Software Park Thailand

</Code Camp>

JS

DataTypes & Variable

Pavit Pimchanagul

หัวข้อ

- DataTypes
  - Values,Types
  - Number *(with arithmatic operator)*
  - String *(with concatenate)*
  - Boolean *(with logical operator)*
  - Null, Undefined
- Expression
- Prototype Based
- Variable (Binding)
  - Declaration, Initialization
  - Operator with variable
  - Scope, Redeclaration, Hoisting

Software Park Thailand
</Code Camp>

DataTypes

Values,Types

Software Park Thailand
</Code Camp>

Environment

Functions

Control Flow

Loop

Branch

Variable

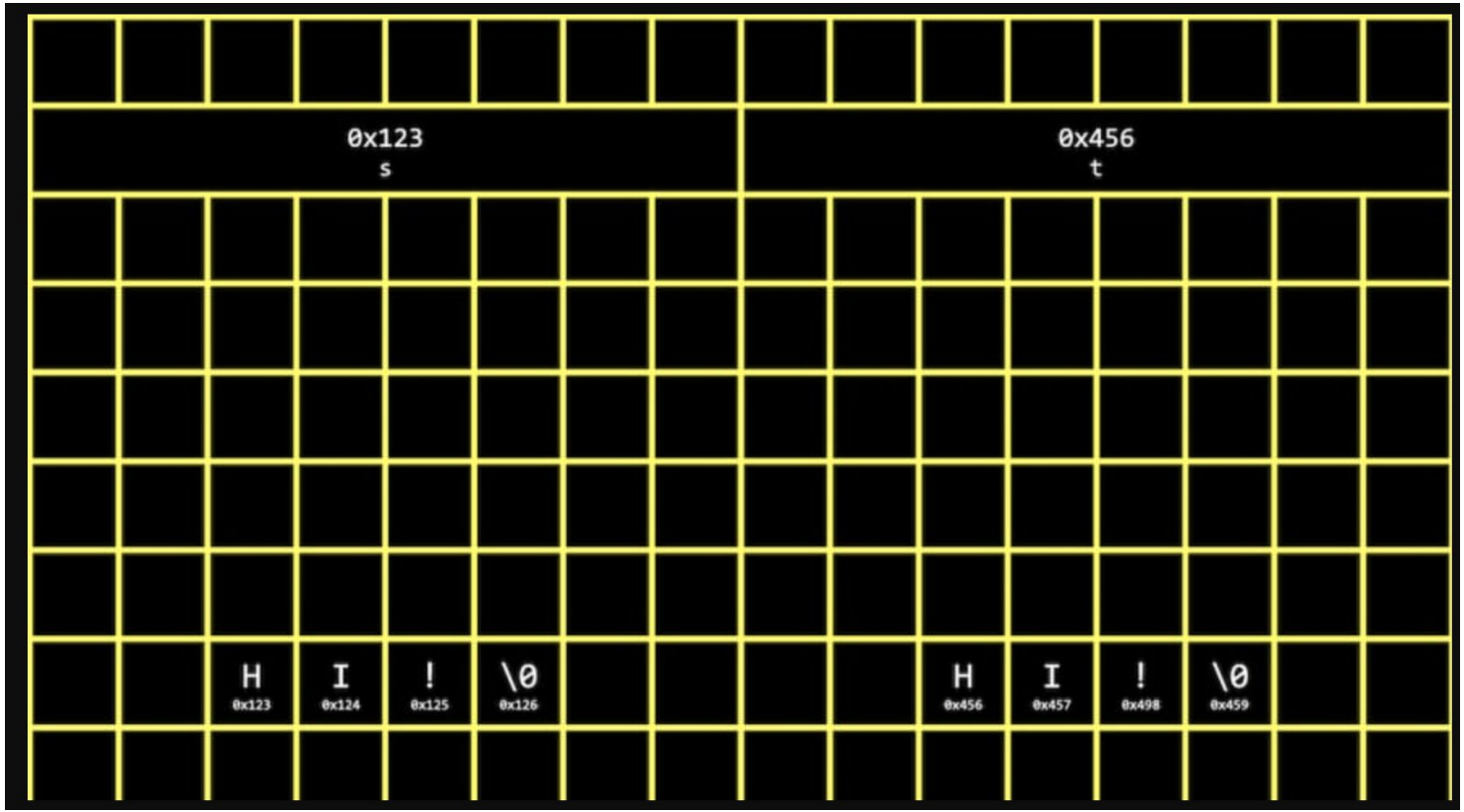
Statements

DataType

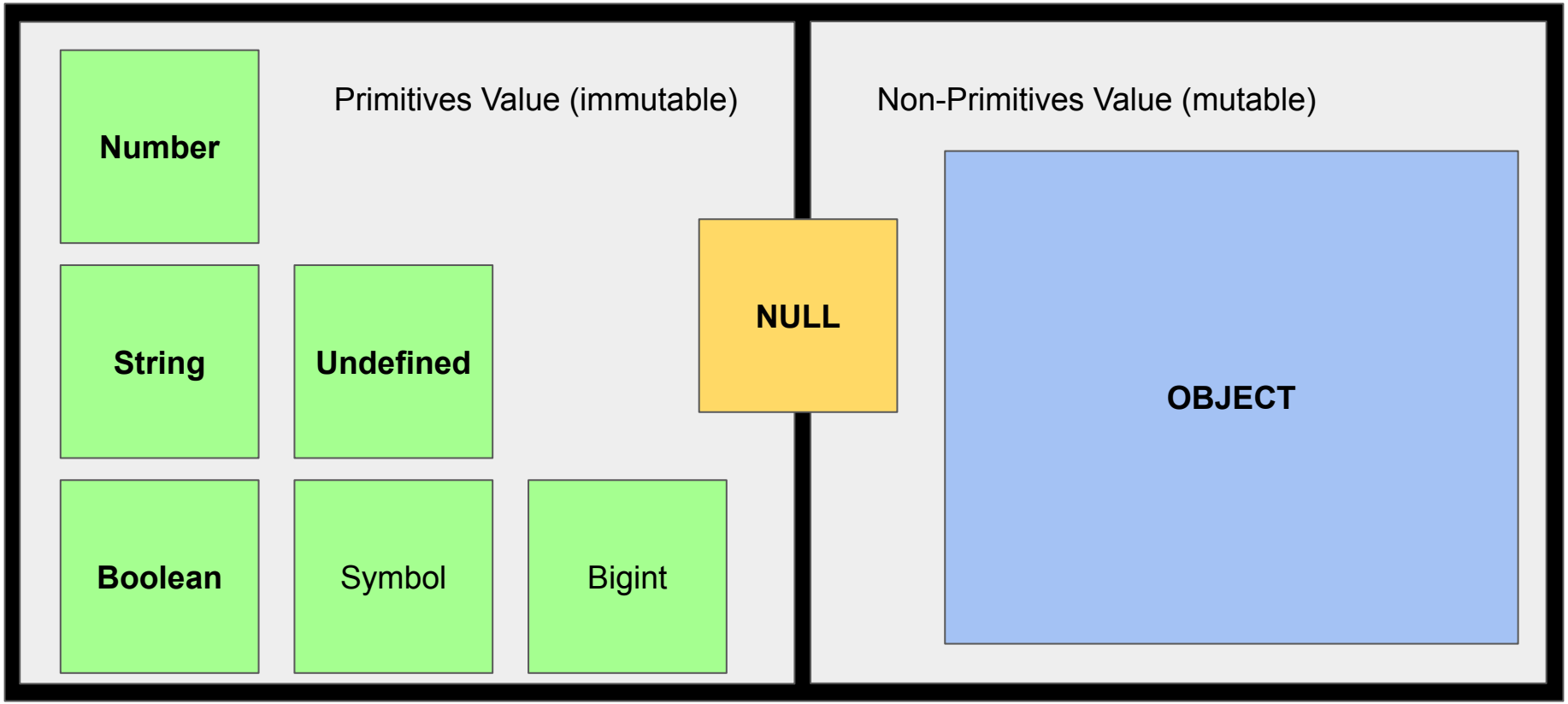
Expression

Comment

Values


0x123
s
0x456
t
H
0x123
I
0x124
!
0x125
\0
0x126
H
0x456
I
0x457
!
0x498
\0
0x459

Values


Primitives Value (immutable)
Number
String
Undefined
Boolean
Symbol
Bigint
NULL
Non-Primitives Value (mutable)
OBJECT

Types

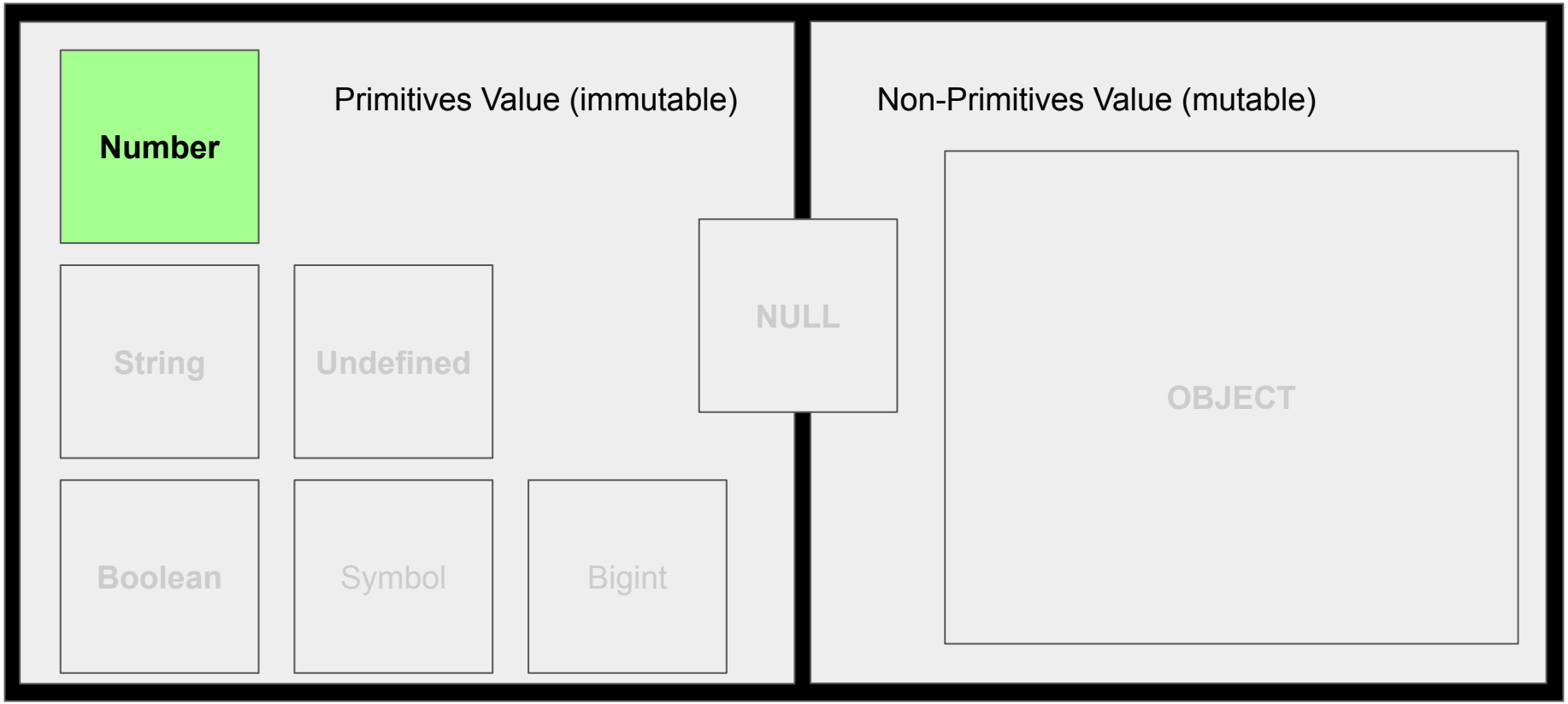
Basic Data Types in JavaScript

- Number คือข้อมูลประเภทตัวเลข
- String คือข้อมูลประเภทข้อความ
- Boolean คือข้อมูลที่มีแค่ true (จริง) และ false (เท็จ)
- Null คือค่าว่างหรือไม่รู้ค่า (nothing, empty, unknown value)
- Undefined คือข้อมูลที่ยังไม่ได้กำหนดค่า (not assigned value)
- BigInt คือข้อมูลประเภทตัวเลขที่มีค่ามากๆ นอกช่วง $-(2^{53}-1)$ ถึง $(2^{53}-1)$
- Symbol คือข้อมูลที่เป็น unique identifiers
- Object คือข้อมูลที่มีโครงสร้างซับซ้อน

Software Park Thailand
</Code Camp>

Number

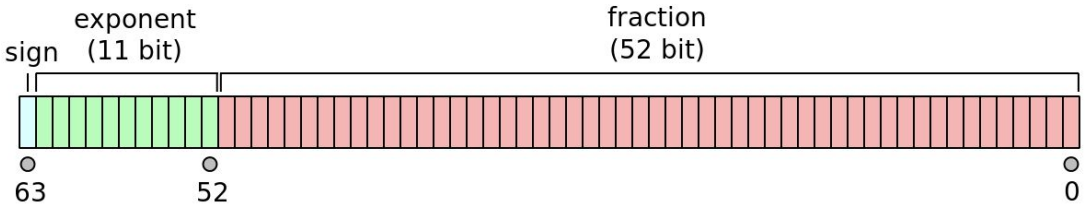
Number


Primitives Value (immutable)
Non-Primitives Value (mutable)
Number
String
Undefined
NULL
Boolean
Symbol
Bigint
OBJECT

Number (In Deep)

- ข้อมูลพื้นฐาน (primitive) ประเภทตัวเลข
- ใช้หน่วยความจำ 64 bits (8bytes)ในการเก็บค่า
- ตามมาตรฐาน IEEE 754 double-precision binary floating-point format: binary64

$$(-1)^{\text{sign}} \left(1 + \sum_{i=1}^{52} b_{52-i} 2^{-i}\right) \times 2^{e-1023}$$

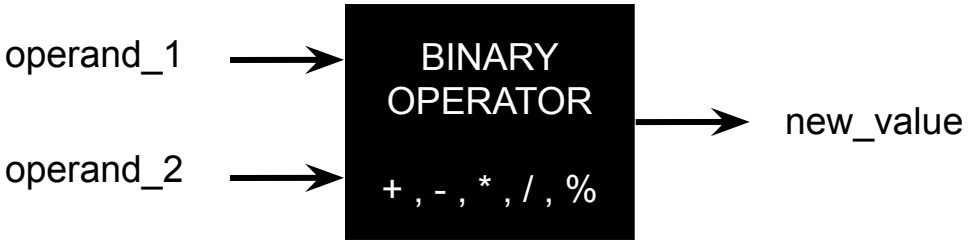


Arithmatic Operator

ตัวดำเนินการทางคณิตศาสตร์

- เราสามารถเอาค่าตัวเลขมา บวก,ลบ,คูณ,หาร, หาเศษเหลือได้
- +, - , *, / , % เป็น *binary operator* docs

```
2 + 3  // 5
21 - 6 // 15
10 * 3 // 30
16 / 3 // 5.333
16 % 3 // 1
-16 % 3 // -1
```



Arithmatic Operator

- หากมีการดำเนินการพร้อมกันหลายตัว ผลลัพธ์จะขึ้นอยู่กับ ***precedence*** ของ operator

  เช่น การคูณ,หาร,หาเศษเหลือ มี precedence สูงกว่าการบวก,ลบ [docs]

- in short : ***PEMDAS*** (Parenthesis, Exponent, Multiply,Divide,Addition,Subtraction)

```
 15 + 4 * 5 //
(15 + 4) * 5 //
 50 - 12 / 2 + 6 * 4 //  50 - 6 + 24
```

Arithmatic Operator

- บวก ใช้เครื่องหมาย +
- ลบ ใช้เครื่องหมาย -
- คูณ ใช้เครื่องหมาย *
- หาร ใช้เครื่องหมาย /
- Remainder (หารเอาเศษ) ใช้เครื่องหมาย %
- Exponentiation (ยกกำลัง) ใช้เครื่องหมาย **

Arithmatic Operator

Software Park Thailand
</Code Camp>

Number

- ตัวอย่างการใช้ค่า number และข้อควรระวัง

```
console.log(1 / 0); // Infinity
console.log(Infinity); // Infinity
console.log('Codecamp' / 2); // NaN
console.log('John' * 2 + 100); // NaN
```

Arithmatic Operator

Software Park Thailand
</Code Camp>

Maths Operations

- % คือการหารเอาเศษ
- ผลลัพธ์ ของ % คือเศษที่ได้จากการหาร

```
console.log(4 % 3); // 1
console.log(14 % 5); // 4
console.log(9 % '5'); // 4
```

Arithmatic Operator

Software Park Thailand
</Code Camp>

Maths Operations

- ** คือการยกกำลัง

```
console.log(2 ** 5); // 32
console.log(7 ** 2); // 49
console.log(0.2 ** 3); // 0.008000000000000002
```

เกิดอะไรขึ้นกับบรรทัดที่ 3 ?

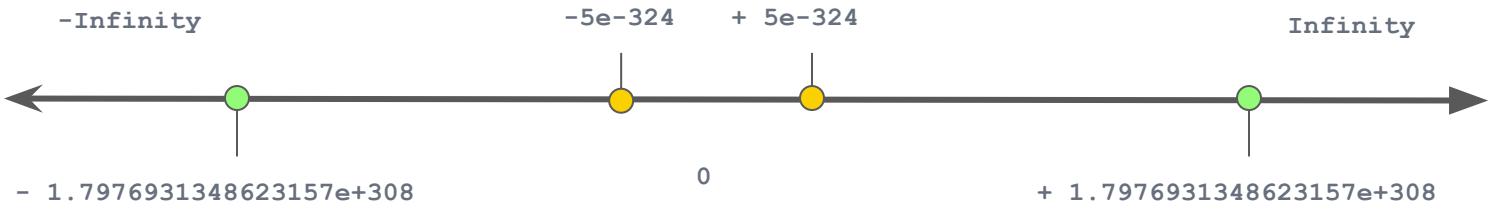
https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html

https://www.patanasongsivilai.com/blog/%E0%B8%9B%E0%B8%B1%E0%B8%8D%E0%B8%AB%E0%B8%B2-ieee-754-floating-point/

Number (In Deep)

- ข้อมูลประเภท number สามารถเก็บค่าได้ในช่วงหนึ่งเท่านั้น ไม่ใช่เท่าไหร่ก็ได้
- ด้วยข้อจำกัดของขนาด memory ของตัวแปร (64 bits)

```
// ##  FLOAT : MAX-MIN
Number.MAX_VALUE // 1.7976931348623157e+308
Number.MIN_VALUE // 5e-324
Number.MAX_VALUE  + 0.00000000000000001e308 // Infinity
```

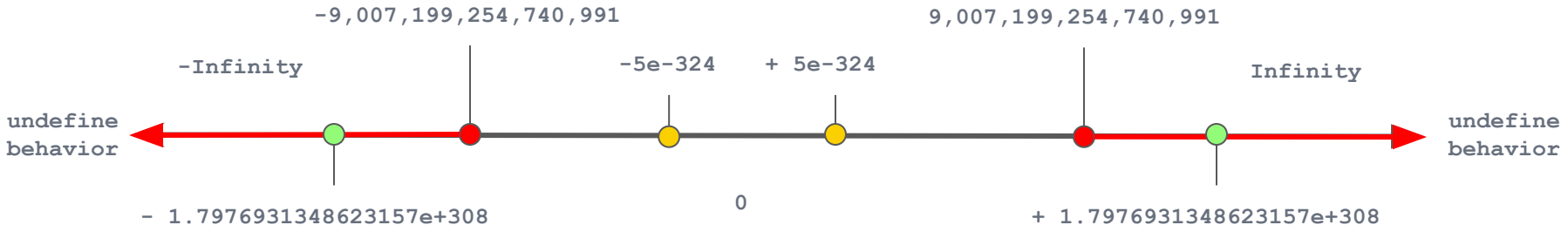


Number (In Deep)

Software Park Thailand
</Code Camp>

- ข้อมูลแบบ Integer (จำนวนเต็ม) ก็มีข้อจำกัดเช่นเดียวกัน
- หากต้องการใช้งานตัวเลขที่สูงกว่าขอบเขตให้ใช้ข้อมูลประเภท BigInt แทน

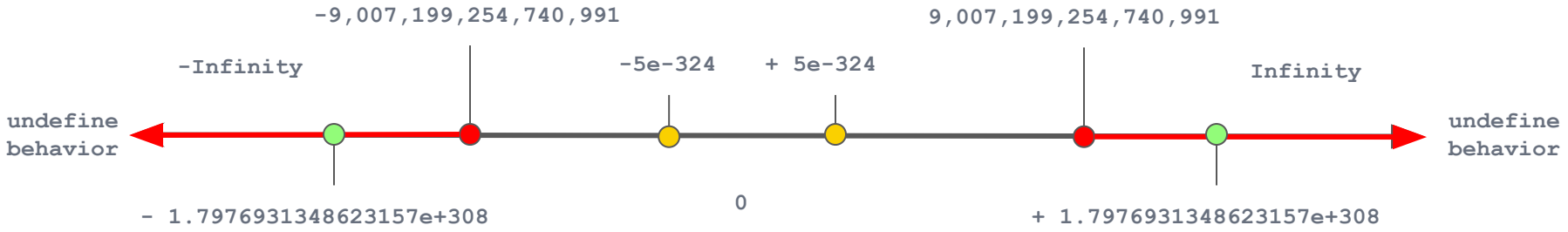
```
// ## INTEGER : SAFE INT
Number.MAX_SAFE_INTEGER //9,007,199,254,740,991
Number.MIN_SAFE_INTEGER // -9,007,199,254,740,991
```



Number (In Deep)

- ข้อมูลแบบ Integer (จำนวนเต็ม) ก็มีข้อจำกัดเช่นเดียวกัน
- หากต้องการใช้งานตัวเลขที่สูงกว่าขอบเขตให้ใช้ข้อมูลประเภท BigInt แทน

```
// Wierd Part
Number.MAX_SAFE_INTEGER + 1 === Number.MAX_SAFE_INTEGER + 2
```

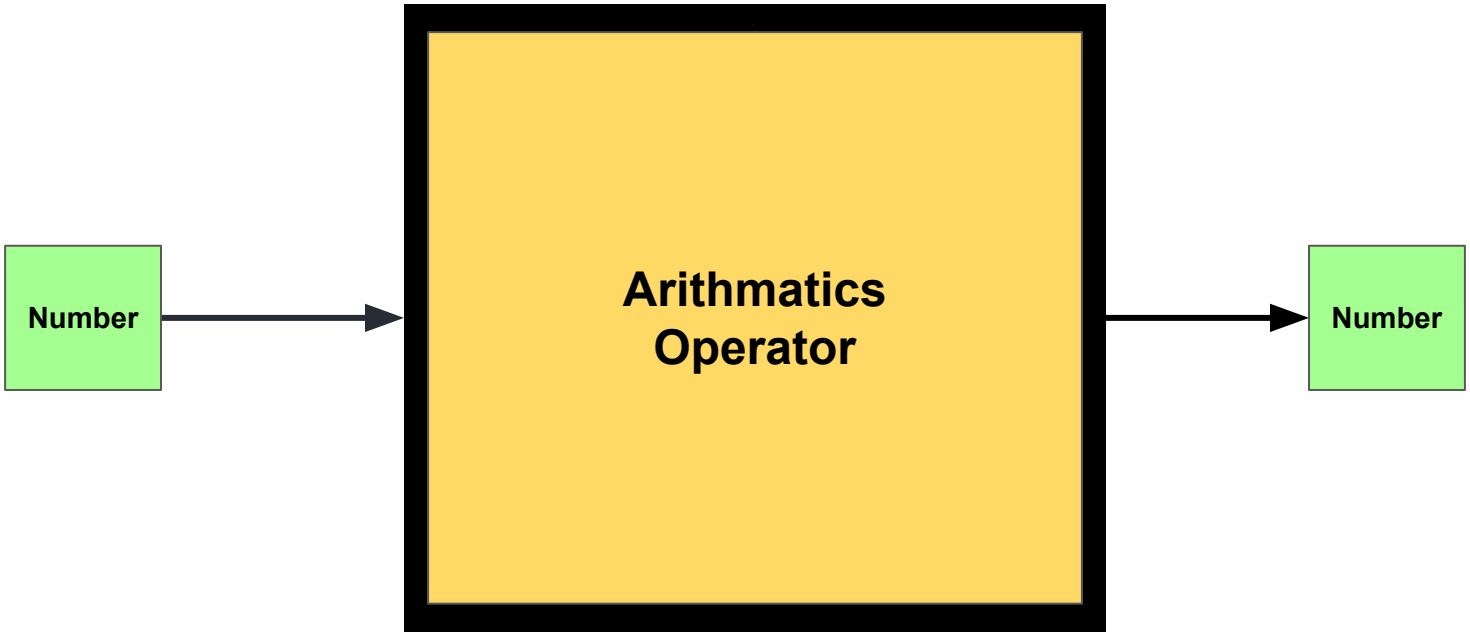


Number (In Deep)

- ค่า Infinity
- ค่า NaN

```
// ## INFINITY
Number.POSITIVE_INFINITY // Infinity
Number.NEGATIVE_INFINITY // -Infinity


// ## NOT A NUMBER
Number.NaN // NaN
```

Summary



Summary

Number

- คือตัวเลขทั้งจำนวนเต็ม (Integer) และ ทศนิยม (Float)
- Special numeric values (ค่าพิเศษที่นอกเหนือจาก Integer และ Float)
  - NaN (Not a number)
  - Infinity (∞)
  - -Infinity (-∞)
- สามารถดำเนินการทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร ได้
- ไม่ว่าจะทำอะไร (บวก, ลบ, คูณ, หาร) กับ NaN ก็จะได้ NaN เสมอ

Summary

Primitives Value (immutable)

- **Number**
- String
- Undefine
- NULL
- Boolean
- Symbol
- Bigint

normal values with some limitation

- **Integer** *(10,0,-5)*
- **Float** *(3.14)*
- **NaN**
- **Infinity, -Infinity**

features

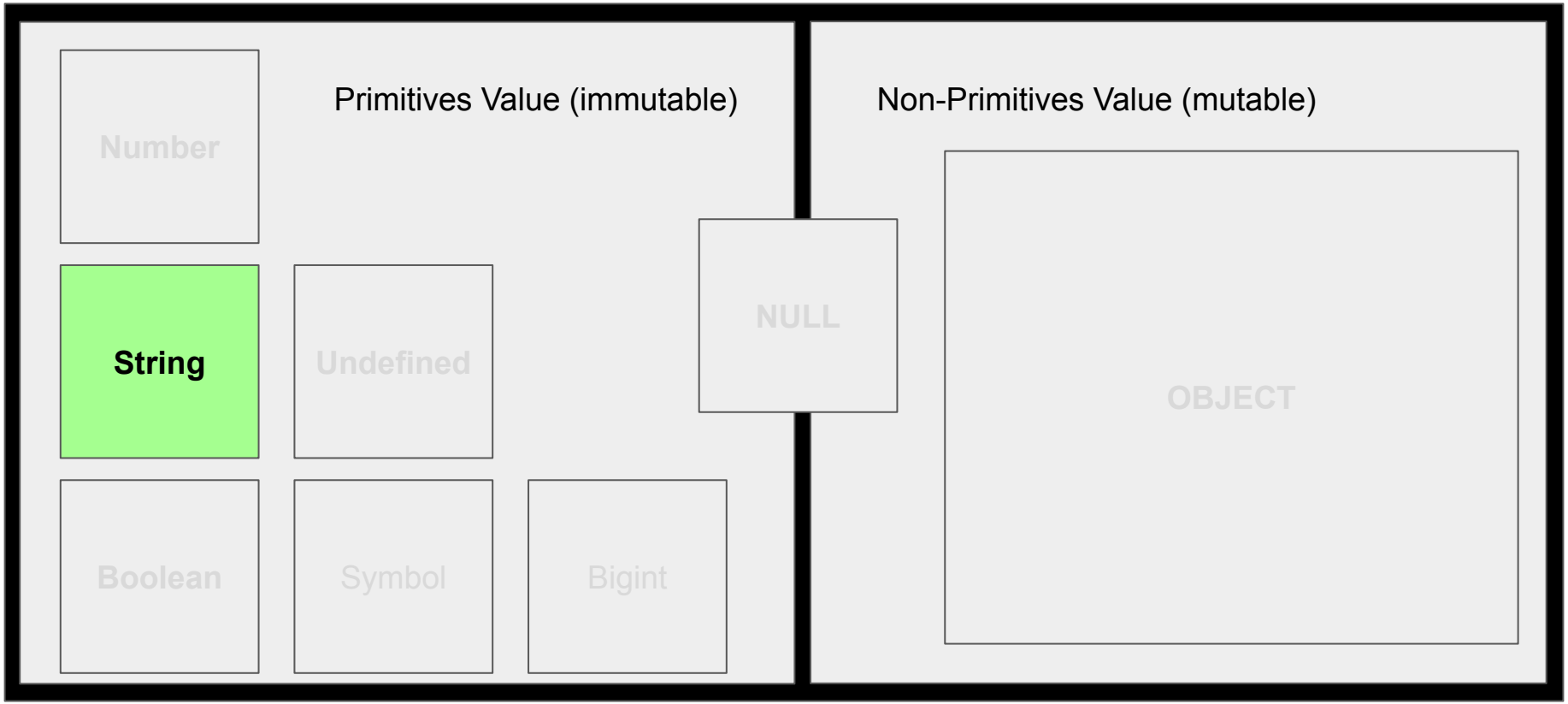
- **arithmatics**

Operator : + , -, *, / , % with Precedence
Note : PEMDAS

Software Park Thailand
</Code Camp>

String

String


Primitives Value (immutable)
Non-Primitives Value (mutable)
Number
String
Undefined
NULL
OBJECT
Boolean
Symbol
Bigint

String

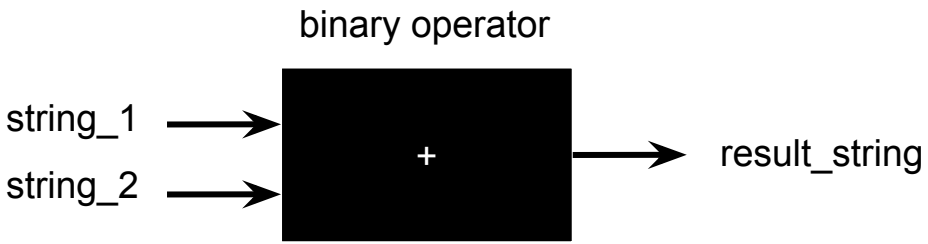
- ในหลายๆภาษามีการแยก อักขระ (character) กับ ข้อความ (string) ออกจากกันอย่างชัดเจน เช่นภาษา C
- ใน javascript จะมีข้อมูลประเภท string เท่านั้น
- การใช้งานค่า string สามารถเรียกใช้ได้ 3 แบบ

```
// Single Qoute
'W'
// Double Qoute
"W"
// Back Tick
`W`
```

```
// Single Qoute
'Why is JavaScript so Easy'
// Double Qoute
"Why is JavaScript so Easy"
// Back Tick
`Why is JavaScript so Easy`
```

String

- เราสามารถต่อ string ได้ด้วย + (addition operator) เรียกว่า ***string concatenate***
- หากดำเนินการด้วย operator อื่นจะได้ ***NaN*** (เช่น - , * , / , %)
- หากดำเนินการกับข้อมูลประเภทอื่นจะเจอกับ *Automatic Type Conversion*

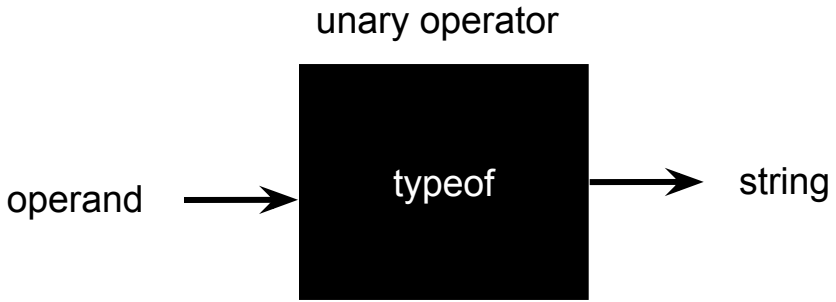



```
'CodeCamp' + ' ' + "Thailand" + "_" + `12` // "CodeCamp Thailand_12"
"CodeCamp" + 12 // "CodeCamp12"
"CodeCamp" + NaN // "CodeCampNaN"
```

String

- เราสามารถตรวจสอบประเภทของข้อมูลได้ด้วย operator ที่ชื่อว่า **typeof**
- typeof เป็น unary operator (ใช้แค่ 1 operand)
- เมื่อ typeof ดำเนินการเสร็จสิ้นจะส่ง***คืนค่า string มาให้*** (string ที่ระบุประเภทของข้อมูล)

```
typeof 5 // "number"
typeof '5' // "string"
typeof "W" // "string"
typeof `ก` // "string"
typeof "CodeCamp" // "string"
typeof 'CodeCamp' // "string"
typeof `CodeCamp` // "string"
typeof typeof 5 //
```



note : สามารถเรียกใช้ได้อีกแบบ typeof(value)

String

- มีเฉพาะการใช้งาน string แบบ backtick เท่านั้นที่สามารถแทรก ***expression*** ลงไปได้
- an **expression** is a syntactic entity in a programming language that may be evaluated to determine its value. (read more)
- การใช้งาน backtick สำหรับ string มีอีกชื่อคือ ***template literal*** , ***string literal***

```
`CodeCamp ${' ' + '12'}` // "CodeCamp 12"
`CodeCamp ${(20-16)*3}` // "CodeCamp 12"
'CodeCamp ${(20-16)*3}' // 'CodeCamp ${(20-16)*3}'
"CodeCamp ${(20-16)*3}" // "CodeCamp ${(20-16)*3}"
```

String (In Deep)

- อีกชื่อนึงของ string (ข้อความ) คือ array of character (แถวลำดับอักษร)
- in short : string คือเอา character มาเรียงต่อกัน
- จริงๆแล้ว computer รู้จักแค่เลข แต่คอมฯ รู้ว่าต้องแสดงผลเป็นอักขระด้วยการ mapping
- ระบบแรกเริ่มคือ ASCII โดย 1 character จะเก็บโดยใช้ความจำ 7 bits ภายหลังเป็น 8 bits (1byte)
- พอคอมพิวเตอร์เริ่มแพร่หลาย มีผู้ใช้หลายภาษามากขึ้น ก็มีความจำเป็นที่ต้องขยายระบบอักขระ
- ระบบเข้ารหัสที่เพิ่มเติมมาคือ UTF-8 ที่ใช้หน่วยความจำได้ถึง **32 bits (4bytes)** ในการเก็บอักขระ
- นอกจากนั้นยังมีระบบ UTF-16 , UTF-32 อีกด้วย
- read more

String (In Deep)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

String

- ASCII Table สามารถแบ่งประเภท character ได้ 2 กลุ่ม
- non-printable character (code : 0-31)
- printble character (code : 32 ขึ้นไป)
- ในกลุ่ม non-printable character ควรรู้จักกลุ่มย่อยที่เป็น white space character

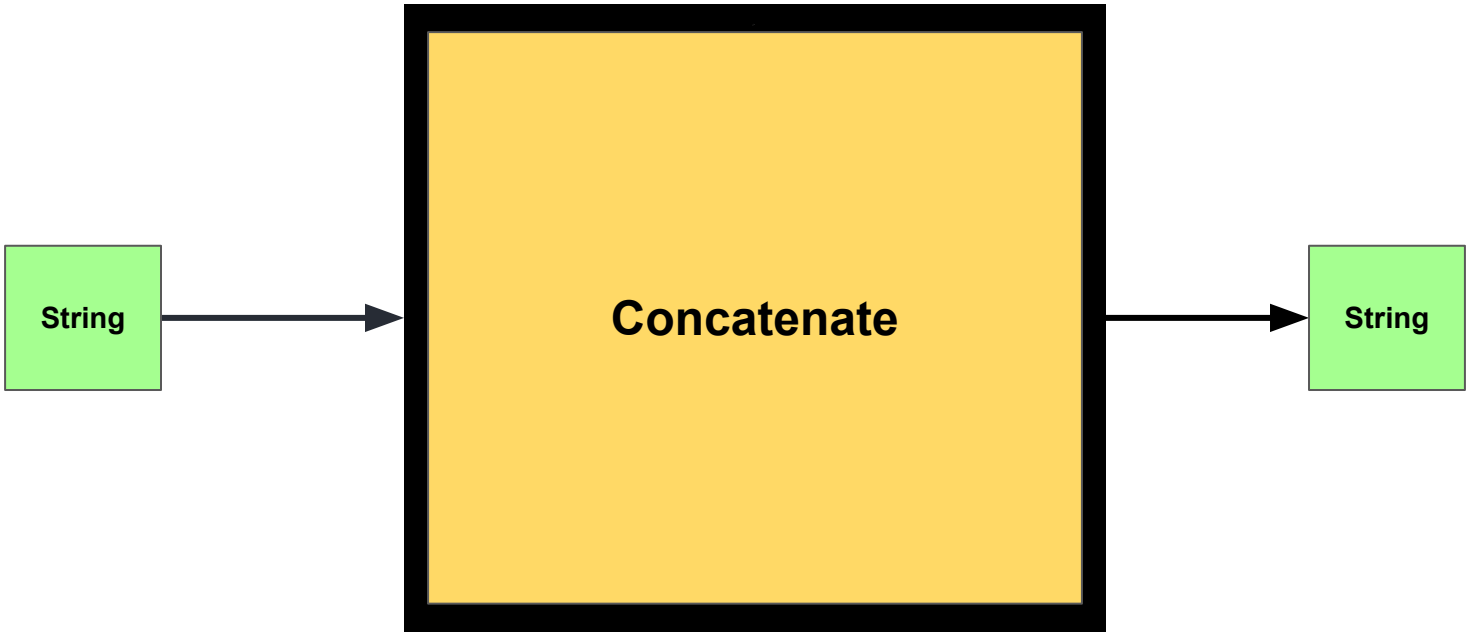
```
"Codecamp" + "\t" + "12" // Codecamp 12
"Codecamp" + "\v" + "12" // Codecamp 12
"Codecamp" + "\f" + "12" // Codecamp 12
"Codecamp" + "\n" + "12" // Codecamp 12
```

String

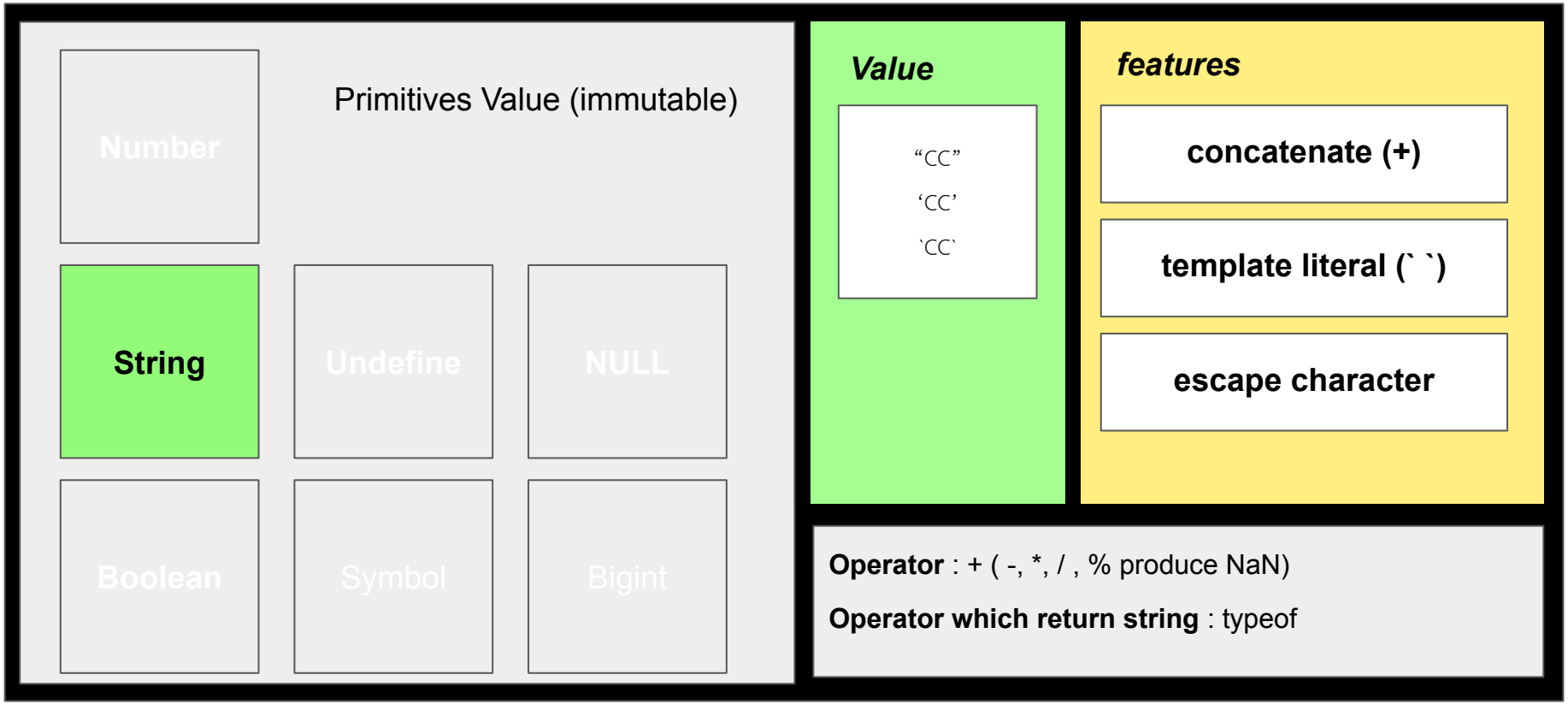
- Escape Character : อักขระที่ไม่ได้ตีความตรงตัว ต้องอาศัยการตีความอีกชั้นนึง
- an **escape character** is a character that invokes an alternative interpretation on the following characters in a character sequence

```
console.log("This is awesome "qoute"")
console.log('This is awesome "qoute"')
console.log(`This is awesome "qoute"`)
console.log("This is awesome \"qoute\"")
console.log("How to print backslash \\")
```

Summary



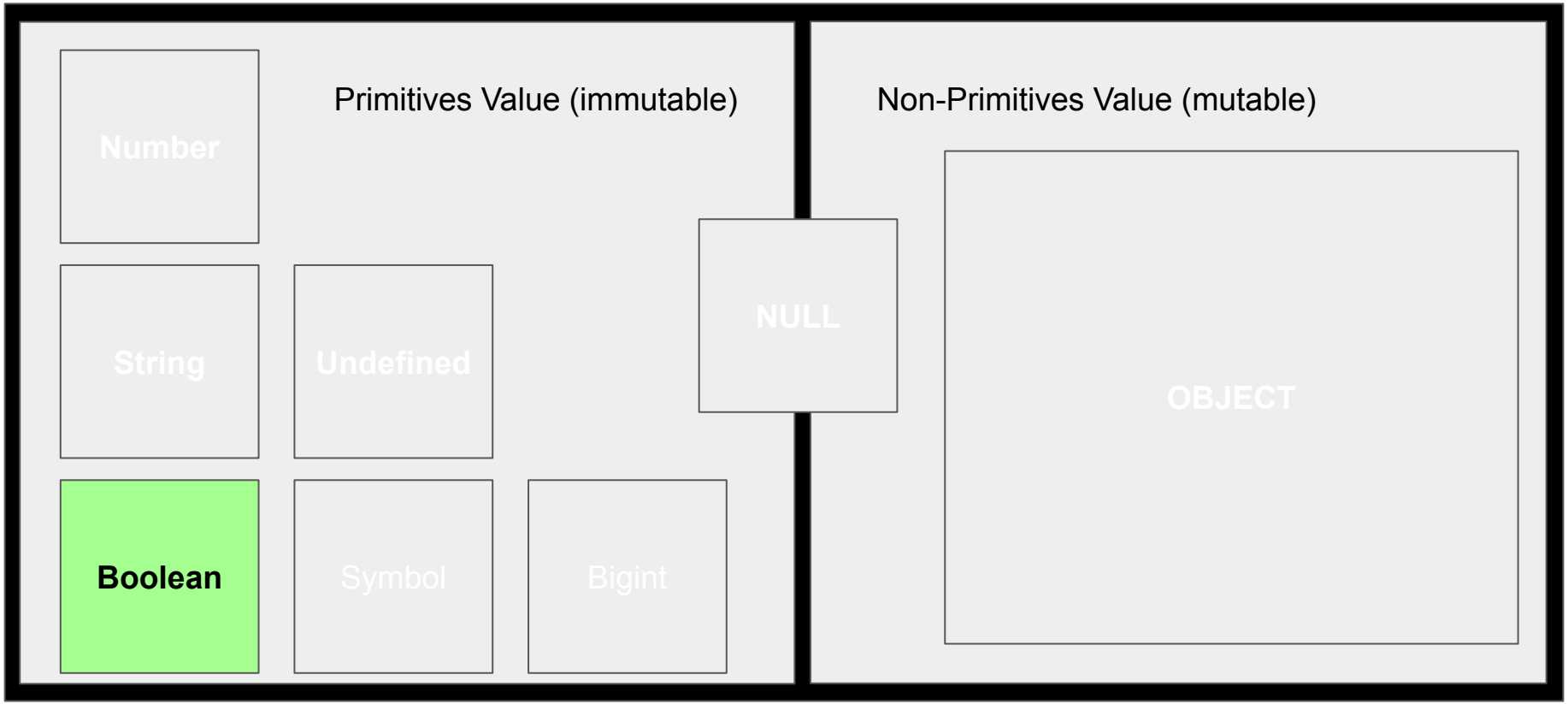
Summary


Primitives Value (immutable)
Number
String
Undefine
NULL
Boolean
Symbol
Bigint
Value
“CC”
‘CC’
`CC`
features
concatenate (+)
template literal (` `)
escape character
Operator : + (-, *, / , % produce NaN)
Operator which return string : typeof

Software Park Thailand
</Code Camp>

Boolean

Boolean


Primitives Value (immutable)
Non-Primitives Value (mutable)
Number
String
Undefined
NULL
OBJECT
Boolean
Symbol
Bigint

Boolean

- Boolean คือชนิดข้อมูลแบบตรรกะ หรือค่าความเป็นจริง
- มีค่าเพียง 2 แบบ คือ จริง (True) หรือเท็จ (False) เท่านั้น

```
true
false
```

Boolean

- Boolean คือชนิดข้อมูลแบบตรรกะ หรือค่าความเป็นจริง
- มีค่าเพียง 2 แบบ คือ จริง (True) หรือเท็จ (False) เท่านั้น

```
true
false
```

Logical Operator

Software Park Thailand
</Code Camp>

Logical Operators

- ตัวดำเนินการแบบตรรกะ
- OR ใช้เครื่องหมาย ||
- AND ใช้เครื่องหมาย &&
- NOT ใช้เครื่องหมาย !

Logical Operator

Logical Operators

- OR ใช้เครื่องหมาย ||
- OR จะให้ผลลัพธ์เป็น false กรณีเดียวเท่านั้นคือ operand ทั้ง 2 ตัว เป็น false นอกนั้นจะให้ผลลัพธ์เป็น true เสมอ

```
console.log(true || true); // true
console.log(true || false); // true
console.log(false || true); // true
console.log(false || false); // false
```

Logical Operator

Logical Operators

- AND ใช้เครื่องหมาย &&
- AND จะให้ผลลัพธ์เป็น true กรณีเดียวเท่านั้นคือ operand ทั้ง 2 ตัว เป็น true นอกนั้นจะให้ผลลัพธ์เป็น false เสมอ

```
console.log(true && true); // true
console.log(true && false); // false
console.log(false && true); // false
console.log(false && false); // false
```

Logical Operator

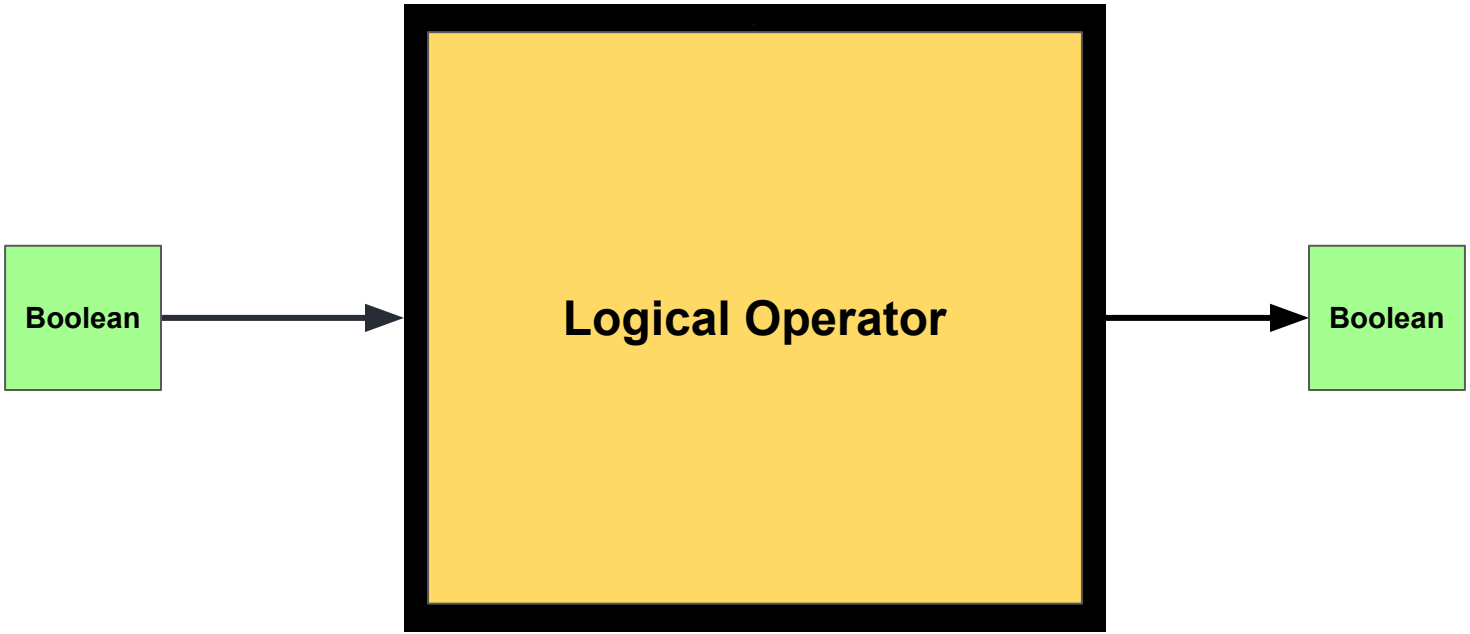
Software Park Thailand
</Code Camp>

Logical Operators

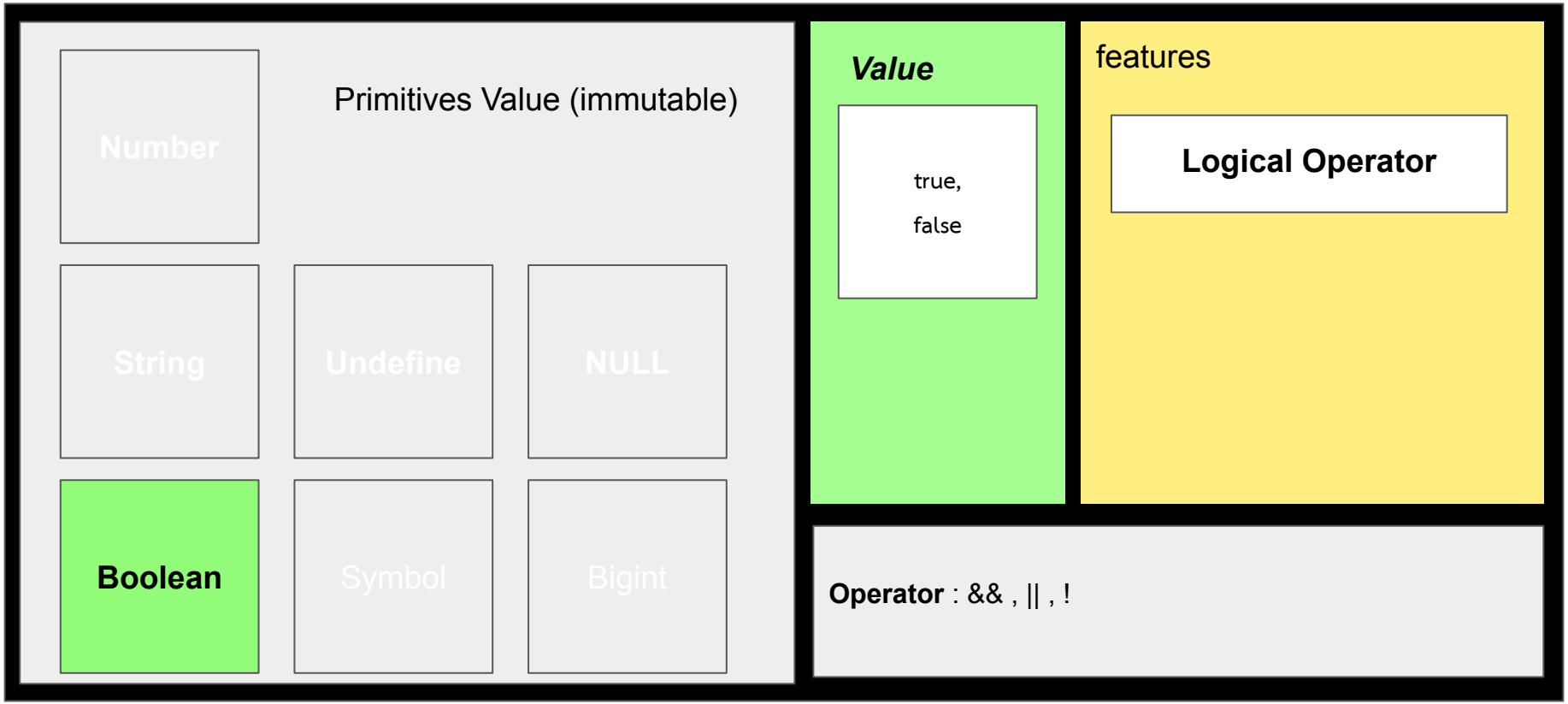
- NOT ใช้เครื่องหมาย !
- NOT จะให้ผลลัพธ์เป็นตรงกันข้ามกับค่าเดิม

```
console.log(!true); // false
console.log(!false); // true
```

Summary

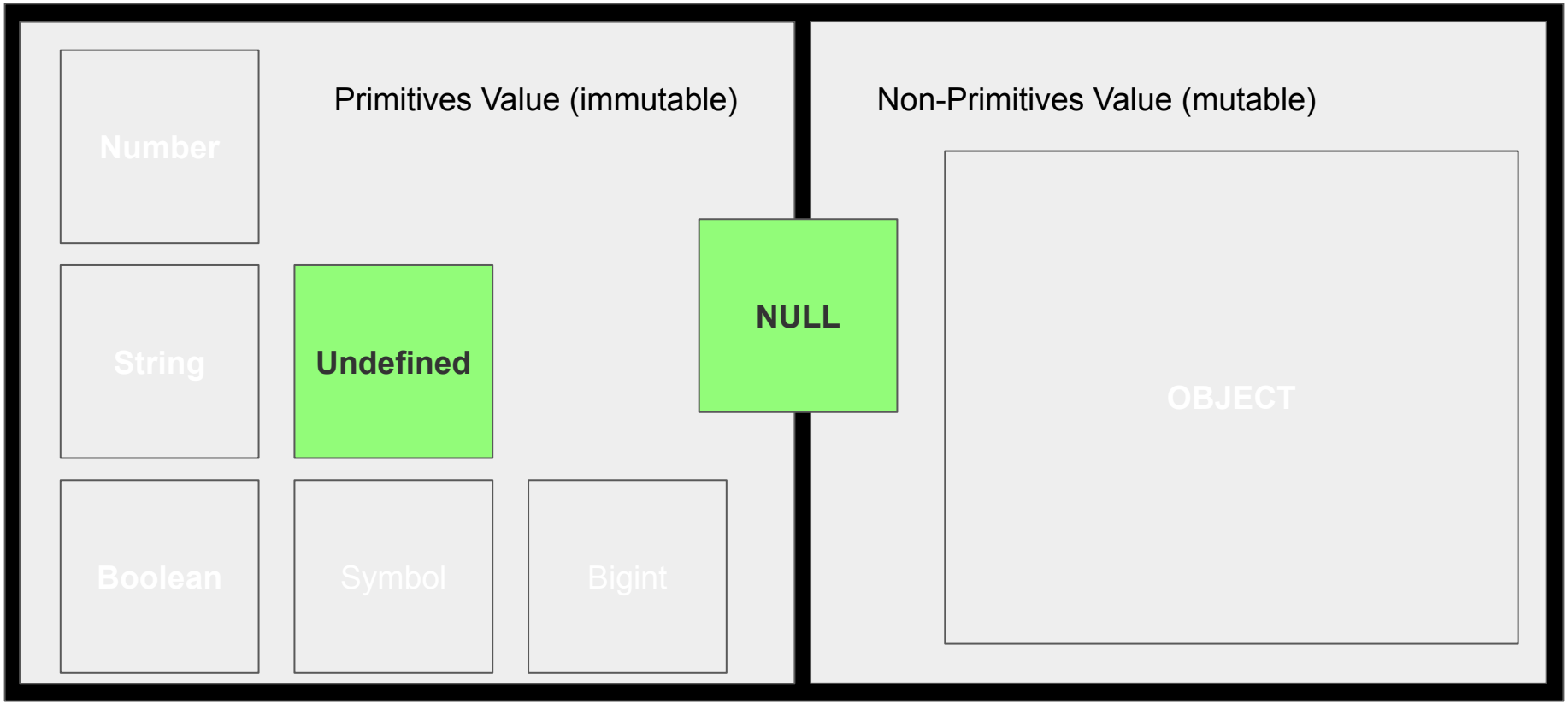


Summary


Primitives Value (immutable)
Number
String
Undefine
NULL
Boolean
Symbol
Bigint
Value
true,
false
features
Logical Operator
Operator : && , || , !

Null, Undefined

Null, Undefined



Null , Undefined

- ใน javascirpt มีชนิดข้อมูลพิเศษอยู่ 2 ตัวที่แสดงถึงการ “ไม่มีค่า” ก็คือ null กับ undefined
- ข้อมูลทั้ง 2 ชนิดมีทั้งความเหมือนและความต่างในเวลาเดียวกัน (หลายๆครั้งทำให้ชวนสับสน)
- เราจะพูดถึงข้อมูลทั้ง 2 แบบนี้ โดยละเอียดในหัวข้อ Variable

```
null

undefined
```

Software Park Thailand
</Code Camp>

Expression

Expression

Software Park Thailand
</Code Camp>

Environment

Functions

Control Flow

Loop

Branch

Variable

Statements

DataType

Expression

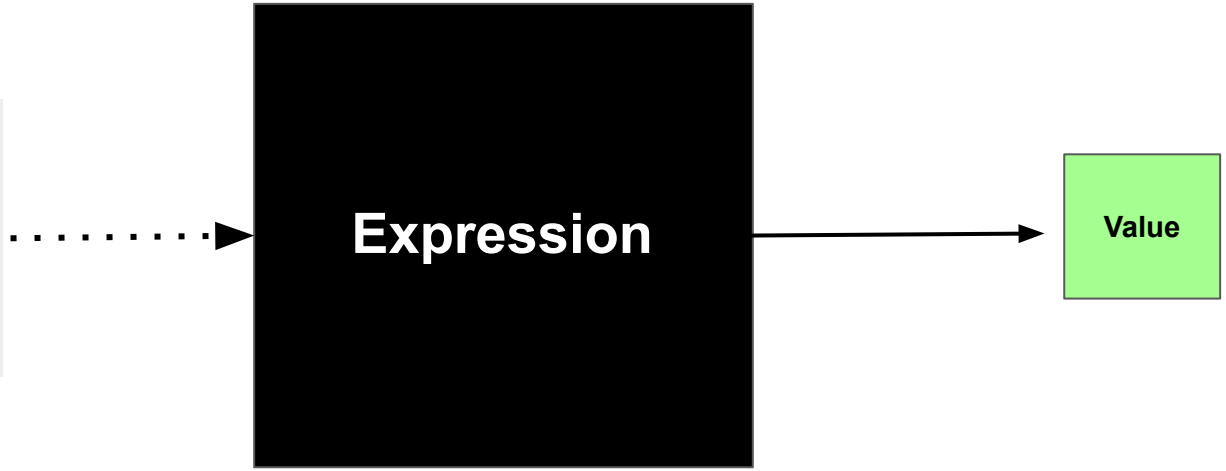
Comment

Expression

- an **expression** is a syntactic entity in a programming language that may be evaluated to **determine its value**
- It is a combination of one or more constants, variables, functions, and operators that the programming language interprets and computes to produce ("to return", in a stateful environment) another value
- expression ส่วนย่อยของโปรแกรม(นิพจน์,ข้อความย่อย) ที่สามารถ “evaluate” ค่าได้
- expression อาจจะประกอบขึ้นจาก ***ค่าคงตัว (constant,value) , ตัวแปร, ฟังก์ชัน*** และ***ตัวดำเนินการ*** อย่างน้อยหนึ่งอย่างหรือมากกว่านั้น ซึ่งตัว programming langauge จะทำการ evaluate ค่าใหม่ให้ (หรืออาจจะเรียกว่า “คืนค่า” กลับมา) เพื่อให้ส่วนอื่นของโปรแกรมเรียกใช้งานต่อไป
- **Expressions produce at least one value.**
- **Statements Do Something** and are often composed of expressions (or other statements)

Expression

- Value
- Values with Operator



Expression

- ตัวอย่าง expression จากค่าคงตัว(value)

```
// ### Expression from value

502;  // 502
"CodeCamp"; // "CodeCamp"
true; // true
undefined; // undefined
null; // null
```

Expression

- ตัวอย่าง expression จาก values และ operator

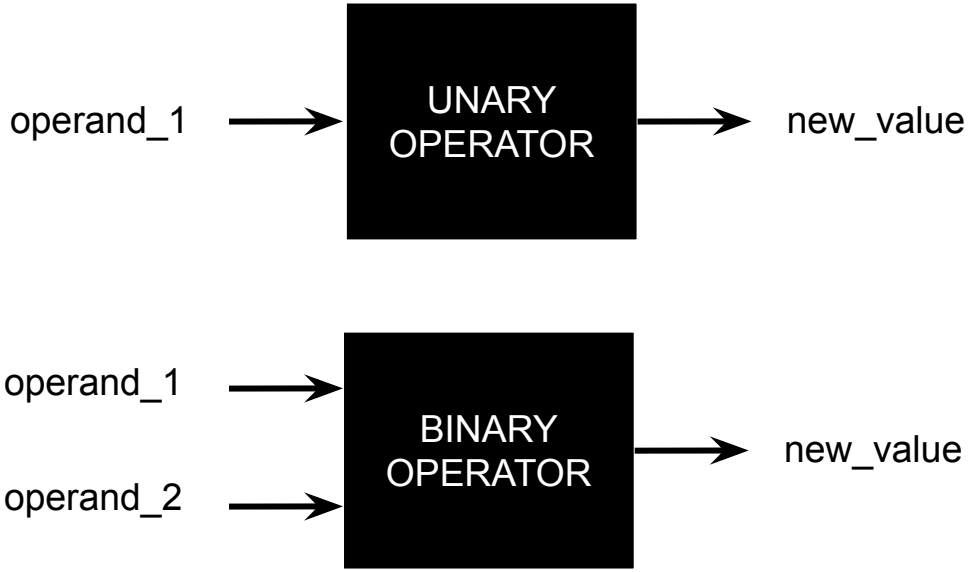
```
//  ### Expression from Values and Operator

5 + 7 - 3; // 9
"CodeCamp" + "12"; // CodeCamp12
true && true; // true
"Hello" - 44; // NaN
!!""; // false
num > 9 ? "Higher" : "Lower"; // result depend on num value
```

Operators

Terms: “unary”, “binary”, “ternary”, “operand”

- An operand is what operators are applied to. For instance, in the multiplication of 5 * 2 there are two operands: the left operand is 5 and the right operand is 2.
- An operator is **unary** if it has a **single operand.**
- An operator is **binary** if it has **two operands.**
- An operator is **ternary** if it has **three operands.**

Operator


operand_1
UNARY OPERATOR
new_value
operand_1
operand_2
BINARY OPERATOR
new_value

Operator

Expression

- ตัวอย่าง expression จาก function

```
// ### Expression from Function

function sayHi() {
 return "Hello,CodeCamp";
}

sayHi(); // Hello, CodeCamp
```

Software Park Thailand
</Code Camp>

Variable
(binding)

Values,Types

Software Park Thailand
</Code Camp>

Environment

Functions

Control Flow

Loop

Branch

Variable

Statements

DataType

Expression

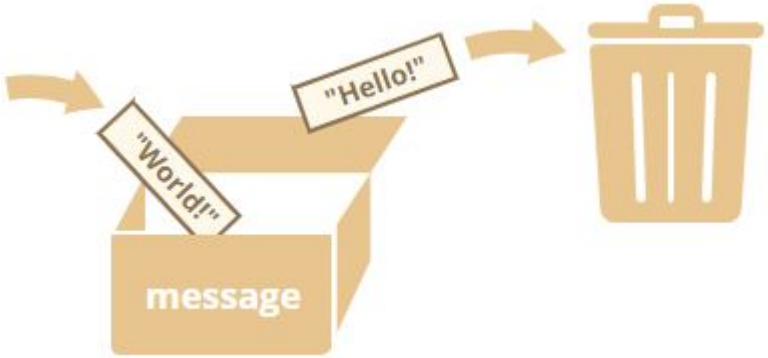
Comment

Variables

What's Variables (name binding)

- ตัวแปรคือสิ่งที่เอาไว้ใช้เก็บข้อมูล
- สามารถแก้ไขหรือเปลี่ยนแปลงข้อมูลที่เก็บไว้ในตัวแปรได้

```
let message = 'Hello!';
message = 'World!';
```



Variable

Terms

- Variable **Declaration** : การประกาศตัวแปรแต่ยังไม่กำหนดค่า
- Variable **Initialization** : การประกาศตัวแปร พร้อมกับกำหนดค่า
- **Assigment** Operator : ตัวดำเนินการที่ใช้กำหนดค่า (Value) ให้กับตัวแปร

```
let name; // Declaration
let age = 27 - 3; // Initialization
name = 'Hello World!'; // Assign,ReAssign
```

Variable

Variable Declaration

- การประกาศตัวแปร คือ การบอกให้โปรแกรมรู้ว่าเราจะใช้ตัวแปรนั้น
- สามารถประกาศตัวแปร โดยเขียน var let หรือ const นำหน้าชื่อตัวแปรที่เราต้องการ
- var let สามารถกำหนดค่าเริ่มต้นให้ตัวแปรตอนประกาศ หรือกำหนดในภายหลังก็ได้

```
var name = 'John Doe';
var age = 27;
var message = 'Hello World!';
```

```
let name = 'John Doe',
age = 27,
message = 'Hello World!';
```

```
const name = 'John Doe', age = 27, message = 'Hello World!';
```

LET

Var, Let, and Const – What's the Difference?

- ตัวอย่างการประกาศตัวแปรแบบ let

```
// No error, value is undefined
let name;

// Assign value John Doe to variable name
name = 'John Doe';
```

```
// define the variable and assign the value
let name = 'John Doe';
```

CONST

Var, Let, and Const – What's the Difference?

- ตัวอย่างการประกาศตัวแปรแบบ const

```
// Uncaught SyntaxError: Missing initializer in const declaration
const name;
```

```
// No error
const name = 'John Doe';
```

Dynamic Type

- ไม่ต้องบอกประเภทตัวแปรตอนประกาศตัวแปร
- สามารถเปลี่ยนแปลงประเภทได้

```
let name = 'John Doe';

let age = 27;
age = 'Twenty seven';
```

Assign to String

String

- การเขียน String โดยใช้ Double quote และ Single quote เหมือนกันทุกอย่าง

```
let firstName = 'John';
let lastName = "Doe";
```

Assign to String

String

- Backticks (Template literals) สามารถ แทรกตัวแปรระหว่าง String ได้

```
let firstName = 'John';
let lastName = "Doe";
let nickname = `John`;
let message = `My name is ${firstName} ${lastName}`;
console.log(message); // My name is John Doe
```

Assign to String

String

- Double quote หรือ Single quote ใส่ตัวแปรไม่ได้

```
let firstName = 'John';
let lastName = 'Doe';
let message = 'My name is ${firstName} ${lastName}';
console.log(message); // My name is ${firstName} ${lastName}
```

Assign to String

String concatenation with binary +

- คือการนำข้อมูล String 2 ค่า มาต่อกันด้วยเครื่องหมาย +

```
let firstName = 'John';
let lastName = 'Doe';
let fullname = firstName + lastName;
console.log(fullname); // JohnDoe
```

Assign to Boolean

Software Park Thailand
</Code Camp>

Boolean

- มีค่าได้แค่ 2 ค่า คือ true และ false

```
let isAllowed = true;
let isActive = false;
```

Assign to Boolean

Software Park Thailand
</Code Camp>

Boolean

- ตัวอย่าง

```
let w = 5;
let x = w > 0; // true
let y = w < 0; // false
```

Assign to Null

Null

- หมายถึง ค่าว่าง ไม่มีค่า หรือไม่ทราบค่า ก็ได้

```
let age = null;
```

Undefined

Undefined

- undefined คือข้อมูลที่ยังไม่ได้กำหนดค่า

```
let x; // undefined
var y; // undefined
```

Null vs Undefined

Software Park Thailand
</Code Camp>

[null vs undefined]

Non-zero value

null

0

undefined

Naming

Variable Naming

- ประกอบด้วย **ตัวอักษร ตัวเลข** หรือ สัญลักษณ์ **$** และ **_** เท่านั้น
- ตัวแรกต้องไม่เป็นตัวเลข
- ต้องไม่เป็นคำสงวน เช่น if, function, break เป็นต้น (Keyword ทั้งหมด)

Naming

Variable Naming (Example)

- c1mp ถูกต้อง
- _codecamp12 ถูกต้อง
- _ ถูกต้อง
- $code ถูกต้อง
- 3codecamp ผิด เพราะขึ้นต้นด้วยตัวเลข
- codecamp#1 ผิด เพราะมี #
- code camp ผิด เพราะมีเว้นว่าง (space)

Naming

Variable Naming (Example)

- function1 ถูกต้อง
- function ผิด เพราะ function เป็นคำสงวน
- if ผิด เพราะ if เป็นคำสงวน
- _if ถูกต้อง
- else ผิด เพราะ else เป็นคำสงวน
- codecamp-1 ผิด เพราะมี -
- โค้ดแคมป์ ไม่ผิด แต่ไม่ควรใช้ภาษาไทย

Naming

Name things right

- ไม่ควรตั้งชื่อด้วยอักษรตัวเดียว เช่น a, b, c
- ชื่อตัวแปรควรอ่านรู้เรื่องและสื่อความหมาย เช่น newUser, myCareer เป็นต้น
- ไม่ควรตั้งชื่อที่ไม่สื่อความหมาย เช่น aaa, bb, af, za เป็นต้น
- ถ้าชื่อตัวแปรมีช่องว่างนิยมใช้แบบ camelCase (ขึ้นต้นด้วยตัวเล็กคำถัดไปให้เริ่มด้วยตัวใหญ่) เช่น current user ให้เขียนเป็น currentUser
- สามารถตั้งชื่อแบบ snake_case (current_user) หรือ PascalCase (CurrentUser) ได้แต่นิยมใช้เป็นบางกรณี

Naming

Case-Sensitive

- JavaScript เป็นภาษาที่ชื่อตัวแปรเป็น case-sensitive
- การพิมพ์ตัวใหญ่และตัวเล็กถือว่าเป็นตัวแปรคนละตัว เช่น
  - name และ Name ถือว่าเป็นตัวแปรคนละตัว
  - codecamp และ CodeCamp ถือว่าเป็นตัวแปรคนละตัวกัน

LAB

Lab 1

- ให้ประกาศตัวแปรชื่อ person และ name
- ใส่ชื่อตัวเองลงในตัวแปร name
- นำค่าที่อยู่ในตัวแปร name ไปใส่ให้ person
- เมื่อ alert(person) ออกมาต้องเป็นชื่อตัวเอง

LAB

Lab 2

- ตั้งชื่อตัวแปรที่ใช้เก็บจำนวนเงินในกระเป๋าตังของคุณ
- ตั้งชื่อตัวแปรที่ใช้เก็บชื่อของ พ่อและแม่ของคุณ
- ตั้งชื่อตัวแปรที่ใช้เก็บที่อยู่ของบ้านคุณ
- ตั้งชื่อตัวแปรที่ใช้เก็บอายุของจักรวาล

Lab 3

- ประกาศตัวแปรเป็นค่าคงที่ชื่อ user และ role
- ใส่ค่า ‘iamhero’ ลงในตัวแปร user และ ‘customer’ ลงในตัวแปร role
- alert ตัวแปร user

LAB

Lab 4

- ประกาศตัวแปรเป็น const ชื่อ firstName มีค่าเป็นชื่อจริงตัวเอง
- ประกาศตัวแปรเป็น let ชื่อ lastName มีค่าเป็นนามสกุลตัวเอง
- ประกาศตัวแปรเป็น var ชื่อ nickname มีค่าเป็นชื่อเล่นตัวเอง
- ประกาศตัวแปรทั้ง 3 ซ้ำอีกรอบโดยใช้ชื่อตัวแปรเดิม และให้ใส่ข้อมูลของเพื่อนลงในตัวแปร
- ประกาศตัวแปรชื่อ birthDate โดยไม่ต้องใส่ const let หรือ var นำหน้าชื่อตัวแปร และกำหนดค่าเริ่มต้นเป็นวันเกิดของตัวเอง

LAB

Lab 5

- ประกาศตัวแปรสำหรับเก็บชื่อจริง และกำหนดค่าเริ่มต้นเป็นชื่อจริงของผู้เรียน
- ประกาศตัวแปรสำหรับเก็บนามสกุล และกำหนดค่าเริ่มต้นเป็นนามสกุลของผู้เรียน
- ประกาศตัวแปรสำหรับเก็บอายุ และกำหนดค่าเริ่มต้นเป็นอายุของผู้เรียน
- ประกาศตัวแปรสำหรับเก็บที่อยู่ และกำหนดค่าเริ่มต้นเป็นที่อยู่ของผู้เรียน

ให้ใช้คำสั่ง console.log โดยให้ได้ผลลัพธ์ออกมาเป็น

Full Name: John Doe, Age: 27, Address: 118 Mint Tower Banthatthong Rd.

LAB

Lab 6

- ประกาศตัวแปรชื่อ BRAND_NAME ให้ค่าเป็น ‘I am a hero’
- สามารถประกาศชื่อตัวแปรชื่อตัวพิมพ์ใหญ่ทั้งหมดได้หรือไม่ ถ้าได้นิยมใช้ในกรณีอะไร

LAB

Lab 7

- จงหาผลลัพธ์ในบรรทัดคำสั่ง console.log ทั้งหมด

```
const country = 'Thailand';
const continent = 'Asia';
console.log(`number is ${2}`);
console.log(`result is ${1 + 3}`);
console.log(`I live in ${country}`);
console.log(`I live in ${country}, ${continent}`);
console.log(`I live in ${country + ', ' + continent}`);
console.log(`I live in ${'country, continent'}`);
```

Software Park Thailand
</Code Camp>

Operator with Variable

Operator with Variable

Software Park Thailand
</Code Camp>

Environment	**Functions**	*Control Flow*
		Loop
Variable	**Statements**	**Branch**
DataType	**Expression**	**Comment**

Modify in-place

Modify-in-place

- We often need to apply an operator to a variable and store the new result in that same variable.

```
let num = 2;
num = num + 2;
console.log(num); // 4
```

```
let num = 2;
num += 2;
console.log(num); // 4
```

การเขียนทั้งสองแบบได้ผลลัพธ์เหมือนกัน แต่การเขียนแบบ Modify-in-place จะสั้นกว่า

Modify in-place

Modify-in-place

- ทำได้ทั้ง บวก(+), ลบ(-), คูณ(*) หาร(/), ยกกำลัง(**) และ หารเอาเศษ(%)

```
let num = 10;
num += 2; // 12
num -= 2; // 10
num *= 2; // 20
num /= 2; // 10
num **= 2; // 100
num %= 7; // 2
```

Increment

- ++ คือการเพิ่มขึ้น 1
- ใช้ได้กับตัวแปรเท่านั้น ไม่สามารถใช้กับ ตัวเลข ได้

```
let num = 2;
num = num + 1;
console.log(num); // 3
```

```
let num = 2;
num++;
console.log(num); // 3
```

การเขียนทั้งสองแบบได้ผลเหมือนกัน แต่การเขียนด้วย ++ จะสั้นกว่า

decrement

- -- คือการลดลง 1
- ใช้ได้กับตัวแปรเท่านั้น ไม่สามารถใช้กับ ตัวเลข ได้

```
let num = 2;
num = num - 1;
console.log(num); // 1
```

```
let num = 2;
num--;
console.log(num); // 1
```

การเขียนทั้งสองแบบได้ผลเหมือนกัน แต่การเขียนด้วย -- จะสั้นกว่า

LAB

Lab 1

- ให้สร้างตัวแปร a มีค่าเท่ากับ 0
- ให้เพิ่มค่า a อีก 1 แล้วเก็บค่าไว้ใน a โดยใช้วิธี Increment
- ให้เพิ่มค่า a อีก 3 แล้วเก็บค่าไว้ใน a โดยใช้วิธี Modify-in-place
- ให้คูณ a ด้วย 17 แล้วเก็บค่าไว้ใน a โดยใช้วิธี Modify-in-place
- ให้หารเอาเศษ a ด้วย 7 แล้วเก็บค่าไว้ใน a โดยใช้วิธี Modify-in-place
- ผลลัพธ์สุดท้ายของ a คือเท่าไหร่

LAB

Lab 2

- ให้ระบุค่าของ a, b, c , d และ e หลังจากจบการทำงานในแต่ละ statements

```
let a = 1;
let b = 2;
let c = a++;
let d = ++c;
let e = ++d + d++ + d;
```

Software Park Thailand
</Code Camp>

Scope,Redeclaration, Hoisting

Scope,Redeclaration,Hoisting

Software Park Thailand
</Code Camp>

Environment	**Functions**	*Control Flow*
		Loop
Variable	**Statements**	**Branch**
DataType	**Expression**	**Comment**

Scope

Var, Let, and Const – What's the Difference?

- scope ของ var เป็นแบบ global scope หรือ function scope ส่วน scope ของ let และ const เป็นแบบ block scope (include function)
- scope ของตัวแปร คือ ที่หรือตำแหน่งที่จะเรียกใช้งานตัวแปรนั้นได้

```
// global scope: variable can be accessed from anywhere
function say() {
  // function scope: variable can only be accessed from within the function
}
{
  // Anything within curly braces is a block
  // block scope: variable can only be accessed from within {}
}
```

Scope

Block Scope : ตัวแปรถูกมองเห็น/เรียกใช้งานได้แค่ภายใน block

Global Scope : ตัวแปรถูกมองเห็น/เรียกใช้งานได้จากทุกตำแหน่ง

```
{
    var name = "john";
    let isMan = true
    const age = 37

    console.log(name) // 'john'
    console.log(isMan) // true
    console.log(age) // 37
}
console.log(name) // 'john'
console.log(isMan) // Reference Error
console.log(age) // Reference Error
```

Scope

Block Scope : ตัวแปรถูกมองเห็น/เรียกใช้งานได้แค่ภายใน block

Global Scope : ตัวแปรถูกมองเห็น/เรียกใช้งานได้จากทุกตำแหน่ง

```
for (var i = 0; i < 3; i++){
   console.log(i)// 0 1 2 3
}
console.log(i) // 3
for (let i = 0; i < 3; i++){
   console.log(i)
}
console.log(i) // Reference Error
```

Scope

Function Scope : ตัวแปรถูกมองเห็น/เรียกใช้งานได้แค่ภายใน Function

```
function newFunction() {
   var hello = 'hello';
   const say = 'say';
 }
 console.log(hello); // Uncaught ReferenceError: hello is not defined
 console.log(say); // Uncaught ReferenceError: say is not defined
```

Redeclaration

Var, Let, and Const – What's the Difference?

- ใน scope เดียวกัน var สามารถประกาศชื่อตัวแปรซ้ำได้ ส่วน const และ let ไม่สามารถประกาศชื่อตัวแปรซ้ำได้

```
var hello = 'hello';
var hello = 're-declare'; // No Error
let say = 'say';
let say = 'new say'; // Uncaught SyntaxError: Identifier 'say' has already been declared
```

Hoisting

- การ Hoisting เกิดขึ้นในจังหวะ Interpret
- ตัวแปรประเภท var สามารถเรียกใช้งานก่อนประกาศได้
- ตัวแปรประเภท let,const ต้องประกาศก่อนเรียกใช้งานเท่านั้น

```
console.log(num); // Returns 'undefined' from hoisted var declaration
(not 6)
var num = 6; // Initialization and declaration.
console.log(num); // Returns 6 after the line with initialization is
executed.
```

Hoisting

- การ Hoisting เกิดขึ้นในจังหวะ Interpret
- ตัวแปรประเภท var สามารถเรียกใช้งานก่อนประกาศได้
- ตัวแปรประเภท let,const ต้องประกาศก่อนเรียกใช้งานเท่านั้น

```
console.log(num); // Returns 'undefined' from hoisted var declaration
(not 6)
var num; // Declaration
num = 6; // Initialization
console.log(num); // Returns 6 after the line with initialization is
executed.
```

Hoisting

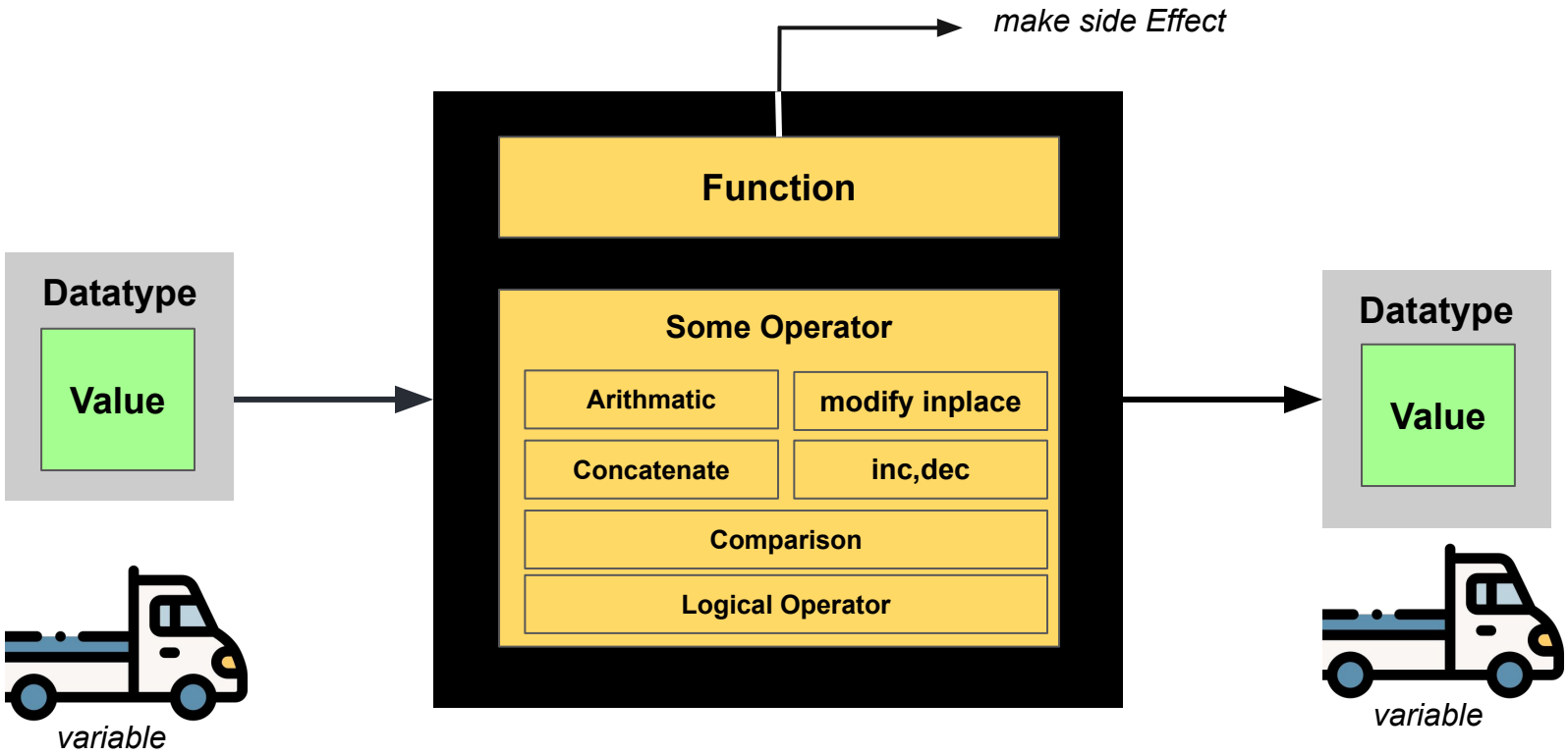
- การ Hoisting เกิดขึ้นในจังหวะ Interpret
- ตัวแปรประเภท var สามารถเรียกใช้งานก่อนประกาศได้
- ตัวแปรประเภท let,const ต้องประกาศก่อนเรียกใช้งานเท่านั้น

```
console.log(num); // Throws ReferenceError exception as the variable
value is uninitialized
let num = 6; // Initialization
```

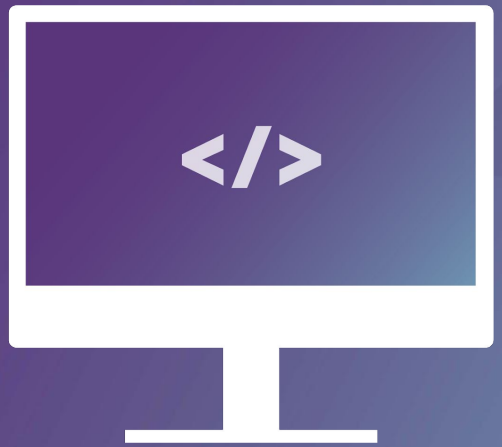
Summary

	global scoped	function scoped	block scoped	reassignable	redeclarable	can be hoisted
var	+	+	-	+	+	+
let	-	+	+	+	-	-
const	-	+	+	-	-	-

Summary


make side Effect
Function
Some Operator
Arithmatic
modify inplace
Concatenate
inc,dec
Comparison
Logical Operator
Datatype
Value
Datatype
Value
variable
variable

Software Park Thailand
</Code Camp>



SOFTWARE PARK THAILAND
Digital Academy
CHANWANICH
THAI PROGRAMMER
ICTI THE FEDERATION OF THAI INDUSTRIES
nipa.cloud
Relearn Solution Co.,Ltd.