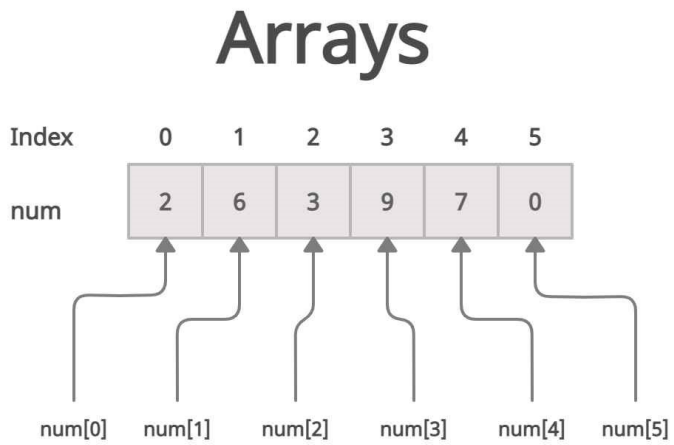


Data Type #3

JavaScript

เจาะลึก อีกหน่อย ตอน 3

Array, Array Method

Data Type

- Method of primitives
- Numbers
- Strings
- Arrays
- Array methods
- Iterables
- Map and Set
- WeakMap and WeakSet
- Object.keys, values, entries
- Destructuring assignment
- Date and time
- JSON method, toJSON

Array อาร์เรยคือ Object ที่อนุญาติ ให้เก็บ ชุดของ ข้อมูล (Value)

การสร้าง Array

```
let arr = new Array(); // แบบใช้ constructor

let arr = []; // แบบใช้ []
```

การเข้าถึง สามารถเข้าถึงแต่ละ ค่าของ อาร์เรย ได้ผ่าน Index

เริ่มต้นด้วย 0 ศูนย์

ตัวอย่าง

```
let fruits = ["Apple", "Orange", "Plum"];

   alert( fruits[0] ); // Apple
   alert( fruits[1] ); // Orange
   alert( fruits[2] ); // Plum
```

Array

- การเปลี่ยนค่า

```
fruits[2] = 'Pear';
// ["Apple", "Orange", "Pear"]
```

การเติมค่าใหม่

```
fruits[3] = 'Lemon';
// ["Apple", "Orange", "Pear", "Lemon"]
```

การหาความยาว

```
alert( fruits.length ); // 3
```

การแสดง ทุกรายการ

```
alert( fruits );     //Apple, Orange, Pear, Lemon
```

การเก็บค่า ใน Array

- สามารถเก็บ ได้ทุกประเภท any type เช่น

```
// ผสมกัน
let arr = [    'Apple'
           , { name: 'John' }
           , true
           , function() { alert('hello'); }
        ];
// ดึงค่า index 1 และ แสดง name
alert( arr[1].name ); // John
// ดึงค่า function ตำแหน่ง index 3 แล้ว run
arr[3](); // hello
```

การดึงค่า สุดท้าย ของ Array

```
// fruits=["Apple", "Orange", "Pear", "Lemon"]

//ใช้  length-1
alert( fruits[fruits.length-1] );  // Lemon

//ใช้  at(-1)
alert( fruits.at(-1) );          // Lemon
```

Array Method

- Add/Remove items

รองรับการเพิ่ม/ลด รายการ ทั้ง Queue/Stack

Queue (FIFO)

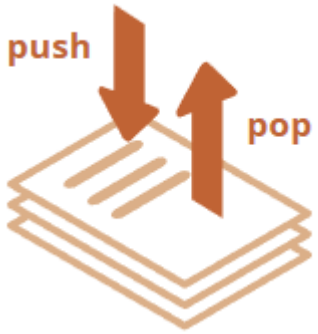
push ต่อท้าย

shift เลื่อนไปข้างหน้า

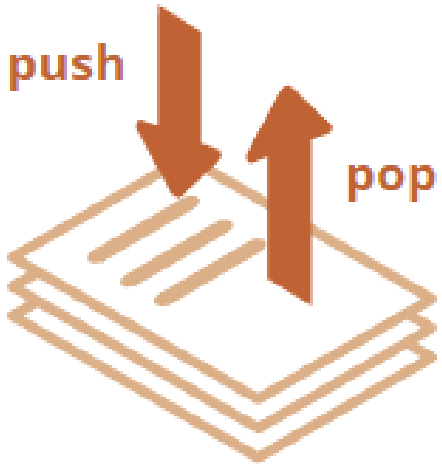
Stack (LIFO)

push ต่อท้าย

pop ดันตัวสุดท้ายออก



ตัวอย่างการใช้งาน Array Method



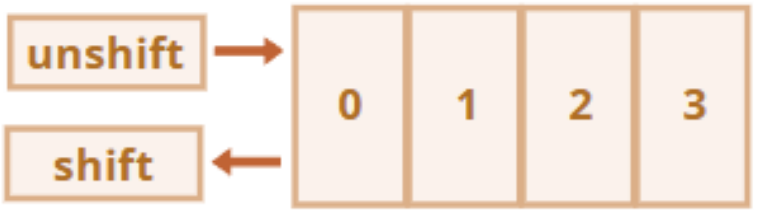
- Pop

```
let fruits = ["Apple", "Orange", "Pear"];
alert( fruits.pop() ); // pop "Pear" ออก แล้ว alert
```

- Push

```
fruits.push("Banana");
alert( fruits ); // Apple, Orange, Banana
```

ตัวอย่างการใช้งาน Array Method



- **shift**

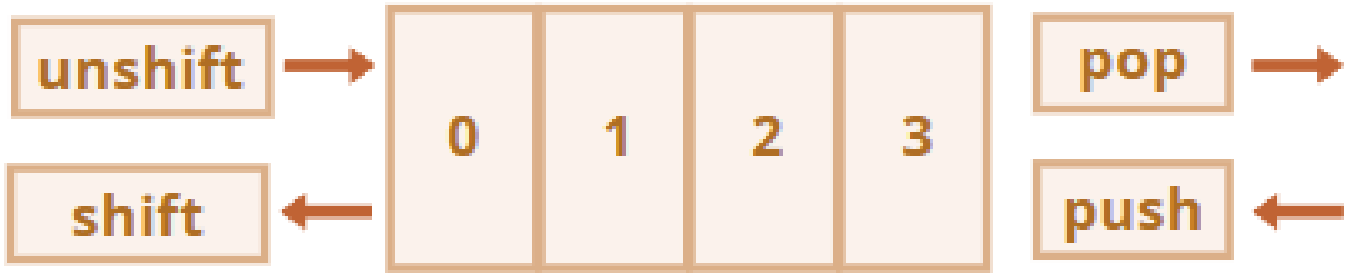
```
let fruits = ["Apple", "Orange", "Pear"];
alert( fruits.shift() ); // ลบ “Apple“ ออก แล้ว alert
alert( fruits ); // Orange,  Pear
```

- **unshift**

```
fruits.unshift("Apple");
alert( fruits ); // Apple, Orange, Pear
```

Performance

- Push/pop เร็ว
- Shift/unshift ช้า



การ Loop

```
// index
let arr = ["Apple", "Orange", "Pear"];
for (let i = 0; i < arr.length; i++) {
  alert( arr[i] );
}
```

```
//  for .. of
for (let fruit of fruits) {
  alert( fruit );
}
```

```
// for .. In
for (let key in arr) {
  alert( arr[key] );
}
```

**** ไม่ควรใช้ for..in กับ array เพราะว่า สร้างมาเพื่อใช้กับ generic object ทำให้ช้ากว่า 10-100 เท่า

length

```
let fruits = [];
fruits[123] = "Apple";

alert( fruits.length );  // 124

let arr = [1, 2, 3, 4, 5];

arr.length = 2;      // ตัดให้เหลือ 2 elements
alert( arr );        // [1, 2]

arr.length = 0;      // clear array
```

Array หลาย มิติ

```
//   อารเรย์ 2 มิติ
let matrix = [
   [1, 2, 3],      < row 0
   [4, 5, 6],
   [7, 8, 9]
];
       ^col1
alert( matrix[0][1] ); // 2    (row 0, col 1)
```

ไม่เปรียบเทียบ Array ด้วย ==

เนื่องจาก Array ในภาษา javaScript ไม่เหมือน ภาษา โปรแกรมมิ่ง อื่น
ให้ใช้การเปรียบเทียบ item-by-item ใน loop โดยใช้ iteration method แทน

```
alert( [] == [] );     // false
alert( [0] == [0] );   // false


alert( 0 == [] );      // true
alert('0' == [] );     // false
```

Splice

- การลบ ค่า จาก **array** ทำอย่างไร
- สามารถใช้ **delete**

```
let arr = ["I", "go", "home"];
delete arr[1]; // remove "go"
alert( arr[1] ); // undefined
// now arr = ["I",  , "home"];
alert( arr.length ); // 3
```

- สามารถใช้ **arr.splice** ในการ จัดการได้ หลายอย่าง เช่น เพิ่ม ลบ, แทนค่า

```
arr.splice(start[,delCount, el1,..., elN])
```

```
let arr = ["I", "study", "JS", "now"];

// ลบ 2 elementsแรก
let removed = arr.splice(0, 2);
```

slice

```
arr.slice([start], [end])
```

```
let arr = ["t", "e", "s", "t"];
```

```
alert( arr.slice(1, 3) ); // e,s (copy จาก 1 ถึง 3 ไม่รวม)
```

```
alert( arr.slice(-2) ); // s,t (copy , ตัว จากท้าย)
```

concat เชื่อม

```
arr.concat(arg1, arg2…)
```

```
let arr = [1, 2];

// สร้าง array ใหม่ จากการนำ : arr และ [3,4] มาต่อกัน
alert( arr.concat([3, 4]) ); // 1,2,3,4

// สร้าง array ใหม่ จากการนำ : [3,4] และ [_,`] มาต่อกัน
alert( arr.concat([3, 4], [5, 6]) ); // 1,2,3,4,5,6

// สร้าง array ใหม่ จากการนำ : [3,4] และ _ และ ` มาต่อกัน
alert( arr.concat([3, 4], 5, 6) ); // 1,2,3,4,5,6
```

Iterate: forEach (วน ทำงานไปทีละ element)

```
arr.forEach(function(item, index, array) {
  // ... do something with an item
});
```

```
// ทำการ alert ทุก element
["Bilbo", "Gandalf", "Nazgul"].forEach(alert);
```

Searching îŋđêẏÔğ ľǎṣţÍŋđêẏÔğ îŋçľụđêṣ

```
let arr = [1, 0, false];

alert( arr.indexOf(0) );       // 1
alert( arr.indexOf(false) );   // 2
alert( arr.indexOf(null) );    // -1

alert( arr.includes(1) ); // true
```

```
let fruits = ['Apple', 'Orange',
'Apple']


alert( fruits.indexOf('Apple') );
// 0 (first Apple)
alert( fruits.lastIndexOf('Apple') );
// 2 (last Apple)
```

Searching ğîŋđ ǎŋđ ğîŋđÍŋđêẏ ğîŋđĽǎṣŧÍŋđêẏ

```
arr.find(fn)
let result = arr.find(function(item, index, array) {
  // if true is return กรณีหาเจอ,
     item is return and stopped กรณีหาเจอ
  // for false scenario returns undefined กรณีหาไม่เจอ
});
```

- item
- index
- array

ตัวอย่าง find

```
let users = [
  {id: 1, name: "John"},
  {id: 2, name: "Pete"},
  {id: 3, name: "Mary"}
];

let user = users.find(item => item.id == 1);

alert(user.name); // John
```

ตัวอย่าง ğîŋđÍŋđêy ğîŋđĽắṣŧÍŋđêy คืนค่า index

```
let users = [
  {id: 1, name: "John"},
  {id: 2, name: "Pete"},
  {id: 3, name: "Mary"},
  {id: 4, name: "John"}
];

// หา index แรก ของ John
alert(users.findIndex(user => user.name == 'John')); // 0

// หา index สุดท้าย ของ John
alert(users.findLastIndex(user => user.name == 'John')); // 3
```

filter

```
arr.filter(fn)


let results = arr.filter(function(item, index,
array) {
  // คืนค่า เก็บ ผลลัพธ์ ที่ผ่านการ filter
  // คืนค่า array ว่าง ถ้าหาไม่เจอ
});
```

ตัวอย่าง filter

```
let users = [
  {id: 1, name: "John"},
  {id: 2, name: "Pete"},
  {id: 3, name: "Mary"}
];

// คืนค่า array of ของ , user แรก
let someUsers = users.filter(item => item.id < 3);

alert(someUsers.length); // 2
```

Transform Array

- `map` เรียก `function` ในแต่ละ `element` และ `return` ค่าผลลัพธ์แต่ละตัว
- `sort(fn)` เรียงค่าใน `array`
- `reverse` เรียง กลับด้าน
- `split` แยก
- `join` รวม
- `reduce` เรียก ในแต่ละ `element` คำนวณ แล้ว คืนค่า ค่าเดียว
- `reduceRight` เหมือน `reduce` แต่ทำงาน ขวา ไป ซ้าย

ตัวอย่าง map

```
let lengths = ["John", “Smith", “Michael"].map(item => item.length);
alert(lengths); // 4,5,7
```

ตัวอย่าง sort(fn)

```
let arr = [ 1, 2, 15 ];
arr.sort();
alert( arr );  // 1, 15, 2
```

```
function compareNumeric(a, b) {
  if (a > b) return 1;
  if (a == b) return 0;
  if (a < b) return -1;
}

let arr = [ 1, 2, 15 ];

arr.sort(compareNumeric);

alert(arr);  // 1, 2, 15
```

ตัวอย่าง reverse

```
let arr = [1, 2, 3, 4, 5];
arr.reverse();

alert( arr ); // 5,4,3,2,1
```

ตัวอย่าง split/Join

Split (แยก)

```
let names = 'John, Tom, Jimmy';

let arr = names.split(', ');
```

Join (ต่อ)

```
let arr = ['John', 'Tom', 'Jimmy'];

let str = arr.join(';');
```

ตัวอย่าง reduce ทำกับทุก êĭêņêŋʧ คืนค่าเดียว

```
let arr = [1, 2, 3, 4, 5];

let result = arr.reduce((sum, current) => sum + current, 0);

alert(result); // 15
```

Array.isArray

- ใช้เช็คว่าเป็น Array ไหม เนื่องจากถ้าใช้ type of เช็คค่า array จะคืนค่าเป็น object

- แต่ใช้ Array.isArray จะตอบค่า true
- เช่น

```
alert(typeof {});           // object
alert(typeof []);           // object (same)
alert(Array.isArray({})); // false
alert(Array.isArray([])); // true
```

สรุป Array = ชุดของข้อมูล

To add/remove elements:

```
push(...items) //เร็วกว่า shift, unshift
pop()          //เร็วกว่า shift, unshift
shift()
unshift(...items)
splice(pos, delCount, ...items)
slice(start, end)
concat(...items)
```

To search :

```
indexOf/lastIndexOf(item, pos)
includes(value)
find/filter(func)
findIndex
```

To iterate elements:

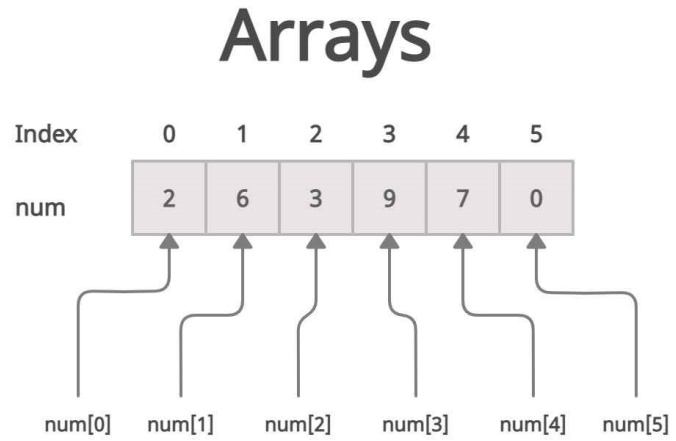
```
forEach(func)
```

To transform :

```
map(func)
sort(func)
reverse()
split/join
reduce/reduceRight(func, initial)
```

Additionally:

```
Array.isArray(value)
```



Data Type #3

JavaScript

Array, Array Method, Iterables

t-live-code