing
7    and typesetting industry.
8    Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
9    when an unknown printer took a galley of type and scrambled
10   it to make a type specimen book.
11 `
12
```

ECMAScript :: ES6

Class

```
// ประกาศคลาสผ่านคีย์เวิร์ด class
class Person {
  constructor(name, age) {
    this.name = name
    this.age = age
  }

  static species = 'Homo sapiens sapiens'

  walk() {
    console.log("I'm walking...")
  }

  print() {
    console.log(`My name is ${this.name}`)
    console.log(`I'm ${this.age}'`)
  }
}

const person = new Person('MyName', 99)
person.walk() // I'm walking...
person.print() // My name is MyName \n I'm 99'

// static method เรียกตรงผ่านคลาสได้เลย
console.log(Person.species) // Homo sapiens sapiens

// สำหรับการทำ inheritance สามารถใช้คีย์เวิร์ด extends ดังนี้
class Female extends Person {

}

```

ECMAScript :: ES6

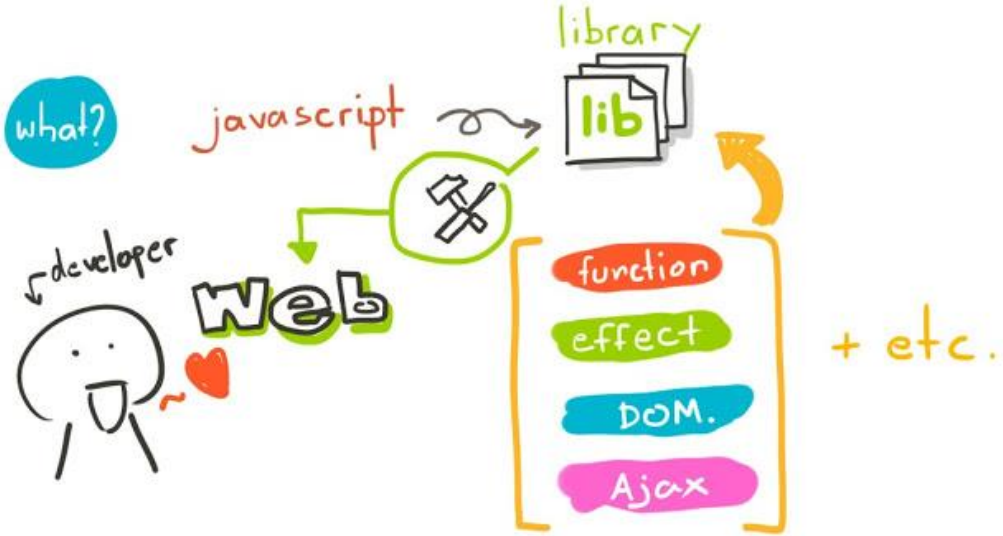
ปัจจุบันการจะใช้งาน ECMAScript ES6 ขึ้นไป ต้องมี JavaScript compiler ก่อน เนื่องจาก Browser ยังไม่รองรับทั้งหมด แต่หากรันด้วย Node.js สามารถรองรับ ES6 ได้ 99%

https://babeljs.io/


jQuery
write less, do more.

jQuery

jQuery คือ JavaScript Library ที่ถูกออกแบบมาให้การเขียน Javascript มีความสะดวกและง่ายขึ้น มีการรวบรวม function ของ JavaScript ให้อยู่ในรูปแบบ Patterns Framework ที่สะดวกและง่ายต่อการใช้งาน มีความยืดหยุ่นรองรับต่อการใช้งาน Cross Browser



jQuery :: Feature

jQuery มีความสามารถที่ทำได้หลักๆ คือ

DOM traversing :: การวิ่งไปตาม DOM ของ HTML ในหน้าเพจนั้นๆ

Event handling :: จัดการกับ Event ต่างๆ

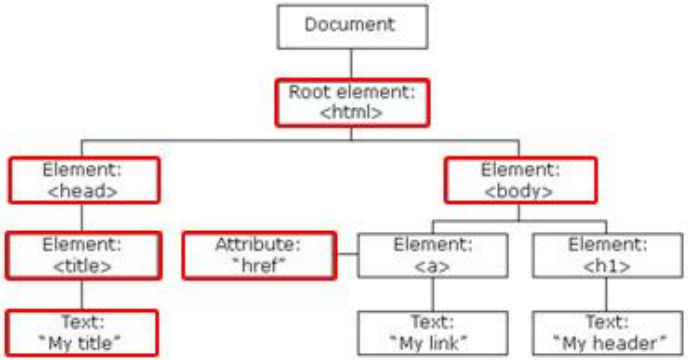
Animation :: ทำให้วัตถุมีการ animate

AJAX :: จัดการกับ AJAX

DOM :: HTML DOM

DOM (Document Object Model) คือ การจัดเอกสาร HTML ให้อยู่ในรูปแบบของ DOM Tree โดยนิยาม objects และ properties สามารถปรับเปลี่ยนโครงสร้าง ข้อมูล และ styles ของ HTML DOM ผ่าน methods ได้

```
<html>
<head>
    <title>My Title</title>
</head>
<body>
    <a>My link</a>
    <h1>My header</h1>
</body>
</html>
```



รู้จัก HTML DOM กับห้องลับ
http://honglub.com

jQuery :: Selectors

วิธีการกำหนด Selector ของ jQuery

$(selector)

CSS Selector

Filter Selector

Traversal Method

** jQuery.noConflict(); **

jQuery :: Filter Selectors

:first	Element ตัวแรกสุด
:last	Element ตัวสุดท้าย
:odd	Element ในลำดับคี่
:even	Element ในลำดับคู่
:header	Element ชนิด h1...h6
:gt(n)	Element ที่อยู่ถัดจาก Element ในลำดับที่ระบุ
:li(n)	Element ที่อยู่ก่อน Element ในลำดับที่ระบุ
:eq(n)	Element ที่อยู่ในลำดับที่ระบุ
:not(n)	เลือก Element อื่นทั้งหมด ยกเว้น Element ในลำดับที่ระบุ
:not(selector)	เลือก Element อื่นทั้งหมด ยกเว้น Element ที่ตรงกับ Selector

jQuery :: Filter Selectors

:button	Element ที่กำหนดด้วย <button>,<input type=”button”>
:input	Element ที่กำหนดด้วย <input>,<textarea>,<select>,<button>
:text	Element ที่กำหนดด้วย <input type=“text”>
:password	Element ที่กำหนดด้วย <input type=“password”>
:checkbox	Element ที่กำหนดด้วย <input type=“checkbox”>
:radio	Element ที่กำหนดด้วย <input type=“radio”>
:image	Element ที่กำหนดด้วย <input type=“image”>
:file	Element ที่กำหนดด้วย <input type=“file”>
:reset	Element ที่กำหนดด้วย <input type=“reset”>
:submit	Element ที่กำหนดด้วย <input type=“submit”>

jQuery :: Filter Selectors

:checked	Element ที่กำหนด Attribute ด้วย checked
:selected	Element ที่กำหนด Attribute ด้วย selected
:disabled	Element ที่กำหนด Attribute ด้วย disabled
:enabled	Element ที่ไม่กำหนด Attribute ด้วย disabled
:focus	Element ที่ถูกโฟกัสขณะนั้น
:hidden	Element ที่ถูกซ่อน <input type="hidden">,display:none,visibility:hidden,height=0:width=0;
:visible	Element ทั้งหมดที่ปรากฏให้เห็น
:first-child	Element ที่เป็น Child ลำดับแรกของ Parent
:last-child	Element ที่เป็น Child ลำดับสุดท้ายของ Parent
:only-child	Element ที่เป็น Child เพียงอันเดียวภายใน Parent

jQuery :: Filter Selectors

:only-of-type	ชนิด Element ที่เป็น Child เพียงอันเดียวภายใน Parent
:first-of-type	ชนิด Element ลำดับแรกของ Parent
:last-of-type	ชนิด Element ลำดับสุดท้ายของ Parent
:nth-child(n)	Element ที่เป็น Child ในลำดับของ Parent
:nth-last-child(n)	Element ที่เป็น Child ในลำดับของ Parent เริ่มนับจากตัวสุดท้าย
:nth-of-type(n)	ชนิด Element ที่เป็น Child ในลำดับของ Parent
:nth-last-of-type(n)	ชนิด Element ที่เป็น Child ในลำดับของ Parent เริ่มนับจากตัวสุดท้าย
:contains(string)	Element ที่มี String ที่ระบุเป็นส่วนประกอบ
:has(selector)	Element ที่มี Element ตรงกับที่ระบุ
:empty	Element ที่ไม่มีเนื้อหาใดๆ
:parent	Element ที่เป็น Parent ของ Element อื่นๆ

jQuery :: Traversal Method

filter(selector)	กรองเอาเฉพาะ Element ที่ระบุ
eq(n)	เลือก Element ในลำดับที่ระบุ
first()	เลือก Element อันแรกสุด
last()	เลือก Element อันสุดท้าย
has(selector)	เลือก Element ที่มี Child ตรงกับที่ระบุ
is(selector)	ใช้ตรวจสอบ Element ตรงกับชนิดที่ระบุหรือไม่ ใช้ร่วมกับ if คืนค่าเป็น Boolean
not(selector)	เลือก Element ที่ไม่ตรงกับที่ระบุ
children(), children(selector)	เลือก Element ทั้งหมดที่เป็น Child หรือตรงกับที่ระบุ
find(selector)	ค้นหา Element ตรงกับที่ระบุ ที่อยู่ภายใน parent นั้น
next(), next(selector)	เลือก Element ที่อยู่ถัดไปที่อยู่ติดกัน หรือตรงกับที่ระบุ

jQuery :: Traversal Method

nextAll(), nextAll(selector)	เลือก Element ที่อยู่ถัดไปทั้งหมด หรือ และตรงกับที่ระบุ
nextUntil(selector)	เลือกทุก Element จนกระทั่งถึง Element ก่อนที่ระบุ
prev(), prev(selector)	เลือก Element ก่อนหน้าที่อยู่ติดกัน หรือตรงกับที่ระบุ
prevAll(), prevAll(selector)	เลือก Element ทั้งหมดที่อยู่ก่อนหน้า หรือ และตรงกับที่ระบุ
prevUntil(selector)	เลือกทุก Element ที่อยู่ก่อนจนกระทั่งถึง Element ที่ตรงกับ Element ที่ระบุ
parent(), parent(selector)	เลือก Element ที่เป็น Parent หรือ เป็น Parent ของ Element ที่ระบุ
parents(), parents(selector)	เลือก Element ที่เป็น Parent ทั้งหมด (กรณีวางซ้อนกันหลายชั้น)
parentsUntil(selector)	เลือก Element ที่เป็น Parent ทั้งหมดจนกระทั่งถึง Parent ที่อยู่ก่อนที่ระบุ
siblings()	เลือก Element ทั้งหมดที่อยู่ ก่อนและหลัง อยู่ใน Parent เดียวกัน

jQuery :: Ready & Load

`$(document).ready()` :: ทำงานเมื่อ content ของไฟล์ HTML โหลดครบและสร้าง DOM เรียบร้อยแล้ว (ไม่สนรูปภาพ ไม่สนCSS)

`$(window).load()` :: ทำงานเมื่อ content ของไฟล์ HTML โหลดเสร็จและสร้าง DOM เรียบร้อยแล้ว และมีการโหลด img, css, iframe and etc ทุกอย่างเสร็จสิ้น

jQuery :: Event Handling

Event ส่วนใหญ่คล้ายกับ JavaScript มีพื้นฐานที่น่าสนใจ ดังนี้

click	เมื่อคลิก
dblclick	เมื่อดับเบิ้ลคลิก
change	เมื่อค่าเปลี่ยนไป หรือเปลี่ยนแปลงการเลือก
focus	เมื่อได้รับโฟกัส
blur	เมื่อย้ายโฟกัสออก
keydown	เมื่อกดปุ่มคีย์บอร์ดลงไป
keypress	เมื่อคีย์บอร์ดถูกกด (เกิดหลัง keydown)
keyup	เมื่อปล่อยมือจากการกดคีย์บอร์ด (เกิดหลัง keypress)
hover	เมื่อเลื่อนเม้าส์เข้าและออก element

jQuery :: Event Handling

mousedown	เมื่อคลิก
mouseup	เมื่อดับเบิ้ลคลิก
mouseseenter	เมื่อค่าเปลี่ยนไป หรือเปลี่ยนแปลงการเลือก
mouseleave	เมื่อได้รับโฟกัส
mousemove	เมื่อย้ายโฟกัสออก
mouseout	เมื่อกดปุ่มคีย์บอร์ดลงไป
mouseover	เมื่อคีย์บอร์ดถูกกด (เกิดหลัง keydown)
resize	เมื่อปล่อยมือจากการกดคีย์บอร์ด (เกิดหลัง keypress)
scroll	เมื่อเลื่อน scrollbar ของ browser
select	เมื่อเลือกข้อความใน element ชนิด text และ textarea
submit	เมื่อส่งข้อมูลจากฟอร์ม

jQuery :: Event Handling

Properties ของ Event ที่น่าสนใจ

pageX	ระยะห่างจากตำแหน่งที่เม้าส์ชี้อยู่ไปทางขอบด้านซ้ายของเอกสาร
pageY	ระยะห่างจากตำแหน่งที่เม้าส์ชี้อยู่ไปทางขอบด้านบนของเอกสาร
target	Element เป้าหมายที่เกิด Event
type	ชนิดหรือชื่อของ Event ที่เกิดขึ้น
which	ตรวจสอบปุ่มเมาส์หรือคีย์บอร์ดที่ถูกกด - เม้าส์ ผลลัพธ์ 1 = ปุ่มซ้าย, 2 = ปุ่มกลาง, 3 = ปุ่มขวา - คีย์บอร์ด ได้ค่ารหัสปุ่มคีย์บอร์ด A = 65
preventDefault()	ใช้ป้องกันไม่ให้เกิดการกระทำที่เป็นลักษณะปกติหรือ Default ของ Element นั้น
stopPropagation()	ป้องกันการเกิด Event ซ้อนกันเมื่อ Element ถูกซ้อนทับกัน แล้วแต่ละ Element มี Event ของตัวเองอยู่

jQuery

Loop Array and Objects

jQuery :: Loop Array

```
var x = [1,2,3,4];
var y = {a:1,b:2,c:3,d:4};

$.each(x, function(i, item){
  console.log(item); // 1 2 3 4
});

$.each(y, function(i, item){
  console.log(i + " : " + item); // a:1 b:2 c:3 d:4
});

console.log(y['d']); // 4
```

jQuery :: Loop Object

```
var z = {"obj" : {"a":"first","b":"second","c":"third","d":"forth"}};

$.each(z.obj, function(i, item){
  console.log(i + " : " + item); // a:first b:second c:third d:forth
});
```

JSON

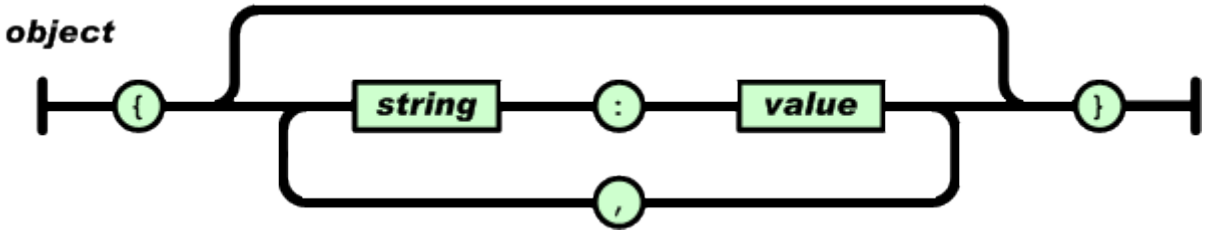
JSON

JSON หรือ Java Script Object Notation คือ รูปแบบข้อมูลที่ใช้สำหรับแลกเปลี่ยนข้อมูลที่มีขนาดเล็ก มีความอิสระอย่างสมบูรณ์ ถูกกำหนดภายใต้ภาษา JavaScript เป็นวิธีการที่ทำให้ JavaScript แลกเปลี่ยนข้อมูลกับ Server ได้อย่างง่ายดาย มีรูปแบบประโยคโดยใช้ [ ] แทน array และใช้ { } แทน hash (associate array) แต่ละสมาชิกคั่นด้วย “ , ” และแต่ละชื่อของสมาชิกคั่นด้วย “ : ”

ใน jQuery ใช้ ฟังก์ชั่น JSON.parse(result) ในการแปลง JSON ให้เป็น Object และในทางกลับกัน ใช้ฟังก์ชั่น JSON. stringify ในการทำให้ข้อมูลอยู่ในรูปแบบของ JSON

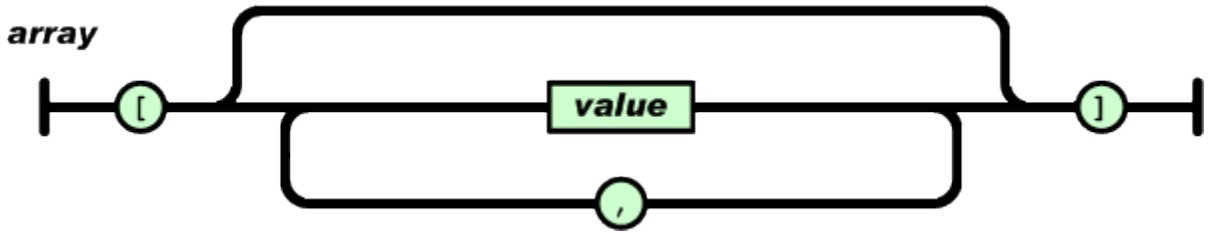
JSON :: Format

Object :: การจัดเก็บในชุดข้อมูลที่มีชื่อข้อมูลและข้อมูลคู่กัน หรือ associative array



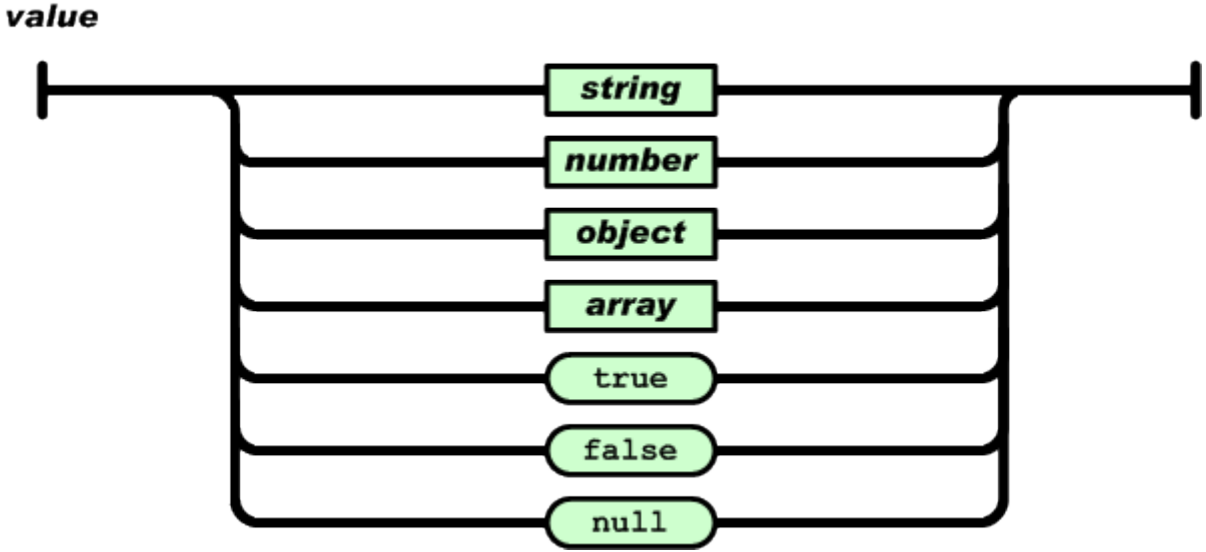
JSON :: Format

Array :: ลำดับของค่าข้อมูล จัดอยู่ในรูปแบบของ array



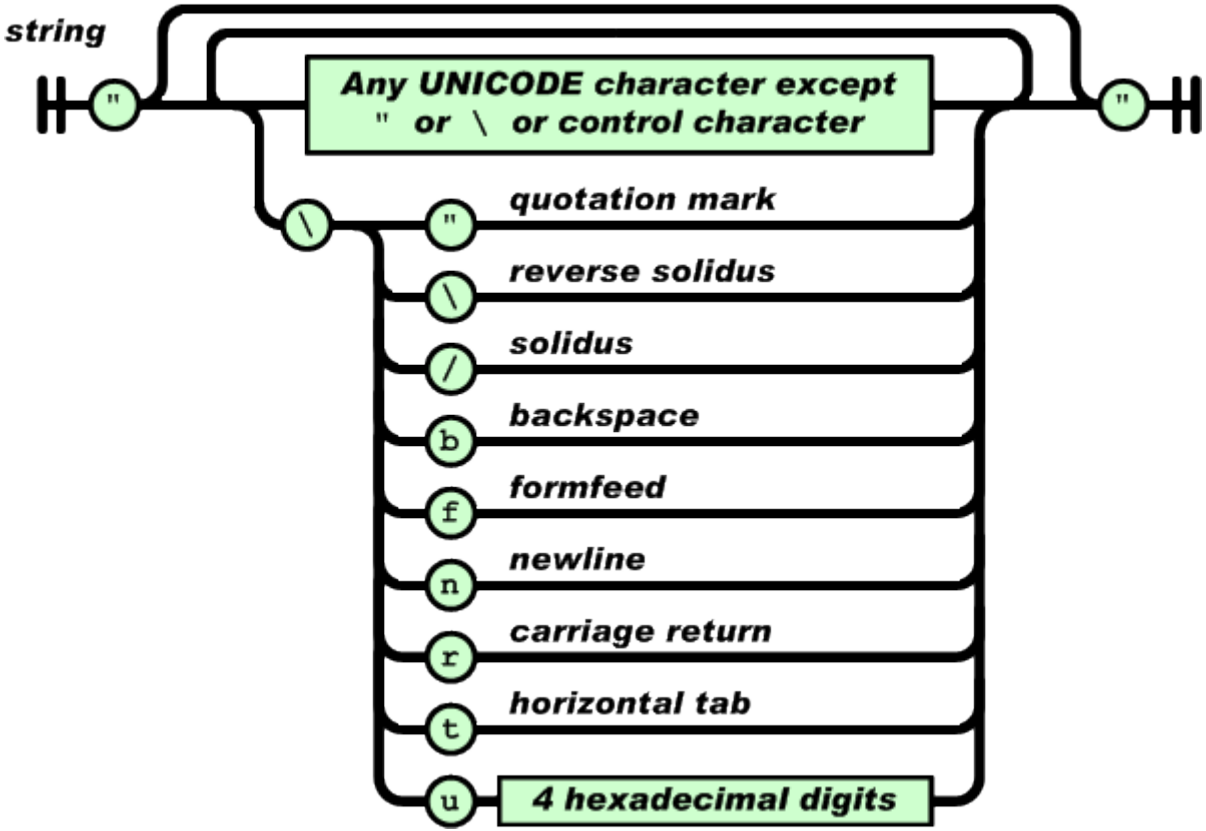
JSON :: Format

Value



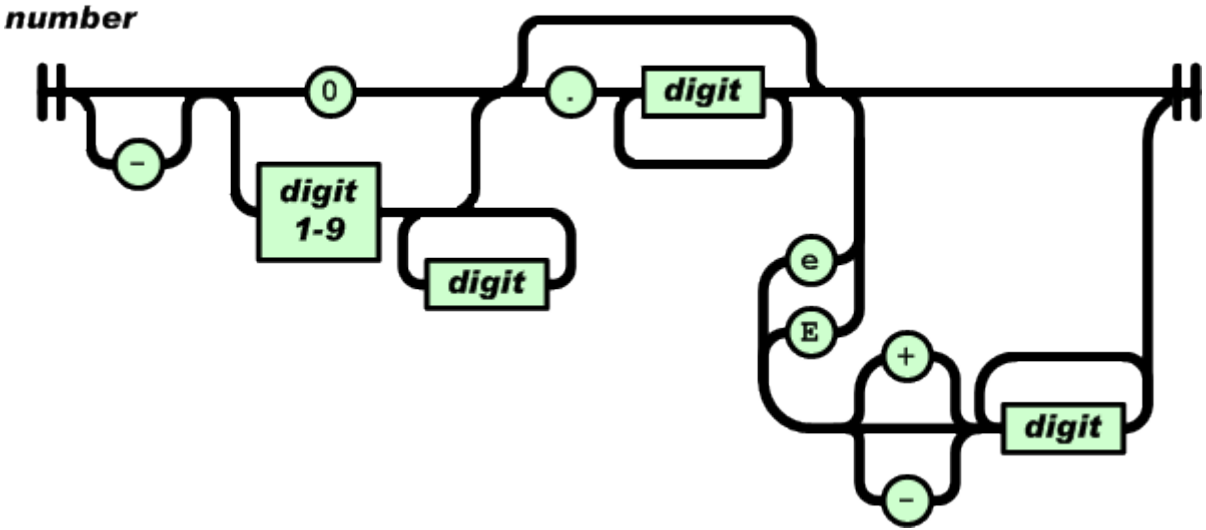
JSON :: Format

String



JSON :: Format

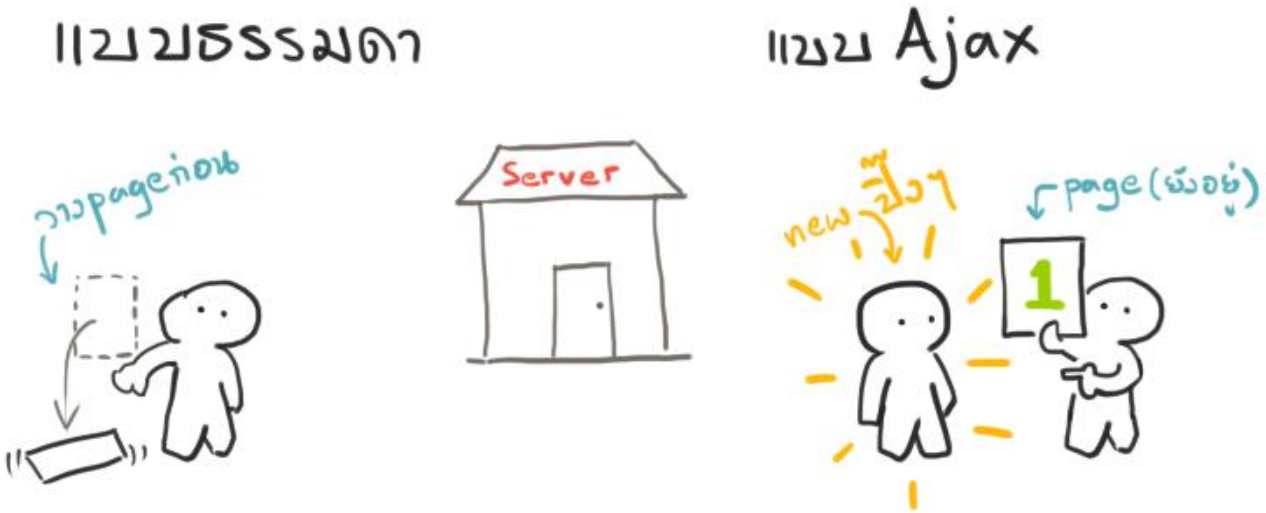
Number

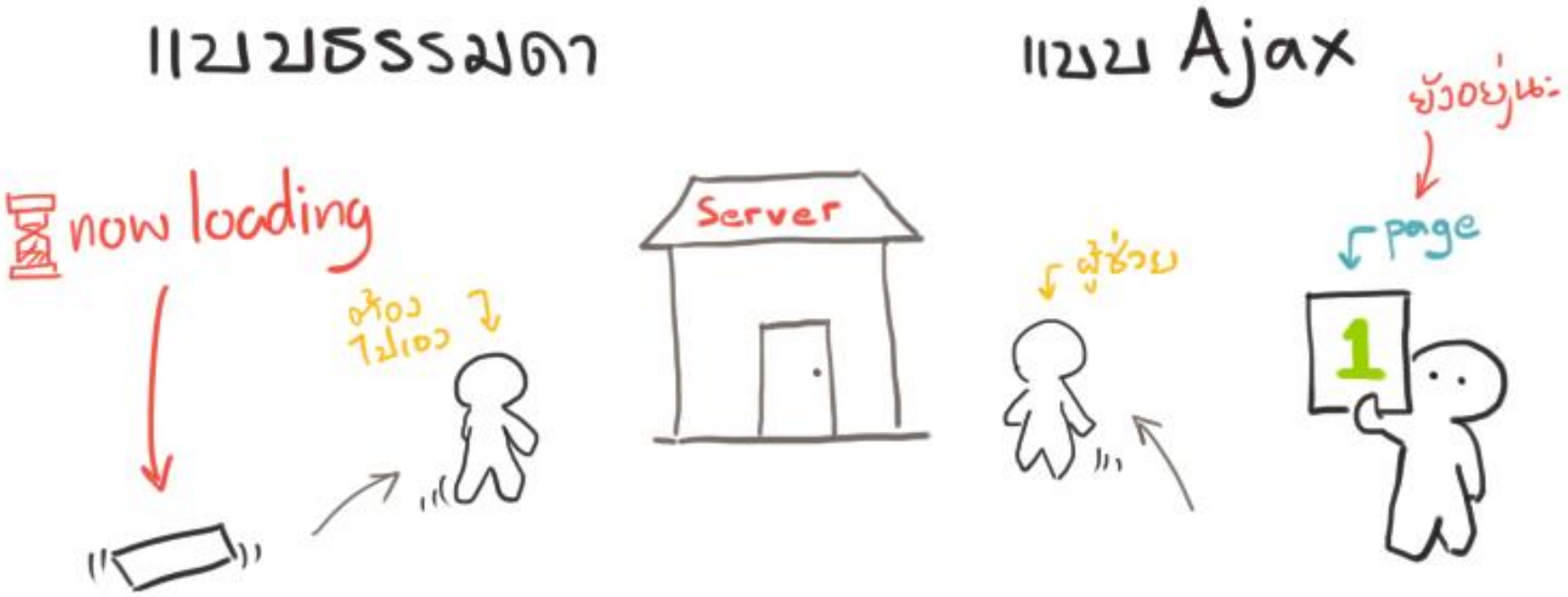


AJAX

AJAX

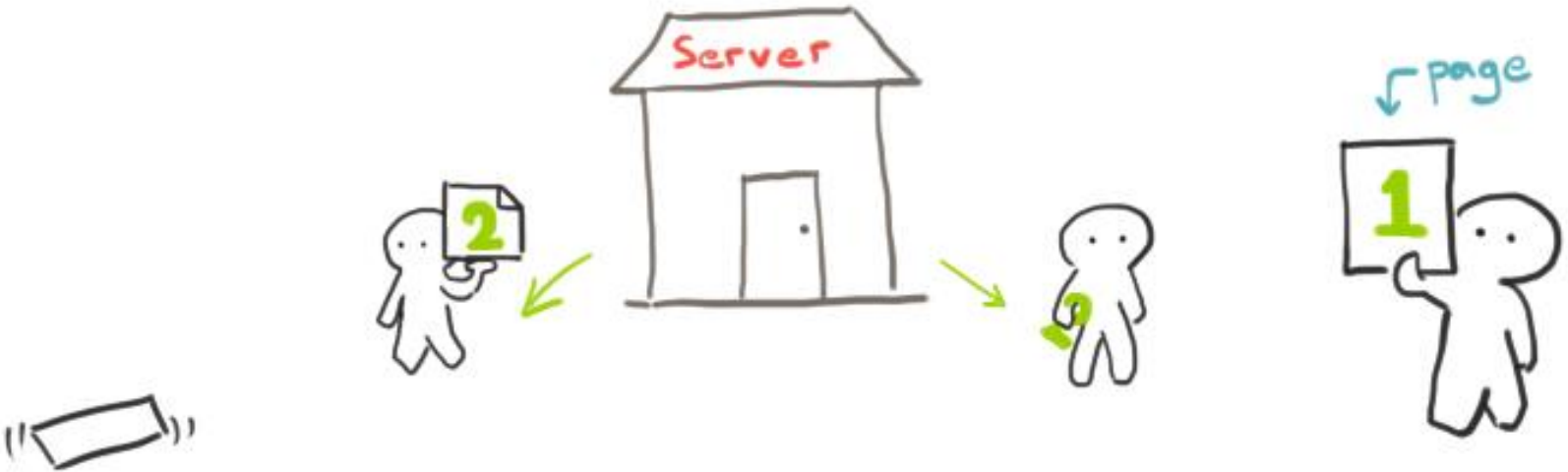
AJAX ย่อมาจาก Asynchronous JavaScript And XML เป็นเทคนิคการสื่อสารระหว่าง Browser กับ Web Server โดยอาศัยพื้นฐานการทำงานอยู่บน JavaScript และ HTTP Requests ทำให้สามารถโหลดข้อมูลจากทางฝั่ง Server เพื่ออัปเดตเนื้อหาเพียงบางส่วนภายในเพจโดยไม่ต้องรีเฟรชหน้าใหม่




แบบธรรมดา
now loading
ห้องไปเลย
Server
แบบ Ajax
ยังงี้ไงล่ะ
page
ผู้ช่วย
1

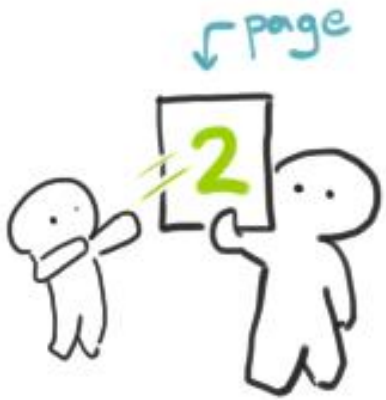
แบบธรรมดา

แบบ Ajax



แบบธรรมดา

แบบ Ajax



jQuery :: AJAX

กำหนดอาร์กิวเมนต์เท่าที่จำเป็นเท่านั้น ไม่สามารถควบคุมกระบวนการต่างๆ

```
$('selector').load('URL', data, callback)

$.get('URL', data, function(result){…})

$.getJson('URL', data, function(result){…})

$.post('URL', data, function(result){…}) **
```

** data เป็น JSON Format

jQuery :: AJAX Advanced

สามารถควบคุมกระบวนการต่างๆ และกำหนดลักษณะปลีกย่อยได้ มี Property ที่น่าสนใจ เช่น

url	กำหนดเพจปลายทาง
type (optional)	ชนิดของ Request ในการส่งข้อมูล GET หรือ POST
datatype (optional)	ชนิดข้อมูลผลลัพธ์ที่จะถูกส่งกลับมา
data (optional)	ข้อมูลที่จะส่งไปเพจปลายทาง
timeout (optional)	เวลาในการรอผลตอบสนอง (มิลลิวินาที) เกินกว่านั้นจะถือว่าผิดพลาด
beforesent (optional)	ฟังก์ชั่นที่ถูกเรียกใช้งานก่อนส่ง Request
success (optional)	ฟังก์ชั่นที่ถูกเรียกทำงาน เมื่อผลลัพธ์ถูกส่งกลับมาได้สำเร็จ
complete (optional)	ฟังก์ชั่นที่ถูกเรียกทำงานเมื่อกระบวนการ AJAX สิ้นสุดไม่ว่าสำเร็จหรือไม่สำเร็จ
error (optional)	ฟังก์ชั่นที่ถูกเรียกทำงาน เมื่อเกิดข้อผิดพลาด

HttpRequest :: State

State คือ สถานะบอกให้รู้ว่า HttpRequest ทำงานไปถึงไหนบ้าง

0 :: request not initialized (คำขอยังไม่ถูกสั่ง)

1 :: server connection established (เริ่มเชื่อมต่อ Server)

2 :: request received (คำขอได้รับแล้ว)

3 :: processing request (Server กำลังทำงาน)

4 :: request finished and response is ready (server ทำงานเสร็จแล้ว และตอบกลับ)

HttpRequest :: Status

Status คือ รหัสสถานภาพในการตอบรับของ HttpRequest จากเครื่องให้บริการ แบ่งเป็นหมวดได้

1xx :: ข้อมูลทั่วไป

2xx :: การร้องขอสำเร็จ

3xx :: การเปลี่ยนทาง

4xx :: ความผิดพลาดของเครื่องลูกข่าย

5xx :: ความผิดพลาดจากเครื่องแม่ข่าย

HttpRequest :: Status

200 OK :: การร้องขอสำเร็จ

201 Created :: มีผลลัพธ์ที่เป็นการสร้างขึ้นใหม่

302 Found :: การเปลี่ยนทางบนหน้าเว็บมากที่สุด

400 Bad Request :: มีความผิดพลาดทางไวยากรณ์ หรือไม่สามารถทำตามคำร้องขอ

401 Unauthorized :: จำเป็นต้องการพิสูจน์ตัวตนก่อน

403 Forbidden :: คำร้องถูกต้อง แต่เครื่องให้บริการปฏิเสธให้บริการ

404 Not Found :: คำร้องขอไม่พบบนเครื่องให้บริการ

HttpRequest :: Status

405 Method Not Allowed :: เครื่องให้บริการไม่รองรับคำสั่งที่ร้องขอ

500 Internal Server Error :: ความผิดพลาดทั่วไป ไม่มีข้อความเฉพาะ

502 Bad Gateway :: เครื่องให้บริการทำหน้าที่เป็น เกตเวย์หรือพร็อกซี และได้รับข้อความตอบรับผิดพลาดจากเครื่องให้บริการเบื้องหลัง

503 Service Unavailable :: เครื่องให้บริการยังไม่ให้บริการ อันเนื่องจากการใช้งานทรัพยากรใช้งานเกินพิกัด หรืออยู่ระหว่างการบำรุงรักษา