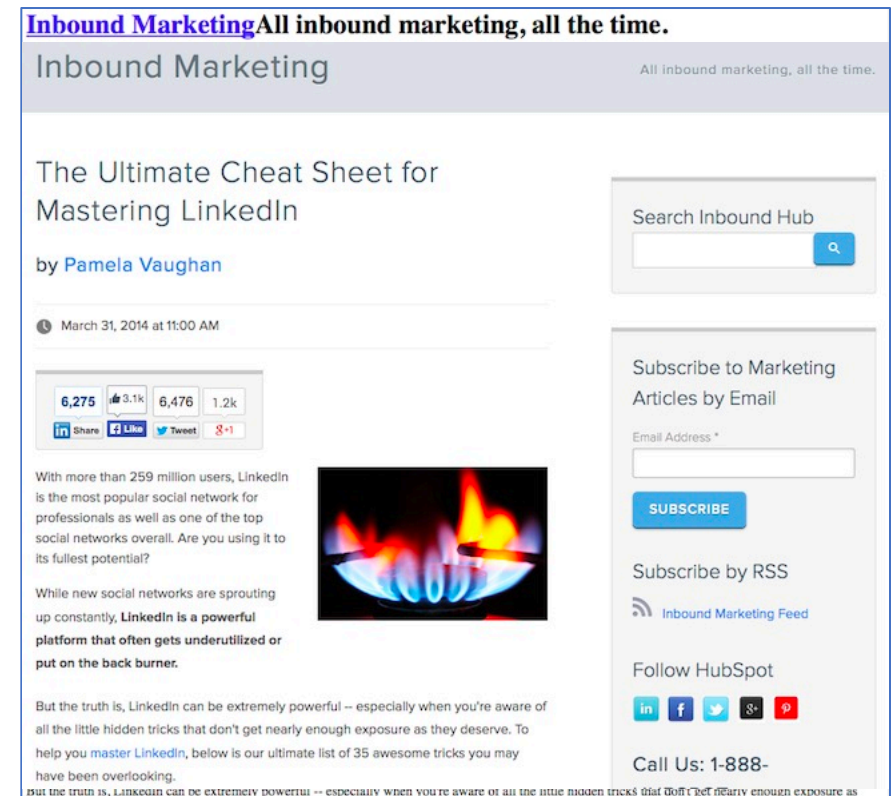


Basic Modern JavaScript

ส่วนประกอบของ Website

- HTML ส่วนแสดงผล โดยการใช้ Tag ต่าง ๆ เช่น <input>, <a>, <head>, <body>
- CSS ส่วนปรับปรุงการแสดงผลให้มีความสวยงาม
- JavaScript ส่วน Logic ของหน้า website ใช้ในการคำนวณค่าต่าง ๆ เพื่อแสดงผล



“Modern” JavaScript

- JavaScript ถูกพัฒนามาตั้งแต่สมัย netscape (199x)
- มาตรฐานที่ใช้การ define JavaScript ชื่อ ECMAScript (ES)
- เมื่อกล่าวถึง Modern JavaScript มักจะหมายถึง JavaScript ที่มี version ตั้งแต่ ES6 เป็นต้นมา (Released on 2015)
- ปัจจุบัน JavaScript ถูกใช้ทั้งภายใน Browser และการเขียนโปรแกรมโดยทั่วไป เช่น Node.js
 - JavaScript ที่ทำงานใน Browser จะมีข้อจำกัดบางอย่าง เช่น การเข้าถึงไฟล์, การสร้าง HTTP Request ไปยัง website ภายนอก เป็นต้น

Code Editor

- ตัวอย่างโปรแกรม Editor ที่สามารถเขียนโปรแกรมภาษา Java Script ได้ เช่น
 - Visual Studio Code
 - Sublime Text



JavaScript Runtime

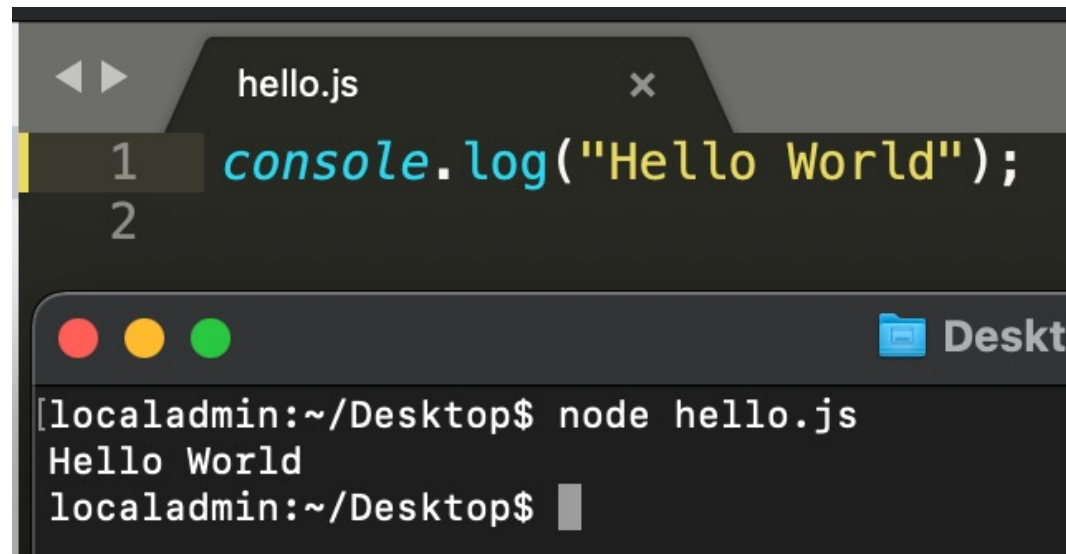
- โปรแกรมที่เขียนด้วยภาษา JavaScript สามารถรันได้โดยผ่านทาง
 - Web Browser
 - Node.js คือโปรแกรมสำหรับรันภาษา JavaScript

Hello World

- พิมพ์คำสั่งต่อไปนี้ลงไปใน Text Editor

```
console.log("Hello World");
```

- รันโปรแกรมด้วยคำสั่ง `node <filename>`



The screenshot shows a code editor window with a tab labeled 'hello.js'. The code inside is `console.log("Hello World");` on line 1. Below the code editor is a terminal window. The terminal shows the command `node hello.js` being executed, which results in the output `Hello World`. The terminal prompt is `localadmin:~/Desktop$`.

Code Structure (1)

- Statement คือ ส่วนของ code ที่พิมพ์ลงไปเพื่อก่อให้เกิด action ใด ๆ ขึ้น เช่น `console.log()` ในหัวข้อก่อนหน้านี้
- Semicolons “;” เป็นอักขระพิเศษสำหรับปิดท้ายแต่ละ statement
 - การเขียนโปรแกรมโดยทั่วไป 1 บรรทัดจะมีเพียง 1 statement ทำให้ semicolon ถูกใช้ปิดบรรทัด
 - JavaScript จะมีการใส่ semicolon ให้อัตโนมัติ แต่ในบางกรณีที่ JavaScript ไม่ใส่ให้อาจเกิด Error ได้ (มือใหม่ควรใส่ semicolon ทุกครั้งที่จบ statement)

Code Structure (2)

- Comment มี 2 แบบคือ
 - `//` สำหรับ comment บรรทัดเดียว
 - `/* */` สำหรับ block comment

```
> // Single Line Comment
< undefined

> /*
  Multiple
  lines
  comment
  */
< undefined
```


การประกาศตัวแปร

- ตัวแปรคือภาชนะสำหรับใส่ค่าต่าง ๆ เพื่อเก็บไว้ใช้ในอนาคต
- JavaScript ประกาศตัวแปรโดยผ่าน keyword ชื่อ let และ const
 - let เป็นการประกาศตัวแปรธรรมดาสามารถเปลี่ยนค่าไปได้เรื่อย ๆ
 - const เป็นการประกาศตัวแปรค่าคงที่

```
> let a = 10;  
< undefined  
> a = 20; alert(a);|
```

```
> const B = 20;  
< undefined  
> B = 50; alert(B)  
✖ ▶ Uncaught TypeError: Assignment to constant variable.  
   at <anonymous>:1:3  
>
```

การประกาศตัวแปร (2)

- Keyword ที่ไม่ได้กล่าวถึงคือ var ซึ่งเป็นการประกาศตัวแปรใน JavaScript แบบเก่า
 - พยายามหลีกเลี่ยงการใช้งาน var เพราะ ไม่สามารถควบคุม scope ของตัวแปรได้

```
> var x = 10;|
```

- การตั้งชื่อตัวแปร ให้ใช้ camelCase เช่น firstName, myBirthDay, pageLoadTime

ชนิดของตัวแปร

- Number สามารถเป็นได้ทั้งเลขจำนวนเต็ม (Integer) และเลขทศนิยม (Float)

```
> let x = 10;  
x = 1.5;  
let y = 20;  
let z = x + y;  
alert(z);
```

- String คือตัวอักษร โดยการประกาศตัวแปรสามารถใช้ Double Quote, Single Quote, และ Backticks ได้

```
> let x = "Hello";  
let y = 'World';  
let z = `Say, ${x} ${y}`;  
alert(z);|
```

ชนิดของตัวแปร (2)

- Boolean แทนค่าด้วยตัวอักษร true, false หรือ เกิดจากผลลัพธ์ของการเปรียบเทียบ

```
> let nameChecked = true;  
   let phoneChecked = false;  
   alert(`Name Checked : ${nameChecked} Phone Checked : ${phoneChecked}`);  
   alert(5 > 10);|
```

- null คือการประกาศตัวแปรที่ไม่มีค่าอะไรอยู่ภายใน (เปรียบเทียบได้กับภาชนะว่างเปล่า)
- undefined คือค่านี้ยังไม่ได้ถูกกำหนดค่า หรือไม่มีตัวแปรนี้อยู่ในระบบ

ชนิดของตัวแปร (3)

- ตัวแปรที่กล่าวมาทั้งหมดก่อนหน้านี้เป็นตัวแปรประเภท Primitive เนื่องจากค่าของตัวแปรมีการเก็บค่าเพียงอย่างเดียว
- ตัวแปรอีกประเภทคือ Object ซึ่งจะพูดถึงในหัวข้อ Object Oriented Programming ต่อไป

- Keyword “typeof” ใช้สำหรับบอกชนิดของตัวแปร

```
> let a = 10;
    typeof(a);
< "number"

> let b = '10';
    typeof(b);
< "string"

> let c = true;
    typeof(c);
< "boolean"
```

การเปลี่ยนค่าชนิดของตัวแปร

- ตัวแปรสามารถเปลี่ยนชนิดไปมาได้ โดยการเรียก function ต่อไปนี้
 - String(value) เปลี่ยนค่าตัวแปรเป็นชนิด String
 - Number(value) เปลี่ยนค่าตัวแปรเป็นชนิด Number
 - Boolean(value) เปลี่ยนค่าตัวแปรเป็นชนิด Boolean

```
> console.log(String(200))  
console.log(Number("100"))  
console.log(Boolean("true"))
```

200

100

true

การเปรียบเทียบค่าตัวแปร

- สามารถใช้เครื่องหมาย ต่อไปนี้ในการเปรียบเทียบ
 - มากกว่า $>$ และน้อยกว่า $<$
 - มากกว่าหรือเท่ากับ $>=$ และน้อยกว่าหรือเท่ากับ $<=$
 - เท่ากับ $==$ (ระวัง อย่าสับสนกับการประกาศตัวแปรที่ใช้ $=$ ตัวเดียว)
 - ไม่เท่ากับ $!=$
- เครื่องหมาย Strict Equality $===$ หรือ $!==$ เป็นการเปรียบเทียบทั้งค่าและชนิดของตัวแปร

```
> console.log(0 == false);  
console.log(0 === false);  
  
true  
false
```

Condition Statement

- JavaScript มีการใช้งาน if และ switch ในการสร้าง condition statement

```
> let accessAllowed;  
let age = 10;  
if (age > 18) {  
    accessAllowed = true;  
}  
else {  
    accessAllowed = false;  
}  
console.log("Allow access ?", accessAllowed);
```

- ประโยค if else สามารถเขียนย่อให้อยู่ในบรรทัดเดียวได้ (Ternary Operation)

```
> let age = 20;  
let accessAllowed = (age > 18) ? true : false;  
console.log("Allow access ?", accessAllowed);  
Allow access ? true
```


Condition Statement (2)

- กรณีมี condition มากกว่า 2 สามารถใช้ keyword "else if" ในการเพิ่ม condition

```
> let year = 2019;  
  if( year < 2019 ) {  
    console.log("Too old");  
  }  
  else if( year > 2019 ) {  
    console.log("Too new");  
  }  
  else {  
    console.log("It's this year");  
  }
```

Condition Statement (3)

- สามารถใช้ keyword ต่อไปนี้ เพื่อเชื่อม logic ได้
 - AND (&&)
 - OR (||)


```
if (true && false) {  
    console.log("1");  
} else {  
    console.log("2");  
}
```

```
if (true || false) {  
    console.log("1");  
} else {  
    console.log("2");  
}
```

Function

- Function คือการแบ่งส่วนของ code ออกมา เพื่อลดความซับซ้อนของโปรแกรมลง และสามารถเรียกใช้งานซ้ำได้

```
1 function name (parameters) {  
2     //body.....|  
3 }
```



```
adder.js  
1 function adder(a, b) {  
2     console.log(a + b);  
3 }  
4  
5 adder(7, 9);
```

- การตั้งชื่อ Function ควรสื่อความหมายที่เข้าใจว่า Function นี้ทำหน้าที่อะไร
- ชื่อ Function อยู่ในรูปแบบ camelCase

Function (2) – Variable Scope

- ตัวแปรที่ถูกประกาศภายใน function จะไม่สามารถเข้าถึงได้จากภายนอก function
 - ตัวแปรประเภทนี้เรียกว่า local variable

```
1  function showMessage() {  
2      let message = "Hello World";  
3      console.log(message);  
4  }  
5  
6  showMessage();  
7  console.log(message);  
8
```

- ตัวแปรที่ประกาศไว้นอก function จะเรียกว่า global variable

```
1  let name = "John";  
2  
3  function showMessage() {  
4      console.log(`Hello ${name}`);  
5  }  
6  
7  showMessage();  
8  
9  console.log(name);
```

Function (3) - Parameters

- ตัวแปรสามารถส่งผ่านไปได้ใน function ได้ เรียกว่า parameter

```
1 function showMessage(message) {  
2     console.log(message);  
3 }  
4  
5 showMessage("Hello");
```

- JavaScript สามารถกำหนดค่า default ให้กับ parameter ได้
 - ทำให้เวลาเรียก function ไม่จำเป็นต้องส่ง parameter ลงไปได้

```
1 function showMessage(message, toUpper=false) {  
2     toUpper ? console.log(message.toUpperCase()) : console.log(message);  
3 }  
4  
5 showMessage("hello 1");  
6 showMessage("hello 2", true);  
7 showMessage("hello 3", toUpper=true);
```

Function (4) – Return value

- Function สามารถส่งค่ากลับมาได้ด้วย keyword "return"

```
1  function multiply(a, b) {  
2      return a * b;  
3  }  
4  
5  console.log(multiply(4, 3));
```



Exercise – google Function

- สร้าง function ชื่อ google โดยรับ parameter เป็นเลข 1 ตัว
- หากตัวเลขสามารถหารด้วย 3 ลงตัวแสดงคำว่า goo
- หากตัวเลขสามารถหารด้วย 5 ลงตัวแสดงคำว่า gle
- หากตัวเลขสามารถหารด้วย 5 และ 3 ลงตัวแสดงคำว่า google
- หากตัวเลขไม่สามารถหารด้วย 5 และ 3 ได้ลงตัวแสดงตัวเลขตามปกติ

- ใช้ operand “%” สำหรับการหารลงตัว

Function (5) – Function Expression

- การประกาศ function ก่อนหน้านี้เรียกว่า function declaration แต่ในภาษา JavaScript สามารถประกาศ function แล้วเก็บในตัวแปรได้ ซึ่งวิธีนี้เรียกว่า Function Expression

```
1 function multiply(a, b) {  
2     return a * b;  
3 }  
4  
5 console.log(multiply(4, 3));
```

Function Declaration

```
1 let multiply = function (a, b) {  
2     return a * b;  
3 }  
4  
5 console.log(multiply(4, 3));
```

Function Expression

Function (6) – Callback Function

- Callback function คือการส่ง function expression มาเป็น parameter ให้กับ function อีกหนึ่ง
 - มักใช้กับการตอบกลับ หลังการทำงานเสร็จสิ้น เช่น การตอบกลับหลังจาก load data จาก website เสร็จสิ้น เป็นต้น

```
1  function loadData(complete, error) {  
2      console.log("Loading Data");  
3      Math.random() >= 0.5 ? complete() : error();  
4  }  
5  
6  let success = function() {  
7      console.log("Success!");  
8  }  
9  
10 let failure = function() {  
11     console.log("Failed!");  
12 }  
13  
14 loadData(success, failure);
```

Function (7) – Arrow Function

- วิธีการประกาศ Function Expression ที่ใช้กันในปัจจุบัน มักใช้ Arrow Function

```
1 let showMessage = function() {  
2   console.log("Hello World");  
3 }  
4  
5 showMessage();
```

```
1 let showMessage = () => {  
2   console.log("Hello World");  
3 }  
4  
5 showMessage();
```

```
1 let sum = function(a, b) {  
2   return a + b;  
3 }  
4  
5 console.log(sum(7, 5));
```

```
1 let sum = (a, b) => {  
2   return a + b;  
3 }  
4  
5 console.log(sum(7, 5));
```

```
1 let power = a => a * a;  
2 console.log(pow(3));
```

```
1 let minus = function(a, b) {  
2   return a - b;  
3 }  
4  
5 console.log(sum(7, 5));
```

```
1 let minus = (a, b) => a - b;  
2  
3 console.log(minus(7, 5));
```

Loop

- JavaScript สามารถสร้าง loop ได้ด้วย keyword 2 ตัวคือ **for** และ **while**

```
1  for(begin; condition; step;) {  
2      //..... Body .....  
3  }  
4  
5  for(let i=0; i<3; i++) {  
6      console.log(i);  
7  }
```

```
1  while(condition) {  
2      //..... Body .....  
3  }  
4  
5  let i = 0;  
6  while(i < 3) {  
7      console.log(i);  
8      i++;  
9  }
```

Loop (2)

- break – ใช้สำหรับการหยุด loop ก่อน condition จะเสร็จสิ้น
- continue – ใช้สำหรับข้ามการทำงานของ code ที่อยู่ด้านล่าง continue แล้วเริ่มทำงาน loop รอบถัดไป

```
1  for(let i=0; i<3; i++) {  
2      if(i == 1) {  
3          break;  
4      }  
5      else {  
6          console.log(i);  
7      }  
8  }
```

```
1  for(let i=0; i<3; i++) {  
2      if(i == 1) {  
3          continue;  
4      }  
5      else {  
6          console.log(i);  
7      }  
8  }
```



Exercise – Prime Number

- เขียนโปรแกรมวน loop เลขตั้งแต่ 2 ถึง 20
- เมื่อเจอตัวเลขจำนวนเฉพาะให้พิมพ์ตัวเลขนั้นออกมา
- จำนวนเฉพาะคือจำนวนที่ไม่สามารถหารด้วยตัวเลขก่อนหน้านี้ได้ลงตัว ยกเว้น 1 และตัวเอง

TIP 1 : หารไม่ลงตัวคือ % ไม่เท่ากับ 0

TIP 2 : ควรเขียนฟังก์ชันชื่อ isPrime(num) เพื่อ return true หรือ false เพื่อตรวจสอบว่าเป็นเลขจำนวนเฉพาะหรือไม่?