JavaScript (ทฤษฎี)

JavaScript

- Interpreted Language – อ่านและ แปล รหัสคำสั่ง ทำงานต่อจากบน
- Single Threaded, do one operation at one time
- Dynamically and weakly typed language – ไม่ต้องประกาศ Type
- Support Object Oriented Programming (Prototyped-based)

เปลี่ยนค่าเองได้

เข้าใจ oop ได้

Asynchronous vs. Synchronous Programming

Synchronous (ทำก่อน)

Asynchonous (ทำหลัง)

```
1  // Asych vs Sych
2  console.log("start");
3  setTimeout(() => console.log("1"), 0);
4  setTimeout(() => console.log("2"), 2000);
5  console.log("3");
6  console.log("end");
7
```

อะไรมาก่อนกัน ?

Primitive Type

- **Number**
  - integer between -2^{53} to 2^{53}
  - floating-point numbers (64 bits floating point format defined by IEEE754 standard)
- **String** - support Unicode characters, use UTF-16 encoding, zero-based indexing
- **Boolean –** true or false
- **BigInt** - for very large integers, created by appending *n* to the end of an integer literal
- **Symbol** - a built-in object, non-string property names, guaranteed to be unique
- **Null** - a missing object
- **Undefined** – a variable that has not been assigned a value

~~**Objects** (mutable) - Collections of properties~~

Immutable Primitive (ค่าที่ไม่เปลี่ยนแปลง)

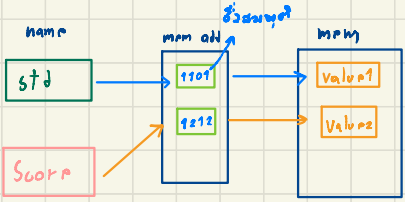
- แยกใครแยกมัน
- มีค่าเป็น Value จริงๆ

```
1  let str = 'hello'
2  str.toUpperCase()
3  console.log(str);
```

Mutable Object (obj)

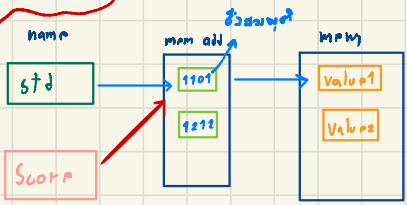
obj : {}
Array: [ ]

```
let std = { firstname: 'Somchai', lastname: 'Rakdee' }
std.lastname = 'Deejai'
console.log(std) //{ firstname: 'Somchai', lastname: 'Deejai' }
let scores = [10, 25, 30, 50]
scores[0] = 5
scores[scores.length - 1] = 100
console.log(scores) //[ 5, 25, 30, 100 ]
```



- ไม่ได้เก็บค่านั้นจริงๆแต่เก็บเป็นตัว Memory address
- ตัวที่มี Memory Address เหมือนกัน เมื่อเปลี่ยน Value ตัวทั้ง 2 ก็เปลี่ยน

```
std = score;
```



* ถึงนั้นจะไม่ใช้การ copy Value แต่เป็นการ copy mem Address

Identifier

ใช้ใน : constants, variables, Properties, functions, classes,
ควร : ตั้งชื่อเริ่มด้วย ตัวอักษร เท่านั้น ยกเว้น (_) และ $ สามารถได้
จำว่า : เป็น Case Sensitive

Declarations and Types

let : หลังประกาศ สามารถ เปลี่ยนค่า และ Type ของตัวแปรได้
const : หลังประกาศไม่สามารถเปลี่ยนค่าได้อีกต่อไป
Var : เหมือน let แต่ไม่ใช้กันแล้วเพราะ ไม่สร้างมาเพื่อ block-scope

Function Scope

- Variables (`let`, `const`, `var`) declared within a JavaScript function, become **local to the function as local variables**
- They can only be accessed from within the function
- Local variables are created when a function starts and deleted when the function is completed.

```
// code here can NOT use all studentNames
function myFunction() {
  let studentName1 = 'Somchai'
  const studentName2 = 'Somsak'
  var studentName3 = 'Somsri'
  console.log(studentName1)
  console.log(studentName2)
  console.log(studentName3)
}
myFunction()
// code here can NOT use all studentNames
```

Global Scope

- A variable declared outside a function or a block, becomes **global**.
- Global variables can be accessed from anywhere in a JavaScript program.

```
let num1 = 10
const num2 = 20
var num3 = 30

function testGlobalScope() {
  console.log(num1)
  console.log(num2)
  console.log(num3)
  num1++
  // num2++ //constant cannot change value
  num3++
}

testGlobalScope()
console.log(num1)
console.log(num2)
console.log(num3)
```

ความสามารถของ Dynamic Type

JavaScript implicit type conversions

Value	to String	to Number	to Boolean
`undefined`	`"undefined"`	`NaN`	`false`
`null`	`"null"`	`0`	`false`
`true`	`"true"`	`1`	
`false`	`"false"`	`0`	
`""` (empty string)		`0`	`false`
`"1.2"` (nonempty, numeric)		`1.2`	`true`
`"one"` (nonempty, non-numeric)		`NaN`	`true`
`0`	`"0"`		`false`
`-0`	`"0"`		`false`

Value	to String	to Number	to Boolean
`1` (finite, non-zero)	`"1"`		`true`
`Infinity`	`"Infinity"`		`true`
`-Infinity`	`"-Infinity"`		`true`
`NaN`	`"NaN"`		`false`
`[]` (empty array)	`""`	`0`	`true`
`[9]` (one numeric element)	`"9"`	`9`	`true`
`['a']` (any other array)	*use join() method*	`NaN`	`true`

```
// Implicit Type Conversions
const conv1 = 10 + ' rooms' //10 rooms
const conv2 = '4' * 5  //20
const conv3 = 'a' - 2  //NAN
const conv4 = !'Hello' //false
```

Source codes: //types-variables/conversion

Explicit Conversions

- Although JavaScript performs many type conversions automatically, you may sometimes need to perform an explicit conversion, or you may prefer to make the conversions explicit to keep your code clearer.
- ~~The simplest way to perform an explicit type conversion~~ is to use the `Boolean()`, `Number()`, and `String()` functions:

แปลง Type

ค่า log จะออกมาเป็นแบบไหน

```
const conv5 = "5" + 5 + 5
const conv6 = 5 + "5" + 5
const conv7 = 5 + 5 + "5"

console.log(conv5);
console.log(conv6);
console.log(conv7);
```

```
console.log(false + true);
```

```
const conv8 = 5 + "5" - "5"
console.log(conv8);
```

JavaScript Operators

Operator precedence and associativity specify the order in which operations are performed in a complex expression.

Precedence (High)

- Optional Chaining `?.` — 1
- Increment and Decrement `++ --` — 2
- Invert Boolean value `!` — 3
- Type of operand `typeof`
- Arithmetic operators
  - `* / %` — 4
  - `+ -` — 5
- Relational operators `< <= > >=` — 6
- Equality operators
  - `== !=` (non strict equality)
  - `=== !==` (strict equality) — 7
- Logical operators `&&` — 8
- Logical operators `||` — 9
- Nullish Coalescing `??`
- Assignment operators `+= -= *= /= %=` — 10
- Conditional Operator `?:` — 11

(Low)

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

forEach

- **`forEach()`** method iterates through an array, invoking a function you specify for each element.

```
let data = [1, 2, 3, 4, 5, 6, 7, 8, 9]
let sum = 0
data.forEach((num) => (sum += num))
console.log(`sum=${sum}`) //sum=45
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach

รวม คำสั่ง ที่อาจารย์สอน

1) ประกาศค่า
   - const
   - let
   - var
2) loop
   - for(i=0; i<..., i++){}
   - while(codition)
3) Array[]
   - length
   - arry[index]
   - arry.of( )
   - 3.1) การ Spared
     - const addone = [...array,1]
   - 3.2) การ distructuring
     - const [first,second,...keep] =[1,2,3,4,5,6]