

ទាត់ក្រា

ទេស្ថែលទី១: សេចក្តីថ្លែងការណ៍នៃ JavaScript	1
1. ការដំឡើងការណ៍នៃ JavaScript	1
1.1. ប្រភេទបេស JavaScript	1
1.2. យល់ពី JavaScript Frameworks និង Libraries	1
1.3. អ្វីទៅជាការណ៍នៃ JavaScript?	2
1.4. តើ Java និង Javascript គឺមិត្តភកទេ?	2
1.5. តើ JavaScript អាចធ្វើអ៊ីហានខ្លះ?	2
1.6. ការប្រើប្រាស់ការណ៍នៃ javascript	3
1.7. ការប្រើប្រាស់ Comments ក្នុង JavaScript	3
ទេស្ថែលទី២: ផលិតផល (Variable) និង ប្រភេទការណ៍នៃខ្លះ (Data Type)	5
2. ការបង្កើតរួមផលិតផល (Variable)	5
2.1. ការការដោករោង Variable	5
2.2. ប្រភេទ Variable	6
2.3. ប្រភេទ let Statement	8
2.4. ប្រភេទ Constants	9
2.5. ប្រភេទទិន្នន័យ (Data Types)	10
ទេស្ថែលទី៣: Control flow Statements	13
3. ការគ្រប់គ្រងការណ៍នៃខ្លះ Operators និង ការគំនែកការណ៍នៃខ្លះ Control Flow	13
3.1. ប្រភេទនៃ Arithmetic Operators	13
3.2. ប្រភេទនៃ Comparison Operators	15
3.3. ប្រភេទនៃ Logical Operators	18
3.4. ប្រភេទនៃ Assignment Operators	21
3.5. ប្រភេទនៃ Conditional Operators	23
3.6. ប្រភេទនៃ typeof Operator	24
3.7. ប្រភេទនៃ Nullish Coalescing Operator	28
3.8. ប្រភេទនៃ if...else Statement	29
3.9. ប្រភេទនៃ if...else if... statement	30
3.10. ប្រភេទនៃ Switch Case	31

3.11.	ប្រកែទេន While Loops	33
3.12.	ប្រកែទេន do...while Loop	34
3.13.	ប្រកែទេន For Loop	36
3.14.	ប្រកែទេន For...in Loop.....	37
3.15.	ប្រកែទេន For...of Loop.....	39
3.16.	ប្រកែទេន Loop Control	40
ចំណោមទី៤: Array.....		43
4.	គាន់យុទ្ធសាស្ត្រ Array	43
4.1.	Array Properties	43
4.2.	Array Methods	44
4.3.	Array instance methods	46
4.4.	របៀបប្រកាស One Dimensional Array	59
4.5.	របៀបប្រកាស Multi Dimensional Array.....	60
4.6.	របៀបប្រកាស Associative Array.....	62
ចំណោមទី៥: Functions និង Events.....		64
5.	គាន់បញ្ជីតែន Functions.....	64
5.1.	ការកំណត់ Functions	64
5.2.	ការកំណត់ Function expression	64
5.3.	ការកំណត់ Function parameters	65
5.4.	ការកំណត់ Default Parameters	67
5.5.	ការកំណត់ Function() Constructor	69
5.6.	ការកំណត់ Arrow Functions.....	70
5.7.	ការកំណត់ Variable Scope	71
5.8.	ការប្រើប្រាស់ប្រភេទ Events	73
ចំណោមទី៦: Form Validation		78
6.	គាន់ទំនាក់ទំនើ Form Validation.....	78
6.1.	ការកំណត់លើ Form Validation ដើម្បី Basic Form Validation.....	79
6.2.	ការកំណត់លើ Form Validation ដើម្បី Data Format Validation.....	80
6.3.	ការប្រើប្រាស់ Regular expressions	82
6.4.	ការកំណត់ប្រកែទេន Document Object Model (DOM).....	84

ទេស្ថន៍ទី៧:	ប្រព័ន្ធមូតិក Objects និង Prototypes	87
7.	និមិត្តសញ្ញាណប្រព័ន្ធមូតិក Objects និង Prototypes	87
7.1.	ការបង្កើត Objects	87
7.2.	របៀបប្រើប្រាស់ Object Prototypes	90
7.3.	របៀបប្រើប្រាស់ Object Properties	93
7.4.	របៀបប្រើប្រាស់ Object Accessors	95
ទេស្ថន៍ទី៨:	JavaScript Classes	96
8.	នៃលានំពីចំណើនការនៃ Object Oriented	96
8.1.	ការកំណត់ JavaScript Classes	97
8.2.	ស្វាល់ពី Encapsulation	100
8.3.	ស្វាល់ពី Inheritance	102
8.4.	ស្វាល់ពី Abstraction	105
8.5.	ស្វាល់ពី Polymorphism.....	107

ទេស្ថែនទី១:

កែវគមន៍ JavaScript

1. ការកែវគមន៍ JavaScript

JavaScript (JS) រួចជា **client site script** និង **light-weight object-oriented programming language** មួយដែលពេញនិយម ដែលមានសម្រាកាតធ្វើឱ្យគេហទំនើមភាពបច្ចនុ (Dynamic) និង **Interactive** នៃគេហទំនើម អ្នកអភិវឌ្ឍន៍គេហទំនើមអស់ (Developers) តែងតែប្រើប្រាស់ Javascript ដើម្បីបន្ថែម animations, pop-up windows, search bars, buttons, audio and video, chat widgets និងបន្ថែម interactive elements ផ្សេងៗទៀតចូលទៅក្នុង Webpage មួយ។ វាបានត្រូវការ Server ដើម្បីធ្វើការ Compile ក្នុងរបស់វានោះទេ វាអាចដំណឹករាលើ Browser តែម្នាច់ហើយជាទុទេវារដ្ឋីការ (Support) ជាមួយ Browser ដូចជា : Internet Explorer, Firefox, Chrome, Opera, Safari និង browsers ជាប្រើប្រាស់ផ្សេងទៀត។

1.1. ប្រវត្តិបស់ JavaScript

JavaScript ត្រូវបានបង្កើតដោយ លោក Brendan Eich ដែលបំផើករនវិក្សុងក្រុមហ៊ុន Netscape ដែលបង្កើតឡើង និងបានគេហទំនើមជាក់ល្អោះថា Mocha ប៉ុន្មោះក្រុមហ៊ុន Mochikit ត្រូវបានបង្កើតឡើង និងបានគេហទំនើម LiveScript បន្ទាប់មកទៀត ក្រុមហ៊ុន Netscape និង Sun Microsystems បានសហការគ្នា ក្នុងការអភិវឌ្ឍន៍ LiveScript រួចកំណត់បានបន្ទាប់ឡើងដោយ JavaScript នៅថ្ងៃខែ ឆ្នាំ 1995។

1.2. យល់ពី JavaScript Frameworks និង Libraries

មកដើងពី **JavaScript Framework** ដែលពេញនិយមបំផុតដែលអ្នកអាចប្រើប្រាស់ម្រាប់ការអភិវឌ្ឍន៍គេហទំនើម ការអភិវឌ្ឍន៍កម្មវិធី ការអភិវឌ្ឍន៍ដែកខាងក្រោម (backend) ។ លើនេះ ដែលវាកែតបច្ចុប្បន្ននៃ Code Javascript មានដូចជា : React, Angular, Vue.js, Ember.js, Node.js, Backbone.js, Next.js, Mocha, Meteor.js, Polymer ។

នៅក្នុង **JavaScript Libraries** មានសំណុំមុខងារដែលងាយស្រួលឱ្យយើងអាចកំណត់ទុកមុន (pre-defined functions, classes, methods, objects) ជាប្រើប្រាស់ដែលអាចប្រើប្រាស់តាមរយៈ CND បូកដើម្បី Download មកដើលក្នុង local web រួចដាក់លើក្នុងបញ្ជី Libraries ដែលពេញនិយមបំផុត ដែលអ្នកអភិវឌ្ឍន៍គេហទំនើម (Developer) យកមកប្រើប្រាស់មានដូចជា : jQuery, Axios, Chart.js, D3.js, Socket.io, Underscore.js, Lodash, Three.js ជាដើម។

1.3. អ្នកដោ JavaScript?

- JavaScript ត្រូវបានគេប្រើប្រាស់វាដើម្បីធ្វើការដោម្បួយ HTML
- JavaScript ជាការសាជែលសរសេរដោយ Script
- Javascript ជាការសាជែលស្ថាបនិយោគនឹង Programming Tools ម្នាយដែលដាយស្ថាបនឹង និងដាយស្ថាបនឹងកែវប្រើប្រាស់
- ទីតាំងរបស់ JavaScript ជាទូទៅស្ថិតនៅក្នុង HTML
- JavaScript ជាការសាជែលដំណើរការដោយខ្លួនដង (មាននំយច្ចាត់ដំណើរការដោយមិនមានអ្នកបកប្រើប្រាស់ compilation)
- អ្នករាល់គ្មានប្រើប្រាស់ JavaScript បានដោយមិនមានការទិញកម្មសិទ្ធិបញ្ហានៅ (Open Source)

1.4. តើ Java និង Javascript ដូចគ្នាទេ?

ស្មូកកំមានយល់ប្រលៃគ្មានរាង Java និង Javascript មិនដូចគ្នាទេ។

- Java (គីបអីតុឡើងដោយ Sun Microsystems) រាជការសាជែលស្ថាបនឹង C, C++ ដើរ ... ។
- JavaScript គីជាឌ Scripting Language ម្នាយសម្រាប់គេហទំនាក់ទំនាក់ គីមិនត្រូវការ Server ដើម្បីធ្វើការ Compile ក្នុងរបស់រាយការណ៍ រាជការសាជែលក្នុងការបញ្ចូលក្នុងរបស់ Javascript នៅក្នុង HTML បានដើរ។

1.5. តើ JavaScript អាចធ្វើអ្នកបានខ្លះ៖?

- JavaScript បានផ្តល់សិទ្ធិឱ្យ HTML សម្រាប់ធ្វើការបែន (animation) ។ បើនិយាយឱ្យបំទេ HTML ជាអ្នកនិពន្ធ មិនមែនជាអ្នកសរសេរកម្មវិធីទីផ្សេយ ប៉ុន្តែ JavaScript ជាការសាជែលដែលមានរូបមន្តសាមញ្ញ ជាទូទៅនរណាក៏ដាយស្ថាបនឹងការបញ្ចូលក្នុងរបស់ Javascript នៅក្នុង HTML បានដើរ។
- JavaScript អាចបញ្ចូល dynamic text នៅក្នុង HTML ។ JavaScript មាន statement : `document.write(" <h1>" + name + " </h1>")` បើយកអាចសរសេរអេឡិតិតិតិ (variable) ទៅឱ្យ text នៅក្នុង HTML បានទៀតដែរ។
- JavaScript អាចកែតមានព្រឹត្តិការណី (events) តបទីផ្សេង ។ JavaScript អាចដាក់ឱ្យមានប្រតិកម្ម (execute) នៅពេលមានអ្នកដោយក្នុង។ ដូចជានៅពេលដែលបញ្ចប់យើងនឹងអាចបានសារ (alert message)ប្រាប់ទៅអ្នកប្រើប្រាស់ (user) បានឱ្យបុច (click) ទៅលើជាតិរបស់ HTML ណាម្បយ ... ។
- JavaScript អាច Read និង Write ឈាតុរបស់ HTML ។ JavaScript អាចអាន និងបញ្ចូរមាតិកា (content) ឈាតុរបស់ HTML បាន។
- JavaScript អាចធ្វើឱ្យទិន្នន័យបានមុនពេលដែលទិន្នន័យរបស់យើងបញ្ចូនទៅកាន់ (Server) ។

- JavaScript អាចប្រើដើម្បីកប់ចំនួនអ្នកទស្សនា (visitor's) នៅលើ browser បានទៀតដឹង ... ។

1.6. ការប្រើប្រាស់ភាសា javascript

- ✓ ដើម្បីសរសេរកូដ JavaScript នៅក្នុងទំព័រ HTML គឺយើងប្រើប្រាស់ tag មួយដែលមានឈ្មោះថា និងត្រូវប្រើ attribute type="text/javascript" ដាច់ម្រាង Internal Javascript

```

1 <html>
2   <head>
3     <title>The JavaScript Example</title>
4   </head>
5   <body>
6     <script language = "javascript" type = "text/javascript">
7       document.write("Hello Web Development-JavaScript!")
8     </script>
9   </body>
10  </html>

```

- ✓ document.write គឺជាកូដបែស់ JavaScript មួយដែលវាមានត្តុនាទីសម្រាប់ឡើយឱ្យអាចសរសេរអ្នកបទដៃង់បានម្រាង ដោយប្រើប្រាស់ HTML Tag សម្រាប់បង្ហាញនៅលើ Browser ។
- ✓ ការសរសេរកូដ JavaScript នៅខាងក្រោម ទំព័រ HTML ដាច់ម្រាង External Javascript ប្រសិនបើអ្នកចង់ដើរការកូដ JavaScript ដូចម្នាន់នៅលើទីតាំងរបីនេះ ដោយមិនមានការសរសេរកូដដែលប្រគល់ទំនួរ អ្នកអាចសរសេរកូដ JavaScript ទាំងនេះនៅក្នុង file ដាច់ដោយទៅកម្ពុយ ដោយ file នៅៗ មាន extension .js ។ បើសិនជាបានប្រើប្រាស់ JavaScript file នៅៗ ត្រូវប្រើប្រាស់ tag ដោយមាន attribute src ហើយតាំងលើបែស់ attribute src គឺជាគុណភាពរបស់ JavaScript file ។

```

1 <html>
2   <head>
3     <title>The JavaScript Example</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <script language = "javascript" type = "text/javascript">
8       document.write("Hello Web Development-JavaScript!")
9     </script>
10    </body>
11  </html>

```

1.7. ការប្រើប្រាស់ Comments ក្នុង JavaScript

- JavaScript Comment គឺគិតប្រើដើម្បីពិពណ៌នាកូដ JavaScript បែស់យើង ដើម្បីផ្តល់ការណាយស្រួល ដល់យើងក្នុងការមេល និងធ្វើការកំក្របផ្តើមដៃង់ក្នុងការប្រើប្រាស់ Comment ទាំងអស់ Browser និងបានទូទាត់ការប្រើប្រាស់ JavaScript នៅទីនេះ។
- ✓ Single Comment

Single Comment យើងប្រើសមាប់ពិពណ៌នាក្នុងបស់យើង ប្រសិនបើ Comment

បស់យើងមានតើ មួយបន្ទាប់។ ដើម្បី Comment យើងប្រើសញ្ញា //

```

1 <html>
2   <head>
3     <title>The JavaScript Example</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <script language = "javascript" type = "text/javascript">
8       //Welcome to javascript using single comment
9       document.write("Hello Web Development-JavaScript!")
10      </script>
11    </body>
12  </html>

```

✓ Multiline Comment

យើងប្រើ Multiline Comment ដើម្បីពិពណ៌នាក្នុងដែលការពិពណ៌នាបស់យើង មានចំនួនបន្ទាត់ ឬ ធ្វើការបិទក្នុងកំឡូងដែលការដោដីមាន ដើម្បី Comment យើងចាប់ផ្តើមដោយសញ្ញា /* និងបញ្ចប់ដោយ សញ្ញា */

```

1 <html>
2   <head>
3     <title>The JavaScript Example</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">Introduction to JavaScript</h1>
8
9     <script language = "javascript" type = "text/javascript">
10    /*
11      CreatedBy:.....
12      CreatedDate:.....
13      ContactInfo:.....
14      Description:.....
15    */
16    document.getElementById("display").innerHTML ="Welcome to JavaScript!";
17    document.write("Hello Web Development-JavaScript!")
18
19   </script>
20 </body>

```

សេចក្តីផ្តើមតាមប្រភេទ (Variable) និង ប្រភេទផលិតផល (Data Type)

2. តាមប្រភេទតាមប្រភេទ (Variable)

Variable ដែលប្រជាកាសាច្នៃរបា «អប់រំ» មានគុណភាពធមិនស្មើទេសត្រាតីអប់រំនៅក្នុងគណិតវិទ្យាទេ នៅលើគឺត្រូវបានគេប្រើសម្រាប់ក្រុងកត្តិទឹនយុទ្ធយុទ្ធយ។ បើនេះអ្នីដែលពិសេសនោះគឺ **Variable** នៅក្នុងកាសាកំពុងរោង ហើយអាចរក្សាទុកទឹនយុទ្ធយប្រើប្រាស់បានទេ មិនមែនត្រឹមតែទឹនយុទ្ធយដាប់នូននោះទេ។

2.1. ការកែដាក់លួយដាក់: Variable

ការកែដាក់លួយដាក់អប់រំនៅក្នុងកាសា Javascript គឺត្រូវធ្វើឡើងដោយគោរពទៅតាមច្បាប់ដែលបានកំណត់ដូចខាងក្រោម៖

- ដើម្បីធ្វើការប្រកាសអញ្ជាត់នៅក្នុង JavaScript គឺយើងប្រើប្រាស់ **var**, **let**, **const** នៅពីមុខអញ្ជាត់របស់យើង។
- លួយដាក់មួយអាចតាំងឈានទូរអប់រំមួយបុង្ញារៈ។ ដូច្នេះលួយដាក់អប់រំមិនអាចដាក់ជាន់ត្រាបានទេ។
- លួយដាក់អប់រំដាការបន្ថែមតាមរយៈក្រុងអក្សរ(អង់គ្លេស) លេខ សញ្ញា **Underscore** និងសញ្ញា **\$**។
- លួយដាក់អាចបាប់ដើមដោយសញ្ញា **\$** និង **_** (បើនេះគឺមិនស្មើរបស់ប្រើប្រាស់ទេ)។
- លួយដាក់ដែលប្រើប្រាស់អក្សរដែលមិនមែនត្រឹមតែលួយដាក់អក្សរតូច (Case sensitive)។
- លួយដាក់មិនត្រូវដាក់ជាន់នឹងពាក្យតូចនៃកាសា Javascript ទេ (if else new class ...)។

- ✓ អ្នកអាចអនុវត្តតាម **syntax** ខាងក្រោមដើម្បីប្រកាសអប់រំដោយមិនប្រើពាក្យតូចនៃកាសា Javascript ទេ

```

1 <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">Introduction to Variable-DataType</h1>
8
9     <script language = "javascript" type = "text/javascript">
10       Money = 100;
11       Name = "Tutorials-JavaScript";
12     </script>
13   </body>
14 </html>
```

- ✓ អ្នកអប់រំពីពាក្យ var keyword ដើម្បីប្រកាសអប់ដូចបង្ហាញខាងក្រោម

```

1 <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">Introduction to Variable-DataType</h1>
8
9     <script language = "javascript" type = "text/javascript">
10    var money;
11    var name;
12    // or
13    var money, name;
14  </script>
15  </body>
16 </html>

```

- ✓ ការចាប់ផ្តើមអប់រំ(Variable Initialization)ដោយប្រើ Assignment Operator សម្រាប់ប្រើប្រាស់នៅពេលក្រាយបាន

```

1 <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">Introduction to Variable-DataType</h1>
8
9     <script language = "javascript" type = "text/javascript">
10    var name = "Alex";
11    var money;
12    money = 2000.50;
13  </script>
14  </body>
15 </html>

```

2.2. ប្រភេទ Variable

- ✓ Variable (អប់រំ) ត្រូវបានប្រើដើម្បីក្រឡុកទិន្នន័យដែលអាចធ្វើសម្រេចនៅពេលក្រាយ។
- ✓ អាបង្រួយកត្តម្លៃជាប្រភេទ number, string, boolean, object, array ជាដឹងម។

```

1 <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">Introduction to Variable-DataType</h1>
8
9     <script language = "javascript" type = "text/javascript">
10    var num = 765; // Number
11    var str = "Welcome"; // String
12    var bool = false; // Boolean
13    const cars = ["Tesla", "Volvo", "BMW"]; // Array
14    const car = {type:"SUV", model:"X", color:"white"}; //object
15  </script>
16  </body>
17 </html>

```

✓ តម្លៃអប់រំដែលមិនបានកំណត់ (Undefined Variable Value)

```

1 <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">Introduction to Variable-DataType</h1>
8
9     <script language = "javascript" type = "text/javascript">
10       var num;
11       document.write("The value of num is: " + num + "<br/>");
12     </script>
13   </body>
14 </html>

```

✓ □វិសាលភាពអប់រំ (Variable scope)

- **Global Variables** – មាននំយច្ចាវអាបត្រូវបានកំណត់យកទៅប្រើគ្រប់ទីកន្លែងនៅក្នុងកូដ JavaScript បែសអ្នក។
- **Local Variables** – ត្រូវបានកំណត់យកទៅប្រើនៅក្នុង Function

```

1 <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">Introduction to Variable-DataType</h1>
8
9     <script language = "javascript" type = "text/javascript">
10       var myVar = "global"; // Declare a global variable
11       function checkscope( ) {
12         var myVar = "local"; // Declare a local variable
13         document.write(myVar);
14       }
15     </script>
16   </body>
17 </html>

```

✓ ការប្រកាសអញ្ជាត់ ដោយមិនប្រើបាស់ var keyword

- នៅពេលដែលយើងកំណត់អប់រំដោយមិនប្រើពាក្យគន្លឹះណាមួយ JavaScript បាត់ទុក អប់រំទាំងនោះជាប្រភេទ global variables ហើយអាបប្រើគ្រប់ទីកន្លែងនៅក្នុងកូដ។

```

1 <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">Variables without var keyword</h1>
8
9     <script language = "javascript" type = "text/javascript">
10       name = "Hello javascript"; // String type variable
11       number = 10.25; // Number type variable
12       document.write("name = " + name + ", number = " + number + "<br>"); 
13     </script>
14   </body>
15 </html>

```

- ឧបាទរណីខាងក្រោម យើងបានប្រកាសអថេរ age ហើយចាប់ផ្តើម (initialized) វាដាមួយ 10 ។ ជាដូចមានឡើត យើងបានប្រកាសអថេរ age បុន្ថែមនាំបានចាប់ផ្តើម (haven't initialized) វាទេ ។ ទៅយើងណាក់ដោយ វានឹងបង្ហាញឡើងម៉ែ 10 នៅក្នុងលទ្ធផលព្រះវា មិនបាត់បង្កែតម៉ែនការចាប់ផ្តើមពីមុនទេ ។ ទៅយើងណាក់ដោយ ប្រសិនបើយើងធ្វើបច្ចុប្បន្នភាពកំម៉ែនអថេរ age នោះ វាអធ្វើបច្ចុប្បន្នភាពដោយដោតជំយ៉ា ។

```

1 <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">Variables with var keyword</h1>
8
9     <script language = "javascript" type = "text/javascript">
10    var age = 10;
11    var age;
12    document.write("age = " + age + "<br>");
13
14  </script>
15 </body>
</html>

```

age = 10

2.3. ប្រភេទ let Statement

- JavaScript let statement ត្រូវបានប្រើដៃមីរប្រកាសអថេរ ។ ជាមួយនឹង let statement យើងអាចប្រកាស variable ដែលត្រូវបាន block-scoped ។ នេះមាននំយប់អថេរ ដែលបានប្រកាសជាមួយ let គឺអាចចូលប្រើបានតែនៅក្នុងបុក្រាណក្នុងដែលវាត្រូវបានកំណត់ ។
 - let keyword ត្រូវបានណែនាំនៅក្នុង ECMAScript ហេកាត់ថា ES6 (2015) នៃ JavaScript វាកីជាដើរមីសជំនួស var keyword
- ✓ ការប្រកាសអថេរជាមួយ let statement

```

1 <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">let statement</h1>
8
9     <script language = "javascript" type = "text/javascript">
10    let name = "John";
11    let age = 35;
12    let x = true;
13
14  </script>
15 </body>
</html>

```

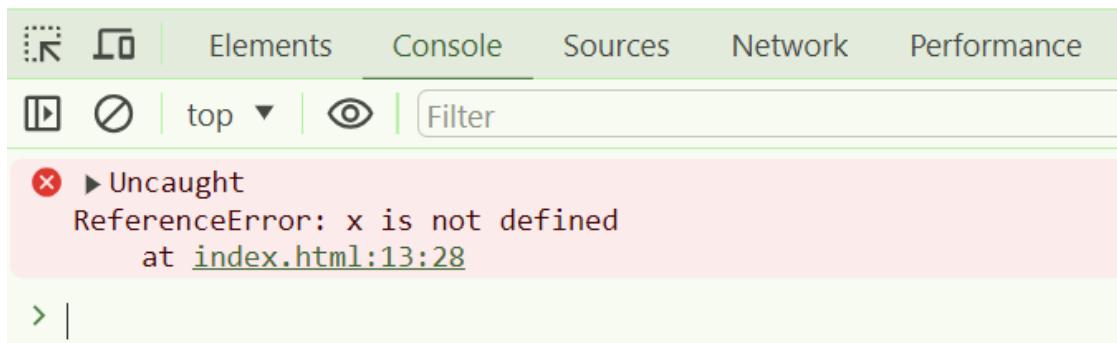
✓ វារីដំនឹង Block Scope និង Function Scope

- **Block Scope:** ជាស្ថាប់ដែលបានប្រកាសដោយប្រើ **let statement** គឺជាពីរ **block-scope** ។ វាមាននូយចំណាំប្រសិនបើអ្នកកំណត់អប់រំដោយប្រើ **let keyword** អនុញ្ញាតខ្លួនទៅក្នុងបុកជាក់លាក់ អ្នកអាចចូលប្រើអប់រំនៅខាងក្នុងបុកជាក់លាក់នៅទេដែលបានប្រើបាន ហើយប្រសិនបើអ្នកព្យាយាមចូលប្រើអប់រំនៅខាងក្រោមបុក វាបានផ្តល់ព័ត៌មានថា 'variable is not defined' .

```

1 <html>
2   <head>
3     <title>Variable-DatType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7     <h1 id="display">let statement</h1>
8
9     <script language = "javascript" type = "text/javascript">
10    {
11      let x = "John";
12    }
13    document.write("x = " + x + "<br/>");
14    //here x can't be accessed
15  </script>
16 </body>
17 </html>

```



2.4. ប្រភេទ Constants

- **Constants** គឺជាមួយប្រភេទដែលតម្លៃកមិនមានការផ្តល់បញ្ជីពីក្នុងបុកជាក់លាក់ នៃកម្មវិធី ។ អ្នកអាចប្រកាស **constants** ដោយប្រើ **const keyword** ។
 - **const keyword** ត្រូវបានណែនាំនៅក្នុង ES6 version នៃ JavaScript
- ✓ របៀបប្រកាស **Constants**
- អ្នកតែងតែត្រូវកំណត់តម្លៃនៅពេលប្រកាស ប្រសិនបើអប់រំត្រូវបានប្រកាសដោយប្រើ **const keyword** ។
 - មិនអាចធ្វើការ **Reassigned** តម្លៃបានទេ អ្នកមិនអាចធ្វើកំហែតម្លៃនៃអប់រំដែលបានប្រកាសដោយប្រើ **const keyword**បានទេ ។

```

1 <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7
8     <h1 id="display">let statement</h1>
9
10    <script language = "javascript" type = "text/javascript">
11      const x = 10; // Correct Way
12
13      const y; // Incorrect way
14      y = 20;
15    </script>
16
17   </body>
18 </html>

```

2.5. ប្រភេទទិន្នន័យ (Data Types)

- នៅក្បាង JavaScript ប្រភេទទិន្នន័យ (Data Types) សំដើលើប្រភេទនេះតម្លៃដែលយើងកំពុងរក្សាទុកប្រធើការជាមួយ។ ប្រភេទទិន្នន័យជាពីរ primitive data types

- **Strings** of text e.g. "This text string" etc.
- **Numbers**, eg. 123, 120.50 etc.
- **Boolean** e.g. true or false.
- **null**
- **undefined**
- **BigInt**
- **Symbol**

- ប្រភេទទិន្នន័យជាបីទីប្រភេទ (composite data type) ជាពីរ Object មានប្រភេទ៖

- **Object**
- **Array**
- **Date**

- ហេតុអ្នគានជាប្រភេទទិន្នន័យមានសារ៖សំខាន់?

នៅក្បាងកាសាសរសរកម្មវិធីណាមួយ ប្រភេទទិន្នន័យមានសារ៖សំខាន់សម្រាប់ការរៀបចំប្រតិបត្តិការ (operation manipulation)។

- **Function Scope:** យើងបានកំណត់អប់រំ x ដោយប្រើ let និង y ដោយប្រើ var ។ ដូច្នោនេះដឹងដឹងបានកំណត់តម្លៃ 10 និង 20 ទៅអប់រំទាំងពីរ

- យើងបានកំណត់មុខងារ **test()** ប្រកាសឡើងវិញនូវអប់រំ x និង y នៅក្នុងវា
ហើយចាប់ផ្តើមពួកវាដាមួយនឹងតម្លៃ 50 និង 100 ដោយត្រា។
យើងបង្ហាញតម្លៃអប់រំនៅខាងក្នុងមុខងារ ហើយវាបង្ហាញលេខ 50 និង 100 ដូចដែលវាដ្ឋាល់នៅ **local variables** ទៅស្ថិត **global variables**.

```

1  <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7
8     <h1 id="display">let statement</h1>
9
10    <script language = "javascript" type = "text/javascript">
11      let x = 10;
12      var y = 20;
13      function test() {
14        let x = 50;
15        var y = 100;
16        document.write("x = " + x + ", y = " + y + "<br/>");
17      }
18      test();
19    </script>
20
21   </body>
22 </html>

```

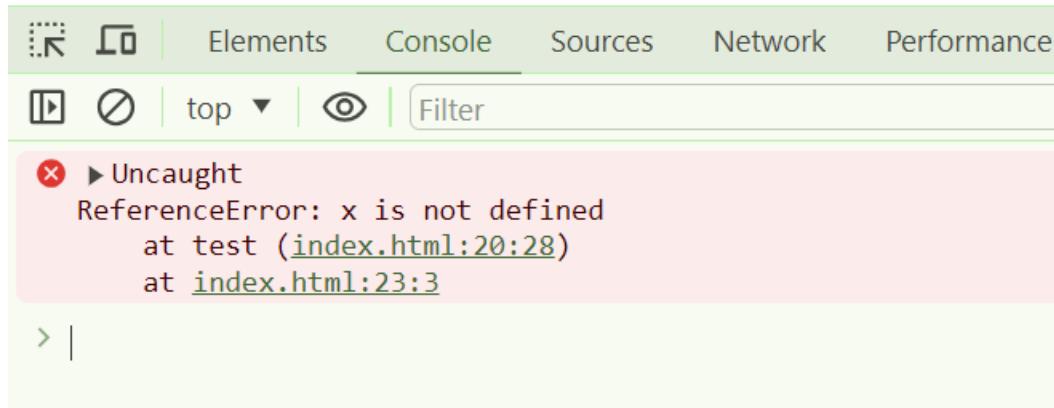
- យើងបង្ហាញតម្លៃនៃអប់រំ x និង y នៅខាងក្នុងបុក 'if' ។ យើងមិនអាចចូលប្រើអប់រំ 'x'
នៅខាងក្រោមបុក 'if' ដូចនឹង **blocked scope** បុន្ថែយើងអាចចូលប្រើអប់រំ y នៅខាងក្រោមបុក 'if'
និងនៅខាងក្នុងបុកមុខងារ ដោយសារ y យើងប្រើប្រាស់ **var keyword**

```

1  <html>
2   <head>
3     <title>Variable-DataType</title>
4     <script type="text/javascript" src="javascript/file_name.js"></script>
5   </head>
6   <body>
7
8     <h1 id="display">let statement</h1>
9
10    <script language = "javascript" type = "text/javascript">
11      function test() {
12        let bool = true;
13        if (bool) {
14          let x = 30;
15          var y = 40;
16          document.write("x = " + x + ", y = " + y + "<br/>");
17        }
18        document.write("y = " + y + "<br/>");
19
20        // x can't be accessible here (x is not defined)
21        document.write("x = " + x + "<br/>");
22
23      }
24      test();
25    </script>
26
27   </body>
28 </html>

```

```
x = 30, y = 40  
y = 40
```



សេចក្តីពណ៌:

Control flow Statements

3. គារប្រើប្រាស់ប្រភេទ Operators និង គារអំណែត់លក្ខខណ្ឌនៃប្រើប្រាស់ Control Flow

នៅក្នុង JavaScript Operator គឺជានិមិត្តសញ្ញាណដែលធ្វើប្រតិបត្តិការណើ **operand** មួយ ប្រចើនដូចជាអប់រំ (variables) ប្រតិបត្តិការណើ **value** ហើយវានឹង **return** លទ្ធផលមួយ។ ឧបាទរណ៍បង្ហាញសមញ្ញមួយ $4 + 5$ ស្មើនឹង 9 ។ នៅទីនេះ 4 និង 5 គ្រឿងបានគេហោថា **operands** ហើយ $'+'$ គ្រឿងបានគេហោថា **operator** ។ ខាងក្រោមនេះជាប្រភេទនៃ Operators

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Bitwise Operators
- Assignment Operators
- Miscellaneous Operators

3.1. ប្រភេទនៃ Arithmetic Operators

- **Arithmetic operators** ជាការប្រើប្រាស់សញ្ញាប្រមាណដើម្បីនួនពារិនិត្យដោបីនូចជាការបូក (+) ដឹក (-) គូណា (*) និងចែក (/)
- យើងបានផ្តល់តារាងមួយដែលមានសញ្ញាប្រមាណដើម្បីគឺតិចឡាយ និងពន្យល់ពីមុខដារបស់ប្រតិបត្តិការនេះមួយ។

Operator	Name	Description
+	Addition	Adds two operands
-	Subtraction	Subtracts the second operand from the first
*	Multiplication	Multiply both operands
/	Division	Divide the numerator by the denominator
%	Modulus	Outputs the remainder of an integer division
++	Increment	Increases an integer value by one
--	Decrement	Decreases an integer value by one

✓ Addition operator (+)

```
let netPrice    = 9.99,  
    shippingFee = 1.99;  
let grossPrice = netPrice + shippingFee;  
  
console.log(grossPrice);
```

Output:

```
11.98
```

✓ Subtraction operator (-)

```
let result = 30 - 10;  
console.log(result); // 20
```

✓ Multiplication operator (*)

```
let result = '5' * 2;  
  
console.log(result);
```

Output:

```
10
```

✓ Divide operator (/)

```
let result = 20 / 10;  
  
console.log(result); // 2
```

✓ Arithmetic operators with objects

```

let energy = {
  valueOf() {
    return 100;
  },
};

let currentEnergy = energy - 10;
console.log(currentEnergy);

currentEnergy = energy + 100;
console.log(currentEnergy);

currentEnergy = energy / 2;
console.log(currentEnergy);

currentEnergy = energy * 1.5;
console.log(currentEnergy);

```

Output:

```

90
200
50
150

```

3.2. ប្រកួតនៃ Comparison Operators

- Comparison Operators នេះគឺជា JavaScript ប្រព័បណ្តុះដោយត្រឡប់តម្លៃ Boolean ទាំងពីត បូមិនពីត ដោយផ្តល់លទ្ធផលប្រព័បណ្តុះដោយទៅលាស់។ ឧបាទរណ៍ យើងអាចប្រើ comparison operators ប្រព័បណ្តុះដោយមីតិនិភ័យមែនបានក្នុង Operator ពីរស្មើប្រចាំថ្ងៃ។

Operator	Description	Example
<code>==</code>	Equal	<code>x == y</code>
<code>!=</code>	Not Equal	<code>x != y</code>
<code>===</code>	Strict equality (equal value and equal type)	<code>x === y</code>
<code>!==</code>	Strict inequality (not equal value or not equal type)	<code>x !== y</code>
<code>></code>	Greater than	<code>x > y</code>
<code><</code>	Less than	<code>x < y</code>
<code>>=</code>	Greater than or Equal to	<code>x >= y</code>
<code><=</code>	Less than or Equal to	<code>x <= y</code>

✓ **Equality (==) Operator**

ពិនិត្យមែលបាត់តែម្នាស់នៃ operand ពីរគឺស្មើគ្មាប់អត់ វិញ return ពិត ប្រសិនបើ operands ស្មើគ្មាប់ បើមិនដូច្នេះទេ វិញ return មិនពិត ។ ប្រសិនបើ operands មានប្រភេទ Type ផ្សេងៗគ្នា វាដែលវិរាងការការបំប្លែងប្រភេទ Type ហើយបន្ទាប់មករៀបចំបានដោយបានដោល operands ។

```

10  |<script language = "javascript" type = "text/javascript">
11  |    //here a,b called operand
12  |    const a = 10;
13  |    const b = 20;
14  |    a == 10; //true
15  |    a == b; // false
16  |    "Hello" == "Hello"; // true
17  |</script>
```

✓ **Inequality (!=) Operator**

ពិនិត្យប្រសិនបើតែម្នាស់នៃ operand ពីមិនស្មើគ្មាប់ វិញ return ពិត ប្រសិនបើ operands មិនស្មើគ្មាប់ បើមិនដូច្នេះទេ វិញ return មិនពិត ។

```

8   |<div id="output"></div>
9   |<script language = "javascript" type = "text/javascript">
10  |    const a = 10;
11  |    const b = 20;
12  |    let result = (a != b);
13  |    document.getElementById("output").innerHTML = "(a != b) => " + result;
14  |</script>
```

✓ **Strict Equality (===) Operator**

ពិនិត្យបាត់តែម្នាស់(value) និងប្រភេទទិន្នន័យ(data type)នៃ operand ទាំងពីរគឺស្មើគ្មាប់អត់ វិញ return ពិត ប្រសិនបើ operand ទាំងពីរស្មើគ្មាប់ និងប្រភេទដូចគ្នា ។

```

8   |<div id="output"></div>
9   |<script language = "javascript" type = "text/javascript">
10  |    const a = 10;
11  |    const b = 20;
12  |    let result = (a === b);
13  |    document.getElementById("output").innerHTML = "(a === b) => " + result;
14  |</script>
15  |<p> Set the variables to different values and then try...</p>
```

✓ **Strict Inequality (!==) Operator**

ពិនិត្យមែលបាត់ operand ទាំងពីរមិនស្មើគ្មាប់នៅក្បងតែម្នាស់(value) បុប្រភេទ(type) ។ វិញ return ពិត ប្រសិនបើ operands មានប្រភេទដូចគ្នា បុន្ថែមិនស្មើគ្មាប់ បុណ្ណាមានប្រភេទផ្សេងៗគ្នា ។

```

10  | 10 !== 10; //returns false
11  | 10 !== 20; // returns true
12  | 'Hello'!=='Hello'; // returns false
13  | 10 !== '10'; //return true
14  | 0 !== false; //returns true
```

```

8   |<div id="output"></div>
9   |<script language = "javascript" type = "text/javascript">
10  |    const a = 10;
11  |    const b = 20;
12  |    let result = (a !== b);
13  |    document.getElementById("output").innerHTML = "(a !== b) => " + result;
14  |</script>
```

✓ **Greater Than (>) Operator**

ពិនិត្យមេលបាត់តែម្វោន៍ **operand** ខាងឆ្លងគីជំដាចកម្មនៃ **operand** ស្ថា ។ ប្រសិនបើ Yes និង return ពីត បើមិនដូច្នះទេ និង return មិនពីត ។

```

8   <div id="output"></div>
9
10 <script language = "javascript" type = "text/javascript">
11     const a = 10;
12     const b = 20;
13     let result = (a > b);
14     document.getElementById("output").innerHTML = "(a > b) => " + result;
15
16 </script>
<p> Set the variables to different values and then try...</p>
```

✓ **Than or Equal (>=) Operator**

ពិនិត្យមេលបាត់តែម្វោន៍ **operand** ខាងឆ្លងគីជំដាច បុស្ថឹនឯងកម្មនៃ **operand** ស្ថា ។ ប្រសិនបើ Yes និងត្រួតពីត បើមិនដូច្នះទេ មិនពីត ។

```

8   <div id="output"></div>
9
10 <script language = "javascript" type = "text/javascript">
11     const a = 10;
12     const b = 20;
13     let result = (a >= b);
14     document.getElementById("output").innerHTML = "(a >= b) => " + result;
15
16 </script>
<p> Set the variables to different values and then try...</p>
```

✓ **Less Than (<) Operator**

return ពីត ប្រសិនបើតម្លៃនៃ **operand** ខាងឆ្លងគីជំដាចកម្មនៃ **operand** ស្ថា បើមិនដូច្នះទេ និង return មិនពីត ។

```

9   <script language = "javascript" type = "text/javascript">
10    10 < 20; // true
11    5 < 5; // false
12    "ab" < "aa"; // true
13    10 < '5'; // false
14
15 </script>
```

```

8   <div id="output"></div>
9
10 <script language = "javascript" type = "text/javascript">
11     const a = 10;
12     const b = 20;
13     let result = (a < b);
14     document.getElementById("output").innerHTML = "(a < b) => " + result
15
16 </script>
<p> Set the variables to different values and then try...</p>
```

✓ Than or Equal (\leq) Operator

ពិនិត្យមេលបាត់តិចម៉ែន operand ខាងឆ្វេងគឺបានដឹងថា a ប្រសិនបើ Yes នៅលក្ខខណ្ឌនឹងភាពយករាជី។

```

8 | <div id="output"></div>
9 | <script language = "javascript" type = "text/javascript">
10|   const a = 10;
11|   const b = 20;
12|   let result = (a <= b);
13|   document.getElementById("output").innerHTML = "(a <= b) => " + result;
14|
15| </script>
16| <p> Set the variables to different values and then try...</p>

```

✓ Comparing null, undefined and NaN

នៅក្បាច់ JavaScript, null, undefined និង NaN

គឺជាតម្លៃក្នុងភាពយកដែលមិនត្រូវបានបំប្លែងទៅជាសូលី (0) សម្រាប់ការប្រៀបធៀប។

```

9 | <script language = "javascript" type = "text/javascript">
10|   0 == null; // returns false
11|   0 == undefined; // returns false
12|   0 == NaN; // returns false
13|
14|   //null and undefined are weakly equal.
15|   null == undefined; // returns true
16|   null === undefined; // returns false
17|
18|   //The type of NaN is number but it is not equal to zero
19|   NaN == NaN; // returns false
20|
21| </script>

```

3.3. ប្រភេទ Logical Operators

- នៅក្បាច់ JavaScript ជាទុទេត្រូវបានប្រើជាមួយ operands ជាមួយ Boolean ហើយ return តម្លៃ boolean។ ប្រភេទនេះ logical operators មានបីប្រភេទទេ: **&&** (AND), **||** (OR), and **!** (NOT)
- Operators ទាំងនេះត្រូវបានប្រើដើម្បីគ្រប់គ្រងលំហោកម្មវិធី។

Operator	Description	Example
&&	Logical AND	(x && y) is false.
 	Logical OR	(x y) is true.
!	Logical NOT	!(x) is false.

✓ Logical AND (**&&**) Operator

The logical AND (**&&**) operator វាយតម្លៃប្រកិបត្ថិករាជីថា តើឆ្វេងទៅស្ថា។ ប្រសិនបើ operand ទីម្ចាស់អាចបំប្លែងទៅជាមិនពិត វានឹង return តម្លៃនៃ operand ដំបូង ហើយនឹងបង្ហាញទៅក្នុងក្រុងក្រោម។

x && y

វានឹង **return** ពីត ប្រសិនបើប្រតិបត្តិករទាំងពីរគឺពិត នៅ៖ **return** មិនពិត។

```
true && true; // returns true
true && false; // returns false
false && true; // returns false
false && false; // returns false
```

```
8 <div id="output"></div>
9 <script language = "javascript" type = "text/javascript">
10   const x = 3;
11   const y = -2;
12   document.getElementById("output").innerHTML = x > 0 && y > 2;
13
14 </script>
```

✓ Logical OR (||) Operator

The logical OR (||) operator ក៏រាយតម្លៃ **operands** ពីច្បងទៅស្តាំ។ ប្រសិនបើ **operand** ទីមួយអាចត្រូវបានបំប្លែងទៅជាទុរាប់ **true** វានឹង **return** តម្លៃនៃ **operand** ដែលបានបំប្លែង ហើយបើមិនជាប្រួលទេ វានឹង **return** តម្លៃនៃ **operand** ទីពីរ។

```
x || y
```

វានឹង **return false** ប្រសិនបើ **operand** ទាំងពីរមិនពិត នៅ៖ វានឹង **return true** ។

```
true || true; // returns true
true || false; // returns true
false || true; // returns true
false || false; // returns false
```

```
8 <div id="output"></div>
9 <script language = "javascript" type = "text/javascript">
10   const x = 3;
11   const y = -2;
12   document.getElementById("output").innerHTML = x > 0 || y > 2;
13
14 </script>
```

✓ Multiple || Operators

យើងអាចមានប្រើប្រាស់ **|| operators** នៅក្នុងកន្លោមបញ្ហា។ នេះ **|| operators**

ក៏រាយតម្លៃកន្លោមពីច្បងទៅស្តាំ ហើយវាបានបំប្លែង **operand** នីមួយនេះជាកំណែតមួយ **boolean**

ប្រសិនបើលទ្ធផលគឺពិត នៅ៖ វានឹង **return** តម្លៃនៃ **operand** នៅ៖ ហើយបញ្ចប់ការប្រតិបត្តិ។ ប្រសិនបើ **operand** ទាំងអស់ត្រូវបានបំប្លែង នៅ៖ វានឹង **return** តម្លៃនៃ **operand** ចុងក្រាយ។

```
null || 10 || false // returns 10
false || null || undefined // returns undefined
```

✓ **Logical NOT (!) Operator**

វិវាទ return មិនពីតិត ប្រសិនបើ **operand** អាចចំណែងទៅជាទិត បើមិនដូច្នេះទេ វិវាទ return ពីតិត។

```
!x
```

```
8 |<div id="output"></div>
9 |<script language = "javascript" type = "text/javascript">
10|    document.getElementById("output").innerHTML =
11|        !true + "<br>" + //false
12|        !false + "<br>" + //true
13|        !0 + "<br>" + //true
14|        !20 + "<br>" + //false
15|        !( 'Hello World' ) //false
16|
17|</script>
```

✓ **Logical Operators Precedence**

- Logical NOT (!)
- Logical AND (&&)
- Logical OR (||)

```
8 |<div id="output"></div>
9 |<script language = "javascript" type = "text/javascript">
10|    document.getElementById("output").innerHTML =
11|        (false || true && !false) // returns true
12|
13|</script>
```

3.4. ប្រកើតទៅ Assignment Operators

- Assignment operators ត្រូវបានប្រើដើម្បីផ្តល់តម្លៃទៅអប់រំ។ ទាំងនេះគឺជាប្រភពបច្ចុករគោលពីរ។ ប្រកើត assignment operators វាយក operand ពីរ ផ្តល់តម្លៃទៅ operand
ខាងឆ្វេងដោយផ្តល់តម្លៃនៃ operand ស្ថា។ operand ខាងឆ្វេងគឺតែងតែជាអប់រំ ហើយ operand ខាងស្ថាអាចជាពួរព្រឹងៗ (literal) អប់រំ(variable) ឬការវិភាគ (expression) ។

Assignment Operator	Example	Equivalent To
= (Assignment)	a = b	a = b
+= (Addition Assignment)	a += b	a = a + b
-= (Subtraction Assignment)	a -= b	a = a - b
*= (Multiplication Assignment)	a *= b	a = a * b
/= (Division Assignment)	a /= b	a = a / b
%= (Remainder Assignment)	a %= b	a = a % b
**= (Exponentiation Assignment)	a **= b	a = a ** b

✓ Assignment (=) Operator

ផ្តល់តម្លៃទៅឱ្យអប់រំរួមយ។ យើងអាចកំណត់តម្លៃតែម្មយទៅអប់រំរបីនេះ នៃត្រូវបានគេហេរីកចាត់ជាដំឡើង។

assignment chaining។

```

8   <div id="output"></div>
9
10  let x = 5;
11  let y = x + 10;
12  document.getElementById("output").innerHTML =
13  "Value of x : " + x + "<br>" + //Value of x : 5
14  "Value of y : " + y; //Value of y : 15
15
16  </script>

```

ខាងក្រោមនេះគឺជាទាមរណ៍ទៅនៃ assignment chaining

```

8   <div id="output"></div>
9
10  var x = 5;
11  var y = x + 10;
12  document.getElementById("output").innerHTML =
13  "Value of x : " + x + "<br>" + //Value of x : 5
14  "Value of y : " + y; //Value of y : 15
15
16  //example of assignment chaining
17  var x = y = 5;
18  document.getElementById("output").innerHTML =
19  "Value of x : " + x + "<br>" + //Value of x : 5
20  "Value of y : " + y; //Value of y : 5
21
22  </script>

```

✓ **Addition Assignment (+=) Operator**

អនុវត្តការបែន្នេមនៅលើ **operand** ពីរ ហើយផ្តល់លទ្ធផលទៅ **operand** ខាងឆ្វេង ។ **addition** នៅទីនេះអាចជា **numeric addition** ឬ **string concatenation** ។

```
x += b;
```

ឧទាហរណ៍ ១ : Numeric Addition Assignment

```
8 | <div id="output"></div>
9 | <script language = "javascript" type = "text/javascript">
10| let x = 5;
11| x += 7;
12| document.getElementById("output").innerHTML = "Value of x : " + x;
13|
14| </script>
```

ឧទាហរណ៍ ២ : String Concatenation Assignment

```
8 | <div id="output"></div>
9 | <script language = "javascript" type = "text/javascript">
10| let x ="Hello";
11| x += " World";
12| document.getElementById("output").innerHTML = "Value of x : " + x;
13|
14| </script>
```

✓ **Subtraction Assignment (-=) Operator**

ជាការដែកតម្លៃនៃ **operand** ស្ថា ពី **operand** ខាងឆ្វេង ហើយផ្តល់លទ្ធផលទៅជា **operand** ខាងឆ្វេង (អប់រំ) ។

```
let x -= b;
```

ឧទាហរណ៍ ៣ :

```
8 | <div id="output"></div>
9 | <script language = "javascript" type = "text/javascript">
10| let x = 15;
11| x -= 5;
12| document.getElementById("output").innerHTML = "Value of x : " + x;
13|
14| </script>
```

✓ **Multiplication Assignment (*=) Operator**

ជាការគូណានឹង **operand** ទាំងពីរ ហើយផ្តល់លទ្ធផលទៅ **operand** ខាងឆ្វេង ។

```
let x *= b;
```

ឧទាហរណ៍ ៤ :

```
8 | <div id="output"></div>
9 | <script language = "javascript" type = "text/javascript">
10| let x = 10;
11| x *= 5;
12| document.getElementById("output").innerHTML = "Value of x : " + x;
13|
14| </script>
```

✓ **Division Assignment (/=) Operator**

Operator នេះបែក **operand** ផ្សេងៗជាយ **operand** ស្ថា ហើយផ្តល់លទ្ធផលទៅ **operand** ផ្សេង ។

```
let x /= b;
```

ឧចាបរណ៍៖

```

8   <div id="output"></div>
9
10  <script language = "javascript" type = "text/javascript">
11    let x = 10;
12    x /= 5;
13    document.getElementById("output").innerHTML = "Value of x : " + x;
14  </script>

```

✓ **Remainder Assignment (%=) Operator**

អនុវត្តប្រព័ន្ធប្រព័ន្ធគ្មាននៃសល់សំណល់នៅលើ operands ហើយផ្តល់លទ្ធផលទៅ operand ខាងឆ្លង។

```
let x %= b;
```

ឧចាបរណ៍៖

```

8   <div id="output"></div>
9
10  <script language = "javascript" type = "text/javascript">
11    let x = 12;
12    x %= 5;
13    document.getElementById("output").innerHTML = "Value of x : " + x;
14  </script>

```

✓ **Exponentiation Assignment (**=) Operator**

Operator នេះអនុវត្តប្រព័ន្ធគ្មាននៃសម្រួលនៅលើ operands ហើយផ្តល់លទ្ធផលទៅ operand ខាងឆ្លង។

```
let x **= b;
```

ឧចាបរណ៍៖

```

8   <div id="output"></div>
9
10  <script language = "javascript" type = "text/javascript">
11    let x = 5;
12    x **= 3; //5x5x5 = 125
13    document.getElementById("output").innerHTML = "Value of x : " + x;
14  </script>

```

3.5. ប្រកើទទេន Conditional Operators

- លក្ខខណ្ឌនៅក្នុង JavaScript ដីបួនភាយតម្លៃកន្លោមសម្រាប់តម្លៃពិត ប្រមិនពិត ហើយបន្ទាប់មកប្រព័ន្ធប្រព័ន្ធអម្ចាស់បំណោមពីដែលបានផ្តល់ អាស្រែយលើលទ្ធផលនៃការវាយតម្លៃ។ លក្ខខណ្ឌត្រូវបានគេបង់ថា ternary operator ។
- ក្នុងលក្ខខណ្ឌ JavaScript (ternary) គ្រាន់តែជាប្រព័ន្ធប្រព័ន្ធបែងការដែលយក operands បីៗ លក្ខខណ្ឌមួយតាមដោយសញ្ញាស្សែវ (?) បន្ទាប់មកកន្លោមទីម្ចាយដែលត្រូវប្រព័ន្ធប្រព័ន្ធប្រសិនបើលក្ខខណ្ឌតីជាការពិតតាមដោយសញ្ញា (:) ហើយបួនភាយទីពី កន្លោមនឹងត្រូវបានប្រព័ន្ធប្រព័ន្ធប្រសិនបើលក្ខខណ្ឌមិនពិត។
- Syntax នៃ conditional (ternary) operator

```
var variable = condition ? exp1 : exp2;
```

នៅទីនេះយើងបានពន្យល់ពីចាបកមែត្រ (parameters) នៅក្នុង statement ខាងលើ ។

- **condition** – It is a conditional statement.
- **exp1** – If the conditional statement evaluates truthy, control flow executes the exp1 expression.
- **exp2** – If the conditional statement evaluates falsy, control flow executes the exp2 expression.

ឧទាហរណ៍ទី១ :

```
8 <div id="output"></div>
9
10 <script language = "javascript" type = "text/javascript">
11     var num1 = 90;
12     var num2 = 67;
13     var res = num1 > num2 ? "num1 is greater than num2" : "num2 is greater than num1";
14     document.getElementById("output").innerHTML = res;
15 </script>
```

ឧទាហរណ៍ទី២ :

```
8 <div id="output"></div>
9
10 <script language = "javascript" type = "text/javascript">
11     const year = 2004;
12     const obj = {
13         name: "Mr.SOK",
14         age: year < 2005 ? "adult" : "minor",
15         city: "Phnom Penh"
16     };
17
18     document.getElementById("output").innerHTML =
19         obj.name + " is " + obj.age + " and lives in " + obj.city;
20 </script>
```

3.6. ប្រភេទនៃ typeof Operator

- ប្រើដើម្បីទទួលបានប្រភេទទិន្នន័យនៃអប់រំ (variable) ដាក់លាក់មួយ ។ វាប្រាក់បានដាក់មុន operand តែមួយរបស់ក្តី ដែលអាចជាប្រភេទណាមួយ ។ វិវាទនៃវានឹងគូនបាននូវការរាយការណ៍ប្រព័ន្ធដែលបង្ហាញពីប្រភេទទិន្នន័យនៃប្រព័ន្ធដែលរាយការណ៍។
- **syntax នៃ typeof operator**

```
typeof (operand);
```

- ននេះគឺជាបញ្ជី return តាម សម្រាប់ប្រភេទ typeof Operator

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"
Undefined	"undefined"
Null	"object"
Symbol	"symbol"
Bigint	"bigint"

- number, string, boolean, bigint, undefined, null, and symbol.

ប្រភេទប្រតិបត្តិដែលសំខាន់ក្នុងការកំណត់អត្ថសញ្ញាណប្រភេទទិន្នន័យបច្ចុប្បន្នជាដំឡើងនេះ។

```
typeof 10; // returns 'number'
typeof 'Hello World'; // returns 'string'
typeof true; // returns 'boolean'
typeof {name:"Tutorialspoint"}; // returns 'object'
typeof function foo(){// returns 'function'
typeof undefined; // returns 'undefined'
typeof null; // returns 'object'
typeof Symbol(); // returns 'symbol'
typeof 10n; // returns 'bigint'
```

- ✓ typeof Operator ដើម្បីពិនិត្យប្រភេទ Number

```
typeof 10; //returns "number";
typeof -10; //returns "number";
typeof 0; //returns "number";
typeof 10.20; //returns "number";
typeof Math.LN10; //returns "number";
typeof Infinity; //returns "number";
typeof NaN; //returns "number";
typeof Number('1'); //returns "number";
typeof Number('hello'); //returns "number";
```

ឧចាបរណ៍៖

```
<div id="output"></div>
<script>
    let num = 42;
    document.getElementById("output").innerHTML = typeof num;
</script>
<p>Set the variable to different value and then try...</p>
```

- ✓ **typeof Operator** ដើម្បីពិនិត្យប្រភេទ string

```
typeof "10"; //returns "string";
typeof ""; //returns "string";
typeof "Hello World"; //returns "string";
typeof String(10); //returns "string";
typeof typeof 2; //returns "string";
```

ឧចាបរណ៍៖

```
<div id="output"></div>
<script>
    let str = "Hello World";
    document.getElementById("output").innerHTML = typeof str;
</script>
```

- ✓ **typeof Operator** ដើម្បីពិនិត្យប្រភេទ Boolean

```
typeof true; //returns "boolean";
typeof false; //returns "boolean";
typeof Boolean(10); //returns "boolean";
```

ឧចាបរណ៍៖

```
<div id="output"></div>
<script>
    let bool = true;
    document.getElementById("output").innerHTML = typeof bool;
</script>
```

- ✓ **typeof Operator** ដើម្បីពិនិត្យប្រភេទ Symbol

```
typeof Symbol(); //returns "symbol";
typeof Symbol("unique values"); //returns "symbol";
```

ឧចាបរណ៍៖

```
<script>
    let sym = Symbol("Hello");
    document.getElementById("output").innerHTML = typeof sym;
</script>
```

- ✓ **typeof Operator** ដើម្បីពិនិត្យប្រភេទ Undefined និង Null

```
typeof undefined; //returns "undefined";
typeof null; //returns "object";
```

ឧទាហរណ៍៖

```
<script>
    let x;
    document.getElementById("output").innerHTML = typeof x;
</script>
```

- ✓ **typeof Operator** ដើម្បីពិនិត្យប្រភេទ Object

```
const obj = {age: 23};
typeof obj; //returns "object";
const arr = [1,2,3,4];
typeof arr; //returns "object";
typeof new Date(); //returns "object";
typeof new String("Hello World"); //returns "object";
```

ឧទាហរណ៍៖

```
<script>
    const person = {name: "John", age: 34};
    document.getElementById("output").innerHTML = typeof person;
</script>
```

- ✓ **typeof Operator** ដើម្បីពិនិត្យប្រភេទ Function

```
const myFunc = function(){return "Hello world"};
typeof myFunc; //returns "function";
const func = new Function();
typeof func; //returns "function";
class myClass {constructor() {}}
typeof myClass; // returns "function";
```

ឧទាហរណ៍៖

```
<script>
    const myFunc = function(){return "Hello world"};
    document.getElementById("output").innerHTML = typeof myFunc;
</script>
```

✓ **typeof Operator** ដើម្បីពិនិត្យប្រភេទ Arrays

```
<script>
    let numbers = [1, 2, 3];
    document.getElementById("output").innerHTML = Array.isArray(numbers);
</script>
```

3.7. ប្រភេទទីនា Nullish Coalescing Operator

- ប្រភេទទីនា Nullish Coalescing គឺជា JavaScript ត្រូវបានតាំងរាល់ដោយសញ្ញាស្ថី (??) ។ វាគ្មានការ operand ពី ហើយ return operand ទីម្ចាយ ប្រសិនបើវាអីមិនមែនជា null ឬ undefined ។ បើមិនជាដ្មីទេ វានឹង return operand ទីពីរ
- syntax** ខាងក្រោមដើម្បីប្រើ Nullish Coalescing operator

```
op1 ?? op2
```

- ប្រភេទ nullish coalescing operator (??) return operand ទីពីរ (op2) ប្រសិនបើ operand ទីម្ចាយ (op1) វិនិយោគ null ឬ undefined ។ បើមិនជាដ្មីទេ អរប់ 'res' នឹងមាន 'op2' ។
- syntax** របស់វា គឺស្របច្បែងនឹងក្នុងខាងក្រោម

```
let res;
if (op1 != null || op1 != undefined) {
    res = op1;
} else {
    res = op2;
}
```

ឧទាហរណ៍៖

```
<script>
    let x = null;
    let y = x ?? 5;
    document.getElementById("output").innerHTML =
        "The value of y is: " + y;
</script>
```

ឧទាហរណ៍៖ null ឬ undefined នៅក្នុង Arrays ៖

```
<div id = "output"></div>
<script>
    const arr = [65, 2, 56, 2, 3, 12];
    const arr1 = arr ?? [];
    document.getElementById("output").innerHTML
        = "The value of arr1 is: " + arr1;
</script>
```

ឧចាបរណ៍ Object :

```

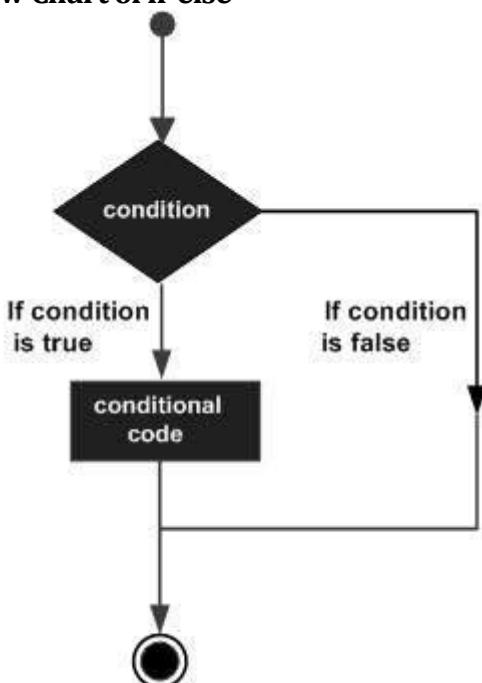
<div id = "output"></div>
<script>
    const obj = {
        product: "Mobile",
        price: 20000,
        color: "Blue",
    }

    let product = obj.product ?? "Watch";
    let brand = obj.brand ?? "Apple";
    document.getElementById("output").innerHTML =
    "The product is " + product + " of the brand " + brand;
</script>

```

3.8. ប្រកិទ្ធន៍ if...else Statement

- **if...else statement** ប្រតិបត្តិប្រកិទ្ធន៍ក្នុង នៅពេលដែលលក្ខខណ្ឌដែលបានបញ្ជាក់គឺពីតិត្ត ឬ នៅពេលដែលលក្ខខណ្ឌមិនពីតិត្ត បុកដោយដោតនឹងត្រូវបានប្រតិបត្តិ ។ **if-else** អាចត្រូវបានប្រើដើម្បីគ្រប់គ្រងលំហេរនៃការអនុវត្តកម្មវិធីដោយផ្តើកលើលក្ខខណ្ឌដោយដោតនឹង។
- **Flow Chart of if-else**



- **syntax នៃ if...else Statement** ត្រូវបានអនុវត្តដូចខាងក្រោម៖

```

if (expression) {
    Statement(s) to be executed if expression is true
} else {
    Statement(s) to be executed if expression is false
}

```

ឧចាបរណ៍៖

```

<div id ='output'> </div>
<script type = "text/javascript">
    let result;
    let age = 15;
    if( age > 18 ) {
        result = "Qualifies for driving";
    } else {
        result = "Does not qualify for driving";
    }
    document.getElementById("output").innerHTML = result;
</script>
<p> Set the variable to a different value and then try... </p>

```

3.9. ប្រភេទនៃ if...else if... statement

- ប្រភេទនៃ if...else if... (ហែងដែរប៉ាជាន់ if...else ladder) គឺជាជម្រៃកម្រិតខ្ពស់នៃ if...else ដែលអនុញ្ញាតឱ្យ JavaScript ធ្វើការសម្រចចិត្តត្រូវបចញ្ចីលក្ខខណ្ឌជាប្រើប្រាស់។ នៅពេលវាត្រូវបានបញ្ជាក់ថាទីតាំងនៃការបញ្ចូនត្រូវបានបញ្ជាក់ឡើង។ ក្នុងការបញ្ចូនត្រូវបានបញ្ជាក់ថាទីតាំងនៃការបញ្ចូនត្រូវបានបញ្ជាក់ឡើង។
- syntax នៃ if...else if... statement ត្រូវបានអនុវត្តដូចខាងក្រោម៖

```

if (expression 1) {
    Statement(s) to be executed if expression 1 is true
} else if (expression 2) {
    Statement(s) to be executed if expression 2 is true
} else if (expression 3) {
    Statement(s) to be executed if expression 3 is true
} else {
    Statement(s) to be executed if no expression is true
}

```

ឧចាបរណ៍៖

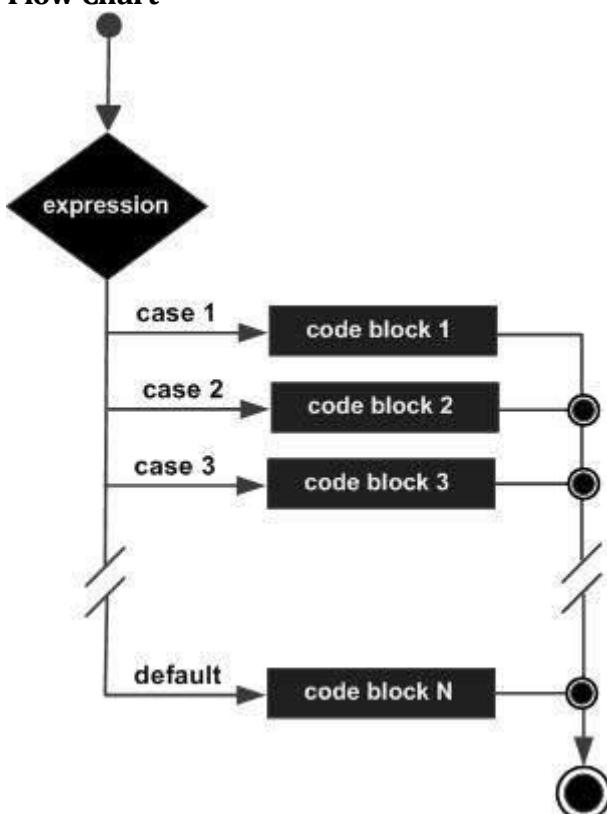
```

<div id ="demo"></div>
<script type="text/javascript">
    const output = document.getElementById("demo")
    let book = "maths";
    if (book == "history") {
        output.innerHTML=<b>History Book</b>";
    } else if (book == "maths") {
        output.innerHTML=<b>Maths Book</b>";
    } else if (book == "economics") {
        output.innerHTML=<b>Economics Book</b>";
    } else {
        output.innerHTML=<b>Unknown Book</b>";
    }
</script>
<p> Set the variable to a different value and then try... </p>

```

3.10. ប្រភេទនៃ Switch Case

- ជាលក្ខខណ្ឌត្រូវបានប្រើដើម្បីប្រពិបត្តិថ្មីក្នុងផ្សេងៗគ្មានស្រែយលើតម្លៃនៃករណ្យមួយ។ ប្រសិនបើវា ត្រូវគ្នានឹងតម្លៃនៃ **case labels** មួយ ឬក្នុងដែលទាក់ទងនឹងករណីនោះត្រូវបានប្រពិបត្តិ។
- អ្នកអាចប្រើប្រើន **if...else...if statements** ដូចក្នុងជំពូកមុន ដើម្បីអនុវត្ត **multiway branch**។ ទៅដើម្បីដោយ នេះមិនជាដឹងឈាន៖ស្រាយដូចតាមតម្លៃដែលបានបង្ហាញ។
- ជាធិសសនោះពេលដែលសាទាចាំងអស់អាស៊ីយទៅលើតម្លៃនៃអប់រំតម្លៃ (**single variable**)។
- Flow Chart**



- **syntax នៃ Switch Case** ត្រូវបានអនុញ្ញាតដូចខាងក្រោម៖
interpreter ពិនិត្យករណីនឹងយករាយដោយបន្ថែមតម្លៃនៃការស្រើសិទ្ធិការងារ ហើយតិចជាបន្ទាត់ការផ្តល់ការផ្តល់ដោយត្រូវបានរកដោយ។
ប្រសិនបើត្រូវបានអ្វីគ្មានទៅ លក្ខណៈណាំដើម (default condition)នឹងត្រូវបានប្រើ។

```
switch (expression) {
    case condition 1: statement(s)
    break;

    case condition 2: statement(s)
    break;

    ...
    case condition n: statement(s)
    break;

    default: statement(s)
}
```

ឧទាហរណ៍៖

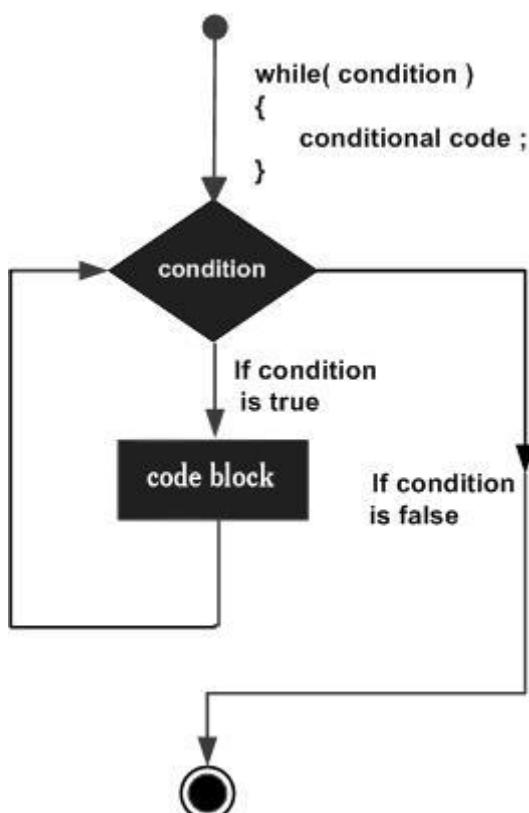
```
<p id = "output"> </p>
<script>
    const output = document.getElementById("output");
    let grade = 'A';
    output.innerHTML += "Entering switch block <br />";
    switch (grade) {
        case 'A': output.innerHTML += "Good job <br />";
        break;
        case 'B': output.innerHTML += "Passed <br />";
        break;
        case 'C': output.innerHTML += "Failed <br />";
        break;
        default: output.innerHTML += "Unknown grade <br />";
    }
    output.innerHTML += "Exiting switch block";
</script>
```

ឧទាហរណ៍ ៖ Strict Comparison

```
<p id = "output"> </p>
<script>
    const output = document.getElementById("output");
    let num = 10;
    switch (num) {
        case '10': output.innerHTML += "10 Marks!";
                    break;
        case '11': output.innerHTML += "11 Marks!";
                    break;
        default: output.innerHTML += "Input is not a string!";
    }
</script>
```

3.11. ប្រកែវនៃ While Loops

- ເນື້ອັກຸຟ JavaScript ບໍ່ເຜົ້າດເຜົ້າລັບ (loop) ໃພລປະຕິບຄືບຸກກູຟມູັງແກ້ຍມູັງເຮັດວຽກ
ຝາກບດາລູ້ຂណ້າໃພລຕານບញ້າກໍຕື່ຕິດ ພູມສູ່ຂណ້າຕຽ່ງຕານກ່າຍຕໍ່ເສີມນະເຕລະປະຕິບຄືໃນບຸກກູຟ ແລ້ວ
 - ເຄາລບໍ່ແດັ່ງນີ້ **while loop** ດີເນີຍປະຕິບຄື **statement** ບັນລຸ **code block** ມູັງແກ້ຍມູັງເຮັດວຽກ
ຝາກບດາ **expression** ດື່ມຕິດ ເນື້ອເຕເລຍໃພລກເຮົາມູັງແກ້ຍມູັງທີ່ ເຜົ້າລັບຜູ້ນີ້ແບ່ງບញ້າປ່ໍ່
 - **Flow Chart**



- syntax នៃ While Loops ត្រូវបានអនុវត្តដូចខាងក្រោម៖

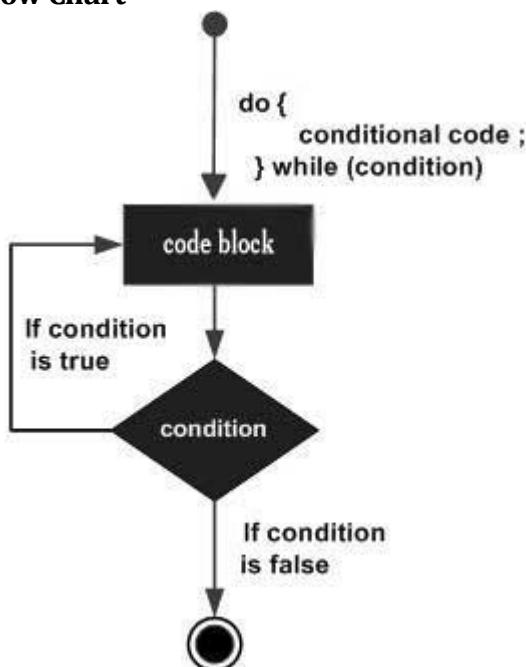
```
while (expression) {
    Statement(s) to be executed if expression is true
}
```

ឧទាហរណ៍៖

```
<div id = 'output'></div>
<script type="text/javascript">
    let output = document.getElementById("output");
    var count = 0;
    output.innerHTML="Starting Loop <br>";
    while (count < 10) {
        output.innerHTML+="Current Count : " + count + "<br>";
        count++;
    }
    output.innerHTML+="Loop stopped!";
</script>
```

3.12. ប្រភេទនៃ do...while Loop

- គឺស្រដែងនឹង while loop លើកលែងតែការពិនិត្យលក្ខខណ្ឌកៅតឡ្វ់នៅចុងបញ្ហប៉ុន្ម័ន្តិលដ្ឋី នៃមាននំយថា ផ្ទិលដ្ឋី (loop) នឹងតែងតែត្រូវបានប្រតិបត្តិយ៉ាងហេចបាលសំម្រេង ទៅបីជាលក្ខខណ្ឌមិនពិតក៏ដោយ។
- Flow Chart



- **syntax នៃ do...while Loop** ត្រូវបានអនុវត្តដូចខាងក្រោម៖

```
do {
    Statement(s) to be executed;
} while (expression);
```

ឧទាហរណ៍៖

```
<div id="output"></div>
<script type="text/javascript">
    let output = document.getElementById("output");
    var count = 0;
    output.innerHTML += "Starting Loop" + "<br />";
    do {
        output.innerHTML += "Current Count : " + count + "<br />";
        count++;
    }
    while (count < 5);
    output.innerHTML += "Loop stopped!";
</script>
```

ឧទាហរណ៍៖ while vs. for Loops

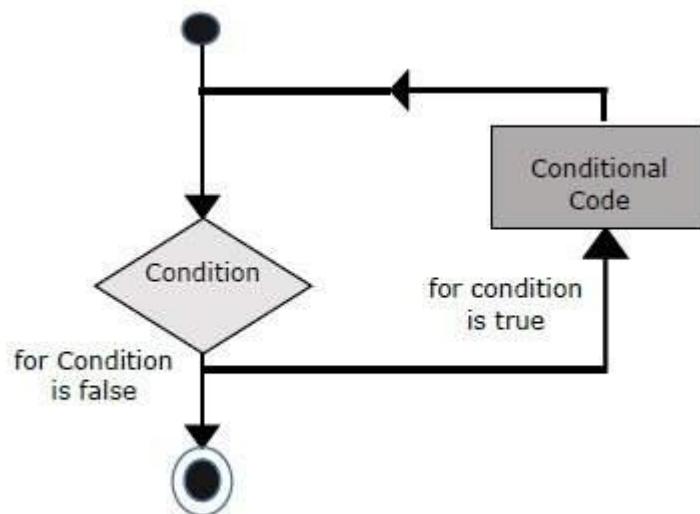
```
<div id = "demo"> </div>
<script>
    const output = document.getElementById("demo");
    for(let i = 1; i <= 5; i++){
        output.innerHTML += i + "<br>";
    }
</script>
```

ឧទាហរណ៍៖ while vs. for Loops (យើងអាចកែវប្រាកដលើសម្រាប់ loop)

```
<div id = "demo"> </div>
<script>
    const output = document.getElementById("demo");
    let i = 1;
    for(; i <= 5; ){
        output.innerHTML += i + "<br>";
        i++
    }
</script>
```

3.13. ប្រកែទន់ For Loop

▪ Flow Chart



- **syntax** នៃ for Loop ត្រូវបានអនុវត្តដូចខាងក្រោម ៖

```
for (initialization; condition; iteration) {  
    Statement(s) to be executed if condition is true  
}
```

ឧចាបរណ៍ទី១

```
<p id = "output"> </p>
<script>
    const output = document.getElementById("output");
    output.innerHTML = "Starting Loop <br>";
    let count;
    for (let count = 0; count < 10; count++) {
        output.innerHTML += "Current Count : " + count + "<br/>";
    }
    output.innerHTML += "Loop stopped!";
</script>
```

ឧចាបរណ៍ទី២ Conditional statement ជា optional

```
<p id = "output"> </p>
<script>
    let output = document.getElementById("output");
    let arr = [10, 3, 76, 23, 890, 123, 54]
    var p = 0;
    for ( ; ; p++) {
        if (p >= arr.length) {
            break;
        }
        output.innerHTML += "arr[" + p + "] -> " + arr[p] + "<br/>";
    }
</script>
```

3.14. ប្រកួតនៃ For...in Loop

- ត្រូវបានប្រើដើម្បី loop តាមរយៈលក្ខណៈសម្រួលបស់ object ។ JavaScript for...in loop គឺជាបំផែលនៃ for loop ។ មិនអាចប្រើសម្រាប់ធ្វើលង្វែង (loop) ដើម្បីផ្តល់ការពិន័យទៅក្នុង object properties បានទេ ។ ដូច្នេះ for...in loop ត្រូវបានណែនាំដើម្បីផ្តល់ការពិន័យទៅក្នុង object properties ទាំងអស់ ។
- syntax នៃ For...in Loop ត្រូវបានអនុវត្តដូចខាងក្រោម៖

```
for (variableName in object) {
    statement or block to execute
}
```

Parameters

- variableName** – It is a property name (key) of the object.
- in** – It is an 'in' operator in JavaScript.
- object** – It is the object to traverse.

ឧទាហរណ៍ ៩: Iterate តាមរយៈ object properties

```
<p id = "output"> </p>
<script>
    let output = document.getElementById("output");
    let car = {
        brand: "OD",
        model: "Q7",
        color: "Black",
    }
    for (key in car) {
        output.innerHTML += key + " -> " + car[key] + "<br>";
    }
</script>
```

លទ្ធផល

```
brand -> OD
model -> Q7
color -> Black
```

ឧទាហរណ៍ ១០: Iterating តាមរយៈ string

```
<p id = "output"> </p>
<script>
    let output = document.getElementById("output");
    let str = "Hello";
    for (key in str) {
        output.innerHTML += key + " -> " + str[key] + "<br>";
    }
</script>
```

លទ្ធផល

```
0 -> H
1 -> e
2 -> l
3 -> l
4 -> o
```

3.15. ប្រកិទន៍ For...of Loop

- ត្រូវបានប្រើដើម្បីផ្តល់ការពិនិត្យអារាសំបុរាណ។ នៅក្នុងការធ្វើឡើងវិញនឹងយុង ការណែនាំនូវ ធម្មតា (element) នៃវត្ថុដែលអាចធ្វើបាន (iterable object) ។ Iterable objects គឺមាន arrays, strings, maps, sets, and generators.
- syntax នៃ For...of Loop ត្រូវបានអនុវត្តដូចខាងក្រោម៖

```
for (ele of iterable) {
    // loop body
}
```

Parameters

- **ele** – It is a current element of the iterable.
- **of** – It is a JavaScript operator.
- **iterable** – It is iterable like an object, array, string, etc.

ឧទាហរណ៍៖ For...of Loop ជាមួយ Arrays

```
<p id="output"> </p>
<script>
    const output = document.getElementById("output");
    const arr = ["JavaScript", "Python", "C", "C++", "HTML", "CSS"];
    for (let ele of arr) {
        output.innerHTML += ele + "<br>";
    }
</script>
```

លទ្ធផល

JavaScript
Python
C
C++
HTML
CSS

ឧចាប់ណើ៖ For...of Loop ដោម្បី Strings

```
<script>
    const output = document.getElementById("output");
    let str = "JavaScript";
    for (let char of str) {
        output.innerHTML += char + ", "; //J, a, v, a, S, c, r, i, p, t,
    }
</script>
```

ឧចាប់ណើ៖ For...of Loop ដោម្បី Set

```
<p id="output"> </p>
<script>
    const output = document.getElementById("output");
    const nums = new Set([10, 20, 30, 30, 30, 40, 50, 60]);
    for (let num of nums) {
        output.innerHTML += num + ", "; //10, 20, 30, 40, 50, 60,
    }
</script>
```

ឧចាប់ណើ៖ For...of Loop ដោម្បី Map

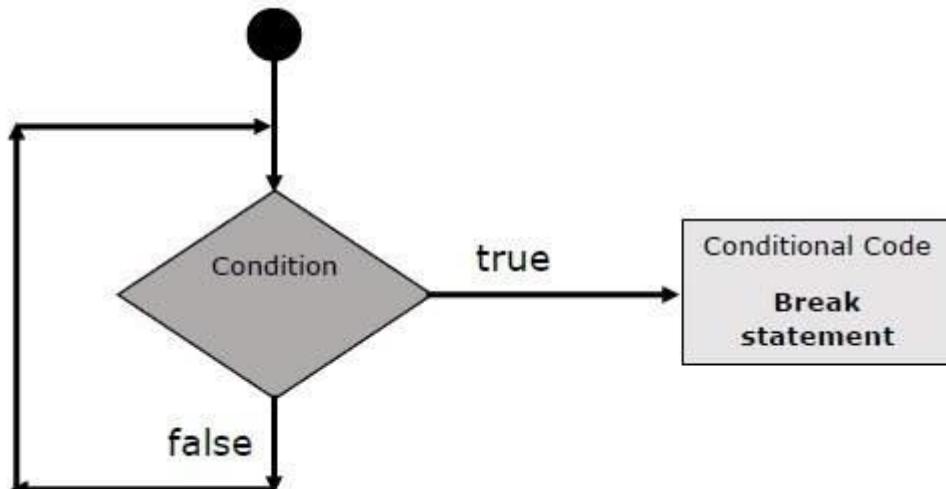
```
<p id="output"> </p>
<script>
    const output = document.getElementById("output");
    const map = new Map();
    map.set("one", 1);
    map.set("second", 2);
    map.set("third", 3)
    for (let [k, v] of map) {
        output.innerHTML += k + " -> " + v + "<br/>";
    }
</script>
```

3.16. ប្រភេទនៃ Loop Control

- JavaScript ផ្តល់ការគ្រប់គ្រងពេញលេញដើម្បីគ្រប់គ្រង loops និង switch statements ។ វាអាចមានស្ថានភាពនៅពេលដែលអ្នកត្រូវការបេញចូលពី loop ដោយមិនយកដល់តម្លៃចុងក្រាយ បែស់វា ។
- ដើម្បីដោះស្រាយស្ថានភាពទាំងអស់នោះ JavaScript ផ្តល់នូវ break និង continue statements ។ statements ទាំងនេះត្រូវបានប្រើដើម្បីបេញចូលពី loop ឬរាយការបន្ទាប់នៃ loop ឬរាយការបន្ទាប់នៃ loop ។ ដូច្នោនេះដឹងដើរ JavaScript អនុញ្ញាតឱ្យអ្នកអភិវឌ្ឍន៍ labels ដើម្បីដាក់រោងចូលដុំ ។

- យើងពន្យល់ពាក្យគីនី៖នៅក្នុងការងារខាងក្រោមដែលអាចត្រូវបានប្រើដើម្បីគ្រប់គ្រងធ្វើលដ្ឋី។
 - **break** ពាក្យគីនី៖ 'break' ត្រូវបានប្រើដើម្បីចេញពីធ្វើលដ្ឋី។
 - **continue** ពាក្យគីនី៖ 'continue' ត្រូវបានប្រើដើម្បីរំលងការបន្លបន្ទាប់នៃធ្វើលដ្ឋី។
 - **label** មិនមែនជាតាក្យគីនី៖ទេ បុន្ថន្ធអ្នកអាចប្រើសម្រាប់សម្រាប់អ្នកប្រើប្រាស់ក្នុងការបន្លបន្ទាប់ពីធ្វើលដ្ឋី។
ដោយសញ្ញា (:) ដើម្បីផ្តល់ **label** ម្មាយទៅធ្វើលដ្ឋី។

- **Flow Chart**



ឧចាបរណ៍៖ **break**

```

<div id = "output"> </div>
<script>
    const output = document.getElementById("output");
    let x = 1;
    output.innerHTML = "Entering the loop<br>";
    while (x < 20) {
        if (x == 5) {
            break; // breaks out of loop completely
        }
        x = x + 1;
        output.innerHTML += x + "<br>";
    }
    output.innerHTML += "Exiting the loop!<br>";
</script>
<p>Set the variable to different value and then try...</p>
  
```

ឧចាបរណ៍នៃ continue

```
<div id = "output"> </div>
<script>
    const output = document.getElementById("output");
    let x = 1;
    output.innerHTML = "Entering the loop<br>";
    while (x < 10) {
        x = x + 1;
        if (x == 5) {
            continue; // skip rest of the loop body
        }
        output.innerHTML += x + "<br>";
    }
    output.innerHTML += "Exiting the loop!<br>";
</script>
```

ឧចាបរណ៍នៃ continue

```
<div id = "output"> </div>
<script>
    const output = document.getElementById("output");
    output.innerHTML = "Entering the loop!<br />";
    outerloop:           // This is the label name
    for (let i = 0; i < 5; i++) {
        output.innerHTML += "Outerloop: " + i + "<br />";
        innerloop:
        for (let j = 0; j < 5; j++) {
            if (j > 3 ) break ;           // Quit the innermost loop
            if (i == 2) break innerloop;   // Do the same thing
            if (i == 4) break outerloop;  // Quit the outer loop
            output.innerHTML += "Innerloop: " + j + " <br />";
        }
    }
    output.innerHTML += "Exiting the loop!<br />";
```

សេចក្តីផ្តើម:

Array

4. គាន់ប្រព័ន្ធមូលដ្ឋាន៖ Array

Array នៅក្នុង JavaScript អនុញ្ញាតឱ្យរក្សាទុកតម្លៃជាប្រើននៅក្នុងអប់រំកម្មយោយកម្លាំងនិមួយ។ ត្រូវមានប្រភេទទីនួនយដ្ឋានច្បាស់។ ធាតុរបស់ Array គឺចាប់ផ្តើមពីលេខ 0 ដែលគេហោថា index។ Array ត្រូវបានប្រើដើម្បីរក្សាទុកបណ្តុះជាបន្ទូបន្ទាប់នៃធាតុជាប្រើននៃប្រភេទទីនួនយដ្ឋានច្បាស់។

Syntax ដើម្បីបង្កើត array object នៅក្នុង JavaScript

```
const arr = new Array(val1, val2, val3, ..., valN)
```

- Parameters

val1, val2, val3, ..., valN – វាត្រូវការពេតម្លៃប្រើនជា argument ដើម្បីចាប់ផ្តើម array ដាម្បួយពួកវា។

- Return value

អ្នកអាចបង្កើលខិត្តជាប្រើនដែលបំបែកដោយសញ្ញាកែវសនេះខាងក្រោមតាមការបង្កើបការដើម្បីបង្កើត array ដោយប្រើពួកណ៍ (array literal)។

```
const fruits = [ "apple", "orange", "mango" ];
```

អ្នកអាចបង្កើលខិត្តជាប្រើនដែលបំបែកដោយសញ្ញាកែវសនេះខាងក្រោមតាមការបង្កើបការដើម្បីបង្កើត array ដោយប្រើពួកណ៍ (array literal)។

```
fruits[0] is the first element  
fruits[1] is the second element  
fruits[2] is the third element
```

4.1. Array Properties

នេះគឺជាបញ្ហាលក្ខណៈ properties នៃ Array object រួមជាមួយនឹងការពិពណ៌នាបែត្រ។

Sr.No.	Name & Description
1	constructor Returns a reference to the array function that created the object.
2	length Reflects the number of elements in an array.

- Array constructor Property ត្រូវបានប្រើដើម្បីត្រួតបង្កើរអត្ថបទ constructor សម្រាប់ array ។ ឧទាហរណ៍៖ ត្រួតបង្កើរអត្ថបទ function Array() { [native code] }.

```
<script>  
    let animals = new Array ("lion", "cheetah", "tiger", "elephant");  
    let result = animals.constructor;  
    document.write(result);  
</script>
```

- **Array length Property** ត្រូវបានប្រើដើម្បីត្រឡប់ចំនួនធាតុដែលមាននៅក្នុង array ម្មយ។ ខាងក្រោមនេះគឺជាសេណាកីយូម្យយចំនួនដែលយើងអាចប្រើ **Array.length Property** ាំ
 - ដើម្បីពិនិត្យមែលចំនួនធាតុដែលមាននៅក្នុង array ម្មយ។
 - ដើម្បីពិនិត្យមែលថាតើ array ទេទ្របាត់។
 - យើងអាចប្រើ **Property** នេះដើម្បីយកធាតុចំពោះចុងនៃ array ។
 - យើងអាចបញ្ចប់ទំហំ array ដោយកំណត់ **length** បើ។

ឧទាហរណ៍ទី១៖

```
<p id="demo"></p>
<script>
  const animals = ["Lion", "Cheetah", "Tiger", "Elephant", "Dinosaur"]
  let result = animals.length;
  document.getElementById("demo").innerHTML = result;
  //Output: 5
</script>
```

ឧទាហរណ៍ទី២៖

```
<p id="demo"></p>
<script>
  const animals = ["Lion", "Cheetah", "Tiger", "Elephant", "Dinosaur"]
  animals.length = 3;
  document.getElementById("demo").innerHTML = animals;
  //Output: Lion,Cheetah,Tiger
</script>
```

4.2. Array Methods

វិធីសារស្ថិតិថានេះត្រូវបានប្រាកដយើងប្រើ **Array class** ខ្លួនវាដាចល់។

Sr.No.	Name & Description
1	from() Creates a shallow copy of the array.
2	isArray() Returns boolean values based on the argument is an array.
3	of() Creates an array from multiple arguments.

Syntax:

```
Array.from(arrayLike, mapFn, thisArg)
```

ឧចាបរណ៍ទី១៖ Array.from() Method

```
const arrayLikeObj = { 0: 'apple', 1: 'banana', 2: 'cherry', length: 3 };
const arr = Array.from(arrayLikeObj);
document.write(arr);
//Output: apple,banana,cherry
```

Array.isArray() Method ប្រើដើម្បីពិនិត្យមិនបានគឺជាមុខត្ថាទីត្រង់ដែលបានផ្តល់កីជាតុ array បុអត់។ ប្រសិនបើវាបានគឺជាមុខត្ថាទីត្រង់នេះនឹងត្រឡប់ "true" បើមិនដូច្នោះទេ វាត្រឡប់ "false" ។

Syntax:

```
array.isArray(object);
```

ឧចាបរណ៍ទី២៖

```
<p id="demo"></p>
<script>
    const animals = ["Lion", "Cheetah", "Tiger", "Elephant", "Dinosaur"];
    const result = Array.isArray(animals);
    document.getElementById("demo").innerHTML = result;
    //Output: true
</script>
```

ឧចាបរណ៍ទី៣៖

```
<p id="demo"></p>
<script>
    const animals = "Lion";
    const result = Array.isArray(animals);
    document.getElementById("demo").innerHTML = result;
    //Output: false
</script>
```

Array.of() Method បង្កើត Array instance ដូចម្នូយនឹងបំណុលអប់រំ arguments ដោយមិនគឺជាប្រភេទទិន្នន័យបស់ពួកគេ។

Syntax:

```
Array.of(element1, element2, ..., elementN)
```

ឧចាបរណ៍ទី១៖

```
<script>
  const numbers = Array.of(1, 2, 3, 4, 5);
  document.write(numbers); //Output: 1,2,3,4,5
</script>
```

ឧទាហរណ៍ទី២៖

```
<script>
  const fruits = Array.of('apple', 'banana', 'cherry');
  document.write(fruits);
  //Output: apple,banana,cherry
</script>
```

4.3. Array instance methods

Sr.No.	Name & Description
1	at() ប្រើដើម្បីទាញយកធានាតុកជាក់លាក់តូចជាមួយ
2	concat() ប្រើដើម្បីភ្លាប់បញ្ជូនជាមួយគ្នា
3	copyWithin() ប្រើដើម្បីចម្លងធានាតុក array ពីទីតាំងមួយទៅទីតាំងមួយទៀតក្នុង array ដើម្បីផ្តល់ជាក់លាក់
4	entries() ដើម្បីទទួលបានធានាតុកនិមួយនៃ array
5	every() ប្រើដើម្បីពិនិត្យធានាតុកទាំងអស់នៅតូចជាមួយ
6	fill() ប្រើដើម្បីបំពេញធានាតុកទាំងអស់នៃ array ជាមួយនឹងតម្លៃដើម្បីផ្តល់បញ្ជាក់
7	filter() ប្រើដើម្បីបង្កើត array ថ្មីដែលផ្តល់តម្លៃការការពារតាមរយៈការត្រួតពិនិត្យ
8	find() ដើម្បីស្វែងរកធានាតុកដែលបំពេញតម្លៃការពារ
9	findLast() ដើម្បីស្វែងរកធានាតុកដែលបំពេញតម្លៃការពារពីចុងក្រោយ
10	findLastIndex() ដើម្បីស្វែងរកលិបិក្រមនៃធានាតុកដែលបំពេញតម្លៃការពារពីចុងក្រោយ។
11	flat() ដើម្បីបង្រួម array
12	flatMap() ដើម្បីទទួលបាន array ថ្មី បន្ទាប់ពីបង្រួម array
13	forEach() ហេរិមុខដាសម្រាប់ធានាតុកនិមួយនៃ array
14	Includes() Return តើម្លៃ boolean ប្រសិនបើ array មានធានាតុជាក់លាក់
15	indexOf() ត្រឡប់ first (least) index នៃធានាតុនៅតូចជាមួយស្ថិតិថ្មីដើម្បីផ្តល់បញ្ជាក់ ឬ -1 ប្រសិនបើកមិនមែន
16	join() ជាប់ធានាតុទាំងអស់នៃ array ទៅក្នុង string
17	Keys() ត្រឡប់ការធ្វើឡើងវិញ array ដើម្បីលើកការណ៍ key សម្រាប់ធានាតុ array និមួយ។

18	lastIndexOf() ប្រសិនបើកមានវត្ថុមានយ៉ាងហេចណាស់ម្នង) ។ ប្រសិនបើធាតុមិនមិនមាននៅក្នុង array រាល់ត្រឡប់ -1
19	map() បង្កើត array ថ្មីដើម្បីយនឹងលទ្ធផលនៃការហេរមុខងារដែលបានផ្តល់នៅលើគ្រប់ធាតុនៅក្នុង array នេះ
20	pop() លួបធាតុចុងក្រោយចេញពី array ហើយត្រឡប់ធាតុនេះ
21	push() បន្ថែមធាតុម្នាយ ប្រចើនទៅចុងបញ្ហាប់នៃ array ម្នាយ ហើយត្រឡប់ length ថ្មីនៃ array
22	reduce() អនុវត្ថុមុខងារក្នុងពេលដំណាលភ្លាមប្រចាំងនឹងតម្លៃពីនៃ array (ពីផ្ទេរទៅស្តាំ) ដើម្បីកាត់បន្ថូយរាយជាដាក់ម្នារម្នាយ
23	reduceRight() អនុវត្ថុមុខងារម្នាយក្នុងពេលដំណាលភ្លាមប្រចាំងនឹងតម្លៃពីនៃ array (ពីស្តាំទៅផ្ទេរ) ដើម្បីកាត់បន្ថូយរាយជាដាក់ម្នារម្នាយ
24	reverse() បញ្ចាំលំដាប់នៃធាតុនៃ array- ទី១ ត្រាយជាបុងក្រាយ ហើយបុងក្រាយត្រាយជាចិះ១
25	shift() យកធាតុទីម្នាយចេញពី array ហើយត្រឡប់ធាតុនេះ
26	slice() ស្របដំឡើកនៃ array ម្នាយ ហើយត្រឡប់ array ថ្មី
27	some() Returns true ប្រសិនបើយ៉ាងហេចណាស់ធាតុម្នាយនៅក្នុង array នេះបំពេញមុខងារសាកល្បងដែលបានផ្តល់
28	toSorted() តាមរយៈបាត់ក្នុង array ក្នុងលំដាប់ជាក់លាក់ម្នាយ
29	sort() តាមរយៈបាត់ក្នុង array
30	splice() បន្ថែម and/or ដកធាតុចេញពី array
31	toLocaleString() ដើម្បីបង្កើតក្នុងធាតុ array ទៅជា string
32	toSpliced() ក្រើមដើម្បីក្រើម array ដើម្បីដកចេញ បុងនូវសាតុដែលមានស្រាប់ and/or បន្ថែមធាតុថ្មី
32	toString() Returns string ដែលត្រួតពិនិត្យបានក្នុង array
33	unshift() បន្ថែមធាតុម្នាយ ប្រចើនទៅដោកខាងមុខនៃarray ហើយត្រឡប់ length ថ្មីនៃarray
34	values() ដើម្បីទទួលបាន iterator ដែលមានតម្លៃនៃ array index នីមួយៗ
35	with() វិធីសាស្ត្រនេះ returns array ថ្មីដើម្បីយកទីតាំង index ដែលបានផ្តល់ឱ្យដូចសារដោយតម្លៃដែលបានផ្តល់ឱ្យ

1. ឧចាបរណ៍៖ Array at() Method

```
<p id="demo"></p>
<script>
  const animals = ["Lion", "Cheetah", "Tiger", "Elephant", "Dinosaur", "I
  let result = animals.at(4);
  document.getElementById("demo").innerHTML = result;
  // OUTPUT: Dinosaur
</script>
```

2. ຂົາທິ່ນດີກໍ : Array concat() Method

```
<script>
  const array1 = ["one", "two"];
  const array2 = ["three", "four", "five", "six"];

  const result = array1.concat(array2);
  document.write(result);
  //JavaScript Array concat() Method
  //one,two,three,four,five,six
</script>
```

3. ຂົາທິ່ນດີກໍ : Array copyWithin() Method

```
<script>
  const array = ['apple', 'banana', 'cherry', 'date', 'grapes'];
  array.copyWithin(3, 0);
  document.write(array);
  //apple,banana,cherry,apple,banana
</script>
```

4. ຂົາທິ່ນດີກໍ : Array entries() Method

```
<script>
  const animals = ["Lion", "Cheetah", "Tiger", "Elephant", "Dinosaur"];

  for (const i of animals.entries()) {
    document.write(i + "<br>");
  }
  //0,Lion
  // 1,Cheetah
  // 2,Tiger
  // 3,Elephant
  // 4,Dinosaur
</script>
```

5. ຂົາທິ່ນດີກໍ : Array every() Method

```
<script>
    const numbers = [2, 4, 6, 8, 10];
    const result = numbers.every(num => num % 2 === 0);
    document.write(result); //true
</script>
```

6. ඇඟරඩ් මෘදු මත Array fill() Method

```
<p id="demo"></p>
<script>
    const animals = ["Lion", "Cheetah", "Tiger", "Elephant"];
    document.getElementById("demo").innerHTML = animals.fill("Dinosaur");
    //Dinosaur,Dinosaur,Dinosaur,Dinosaur
</script>
```

7. ඇඟරඩ් මෘදු මත Array filter() Method

```
<script>
    const ages = [18, 25, 13, 16, 22, 15];

    const result = ages.filter(function (age) {
        return age >= 18;
    });
    document.write(result); //18,25,22
</script>
```

8. ඇඟරඩ් මෘදු මත Array find() Method

```
<script>
    const players = [
        { name: 'Kohli', age: 35 },
        { name: 'Ponting', age: 48 },
        { name: 'Sachin', age: 50 }
    ];

    const result = players.find(item => item.age > 40);
    document.write(JSON.stringify(result));
    //{"name":"Ponting","age":48}
</script>
```

9. ඇඟරඩ් මෘදු මත Array findLast() Method

```
<script>
    const numbers = [5, 8, 12, 15, 20, 4];
    const lastValue = numbers.findLast((num) => num > 10);
    document.write(lastValue); //20
</script>
```

10. ຊາຍរណີ່ : Array flat() Method

```
<p id="demo"></p>
<script>
  const animals = [1, 2, [3, [4, [5, 6]]]];
  const result = animals.flat();
  document.getElementById("demo1").innerHTML = result;
  //Expected Output: Array [1, 2, 3, Array [4, Array [5, 6]]]
  animals.flat(2);
  document.getElementById("demo2").innerHTML = result;
  //Expected Output: Array [1, 2, 3, 4, Array [5, 6]]]
</script>
```

11. ຊາຍເກົດີ້ : Array flatMap() Method

```
<p id="demo"></p>
<script>
  const numbers = [5, 10, 20, 30, 40];
  let result = numbers.flatMap( num => [num * 2]);
  document.getElementById("demo").innerHTML = result;
  //10,20,40,60,80
</script>
```

12. ຂາຍກາຽົງົດີ່ : Array forEach() Method

```
<p id="demo"></p>
<script>
  const animals = ["Lion", "Cheetah", "Tiger", "Elephant", "Dinosaur"];
  const newArray = [];
  animals.forEach((element) => {
    newArray.push(element);
  });
  document.getElementById("demo").innerHTML = newArray;
  //Lion,Cheetah,Tiger,Elephant,Dinosaur
</script>
```

13. ຂາບໂຄງກົດ : Array includes() Method

```
<p id="demo"></p>
<script>
  const animals = ["Lion", "Cheetah", "Tiger", "Elephant", "Dinosaur"]
  let result = animals.includes("Dinosaur");
  document.getElementById("demo").innerHTML = result; //true
</script>
```

14. ຂອບເຮົດໃຫຍ່ Array indexOf() Method

```
<p id="demo"></p>
<script>
  const numbers = [2, 5, 6, 2, 7, 9, 6];
  const result = numbers.indexOf(10);
  document.getElementById("demo").innerHTML = result; // -1
</script>
```

15. ຂອບເຮົດໃຫຍ່ Array join() Method

```
<p id="demo"></p>
<script>
  const animals = ["Lion", "Chetaah", "Tiger", "Elephant", "Dinosaur"];
  let result = animals.join(" and ");
  document.getElementById("demo").innerHTML = result;
  // Lion and Chetaah and Tiger and Elephant and Dinosaur
</script>
```

16. ຂອບເຮົດໃຫຍ່ Array keys() Method

```
<script>
  const numbers = [18, 25, 13, 16, 22, 15];
  const result = Array.from(numbers.keys());
  document.write(result); // 0,1,2,3,4,5
</script>
```

17. ຂອບເຮົດໃຫຍ່ Array lastIndexOf() Method

```
<p id="demo"></p>
<script>
  const numbers = [2, 5, 6, 2, 7, 9, 6];
  const result = numbers.lastIndexOf(2);
  document.getElementById("demo").innerHTML = result; // 3
</script>
```

18. උග්‍රයාදා සඳහා Array map() Method

```
<p id="demo"></p>
<script>
    const numbers = [5, 10, 15, 20];
    const result = numbers.map(multiplication);
    function multiplication(num){
        return num * 10;
    }
    document.getElementById("demo").innerHTML = result;//50,100,150,200
</script>
```

19. උග්‍රයාදා සඳහා Array pop() Method

```
<p id="demo"></p>
<script>
    const animals = ["Lion", "Cheetah", "Tiger", "Elephant"];
    animals.pop();
    document.getElementById("demo").innerHTML = animals;
    //Lion,Cheetah,Tiger
</script>
```

20. උග්‍රයාදා සඳහා Array push() Method

```
<script>
    const animals = ["Lion", "Cheetah", "Tiger", "Elephant"];
    animals.push("Dinosaur");
    document.write(animals); //Lion,Cheetah,Tiger,Elephant,Dinosaur
</script>
```

21. උග්‍රයාදා සඳහා Array .reduce() Method

```
<script>
    const numbers = [10, 20, 30, 40, 50];
    const sum = numbers.reduce((accumulator, currentValue) =>
        accumulator + currentValue, 0);
    document.write(sum); //150
</script>
```

22. උග්‍රයාදා සඳහා Array .reduceRight() Method

```

const numbers = [10, 20, 30, 40, 50];
const reversedNumbers = numbers.reduceRight((accumulator, currentValue)
  => {
  accumulator.push(currentValue);
  return accumulator;
}, []);
document.write(reversedNumbers); //50,40,30,20,10

```

23. උජාපරිභාව : Array reverse() Method

```

<p id="demo"></p>
<script>
  const animals = ["Lion", "Cheetah", "Tiger", "Elephant"];
  document.getElementById("demo").innerHTML = animals.reverse();
  //Elephant,Tiger,Cheetah,Lion
</script>

```

24. උජාපරිභාව : Array shift() Method

```

<p id="demo"></p>
<script>
  const animals = ["Lion", "Cheetah", "Tiger", "Elephant"];
  animals.shift();
  document.getElementById("demo").innerHTML = animals;
  //Cheetah,Tiger,Elephant
</script>

```

25. උජාපරිභාව : Array slice() Method

```

<p id="demo"></p>
<script>
  const animals = ["Lion", "Cheetah", "Tiger", "Elephant", "Dinosaur"];
  let result = animals.slice(2);
  document.getElementById("demo").innerHTML = result;
  //Tiger,Elephant,Dinosaur
</script>

```

26. උජාපරිභාව : Array.toSorted() Method

```

<script>
  let numbers = [4, 2, 5, 1, 3];
  let result = numbers.toSorted();
  document.write(result); //1,2,3,4,5
</script>

```

27. උජාපරිභාව : Array sort() Method

```
const array = [10, 25, 5, 20, 15];
array.sort((a,b) => {return a-b}); //array.sort((a,b) => {return b-a});
document.getElementById("demo").innerHTML = array;
//5,10,15,20,25
```

28. ຂາຍົມເດີກ່າວໂສ : Array splice() Method

```
<p id="demo"></p>
<script>
  const animals = ["Lion", "Cheetah", "Tiger", "Elephant", "Dinosaur"];
  const result = animals.splice(2);
  document.getElementById("demo").innerHTML = animals;//Lion,Cheetah
</script>
```

29. ຂາຍົມເດີກ່າວໂສ : Array toLocaleString() Method

```
<script>
  const currency = [1000, 2000, 3000];
  document.write(currency.toLocaleString('en-US',
  { style: 'currency', currency: 'USD' }));
  //,$1,000.00,$2,000.00,$3,000.00
</script>
```

30. ຂາຍົມເດີກ່າວໂສ : Array toSpliced() Method

```
<script>
  let fruits = ['apple', 'banana', 'cherry', 'dates'];
  let result = fruits.toSpliced(2);
  document.write(result);//apple,banana
</script>
```

31. ຂາຍົມເດີກ່າວໂສ : Array toString() Method

```
<p id="demo"></p>
<script>
  const MatrixArray = [
    [10, 20, 30],
    [40, 50, 60],
    [70, 80, 90],
  ];
  let result = MatrixArray.toString();
  document.getElementById("demo").innerHTML = result;
  //10,20,30,40,50,60,70,80,90
</script>
```

32. ຂາຍົມເດີກ່າວໂສ : Array unshift() Method

```

<p id="demo1"></p>
<p id="demo2"></p>
<script>
    const animals = ["Tiger", "Elephant", "Dinosaur"];
    animals.unshift("Lion", "Cheetah");
    document.getElementById("demo1").innerHTML = animals;
    //Lion,Cheetah,Tiger,Elephant,Dinosaur
    document.getElementById("demo2").innerHTML = animals.length;
    //5
</script>

```

33. ឧទាហរណ៍៖ Array values() Method

```

let array = ['apple', 'banana', 'cherry'];
let iterator = array.values();
document.write(iterator.next().value, "<br>");
document.write(iterator.next().value, "<br>");
document.write(iterator.next().value);
/*apple
banana
cherry*/

```

34. ឧទាហរណ៍៖ Array with() Method

```

<script>
    let numbers = [11, 22, 44, 55];
    let result = numbers.with(2, "Tutorials_JS");
    document.write(result); //11,22,Tutorials_JS,55
</script>

```

- **The Number Object:** តំណាងឱ្យទិន្នន័យជាលេខជាលេខ floating-point numbers ។
 - **Number Methods**

Sr.No.	Name & Description
1	toExponential() តំណាងឱ្យលេខនៅក្នុងសញ្ញាអិបស្ថុណាដៃសៀល (exponential notation) Syntax: toExponential(fractionDigits)
2	toFixed() ធ្វើឡើងត្រាយលេខដែលមានចំនួនខ្ពស់ជាក់លាក់មួយនៅខាងស្តាំទៅកាត់។ Syntax: toFixed(digits)
	toLocaleString() ប្រើបើអ្នកតំណាងឱ្យលេខជាប្រភេទ string ដោយប្រើប្រាស់ការសាមូលដ្ឋាន

	Syntax: <code>toLocaleString(locales, options)</code>
4	toPrecision() កំណត់ចំនួនខ្ពស់សរុប (កប់បញ្ចូលចាំងខ្ពស់នៅខាងមុខ និងខាងស្តាំនៃទសភាគ) ដើម្បីបង្ហាញព្រលេខម្មួយ Syntax: <code>toPrecision(precision)</code>
5	toString() Return string តិ៍ណាងនៃតម្លៃលេខ Syntax: <code>toString(radix)</code>
6	valueOf() Return តម្លៃលេខ Syntax: <code>valueOf()</code>

▪ The Boolean Object

តំណាងឱ្យតម្លៃគឺ "true" ឬ "false" ។ អ្នកអាចបង្កើត boolean object ដោយប្រើ Boolean constructor ជាមួយនឹងពាក្យរីនេះ: 'new'

```
<p id = "output"> </p>
<script>
    let res = Boolean(100 > 90);
    document.getElementById("output").innerHTML = "Boolean(100 > 90) : "
    + res + "<br>";
    res = Boolean(100 < 90);
    document.getElementById("output").innerHTML = "Boolean(100 < 90) : "
    + res + "<br>";
    res = 100 == 90;
    document.getElementById("output").innerHTML = "100 == 90 : "
    + res + "<br>";
    /*
        Boolean(100 > 90) : true
        Boolean(100 < 90) : false
        100 == 90 : false
    */
</script>
```

- Strings អាប្រក្សាបានបង្កើតជា objects ដោយប្រើ String() constructor បុងបុញបទដោយប្រើ string literals
 - វិធីសាស្ត្រ instance ត្រូវបានហេរដោយប្រើ instance នៃ String class

Sr.No.	Name & Description
--------	--------------------

1	at() ប្រើដើម្បីទាញយក single character ពី string នៅទីតាំងដែលបានបញ្ជាក់ Syntax: <code>at(index)</code>
2	charAt() return string បីដែលមាន single character ពី string ដើម្បីនៅ <code>index</code> ដែលបានផ្តល់ឱ្យ Syntax: <code>charAt(index)</code>
3	charCodeAt() ត្រួចបែកចំនួនគត់ចរន្តនៃពី 0 ដល់ 65535 នៅ <code>index</code> ដែលបានបញ្ជាក់ Syntax: <code>charCodeAt(index)</code>
4	codePointAt() ត្រួចបែកចំនួនគត់មិនអវិជ្ជមានដែលតាំងចុងឱ្យបំណុល Unicode នៃក្នុងក្នុង string ដែលបានដើម្បីនៅ <code>index</code> ដែលបានបញ្ជាក់ Syntax: <code>codePointAt(index)</code>
5	concat() ភាពបែកចុះដើម្បីអក្សរពី ឬប្រើប្រាស់ទៅ string បីម្នាយ Syntax: <code>concat(str1, str2,....., strN)</code>
6	endsWith() ប្រើដើម្បីកំណត់ថាជាន់ string ត្រូវបានបញ្ចប់ដោយត្រូវក្នុងដែលបានបញ្ជាក់ (<code>substring</code>)
7	indexOf() ប្រើដើម្បីស្វែងរក <code>index</code> នៃការកើតឡើងដើម្បីនៅ <code>substring</code> ដែលបានបញ្ជាក់នៅក្នុង string ដើម Syntax: <code>indexOf(searchString, position)</code>
8	includes() ដើម្បីពិនិត្យមិនមែនជាជាន់ string ម្នាយនៅក្នុង string ដើម្បីទៅតិច Syntax: <code>includes(searchString, position)</code>
9	localeCompare() ប្រើដើម្បីប្រែបង្រៀប strings ពីនៅក្នុង <code>current locales</code> ដែល browser បែងចែកកំពុងប្រើបច្ចុប្បន្ន Syntax: <code>localeCompare(compareString, locales, options)</code>
10	match() ប្រើដើម្បីផ្តល់ព័ត៌មាន (regular expression) រាយការណ៍ string Syntax: <code>match(regexp)</code>
11	matchAll() ទាញយកពាក្យដែលបានបញ្ជាក់នៃលទ្ធផលទាំងអស់ដែលត្រូវនឹងការរោងចាយជម្លាត់ដែលបានបញ្ជាក់ (<code>regex</code>) ក្នុង string បច្ចុប្បន្ន Syntax: <code>matchAll(regex)</code>
12	normalize() ប្រើដើម្បីទាញយកទម្រង់ Unicode normalization នៃ string បញ្ចប់ដែលបានផ្តល់ឱ្យ Syntax: <code>normalize(form)</code>
13	padEnd() ដើម្បីបន្ថែមទាមតារាប់ទៅ string បច្ចុប្បន្នដែលមាន string ដើម្បីត្រូវនៅចុងបញ្ចប់ Syntax: <code>padEnd(targetLength, padString)</code>
14	padStart() ដើម្បីបន្ថែមទាមតារាប់ទៅ string បច្ចុប្បន្នជាមួយនឹង string ដើម្បីត្រូវនៅពេលបានដើម្បី Syntax: <code>padStart(targetLength, padString)</code>
15	raw() ត្រួចបែកទម្រង់ string ដើម្បីនៅក្នុងដែលបានផ្តល់ឱ្យតាម <code>literal</code> Syntax: <code>String.raw(strings, sub1, sub2, /* ..., */ subN)</code>
16	repeat() ដើម្បីទូទាត់បាន string បីដែលបានលេខ N នៃច្បាប់ចម្លងនៃ string បច្ចុប្បន្ន Syntax: <code>repeat(count)</code>
17	replace() ប្រើដើម្បីស្វែងរកការផ្តល់ព័ត៌មានរាយការណ៍ regular expression និង string ហើយដំឡើស <code>substring</code> ដែលត្រូវត្រូវជាមួយបីម្នាយ Syntax: <code>replace(pattern, replacement)</code>

18	replaceAll() ប្រើដើម្បីស្វែងរកការផ្តល់រាង regular expression និង string ហើយជួន substring ដែលត្រូវត្រូវចាត់ដាមួយប្រើមួយ Syntax: <code>replaceAll(pattern, replacement)</code>
19	search() ប្រពិបត្តិការស្វែងរកការផ្តល់រាង regular expression និង string ដែលបានបញ្ជាក់ Syntax: <code>search(regexp)</code>
20	slice() ប្រើដើម្បីស្រាវជ្រាវ sub-string ពី string ដើម ហើយត្រួតទូប់ string បី Syntax: <code>slice(indexStart, indexEnd)</code>
21	split() ប្រើដើម្បីបងចំការងារ string ទាំងអស់ជាពាក្យជាប់នៃ substrings ដោយផ្តល់តម្លៃថាទាំងអស់ជាក់លាក់មួយ Syntax: <code>split(separator, limit)</code>
22	substr() ត្រួតទូប់ថាគ្នុង string ដែលបានបងចំឡើងនៅទីតាំងដែលបានបញ្ជាក់តាមរយៈចំណាំនូវក្នុងដែលបានបញ្ជាក់ Syntax: <code>substr(start, length)</code>
23	substring() ត្រួតទូប់ថាគ្នុង string រាង indexes ពីនូវក្នុង string Syntax: <code>substring(indexStart, indexEnd)</code>
24	toLocaleLowerCase() ត្រួតទូប់ថាគ្នុង string ត្រូវបានបំប្លែងទៅជាអក្សរក្រុចបាប Syntax: <code>toLocaleLowerCase(locales)</code>
25	toLocaleUpperCase() ត្រួតទូប់ថាគ្នុង string ត្រូវបានបំប្លែងទៅជាអក្សរក្រុចបាប Syntax: <code>toLocaleUpperCase(locales)</code>
26	toLowerCase() ត្រួតទូប់ថាគ្នុង string ដែលបានបំប្លែងទៅជាអក្សរក្រុចបាប Syntax: <code>toLowerCase()</code>
27	toString() ត្រួតទូប់ថាគ្នុង string ដែលត្រូវបានបំប្លែងទៅជាអក្សរក្រុចបាប Syntax: <code>toString()</code>
28	toUpperCase() ត្រួតទូប់ថាគ្នុង string ដែលបានបំប្លែងទៅជាអក្សរក្រុចបាប Syntax: <code>toUpperCase()</code>
29	trim() រាយការណ៍: white spaces ចេញពីចុងទាំងពីរ Syntax: <code>trim()</code>
30	trimEnd() រាយការណ៍: white spaces ចេញពីដំបូង
31	trimStart() រាយការណ៍: white spaces ចេញពីចុង
32	valueOf() ត្រួតទូប់តម្លៃបច្ចេក (primitive value)នៃ object ដែលបានបញ្ជាក់ Syntax: <code>valueOf()</code>

- **The Date Object:** គឺជាដatatype បាន built ឡើងនៅក្នុងភាសា JavaScript ។ Date objects ត្រូវបានបង្កើតជាមួយនឹង `new Date()` ។ methods ភាគចំនួនដែលអនុញ្ញាតឱ្យអ្នក get និង set ជារៀងរាល់ជាមួយនឹង `year, month, day, hour, minute, second, millisecond` ដោយប្រើពេលវេលាក្នុងស្រុក (local time) ឬ UTC (universal, or GMT) ។
- **Syntax:**

```

new Date( )
new Date(milliseconds)
new Date(datestring)
new Date(year,month,date[,hour,minute,second,millisecond ])

```

ឧចាបរណ៍៖ ការបង្កើតកាលបរិច្ឆេទប្រើប្រាស់ Date Object

```

<p id = "output"> </p>
<script>
    const date = new Date();
    document.getElementById("output").innerHTML =
    "Today's date is : " + date;
    //Today's date is : Mon Jul 01 2024 15:03:51 GMT+0700 (Indochina Time)
</script>

```

4.4. របៀបប្រកាស One Dimensional Array

JavaScript one Dimensional Array គ្រឿនប្រើដើម្បីរក្សាតម្លៃជាប្រើប្រាស់នៃប្រភេទទិន្នន័យ ធ្វើដោយគ្មាន (បើចាំបាច់) ត្រូវអប់រំតម្លៃ។

- ឧបាទក្រាមនេះគឺជា syntax ដើម្បីបង្កើត array

```

<script>
    // Using array literal
    var array_name= [item1, item2, ..., itemN];
    // Using new Keyword
    var array_name = new Array(item1, item2, ..., itemN);
</script>

```

ឧចាបរណ៍៖ ដោយប្រើ Array Literal

```

<script>
    var myCar = ["Tesla", "BYD", "TOYOTA"];
    document.write("I like " +myCar[0]);
    //I like Tesla
</script>

```

ឧចាបរណ៍៖ ដោយប្រើ new Keyword

```

<script>
    var myCar = new Array("BYD", "Tesla", "Jetour");
    document.write("I love "+myCar[2]);//I love Jetour
</script>

```

ឧទាហរណ៍៖ ដោយប្រើ Index Value

```
<script>
var myCar = [];
myCar[0] = "Tesla";
myCar[1] = "BYD";
myCar[2] = "Jetour";
document.write("I too like " + myCar[2]);
// I too like Jetour
</script>
```

ឧទាហរណ៍៖ ដោយប្រើ Access Full Array

```
<script>
var myCar = ["BYD", "Tesla", "Jetour"];
document.write("I have " + myCar);
// I have BYD, Tesla, Jetour
</script>
```

ឧទាហរណ៍៖ ដោយប្រើ Array ជាមួយប្រកែទិន្នន័យផ្សេងៗគ្នា

```
<script>
var myCar = ["BYD", "Tesla", {"car": "Jetour", "color": "red"}];
document.write("I like " + myCar[2].color); // I like red
</script>
```

4.5. របៀបប្រកាស Multi Dimensional Array

Javascript មិនគ្រាន់ទៅតាមច្បាស់ប៉ុណ្ណោះ (multi-dimensional arrays) ដូចបាន C, C++ ដោយមិនបានគ្រាន់ទៅតាមច្បាស់ប៉ុណ្ណោះ (multi-dimensional arrays) តុលាត់ JavaScript អ្នកត្រូវប្រើ array តុលាត់ array ម្ខយា អ្នកត្រូវតែប្រកាសបេនាសម្ព័ន្ធជាក់លាក់នៃ multi-dimensional array មុនពេលចាប់ផ្តើមតម្លៃទៅរា

ឧទាហរណ៍

```
<html>
<body>
    <script>
        var myCar = [["Tesla", "3", "White"], ["BYD", "Sedans", "red"]];

        document.write("Car - " +myCar[0][0]);
        document.write("<br>Model - " +myCar[0][1]);
        document.write("<br>Color - " +myCar[0][2]);
    </script>
</body>
</html>
```

Car - Tesla
Model - 3
Color - White



ឧទាហរណ៍ នឹងបាន error ត្រូវយើងបានចាប់ផ្តើមដូរទីបីដោយមិនបានប្រកាសវា

```

<script>
//Only two rows
var myCar = [[],[]];

// first row
myCar[0][0] = "Jaguar";
myCar[0][1] = "White";

// Second row
myCar[1][0] = "BMW";
myCar[1][1] = "red";

// Third row (Error Occurs)
myCar[2][0] = "Benz";
myCar[2][1] = "Brown";

document.write("Car1 - " +myCar[0][0]);
document.write("<br>Car2 - " +myCar[1][0]);
document.write("<br>Car3 - " +myCar[2][0]);
</script>

✖ Uncaught
TypeError: Cannot set properties of undefined (setting '0')
at index.html:22:13
> |

```

4.6. របៀបប្រកាស Associative Array

Array ដែលមានឈ្មោះ **named indexes** គ្រឿងានគេហ៊ា **Associative Array**។

JavaScript មិនមាន concept នៃ Associative Array ទេ ប៉ុន្តែវាបាត់ទូកពួកគេជា Object ម្មយ

(as array itself an object)។

ឧទាហរណ៍៖ ដោយប្រើ **Associative Array**

```

<script>
var bike = [];
// bike[key] = value;
bike["company"] = "Honda";
bike["model"] = "CBR 1000RR";
bike["color"] = "Red";
document.write("I bought " +bike["model"]);
//I bought CBR 1000RR
</script>

```

ឧទាហរណ៍៖ ដោយប្រើ **Object**

```
<script>
    var bike = {
        company: "Honda",
        model: "CBR 1000RR",
        color: "red"
    };
    document.write("I bought " +bike.model);
    //I bought CBR 1000RR
</script>
```

សេវាសម្រាក:

Functions និង Events

5. តាមបញ្ជីៗ Functions

Function នៃក្នុង JavaScript គឺជាក្រុមទៀតដែលអាចប្រើប្រាស់ឡើងវិញបាន ដែលអាចហោចានត្រប់ទីកន្លែងនៃក្នុងកម្មវិធីរបស់អ្នក។ វាលូបបំបាត់តម្លៃការនៃការសេវាក្នុងដែលម្នាច់ហើយម្នាច់ឡើត។ វាបានយកម្នាច់ការសេវាក្នុងការសេវាក្នុងមួយខ្លួន។ **function** អនុញ្ញាតឱ្យអ្នកសេវាក្នុងវិធីបែងចែកកម្មវិធីជំម្លែយទៅជាអាជីវកម្ម។

5.1. ការកំណត់ Functions

មុនពេលយើងប្រើ **function** ម្នាច់ យើងត្រូវកំណត់វា វិធីសមញ្ញបំផុតដើម្បីកំណត់ **function** នៃក្នុង JavaScript គឺដោយប្រើពាក្យគិន៍៖ **function keyword** អមដោយណ៍៖ **function** តែម្នាច់តែម្នាច់ (**unique function name**) ហព្វើថាការប្រើប្រាស់ឡើងត្រូវបានបញ្ជាផ្ទាល់ឡើង។

- Syntax: ដើម្បីកំណត់**function** នៃក្នុង JavaScript មានដូចខាងក្រោម៖

```
function functionName(parameter-list) {
    statements
}
```

ឧបាហរណ៍៖ កំណត់ **function** ម្នាច់បែងថា **sayHello** ដែលមិនប្រើបានប្រើប្រាស់ឡើងត្រូវបានបញ្ជាផ្ទាល់ឡើង។

```
function sayHello() {
    alert("Hello there");
}
```

5.2. ការកំណត់ Function expression

- Function expression នៃក្នុង JavaScript អនុញ្ញាតឱ្យអ្នកកំណត់ **function** ជាការនេរម្នាច់។ Function expression គឺស្របជោងនឹងការ **anonymous function declaration** ។ Function expression អាចប្រើបានកំណត់ថាមីរោច (**variable**) ។
- Syntax នៃ **function expression**

```
const varName = function (parameter-list) {
    statements
};
```

ឧបាហរណ៍ខាងក្រោម៖

```
const myFunc = function (x, y){
    return x + y;
};
```

ឧទាហរណ៍៖ sayHello() function បង្ហាញប្រអប់សារ "Hello there!"

```
<html>
<head>
    <script type="text/javascript">
        function sayHello() {
            alert("Hello there!");
        }
    </script>
</head>
<body>
    <p>Click the following button to call the function</p>
    <form>
        <input type="button" onclick="sayHello()" value="Say Hello">
    </form>
    <p> Use different text in the write method and then try... </p>
</body>
</html>
```

5.3. ការកំណត់ Function parameters

- Function parameters នឹងក្នុង JavaScript គឺជាមធ្យបែវដែលបានរាយបញ្ជី (variables listed) ក្នុងវិភាគក្នុងការកំណត់function ។ function មួយអាចមានចាត់កំមេថ្មត្របើនដែលបំបែកជាយសញ្ញារក្បែស ។ function arguments គឺជាកាលម៉ែនដែលត្រូវបានបញ្ជីឡើង ការនៅពេលវាត្រូវបានហេតុ ។ យើងកំណត់ function រាយបញ្ជីចាត់កំមេថ្មត្រ ហើយហេតុ function ផ្តល់ការត្រូវបានហេតុ ។
- Syntax នៃ function expression

```
function functionName (parameter1, parameter2, parameter3) {
    //statements
}
```

- JavaScript function arguments គឺជាមធ្យបែវ ប្រុតម៉ែនដែលបានបញ្ជីឡើង នៅពេលវាត្រូវបានហេតុ ។

```
functionName (10, b, 'Hello');
```

ឧចាបរណី៖ Function Parameters និង Arguments

```
<p id = "output"> </p>
<script>
    function mult(a, b, c) {
        let res = a * b * c;
        return res;
    }
    let ans = mult(2, 3, 4);
    document.getElementById("output").innerHTML =
        "The multiplication of 2, 3, and 4 is " + ans;
        //The multiplication of 2, 3, and 4 is 24
</script>
```

ឧចាបរណី៖ Argument Object

```
<p id = "output"> </p>
<script>
    function merge() {
        let final = "";
        for (let p = 0; p < arguments.length; p++) {
            final += arguments[p] + " ";
        }
        return final;
    }
    let ans = merge("Hi", "I", "am", "John!");
    document.getElementById("output").innerHTML =
        "The merged string is: " + ans;
        //The merged string is: Hi I am John!
</script>
```

ឧទាហរណ៍៖ Passing Arguments by Value

```
<p id = "output"> </p>
<script>
    const output = document.getElementById("output");
    function update(val1, val2) {
        val1 = 20;
        val2 = 30;
    }

    var val1 = "Hello";
    var val2 = "World";

    output.innerHTML += "Before calling the function! <br>";
    output.innerHTML += "val1 = " + val1 + ", val2 = " + val2 + "<br>";

    update(val1, val2);

    output.innerHTML += "After calling the function! <br>";
    output.innerHTML += "val1 = " + val1 + ", val2 = " + val2 + "<br>";
</script>
```

Before calling the function!
 val1 = Hello, val2 = World
 After calling the function!
 val1 = Hello, val2 = World

5.4. ការកំណត់ Default Parameters

- Default parameters នៃក្នុង JavaScript គឺជាលក្ខណៈពិសេសដែលអនុញ្ញាតឱ្យអ្នកបញ្ជាក់តម្លៃ default សម្រាប់ function parameter ។ concept នៃ default parameters ត្រូវបានរំណៀនៅក្នុង ES6 ។ យើងអាចចាប់ផ្តើម parameters ជាមួយនឹងតម្លៃ default values ។ ជូនឯងបានបង្កើតឡើងនៅក្នុង function ។ ត្រូវបានហេរដោយបានកំណត់ រាប់ពីតម្លៃ default value នៃ parameter នៅក្នុងfunction ។
- ស្ថិតិយល់ភាមរយៈឧទាហរណ៍ខាងក្រោម៖

```
function sum(p, q) {
    return p + q;
}
sum(10, 20); // 30
sum(10); // NaN
sum(); // NaN
```

■ Default Parameters Syntax

```
function functionName(param1 = defaultValue1, param2 = DefaultValue2, ...) {
    // Use parameters here
}
```

၃၂၁ပြောဂျီ၏ Default parameters

```
<p id = "output"> </p>
<script>
    let output = document.getElementById("output");
    function sum(p = 30, q = 40) {
        return p + q;
    }
    output.innerHTML += "sum(10, 20) -> " + sum(10, 20)+"<br>";//10+20=30
    output.innerHTML += "sum(10) -> " + sum(10) + "<br>";//10+40=50
    output.innerHTML += "sum() -> " + sum() + "<br>"; // 30+40=70
</script>
```

၃၂၁ပြောဂျီ၏ Passing Undefined Argument

```
<p id="output"> </p>
<script>
    let output = document.getElementById("output");
    function sum(p = 24, q = 26) {
        return p + q;
    }
    output.innerHTML += "sum(5, undefined) -> " +sum(5, undefined)+"<br>";
    // 5 + 26 = 31
    output.innerHTML += "sum(undefined) -> " + sum(undefined) + "<br>";
    // 24 + 26 = 50
</script>
```

၃၂၁ပြောဂျီ၏ Function expression as a default parameter

```
<p id = "output"> </p>
<script>
    let output = document.getElementById("output");
    function getNum() {
        return 5;
    }
    function mul(p = 5, q = getNum()) {
        return p * q;
    }
    output.innerHTML += "mul(10) -> " + mul(10) + "<br/>";//mul(10) -> 50
    output.innerHTML += "mul() -> " + mul() + "<br/>";//mul() -> 25
</script>
```

ឧទាហរណ៍ទី ៣ : Optional Parameters

```
<p id = "output"> </p>
<script>
    let output = document.getElementById("output");
    function func(p, q=10) {
        return p + q;
    }
    output.innerHTML += "func(10, 20) -> " + func(10, 20);
    //func(10, 20) -> 30
</script>
```

5.5. ការកំណត់ Function() Constructor

- Function() constructor អាចប្រើបានប្រើដើម្បីកំណត់ function នៅពេលដែលការប្ដូនអ្នកគួរតែប្រើ Function() constructor ដាក់យប្បុងប្រយ័ត្នប្រចាំថ្ងៃនៃការការពាយដែលគ្មានក្នុងក្នុង។
- Syntax ដើម្បីប្រើ Function() constructor ទៅបង្កើត object

```
const obj = new Function(arg1, arg2..., functionBody);
```

ឧទាហរណ៍ទី ៤ :

```
<p id = "output"> </p>
<script>
    const func = new Function("p", "q", "return p * q");
    document.getElementById("output").innerHTML =
    "The value returned from the function is: " + func(5, 7);
    //The value returned from the function is: 35
</script>
```

ឧទាហរណ៍ទី ៥ :

```
<p id = "output"> </p>
<script>
    const func = new Function("return 30 + 20");
    document.getElementById("output").innerHTML =
    "The value returned from the function is: " + func(); //50
</script>
```

ឧទាហរណ៍ទី ៦ :

```
<p id = "output"> </p>
<script>
    const add = new Function(
        "const sum = function (a, b) {return a+ b};  return sum",
    )();
    document.getElementById("output").innerHTML = add(5,10) // 15
</script>
```

5.6. ការកំណត់ Arrow Functions

- The arrow functions នៅក្នុង JavaScript អនុញ្ញាតឱ្យយើងបានឱ្យតិច function ខ្លួចជានិង anonymous function ។ Arrow functions ត្រូវបានសរសរដោយតាមពាក្យគឺថា "function" ។ Arrow functions ត្រូវបានណែនាំនៅក្នុង ES6 ។
- កាតខុសត្រាង syntax function expression និង arrow function

```
const varName = function(parameters) {
    // function body
};
```

```
const varName = (parameters) => {
    // function body
};
```

- Syntax arrow function

```
const varName = (p1, p2, ... pN) => Expression;
OR
const varName = (p1, p2, ...pN) => {
    // function body
};
```

ឧទាហរណ៍ : Arrow Function with Single Statement

```
<p id = "output"> </p>
<script>
    const divide = (x, y) => x / y;
    document.getElementById("output").innerHTML = divide(10, 5); //2
</script>
```

ឧទាហរណ៍ : Arrow Function with Multiple Statements

```
<p id = "output"> </p>
<script>
    const divide = (x, y) => {
        let res = x / y;
        return res;
    };
    document.getElementById("output").innerHTML = divide(10, 5); //2
</script>
```

ឧទាហរណ៍ទី១ : Arrow Functions Without Parameters

```
<p id = "output"> </p>
<script>
    //const greet = () => "Hello World!"; //same below
    const greet = () => {return "Hello World!";};
    document.getElementById("output").innerHTML = greet(); //Hello World!
</script>
```

ឧទាហរណ៍ទី២ : Arrow Function with Parameters

```
<p id = "output"> </p>
<script>
    const sum = (a, b, c, d) => {
        let sum = a + b + c + d;
        return sum;
    };
    let res = sum(10, 30, 45, 60);
    document.getElementById("output").innerHTML =
    "The sum of 10, 30, 45, and 60 is: " + res; // 145
</script>
```

ឧទាហរណ៍ទី៣ : Arrow Function with Default Parameters

```
<p id = "output"> </p>
<script>
    const output = document.getElementById("output");
    let isMul = true;
    const mul = (a = 10, b = 15) => a * b;
    output.innerHTML += "mul(5, 8) = " + mul(5, 8) + "<br>"; //40
    output.innerHTML += "mul(6) = " + mul(6) + "<br>"; //90
    output.innerHTML += "mul() = " + mul() + "<br>"; // 150
</script>
```

- អត្ថប្រយោជន៍នៃការប្រើប្រាស់ Arrow Functions
 - Shorter syntax – បន្ទាយទំហំក្នុងដើម្បីកំណត់មុខងារ (function) ។
 - Implicit return – ដើម្បីត្រួតពេញលទ្ធផលនៃការស្វែងរកនូវមិន arrow function ដែលមាន single statement តែម្មប្រើប្រាស់ keyword return នៅលើទី១។
 - Ease to use as expression – អាបីប្រើប្រាស់ដាក់ស្រួលដោកនេរភាព។

5.7. ការកំណត់ Variable Scope

- The variable scope នៅក្នុង JavaScript កំណត់ចំពោះការធាយស្រួលនិងលទ្ធការមិនមែនបានផ្តល់នូវការប្រើប្រាស់ដើម្បីកំណត់ថាបានការប្រើប្រាស់នៅក្នុង ឬជាប្រព័ន្ធដែលបានការប្រើប្រាស់នៅក្នុងវា។

ប្រតិបត្តិមិនមែនដោយកន្លែងដែលវាគ្វោរបានប្រកាស។

- ប្រភព variable scope នៃការប្រាកំ
 - Block scope
 - Function scope
 - Local scope
 - Global scope

ឧទាហរណ៍នៃ Block Scope

```
<p id = "output"> </p>
<script>
    if (true) {
        let a = 10;
        document.getElementById("output").innerHTML = "a = " + a;//a = 10
    }
    // a can't be accessed here
</script>
```

ឧទាហរណ៍នៃ Function Scope

```
<p id = "demo"> </p>
<script>
    const output = document.getElementById("demo");
    function func() {
        var x = 30; // function scope
        let y = 20; // Block scope
        const z = 10; // Block scope
        output.innerHTML += "x -> Inside the block = " + x + "<br>";
        output.innerHTML += "y -> Inside the block = " + y + "<br>";
        output.innerHTML += "z -> Inside the block = " + z + "<br>";
    }
    output.innerHTML += "x -> Outside the block = " + x + "<br>";
    // y and z can't be accessible here
}
func();
</script>
```

ឧទាហរណ៍ទី៣៖ Local scope

```
<p id = "demo"> </p>
<script>
    const output = document.getElementById("demo");
    function func() {
        let first = 34;
        var second = 45;
        const third = 60;

        output.innerHTML += "First -> " + first + "<br>";
        output.innerHTML += "Second -> " + second + "<br>";
        output.innerHTML += "Third -> " + third + "<br>";
    }
    func();
</script>
```

- The global variables នៅក្នុង JavaScript គឺជាអប់រំដែលត្រូវបានកំណត់នៅខាងក្រោម **function** ប្រចាំកាត់លាក់ណាមួយ។ អាចយកទៅប្រើបានពីគ្រប់ទិសទីក្នុងក្នុង។
- អ្នកអាចកំណត់ **global variables** ដោយប្រើពាក្យគ្រឿនេះ: **var, let, ឬ const**។
អប់រំដែលបានកំណត់ដោយមិនប្រើ **keywords var, let ឬ const** ណាមួយត្រូវជាប្រភេទ **global variables** ដោយស្ម័គ្រាន់។

ឧទាហរណ៍ទី៤៖ Global Scope

```
<p id = "demo"> </p>
<script>
    const output = document.getElementById("demo");
    var a = 10;
    let b = 20;
    function test() {
        output.innerHTML += "a -> Inside the function = " + a + "<br>";//10
        output.innerHTML += "b -> Inside the function = " + b + "<br>";//20
    }
    test();
    output.innerHTML += "a -> Outside the function = " + a + "<br>";//10
    output.innerHTML += "b -> Outside the function = " + b + "<br>";//20
</script>
```

5.8. ការប្រើប្រាស់ប្រភេទទី៤ Events

- នៅពេលដែល **page loads** រាយការណ៍ត្រូវបានគេហេត្តិថា **event**។ នៅពេលដែលអ្នកប្រើប្រាស់ចូលលើបូត្រូង **clicks** មួយ ការ **clicks** នៅក្នុង **event** មួយ។ ឧទាហរណ៍ធ្វើត្រូមមាន **event** ដូចជាការចូច **pressing any key, closing a window, resizing a window** ។

- អ្នកអកិវឌ្ឍនភាពរបៀ events ទាំងនេះដើម្បីប្រតិបត្តិការឆ្លើយតប (responses)
ដែលបានសរសើរក្នុង JavaScript ដែលធ្វើឱ្យ close windows
សារត្រូវបានបង្ហាញជាមួយអ្នកប្រើប្រាស់ (messages to be displayed to users)
បានពិនិត្យនិងផ្តល់ដំឡើតទិន្នន័យ (data to be validated) និងស្នើសុំការគ្រប់ប្រកច
នៃការឆ្លើយតបដោយទេរ្តៃត។
- Events គឺជាដំឡើកម្លាយនៃ Document Object Model (DOM) Level 3 ហើយការណាត់ HTML
មានសំណុំនៃ events ដែលអាច trigger ក្នុង JavaScript ។
- អ្នកប្រើប្រាស់អាចអនុវត្តតាម syntax ខាងក្រោម ដើម្បីប្រើ event handlers ជាម្លាយ HTML

```
<div eventHandler = "JavaScript_code"> </div>
```

ឧទាហរណ៍៖ Inline JavaScript with Event Handlers

```
<html>
<body>
    <h2> Click the button to Change its text's color </h2>
    <button onclick = "this.style.color='red'"> Click Me </button>
    <div id = "output"> </div>
</body>
</html>
```

Click the button to Change its text's color

Click Me

ឧទាហរណ៍៖ Function with Event Handlers

```
<style>
    #test {width: 600px; height: 100px; background-color: red;}
</style>
</head>
<body>
    <div id = "test"> </div> <br>
    <button onclick = "handleClick()"> Change Div Color </button>
    <script>
        function handleClick(event) {
            var div = document.getElementById("test");
            div.style.backgroundColor = "blue";
        }
    </script>
```

ឧទាហរណ៍ទី២ : Multiple Functions with Event Handlers

```

<head>
    <style>
        #test {font-size: 15px;}
    </style>
</head>
<body>
    <h2> Hover over the below text to customize the font. </h2>
    <div id = "test" onmouseenter = "changeFontSize(); changeColor();">
        Hello World!
    </div> <br>
    <script>
        function changeFontSize(event) {
            document.getElementById("test").style.fontSize = "25px";
        }
        function changeColor(event) {
            document.getElementById("test").style.color = "red";
        }
    </script>

```



- នេះគឺជាមធ្យប័ណ្ណមួយបំផុននៃ DOM events
 - Click – event នេះកើតឡើងនៅពេលដែលអ្នកប្រើបុចលើជាតិ HTML
 - Load – event នេះកើតឡើងនៅពេលដែលជាតិ HTML ត្រូវបាន loaded
 - Change – event នេះកើតឡើងនៅពេលដែលតម្លៃនៃជាតិ HTML ត្រូវបានផ្តល់បញ្ជី
 - Submit – event នេះកើតឡើងនៅពេលដែលទទួលទៅបញ្ចប់ HTML ត្រូវបានបញ្ចប់

ឧទាហរណ៍ទី៣ : onclick Event Type

```

<script>
    function sayHello() {
        alert("Hello World")
    }
</script>
</head>
<body>
    <p>Click the following button and see result</p>
    <form>
        <input type = "button" onclick = "sayHello()" value = "Say Hello" />
    </form>
</body>

```

ឧចាបរណ៍ទៅលើ onkeydown Event Type

```
<p> Enter charater/s by pressing any key </p>
<input type = "text" onkeydown = "customizeInput()">
<script>
    function customizeInput() {
        var ele = document.getElementsByTagName("INPUT")[0];
        ele.style.backgroundColor = "yellow";
        ele.style.color = "red";
    }
</script>
```

Enter charater/s by pressing any key

d

ឧចាបរណ៍ទៅលើ onmouseenter and onmouseleave Events

```
<div id = "text" style = "font-size: 20px;" 
    onmouseenter = "changeRed()" onmouseleave = "changeBlack()"
    Hover over the text.
</div>
<script>
    function changeRed() {
        document.getElementById("text").style.color = "red";
    }
    function changeBlack() {
        document.getElementById("text").style.color = "black";
    }
</script>
```

- HTML 5 Standard DOM Events

Attribute	Value	Description
Onabort	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលដែលការធ្វើក audio ឬ video ត្រូវបានបញ្ចប់(aborted) នៅក្នុង Loading
onafterprint	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលដែលបាន Load ហើយ Printed ឱ្យ (The keyboard shortcut Ctrl+P prints a page)
onbeforeonload	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលដែលតម្លៃខ្សោចកចេញ ទំនាក់ទំងមូល
onbeforeprint	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលដែលដែលវារ Load មុន Print
onblur	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលដែលដែលជក cursor ចាកចេញពី input field
oncanplay	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល media អាចចាប់ផ្តើម Play
onchange	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល element ផ្ទា ស់ប្បរ
onclick	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល click

ondblclick	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល click mouse double-click
onerror	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលកំបាតសកើតឡើង (error occur)
onfocus	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល input field បង្ហាញ focus
fullscreenchange	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល ចុចបូគុង ដើម្បីបាន fullscreen
oninput	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល សរសអក្សរកុង input tex
onhaschange	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល ចុចបូគុង ដើម្បីបង្ហាញ page
oninvalid	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល element គឺ invalid
onkeydown	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល ចុច Keypress នៅក្នុង Input
onkeypress	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល ចុច pressed និង released
onkeyup	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល released បន្ទាប់ពីសរសួបកាលក្នុង input
onload	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល ចូល Page និង Load
onmessage	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល message គឺ triggered
onmousedown	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលចុច mouse
onmousemove	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលចុច mouse pointer ផ្ទាសប្បរ
onmouseout	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលចុច mouse pointer ជាក់លើ element
onmouseover	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលចុច mouse pointer គឺកំណើលើ element
onpause	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលចុចលើ media data គឺ paused
onplay	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេលចុចលើ media data គឺកំពុងដំណើរការ
onplaying	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល media data គឺកំពុងដំណើរការ
onresize	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល window គឺ resized
onselect	script	ព្រឹត្តិការណ៍កើតឡើងនៅពេល element គឺ selected

ແຈ້ງລາຍເຊີວ:

Form Validation

6. ຂາຍຄໍ້ນຫາແຈ້ງ Form Validation

ជាជម្រើសការកែវគ្រឿងនៃមេដាប់ទូទាត់ (server) បន្ទាប់ពី client បានបញ្ចប់ទិន្នន័យចាំបាច់ទាំងអស់ រូបចុចបូតុងបញ្ញនា ប្រសិនបើទិន្នន័យដែលបានបញ្ចប់ដោយម៉ាសីន client មិនត្រឹមត្រូវ បុគ្គល់តែបានតែ បុខ្នោះពីកំណាមានណាមួយ នោះម៉ាសីនមេនឹងត្រូវបញ្ចប់ទិន្នន័យទាំងអស់ត្រូវដែរទៀត ជាបុរាណ ហើយស្មើសំខីទម្រង់ត្រូវបានបញ្ញនា ឡើងវិញជាមួយនឹងពីកំណាមានត្រឹមត្រូវ។

JavaScript ផ្តល់នូវវិធីម្យយដើម្បីធ្វើ validate form's data នៅលើកុំព្យូទ័រ client's មុនពេលបញ្ចូនការទៅម៉ាស៊ីនមេ (web server)។ Form validation ទម្រង់ជាងួរអនុវត្តមុខងារពីរៀះ

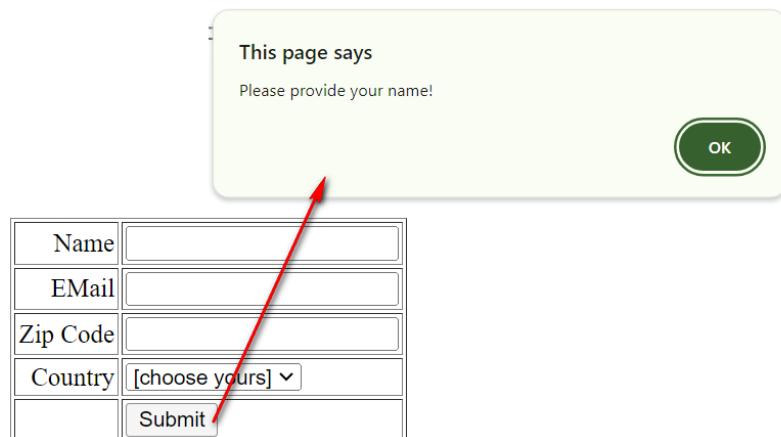
- **Basic Validation** - ជាដំបូង ទម្រង់ត្រូវកែតិនិត្យ ដើម្បីប្រាកដថា រាល់រាល់ចាំបាច់ទាំងអស់ត្រូវបានបំពេញ។ វាគ្មោះការគ្រាន់តែផ្តើលដីផ្សេងកាត់ **filled in** នឹមួយនក្នុងទម្រង់ ហើយពិនិត្យមើលទិន្នន័យ។
 - **Data Format Validation** – ទីតី ទិន្នន័យដើម្បីបានបញ្ហាលត្រូវកែតិនិត្យរកមើលទម្រង់និងតម្លៃត្រឹមត្រូវ។ ក្នុងរបស់អ្នកត្រូវក្នុមបញ្ហាលតក្នុងធាតុសម្រប (**appropriate logic**) ដើម្បីតែនឹងការបញ្ហាលត្រឹមត្រូវនេះទិន្នន័យ។
 - **Example:** លើកខាងក្រោមណាមួយដើម្បីយល់ពីដំណឹងការនៃការបញ្ហាក់។
នេះគឺជាអ្នកប្រើប្រាស់សាមញ្ញក្នុងទម្រង់ **html**

Name	
EMail	
Zip Code	
Country	[choose yours] ▾
	Submit

```
1 <html>
2   <head>
3     <title>Form Validation</title>
4     <script type = "text/javascript">
5       // Form validation code will come here.
6     </script>
7   </head>
8
9   <body>
10    <form action = "/cgi-bin/test.cgi" name = "myForm" onsubmit = "return(v
11      <table cellspacing = "2" cellpadding = "2" border = "1">
12
13        <tr>
14          <td align = "right">Name</td>
15          <td><input type = "text" name = "Name" /></td>
16        </tr>
17        <tr>
18          <td align = "right">EMail</td>
19          <td><input type = "text" name = "EMail" /></td>
20        </tr>
21        <tr>
22          <td align = "right">Zip Code</td>
23          <td><input type = "text" name = "Zip" /></td>
24        </tr>
25        <tr>
26          <td align = "right">Country</td>
27          <td>
28            <select name = "Country">
29              <option value = "-1" selected>[choose yours]</option>
30              <option value = "1">USA</option>
31              <option value = "2">UK</option>
32              <option value = "3">INDIA</option>
33            </select>
34          </td>
35        </tr>
36        <tr>
37          <td align = "right"></td>
38          <td><input type = "submit" value = "Submit" /></td>
39        </tr>
40      </table>
41    </form>
42  </body>
43</html>
```

6.1. ការកំណត់លើ Form Validation ជាមួយ Basic Form Validation

ជាគំបងអនឡាត់ខ្សោយើងមែនពីរបៀបធ្វើទម្រង់មូលដ្ឋាននេះ form validation។ ក្នុងទម្រង់ខាងលើ យើងកំពុងហេវ validate() ដើម្បីធ្វើ validation ទិន្នន័យនៅពេលត្រួតពាណិជ្ជការណ៍ onsubmit កើតឡើង។ ក្នុងខាងក្រោមបង្ហាញពីការអនឡាត់មុខងារ validate() នេះ។



```
// Form validation code will come here.

function validate() {

    if( document.myForm.Name.value == "" ) {
        alert( "Please provide your name!" );
        document.myForm.Name.focus() ;
        return false;
    }

    if( document.myForm.EMail.value == "" ) {
        alert( "Please provide your Email!" );
        document.myForm.EMail.focus() ;
        return false;
    }

    if( document.myForm.Zip.value == "" ||
        isNaN( document.myForm.Zip.value ) ||
        document.myForm.Zip.value.length != 5 ) {

        alert( "Please provide a zip in the format #####." );
        document.myForm.Zip.focus() ;
        return false;
    }

    if( document.myForm.Country.value == "-1" ) {
        alert( "Please provide your country!" );
        return false;
    }

    return( true );
}
```

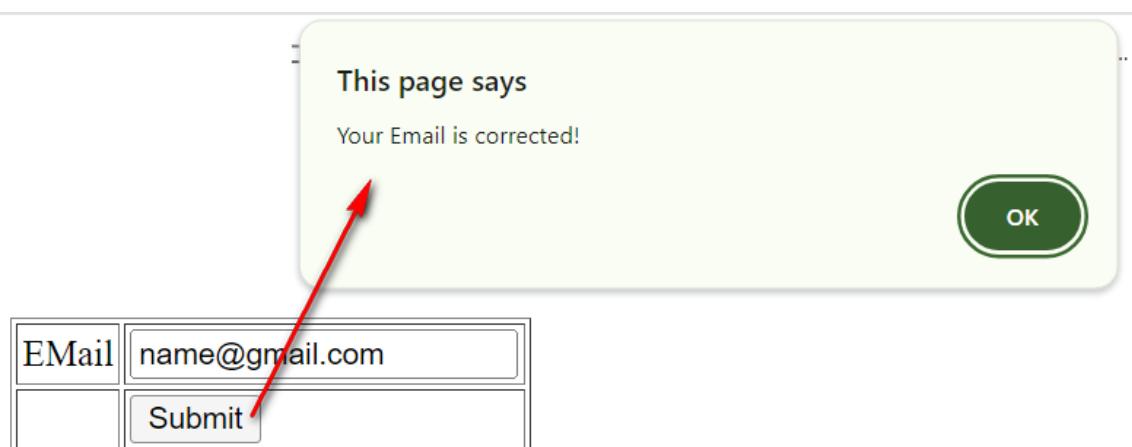
6.2. ការកំណត់លើ Form Validation ជាមួយ Data Format Validation

- របៀបដែលយើងអាចធ្វើដោយទិន្នន័យទម្រង់ដែលបានបញ្ចប់យើង
មុនពេលបញ្ចូនការទៅម៉ាស៊ីនមេ (web server)។
- ឧទាហរណ៍ខាងក្រោមបង្ហាញពីរបៀបធ្វើដោយធ្វើនឹងអីមែនដែលបានបញ្ចប់។
អាសយដ្ឋានអីមែនត្រូវតែមានយ៉ាងហេចណាស់សញ្ញា '@' និង dot (.) ។
ដូច្នោនេះដឹងដឹរ '@' មិនត្រូវដោតអក្សរដើម្បីនៃអាសយដ្ឋានអីមែននោះទេ
ហើយចំនួចចុងក្រោយត្រូវតែមានយ៉ាងហេចណាស់មួយត្រូវបន្ទាប់ពីសញ្ញា '@' ។

```
<script type = "text/javascript">
    function validateEmail() {
        var emailID = document.myForm.EMail.value;
        atpos = emailID.indexOf("@");
        dotpos = emailID.lastIndexOf(".");
        
        if (atpos < 1 || ( dotpos - atpos < 2 )) {
            alert("Please enter correct email ID")
            document.myForm.EMail.focus() ;
            return false;
        }
        return( true );
    }
</script>
```



អាសយដ្ឋានអីមែលដែលត្រួមត្រូវ៖



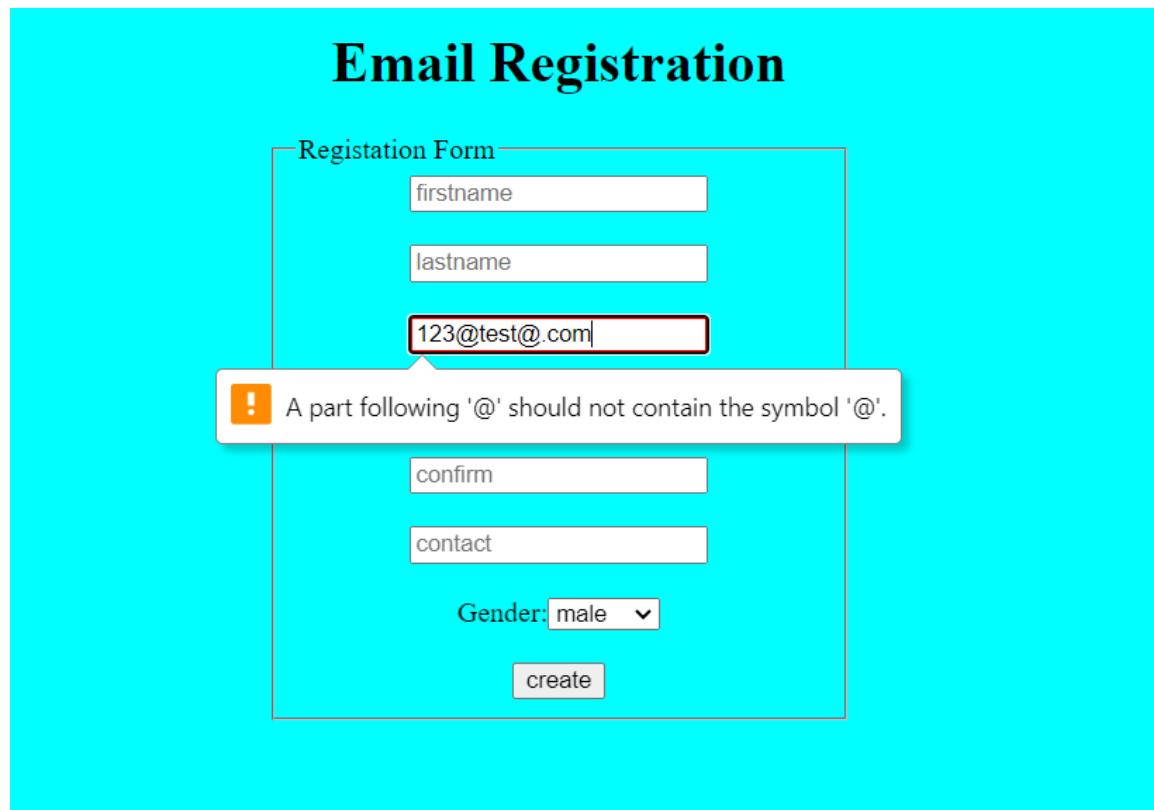
6.3. ការរៀបចំត្រាស់ Regular expressions

ការរោមទៀតជាកំរើដែលប្រើដើម្បីផ្តល់ព័ត៌មានច្បាស់នៃកូដខ្លួន។ នៅក្នុង JavaScript ការរោមធ្វើតាក់ជាក្នុង (objects)ដើរ។ វាលំដាប់គ្មានរក្សាទុកដំណឹង (search pattern)

- ✓ ດາວໂຫຼດໃນ regular expressions ໂອດຖຸກສັບສົນ JavaScript

Method	Description
exec()	ប្រតិបត្តិការដែលរកការផ្តើមផ្តើមក្នុងខ្សោយអក្សរ។ វាប្រគល់បញ្ជី array នៃព័ត៌មាន បុន្ណោះទូទាត់ជាមួយ។
test()	ផ្តើមផ្តើមក្នុងខ្សោយអក្សរ។ វាប្រគល់បញ្ជីពិត បុន្ណោះមិនពិត
match()	ប្រគល់បញ្ជី array ដែលមានការផ្តើកចាំងអស់ រួមចាំងការពារប់យក groups បុន្ណោះប្រសិនបើក្នុងការផ្តើមផ្តើមត្រូវបានរកយើង
matchAll()	ប្រគល់ iterator ផ្តើមផ្តើមដែលមានការផ្តើកចាំងអស់ រួមចាំងការពារប់ក្រុម
search()	ប្រតិបត្តិការ search សម្រាប់ការផ្តើមរាង regular expression និង string នៃ index នៃការផ្តើមផ្តើមដំបូងក្នុងខ្សោយអក្សរ។
replace()	ប្រតិបត្តិការដែលរកសម្រាប់ការផ្តើមផ្តើមក្នុង string ម្ខាយ ហើយដំឡើស substring ដែលត្រូវត្រូវជាមួយនឹង substring ដំឡើស
replaceAll()	ប្រតិបត្តិការដែលរកសម្រាប់ការផ្តើមផ្តើមចាំងអស់ក្នុង string ម្ខាយ ហើយដំឡើស substrings ដែលត្រូវជាមួយនឹង substrings ដំឡើស
split()	ប្រើ regular expression ឬ fixed string ដើម្បីបំបែក string ចូលទៅក្នុង array នៃ substrings

✓ ຂໍາເນົດ Form validation (validating an email)



```

14   <script type="text/javascript">
15     function validate() {
16       let user = document.getElementById("e").value;
17       let user2 = document.getElementById("e");
18       let re = /^w+([.-]?\w+)*@\w+([.-]?\w+)*(\.\w{2,3})+$/;
19       if (re.test(user)) {
20         alert("done");
21         return true;
22       } else {
23         user2.style.border = "red solid 3px";
24         return false;
25       }
26     }
27   </script>
28 </head>
29
30 <body bgcolor="cyan">
31   <center>
32     <h1>Email Registration</h1>
33     <form>
34       <fieldset style="width:300px">
35         <tr>
36           <input type="email" placeholder="username@gmail.com" id="e">
37         </tr>
38         <tr>
39           <input type="submit" onclick="validate()" value="create">
40         </tr>
41       </table>
42     </fieldset>
43   </form>
44 </center>

```

✓ ຂໍາເນົດ Form Validation (validating phone number)

```

14 <script type="text/javascript">
15     function validate() {
16         let user = document.getElementById("c").value;
17         let user2 = document.getElementById("c");
18         let re = /^[7-9][0-9]{9}$/;
19         if (re.test(user)) {
20             alert("done");
21             return true;
22         } else {
23             user2.style.border = "red solid 3px";
24             return false;
25         }
26     }
27 </script>
28 </head>
29
30 <body bgcolor="cyan">
31     <center>
32         <h1>Email Registration</h1>
33         <form>
34             <fieldset style="width:300px">
35                 <legend>Registration Form</legend>
36                 <table>
37                     <tr>
38                         <input type="text" placeholder="contact" id="c">
39                     </tr>
40                     <tr>
41                         <input type="submit" onclick="validate()" value="create">
42                     </tr>
43                 </table>
44             </fieldset>
45         </form>
46     </center>

```

6.4. ការកំណត់ប្រភេទនៃ Document Object Model (DOM)

HTML DOM អនុញ្ញាតឱ្យ JavaScript ចូលប្រើ និងកែប្រែខ្លួនជាតុ HTML ។ JavaScript អាបន្នាស់ប្រជាតុ HTML elements, attributes, CSS styles នៅក្នុងទំព័រ ។ JavaScript កំអប add, remove ជាតុ HTML និង attributes ដើម្បី ដោយប្រើ JavaScript យើងបែមទាំងអាចបង្កើតព្រឹត្តិការណ៍បន្ថីមនៅក្នុងទំព័រ។

- **តើ DOM ជាអ្វី? DOM គឺជាអក្សរកាត់សម្រាប់ Document Object Model ។** វាគឺជាបំណុលប្រទាក់សនេសកម្មវិធីសម្រាប់ Core, XML, និង HTML DOM
- **តើ HTML DOM ជាអ្វី? HTML បង្កើតរបនាសម្ព័ន្ធរបស់គេហទំព័រ ហើយ JavaScript បន្ថែមអន្តរកម្មទៅគេហទំព័រដោយរៀបចំជាតុ HTML (manipulating the HTML elements) ។**
- **នៅក្នុង JavaScript វិធីសាស្ត្រ DOM ត្រូវបានប្រើដើម្បីធ្វើសកម្មការជាក់លាក់មួយលើជាតុ HTML ។ DOM ត្រូវបានប្រើដើម្បីដោះស្រាយការណ៍ប្រព័ន្ធដែលមានមេការងារឡើង។** នៅក្នុងមេការងារ សាខានីមួយុបញ្ចប់ដោយចុះឯកសារ ហើយចុះឯកសារនៅក្នុងការងារ។ វិធីសាស្ត្រ DOM អនុញ្ញាតឱ្យយើងចូលដើរការដោកម្មវិធីទៅការនៃមេការងារ។ ដោយប្រើវិធីសាស្ត្រ DOM អ្នកអាបន្នាស់ប្រជាតុសម្ព័ន្ធ របនាប៉ុន្ម័េប្រស់ Page ។

ឧទាហរណ៍ getElementById() method

```
<button onclick = "accessEle()"> Access output and write </button>
<p id = "output"> </p>
<script>
    function accessEle() {
        document.getElementById("output").innerHTML =
            "Hello User! You have just clicked the button!";
    }
</script>
```

វិធីសារស្តូវ addEventListener() ត្រូវបានប្រើដើម្បីបន្ថែមព្រឹត្តិការណ៍ទៅក្នុង Page

```
<h3>JavaScript – document.addEventListener() Method </h3>
<p> Hover over here to change background color </p>
<script>
    document.addEventListener('mouseover', function () {
        document.body.style.backgroundColor = 'red';
    });
</script>
</body>
</html>
```



JavaScript – document.addEventListener() Method

Hover over here to change background color

- បញ្ជីនៃ DOM Methods

addEventListener()	adoptNode()	append()
caretPositionFromPoint()	close()	createAttribute()
createAttributeNS()	createComment()	createDocumentFragment()
createElement()	createElementNS()	createEvent()
createTextNode()	elementFromPoint()	elementsFromPoint()
getAnimations()	getElementById()	getElementsByName()
getElementsByName()	getElementsByTagName()	getElementsByClassName()
importNode()	normalize()	hasFocus()
prepend()	querySelector()	querySelectorAll()
removeEventListener()	replaceChildren()	write()
writeln()		

- The DOM document មានលក្ខណៈសម្រាតិ និងវិធីសាស្ត្រដៃង់ដែលអ្នកអាចប្រើប្រើម្នូនទូលបានព័ត៌មានលម្អិតអំពីជាតិ HTML និងកំណត់ពួកវាតាមបំណង ឧបាទរណី Document childElementCount Property

```

<body>
    <div>First Element</div>
    <div>Second Element</div>
    <div>Third Element</div>
    <div id = "output"> </div>
    <script>
        document.getElementById('output').innerHTML =
            "Total number of child elements in the document is: " +
            document.childElementCount;
    </script>
</body>
</html>

```

First Element
 Second Element
 Third Element
 Total number of child elements in the document is: 1

- HTML DOM document forms គ្រឿងបានប្រើប្រាស់ដើម្បីទទួលបានទិន្នន័យអ្នកប្រើប្រាស់។

```

<form onsubmit = "submitForm(event)" id = "emailform">
    <p>Email: <input type = "email" name = "email"> </p>
    <p><input type = "submit"></p>
</form>
<div id = "output"> </div>
<script>
    function submitForm(e) {
        e.preventDefault();
        const email = document.forms['emailform']['email'].value;
        //Validate email using regex
        const emailRegex = /^[a-zA-Z]+@[a-zA-Z]+\.[a-zA-Z]{2,3}\$/;
        if (emailRegex.test(email)) {
            document.getElementById('output').innerHTML = "Email is valid!";
        } else {
            document.getElementById('output').innerHTML = "Email is not valid!";
        }
    }
</script>

```

7. ຕີ້ມີສາງສູ່ແຮມື່ງໝາສ່ Objects ລືລ Prototypes

Methods ເພີ່ມື່ງກຳພາດໆ Objects ມານຜູ້ປັດ້ວັນ:

- ແບໍ່ແຕ່ລັດ Object Literal
- ແບໍ່ແຕ່ລັດ ຕາກຸງຄົນ: new
- ແບໍ່ແຕ່ລັດ Object Constructor
- ແບໍ່ແຕ່ລັດ Object.assign()
- ແບໍ່ແຕ່ລັດ Object.create()
- ແບໍ່ແຕ່ລັດ Object.fromEntries()

7.1. ກາບເຜິດ Objects

➤ ຂາທາຣດີ: ແບໍ່ແຕ່ລັດ Object Literal

ຕູ້ກົດ (object literal) ອີ່ຕົວໃຫຍ້: property ດີວຽກຂອງກົດແຜ່ງປ່ອມກັດ

```
<p id = "demo">Iterating the Set: </p>
<script>
    const person = {
        name: 'John Doe',
        age: 30,
        address: '123 Main Street',
    };
    document.getElementById("demo").innerHTML =
        person.name + " is " + person.age + " years old.";
</script>
```

➤ ຂາທາຣດີ: ແບໍ່ແຕ່ລັດ ຕາກຸງຄົນ: new

ເພີ່ມື່ງກຳພາດໆ new Object() ແກ້ໄຂຍບໍ່ໂສ່ມ 4 properties:

```
<p id = "demo"></p>
<script>
    // Create an Object
    const person = new Object();
    person.firstName = "John";
    person.lastName = "Doe";
    person.age = 50;
    person.eyeColor = "blue";

    // Display Object Content
    document.getElementById("demo").innerHTML =
        person.firstName + " is " + person.age + " years old.";
</script>
```

- #### ➤ ឧទាហរណ៍៖ ក្រឹមប្រាស់ Object Constructor

ពេលខ្លះយើងត្រូវបានធ្វើតាតក្នុង (objects)ដាច់ប្រើនដែលមានប្រភេទជូចច្បាស់ ។ដើម្បីបានធ្វើតាតប្រភេទទឹកម្មយោ យើងប្រើ **object constructor function** ។ វាព្យាបានគេចាត់ទុកបានដាការអនុវត្តលើក្នុងការដោក់លើ៖ **constructor functions** ជាមួយនឹងសាងសង់ដោយអក្សរជិំបុង (upper-case) ។

```
<p id = "demo"></p>
<script>
    // Constructor Function for Person objects
    function Person(first, last, age, eye) {
        this.firstName = first;
        this.lastName = last;
        this.age = age;
        this.eyeColor = eye;
    }

    // Create a Person object
    const myFather = new Person("John", "Doe", 50, "blue");
    const myMother = new Person("Sally", "Rally", 48, "green");
    const mySister = new Person("Anna", "Rally", 18, "green");
    const mySelf = new Person("Johnny", "Rally", 22, "green");

    // Display age
    document.getElementById("demo").innerHTML =
        "My father is " + myFather.age + ".";
</script>
```

- ឧទាហរណ៍ ៖ ប្រើប្រាស់ `Object.assign()`

ដើម្បីសារក្នុង **Object.assign()** បានផ្តល់ក្នុងរបៀបខាងក្រោមនេះ

```
<p id = "demo"></p>
<script>
    /// Create Target Object
    const person1 = {
        firstName: "John",
        lastName: "Doe",
        age: 50,
        eyeColor: "blue"
    };
    // Create Source Object
    const person2 = {firstName: "Anne", lastName: "Smith"};
    // Assign Source to Target
    Object.assign(person1, person2);
    // Display Target
    let text = Object.entries(person1);
    document.getElementById("demo").innerHTML = text;
</script>
```

➤ ឧចាបរណ៍ទី២ ប្រើប្រាស់ `Object.create()`

`Object.entries()` ត្រួតព្រម array នៃកូដ key/value ក្នុងវត្ថុមួយ៖

```
<p id = "demo"></p>
<script>
    const person = {
        firstName : "John",
        lastName : "Doe",
        age : 50,
        eyeColor : "blue"
    };
    let text = Object.entries(person);
    document.getElementById("demo").innerHTML = text;
    //output: firstName,John,lastName,Doe,age,50,eyeColor,blue
</script>
```

➤ ឧចាបរណ៍ទី៣ ប្រើប្រាស់ `Object.fromEntries()`

វិធីសារស្តី `fromEntries()` បង្កើតវត្ថុមួយពីបញ្ជីនៃកូដ key/value ។

```
<p id = "demo"></p>
<script>
    const fruits = [
        ["apples", 300],
        ["pears", 900],
        ["bananas", 500]
    ];
    const myObj = Object.fromEntries(fruits);
    document.getElementById("demo").innerHTML = myObj.pears;//900
</script>
```

7.2. របៀបប្រើប្រាស់ Object Prototypes

- វិញ្ញា JavaScript ទាំងអស់ទូលបមទេកលក្ខណៈសម្រាតិ និងវិធីសារត្រួតពិត្យដើម្បី

យើងបានរៀនពីរបៀបប្រើប្រាស់ **Object constructor** ខាងក្រោម៖

```
<p id = "output"></p>
<script>
    function Person(first, last, age, eye) {
        this.firstName = first;
        this.lastName = last;
        this.age = age;
        this.eyeColor = eye;
    }

const myFather = new Person("John", "Doe", 50, "blue");
const myMother = new Person("Sally", "Rally", 48, "green");

document.getElementById("output").innerHTML =
"My father is " + myFather.age + ". My mother is " + myMother.age;
//My father is 50. My mother is 48
</script>
```

- អ្នកកំណតនរៀនដើម្បី អ្នកមិនអាចបន្ថែម **property** ថ្មីទៅ **object constructor** ដែលមានក្រាប់បានទេ៖

```
<p id = "output"></p>
<script>
    function Person(first, last, age, eye) {
        this.firstName = first;
        this.lastName = last;
        this.age = age;
        this.eyeColor = eye;
    }

Person.nationality = "English";

const myFather = new Person("John", "Doe", 50, "blue");
const myMother = new Person("Sally", "Rally", 48, "green");

document.getElementById("output").innerHTML =
"The nationality of my father is " + myFather.nationality;
//The nationality of my father is undefined
</script>
```

- ដើម្បីបន្ថែម property បីទេស constructor អ្នកត្រូវតែបន្ថែមវាទៅ constructor function :

```

<p id = "output"></p>
<script>
    function Person(first, last, age, eye) {
        this.firstName = first;
        this.lastName = last;
        this.age = age;
        this.eyeColor = eye;
        this.nationality = "English";
    }

    const myFather = new Person("John", "Doe", 50, "blue");
    const myMother = new Person("Sally", "Rally", 48, "green");

    document.getElementById("output").innerHTML =
        "The nationality of my father is " + myFather.nationality +
        ". The nationality of my mother is " + myMother.nationality;
        //The nationality of my father is English.
        //The nationality of my mother is English
</script>

```

- របៀបប្រើ Prototype Inheritance
- គ្រប់វត្ថុ (objects) JavaScript ទាំងអស់ទទួលមរតកទ្រព្យសម្រាតិ (properties) និង methods ពីគ្រប់រឹងម (prototype) :
 - Date objects inherit from Date.prototype
 - Array objects inherit from Array.prototype
 - Person objects inherit from Person.prototype
- Date objects, Array objects, and Person objects ទទួលមរតកពី Object.prototype.

ការប្រើប្រាស់ Prototype Property

```

<p id = "output"></p>
<script>
    function Person(first, last, age, eye) {
        this.firstName = first;
        this.lastName = last;
        this.age = age;
        this.eyeColor = eye;
    }

    Person.prototype.nationality = "English";

const myFather = new Person("John", "Doe", 50, "blue");
document.getElementById("output").innerHTML =
"The nationality of my father is " + myFather.nationality;
//The nationality of my father is English
</script>

```

- Prototype property គឺអនុញ្ញាតឱ្យអ្នកបង្កើម method ដូចជាការ់ object constructors ទៅ

```

<p id = "output"></p>
<script>
    function Person(first, last, age, eye) {
        this.firstName = first;
        this.lastName = last;
        this.age = age;
        this.eyeColor = eye;
    }

    Person.prototype.name = function() {
        return this.firstName + " " + this.lastName
    };

```

```

const myFather = new Person("John", "Doe", 50, "blue");
document.getElementById("output").innerHTML =
"My father is " + myFather.name();
//My father is John Doe
</script>

```

7.3. របៀបប្រើប្រាស់ Object Properties

- **Property គិតប្រព័ន្ធ Methods**

```

// Adding or changing an object property
Object.defineProperty(object, property, descriptor)

// Adding or changing object properties
Object.defineProperties(object, descriptors)

// Accessing a Property
Object.getOwnPropertyDescriptor(object, property)

// Accessing Properties
Object.getOwnPropertyDescriptors(object)

// Returns all properties as an array
Object.getOwnPropertyNames(object)

// Accessing the prototype
Object.getPrototypeOf(object)

```

- The `Object.defineProperty()` method can be used to:
 - Adding a new property to an object
 - Changing property values
 - Changing property metadata
 - Changing object getters and setters

ឧទាហរណ៍ ការប្រើប្រាស់ **Property** គិតប្រព័ន្ធដែលបានបង្កើតឡើងជាអំពី `Object.defineProperty(object, property, descriptor)`

```

<p id = "output"></p>
<script>
    // Create an Object:
    const person = {
        firstName: "John",
        lastName: "Doe",
        language: "EN"
    };

    // Add a Property
    Object.defineProperty(person, "year", {value:"2008"})

    // Display the Property
    document.getElementById("output").innerHTML = person.year;
    //2008
</script>

```

ឧទាហរណ៍ កួយបញ្ជី Object Properties ទាំងអស់ ដោយប្រើ `Object.getOwnPropertyNames(object)`

```
<p id = "output"></p>
<script>
    // Create an Object:
    const person = {
        firstName: "John",
        lastName: "Doe",
        language: "EN"
    }

    // Display all Properties
    document.getElementById("output").innerHTML =
        Object.getOwnPropertyNames(person);
        //firstName,lastName,language
</script>
```

ឧទាហរណ៍ កួយបញ្ជី enumerable object properties ដែលអាចរកប់បាន ដោយប្រើ `Object.keys()`

```
<p id = "output"></p>
<script>
    // Create an Object:
    const person = {
        firstName: "John",
        lastName: "Doe",
        language: "EN"
    }

    // Change the language Property
    Object.defineProperty(person, "language", {enumerable:false});

    // Display all Enumerable Properties
    document.getElementById("output").innerHTML = Object.keys(person);
        //firstName,lastName
</script>
```

7.4. របៀបប្រើប្រាស់ Object Accessors

- JavaScript Accessors (Getters and Setters)
- Getters និង setters អនុញ្ញាតឱ្យអ្នកកំណត់ Object Accessors (Computed Properties)

ឧទាហរណ៍ Getter (The get Keyword) :

```
<p id = "output"></p>
<script>
    // Create an object:
    const person = {
        firstName: "John",
        lastName: "Doe",
        language: "en",
        get lang() {
            return this.language;
        }
    };

    // Display data from the object using a getter:
    document.getElementById("output").innerHTML = person.lang;
//en
</script>
```

ឧទាហរណ៍ Setter (The set Keyword) :

```
<p id = "output"></p>
<script>
    // Create an object:
    const person = {
        firstName: "John",
        lastName: "Doe",
        language: "NO",
        set lang(value) {
            this.language = value;
        }
    };

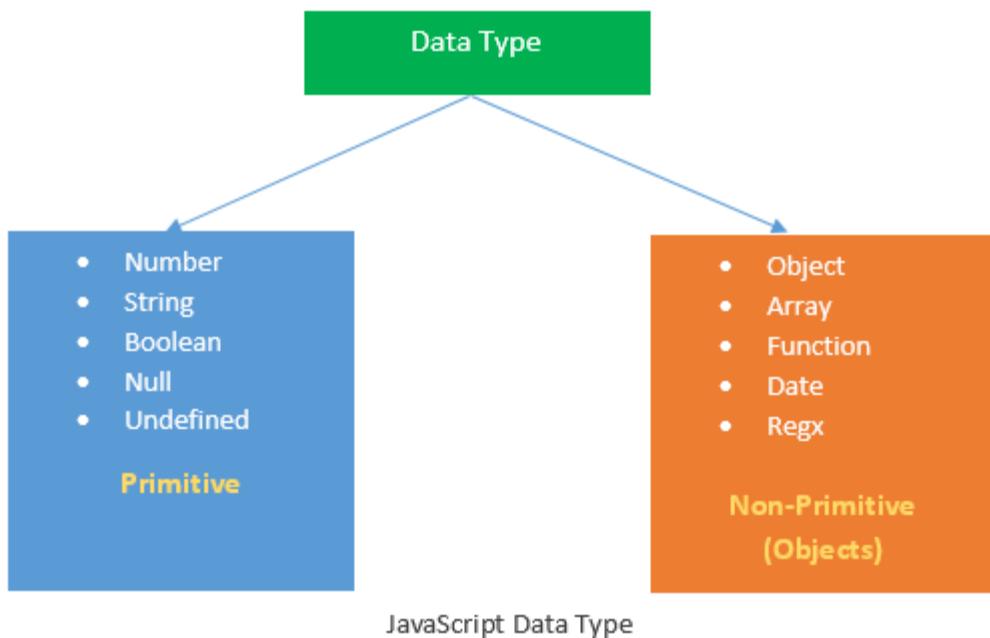
    // Set a property using set:
    person.lang = "en";

    // Display data from the object:
    document.getElementById("output").innerHTML = person.language;//en
</script>
```

8. នៃលវំពីជំនើវការនៃ Object Oriented

វត្ថុ (Objects) ត្រូវបានបង្កើតដោយប្រើប្រាស់ក្រោមអង្វោះ (curly braces) ហើយត្រូវបានរំបៀកដោយសញ្ញាក្បែស (comma)។ property នឹមួយនៅត្រូវបានសរស់ដោយសញ្ញា colon (:) បន្ថែមដោយតម្លៃ property ។ គ្មានៗ គឺជាការប្រមូលដីនៃលក្ខណៈសម្រាតិដែលមានលំដាប់លំដោយដែលសរស់ដោយ key: value pairs ។

Object គឺជាប្រភេទទិន្នន័យ non-primitive ដែលត្រូវបានប្រើប្រាស់ក្នុងទិន្នន័យដោយគឺជាកូលម៉ែនដែលមានលំដាប់លំដោយដែលសរស់ដោយ key-value ។



JavaScript គឺជាការសារ Object Oriented Programming (OOP) ។ OOP មាននូវសមាសធាតុ សំខាន់ឱ្យយកចំណុច ដែលត្រូវប្រើប្រាស់ម្រាប់ការបង្កើត Website និង Web Application ។ ការសារសរសេរកម្មដើម្បីអាចត្រូវបានគេហេត្តិថា object-oriented ប្រសិនបើវាដឹងលំសមត្ថភាពជាមូលដ្ឋាន ចំណុចបានដឹងលំអ្នកអភិវឌ្ឍន៍ ។

- Encapsulation:
- Abstraction
- Inheritance
- Polymorphism

8.1. ការកំណត់ JavaScript Classes

Classes គឺជាប្លង់មេ ប្រតិបត្តិសម្រាប់ការបង្កើតវត្ថុ (object) ។ ពួកវាបំប្លែងទិន្នន័យ និងមុខងារដើម្បីរៀបចំទិន្នន័យនៅ៖ ។ យើងអាចបង្កើត Object ដោយប្រើ classes ។

Syntax

```
// class declaration
class ClassName {
    // Class body
}

//Class expression
const ClassName = class {
    // class body
}
```

Type of JavaScript Classes

```
<p id = "output"> The type of the car class is: </p>
<script>
    class Car {
        // Class body
    }
    document.getElementById("output").innerHTML += typeof Car;
    //The type of the car class is: function
</script>
```

The constructor() method

ក្នុងឧទាហរណ៍ខាងក្រោម យើងប្រើធីថា constructor() ធើម្នូបង្កើត Car class

```
class Car {
    constructor(brand) { // Defining the constructor
        this.brand = brand;
    }
}
```

Creating JavaScript Objects

```
const myCar = new Car("Audi");
```

ឧទាហរណ៍៖ ការបង្កើត class objects ដោយគ្នាតាមអាជីវម៉ាស់(arguments)

```
<p id = "output"> </p>
<script>
    // creating Car class
    class Car {
        constructor() {
            this.brand = "BMW";
            this.model = "X5";
            this.year = 2019;
        }
    }
    // instantiate myCar object
    const myCar = new Car();
    // display the properties
    document.getElementById("output").innerHTML =
    "Car brand is : " + myCar.brand + "<br>" +
    "Car model is : " + myCar.model + "<br>" +
    "Car year is : " + myCar.year + "<br>";
</script>
```

ឧទាហរណ៍៖ ការបង្កើត class objects ដោយមានអាជីវម៉ាស់(arguments)

```
<p id = "output"> </p>
<script>
    class Car {
        constructor(brand, model, price, year) {
            this.brand = brand;
            this.model = model;
            this.price = price;
            this.year = year;
        }
    }
    const carObj = new Car("BMW", "X5", 9800000, 2019);
    document.getElementById("output").innerHTML +=
    "Car brand : " + carObj.brand + "<br>" +
    "Car model : " + carObj.model + "<br>" +
    "Car price : " + carObj.price + "<br>" +
    "Car year : " + carObj.year + "<br>";
</script>
```

JavaScript Class Methods

អ្នកក៏អាចកំណត់ method នៅខាងក្រុង class ដែលអាចចូលរួមជីថយក class instance ។

Syntax

```
class car {
    methodName(params) {
        // Method body
    }
}
obj.methodName();
```

ឧបាទរណ៍ខាងក្រោមបង្ហាញពីធីបញ្ជាផ្ទាល់ថាការមែនជារបៀបទាមទារ class methods

នៅទីនេះ យើងបានកំណត់ updateprice() method ដើម្បីធ្វើបច្ចុប្បន្នការពារថ្មីរបៀបយន្ត ។

```
<p id = "output"> </p>
<script>
class Car {
    constructor(brand, model, price, year) {
        this.brand = brand;
        this.model = model;
        this.price = price;
        this.year = year;
    }
    updateprice(newPrice) {
        this.price = newPrice;
    }
}

const myCar = new Car("BMW", "X5", 9800000, 2019);
document.getElementById("output").innerHTML +=
"The car price is : " + myCar.price + "<br>";

myCar.updateprice(8800000); // updating price

document.getElementById("output").innerHTML +=
"After updating the car price is : " + myCar.price + "<br>";
//The car price is : 9800000
//After updating the car price is : 8800000
</script>
```

8.2. ស្ថាល់ពី Encapsulation

Encapsulation ជាស្ថី? គឺជាគោលគំនិតដាមូលដ្ឋាននៅក្នុងការសែរកម្មវិធីជាទម្រង់ **Object-oriented programming languages** រួមជាមួយនឹងមេតកិ (inheritance) និងពហុនិយម (polymorphism) ។ វាប្រើប្រាណបច្ចុប្បន្ននៃយុទ្ធសាស្ត្រភាពទាំងអ្វី និងផ្តល់សិទ្ធិចូលប្រើទិន្នន័យដែលត្រូវការកែតប់ណែនាំ: ដើម្បីបង្កើនការព្រឹមត្រូវ និងសុវត្ថិភាពនៃទិន្នន័យ។ វាបាបច្រើកកទេស ធ្វើអាយ Fields របស់ Class ជាការ **Private** ដែលអាច អាយ Methods របស់ Class នេះដែលមានលក្ខណៈជាការ **Public** អាចចូលដំណើរការបាន បើនេះ មិនគីឡូ Class ពីខាងក្រោមឯុទ្ធសាស្ត្រ មកប្រើ Field របស់ Class នេះបានទេ ។

- តើអ្នីជាតម្លៃការសម្រាប់ encapsulation?

ឧទាហរណ៍ អ្នកបានកំណត់ត្រូវខាងក្រោមនេះក្នុងក្នុងក្នុងរបស់អ្នក។

```
const car = {
    Brand: "Honda city",
    model: "sx",
    year: 2016,
}
```

អ្នកណាក៏អាចចូលប្រើប្រាស់លក្ខណៈសម្រួលរបស់ car object ដូចបង្ហាញខាងក្រោម។

```
car.Brand
```

ដូចត្រូវនេះដឹងដើរ, នរណាម្នាក់អាចផ្តល់បញ្ជីត្រូវកែល្អែនទ្រព្យសម្រួលិណាមួយនៃ car object ដូចដែលបានបង្ហាញខាងក្រោម។

```
car.Brand = true;
```

នៅក្នុងក្នុងខាងក្រោម អនុគមន៍ **shoppingCart()** គឺជាមុខងារខាងក្រោមដែលមានអប់រំ (variables) និងមុខងារ។ មុខងារខាងក្រោមនេះសាលាកាត **private scope** ។

```
<p id = "output"> </p>
<script>
let output = document.getElementById("output");
function shoppingCart() {
    const cartItems = [];
    function add(item) {
        cartItems.push(item);
        output.innerHTML += `${item.name} added to the cart. <br>`;
    }
    function remove(itemName) {
        const index = cartItems.findIndex(item => item.name === itemName);
        if (index !== -1) {
            const removedItem = cartItems.splice(index, 1)[0];
            output.innerHTML += `${removedItem.name} removed from the cart.`;
        } else {
            output.innerHTML += `Item ${itemName} not found in the cart. <br>`;
        }
    }
    return {
        add,
        remove,
    };
}

```

```
// Defining items
const item1 = { name: 'Car', price: 1000000 };
const item2 = { name: 'Bike', price: 100000 };
// Create a new Shopping cart
const cart = shoppingCart();
// Adding items to the cart
cart.add(item1);
cart.add(item2);
// Remove bike from the cart
cart.remove('Bike');
/*Car added to the cart.
Bike added to the cart.
Bike removed from the cart.*/
</script>
```

ក្នុងខាងក្រោម យើងបានកំណត់ car class

```

<div id = "output1">The car mileage is: </div>
<div id = "output2">After updating the car mileage is: </div>
<script>

    class Car {
        #brand = "TATA"; // Private field
        #name = "Nexon"; // Private field
        #milage = 16; // Private field

        getMilage() {
            return this.#milage; // Accessing private field
        }

        setMilage(milage) {
            this.#milage = milage; // Modifying private field
        }
    }

    let carobj = new Car();
    document.getElementById("output1").innerHTML += carobj.getMilage()
    carobj.setMilage(20);
    document.getElementById("output2").innerHTML += carobj.getMilage()
    // carobj.#milage); will throw an error.
    |
    //result:The car mileage is: 16
    //After updating the car mileage is: 20

```

8.3. ស្ថាល់ពី Inheritance

គោលគំនិតនៃការទទួលមរតក (inheritance) នៅក្នុង JavaScript អនុញ្ញាតឱ្យបានកំណត់ក្នុង (child class)ទទួលមរតកនូវលក្ខណៈសម្រាប់ និងដឹងសារស្ថាបន្ទាក់មែ (parent class)។
ការទទួលមរតកក៏ដោយគោលគំនិតជាមូលដ្ឋាននៃ object-oriented programming ដូចនឹង encapsulation និង polymorphism.

- បានកំណត់មែ (parent class) -
ការដាក់ជាប្រភេទដែលលក្ខណៈសម្រាប់ទាំងទទួលមរតកដោយបានកំណត់ឡើងទៀត។
- បានកំណត់ក្នុង (child class)- ការដាក់ជាប្រភេទដែលទទួលមរតកលក្ខណៈសម្រាប់នៃប្រភេទដែលកំណត់ឡើងទៀត។

អ្នកអាបអនុវត្តតាម syntax នៃក្រោមសម្រាប់ការទទួលមនេតកប្បាក់តែម្មយ។

```
class childClass extends parentClass {  
    // Child class body  
}
```

ឧទាហរណ៍៖ មនេតកប្បាក់តែម្មយ (Single Class)

```
<div id = "output1">The brand of the bike is: </div>  
<div id = "output2">Total gears in the bike is: </div>  
<script>
```

```
    // Parent class  
    class Bike {  
        constructor() {  
            this.gear = 5;  
        }  
  
        getGears() {  
            return this.gear;  
        }  
    }
```

```
    // Child class  
    class suzuki extends Bike {  
        constructor() {  
            super();  
            this.brand = "Yamaha"  
        }  
  
        getBrand() {  
            return this.brand;  
        }  
    }
```

```
const suzukiBike = new suzuki();  
document.getElementById("output1").innerHTML += suzukiBike.getBrand();  
document.getElementById("output2").innerHTML += suzukiBike.getGears();  
//The brand of the bike is: Yamaha  
//Total gears in the bike is: 5
```

ឧបាទរណ៍៖ ការប្រើពាក្យគន្លឹះ `super()` ដើម្បីចាប់ផ្តើមលក្ខណៈសម្រតិថ្នាក់មេ (parent class)

Bike class មាន **constructor** ដោយយក `gears` ជា **parameters** ហើយប្រើវា ចាប់ផ្តើមលក្ខណៈ `gears` ។

ថ្នាក់ 'suzuki' ក៏មាន **constructor** ដោយយក `brand` និង `gears` ជាតាក់មែន្រត់។ ដោយប្រើពាក់មែន្រត់ `brand` វាតាប់ផ្តើម `brand property` ហើយផ្តល់កាត់ពាក់មែន្រត់ `gears` ជា **argument** នៃពាក្យគន្លឹះ `super()`

```
<div id = "output1">The brand of the bike is: </div>
<div id = "output2">Total gears in the bike is: </div>
<script>
    // Parent class
    class Bike {
        constructor(gears) {
            this.gears = gears;
        }
    }
    // Child class
    class suzuki extends Bike {
        constructor(brand, gears) {
            super(gears);
            this.brand = brand;
        }
    }
    const suzukiBike = new suzuki("Suzuki", 4);
    document.getElementById("output1").innerHTML += suzukiBike.brand;
    document.getElementById("output2").innerHTML += suzukiBike.gears;
</script>
```

Multilevel inheritance គឺជាប្រភេទនៃមន្តតកន្លែកក្នុង JavaScript ។ នៅក្នុងការទទួលមន្តតកពហុកម្រិត ថ្នាក់ម្អាយទទួលមន្តតកលក្ខណៈសម្រតិនៃថ្នាក់ម្អាយទៀតហើយថ្នាក់ម្អាយទៀតទទួលមន្តតកលក្ខណៈសម្រតិថ្នាក់បច្ចប្បន្ន ។

```
class A {
}
class B extends A {
}
class C extends B {
```

ក្នុងទេរាប់ខាងក្រោម Honda class ទទួលមកតិច Bike class និង Shine class ទទួលមកតិច
Honda class ។

```
<p id = "output"> </p>
<script>
    // Parent class
    class Bike {
        constructor(gears) {
            this.gears = gears;
        }
    }
    // Child class
    class Honda extends Bike {
        constructor(brand, gears) {
            super(gears);
            this.brand = brand;
        }
    }
    class Shine extends Honda {
        constructor(model, brand, gears) {
            super(brand, gears);
            this.model = model;
        }
    }
    const newBike = new Shine("Shine", "Honda", 5);
    document.getElementById("output").innerHTML = `The ${newBike.model}
model of the ${newBike.brand} brand has total ${newBike.gears} gears. `;
    //The Shine model of the Honda brand has total 5 gears.
</script>
```

8.4. ស្ថាល់ពី Abstraction

នៅក្នុង programming languages ភាគប្រើន អ្នកអាចសម្រចចាននូវ abstraction ដោយប្រើ abstract class ។ abstract class មានព័ត៌មានប្រកាសដើម្បីសាស្ត្រ ប៉ុន្តែមិនត្រូវបានអនុវត្ត។ លើសពីនេះ ទៀត អ្នកត្រូវអនុវត្តដើម្បីសាស្ត្រដែលបានប្រកាសក្នុងថ្មាក់អ្វីបី abstract class ទិញក្នុង child class ។ Abstraction អនុញ្ញាតឱ្យអ្នកលាក់ព័ត៌មានលម្អិតនៃការអនុវត្តនិងបង្ហាញលក្ខណៈពិសេសបំពេះ អ្នកប្រើប្រាស់ប៉ុណ្ណោះ។

➤ ការបង្កើតប្រាក់អរូបី (Abstract Class)

```
<div id = "output"> </div>
<script>
    try {
        // Object constructor
        function fruit() {
            this.name = "Fruit";
            if (this.constructor === fruit) {
                // Preventing the instance of the object
                throw new Error("You can't create the instance of the fruit.")
            }
        }
        // Implementing method in the prototype
        fruit.prototype.getName = function () {
            return this.name;
        }
        const apple = new fruit();
    } catch (error) {
        document.getElementById("output").innerHTML = error;
    }
    //Error: You can't create the instance of the fruit.
</script>
```

➤ ការពន្លឹកប្រាក់អរូបី

```
<div id = "output">The name of the fruit is: </div>
<script>
    // Abstract class
    function fruit() {
        this.name = "Fruit";
        if (this.constructor === fruit) { // Preventing the instance of t
            throw new Error("You can't create the instance of the fruit.")
        }
    }
    // Implementing method in the prototype
    fruit.prototype.getName = function () {
        return this.name;
    }
    // Child class
    function Apple(fruitname) {
        this.name = fruitname;
    }
    // Extending the Apple class with the fruit class
    Apple.prototype = Object.create(fruit.prototype);

    const apple = new Apple("Apple");
    document.getElementById("output").innerHTML += apple.getName();
    //The name of the fruit is: Apple
```

8.5. ស្ថាល់ពី Polymorphism

Polymorphism នៅក្នុង JavaScript អនុញ្ញាតឱ្យអ្នកកំណត់វិធីសារស្ថាប័នដែលមាន

លេខាជូបត្រា និងមុខងារផ្សេងៗគ្នា ។ Polymorphism ត្រូវបានសម្រេចដោយប្រើវិធីសារស្ថាប័ន **overloading** និង **overriding** ។ JavaScript មិនគំទ្រូវឯកសារផ្ទុកលើសទម្យន់ (overloading) ដើមទៃ ។ Method overriding អនុញ្ញាតឱ្យប្រាក់រៀង (subclass) ប្រាក់កូន (child class) កំណត់ឡើងវិញ្ញុវិធីសារស្ថាប័ននៃ superclass ប្រាក់មេ ។ នៅក្នុងជំពូកនេះ យើងនឹងអនុវត្ត polymorphism ដោយប្រើគំនិតនៃវិធីសារស្ថាប័ន overriding ។

➤ Method Overriding

ប្រសិនបើអ្នកកំណត់វិធីសារស្ថាប័នលេខាជូបត្រានៅក្នុងប្រាក់កូន (parent) និងប្រាក់កូន (child class) វិធីសារស្ថាប័ន child class នឹងបងបីសែរវិធីសារស្ថាប័នបស់ប្រាក់មេ ។

ឧទាហរណ៍ទី 1: ការបង្ហាញពី Polymorphism នៅក្នុង JavaScript

```
<div id = "output1"> </div>
<div id = "output2"> </div>
<script>
    class Shape {
        area(a, b) {
            return "The area of each Geometry is different! <br>";
        }
    }

    class Circle extends Shape {
        area(r) { // Overriding the method of the Shape class
            return "The area of Circle is " + (3.14 * r * r) + "<br>";
        }
    }

    class Rectangle extends Shape {
        area(l, b) { // Overriding the method of the Shape class
            return "The area of Rectangle is " + (l * b) + "<br>";
        }
    }

    const circle = new Circle();
    // Calling area() method of Circle class
    document.getElementById("output1").innerHTML = circle.area(5);

    const rectangle = new Rectangle();
    // Calling area() method of Rectangle class
    document.getElementById("output2").innerHTML = rectangle.area(5, 10);
    //The area of Circle is 78.5
    //The area of Rectangle is 50
</script>
```

ឧទាហរណ៍ទី 2: Parent Class Method's បន្ថែមមុខងារនៅក្នុង Child Class

```

<p id = "output1"> </p>
<p id = "output2"> </p>
<script>
function _(obj){
    return document.getElementById(obj);
}
class Math {
    mathOperations(a, b) {
        _("output1").innerHTML = "Addition: " + (a+b) + "<br>";
        _("output1").innerHTML += "Subtraction: " + (a-b);
    }
}
class AdvanceMath extends Math {
    mathOperations(a, b) {
        super.mathOperations(a, b);
        _("output2").innerHTML += "Multiplication: " + (a*b) + "<br>";
        _("output2").innerHTML += "Division: " + (a/b);
    }
}
const A_math = new AdvanceMath();
A_math.mathOperations(10, 5); // Calls method of AdvanceMath class
/*Addition: 15
Subtraction: 5

Multiplication: 50
Division: 2*/

```

អត្ថប្រយោជន៍នៃការប្រើប្រាស់ Polymorphism នៅក្នុង JavaScript

- ភាពអាជប្រើប្រួលក្នុងវិញ្ញាន (Code reusability) - Polymorphism
អនុញ្ញាតឱ្យអ្នកប្រើក្នុងវិញ្ញាន។ ក្នុងឧទាហរណ៍ទីពីរ យើងបានប្រើក្នុងនៅវិធីសាស្ត្រ **mathOperations()** នៃ **math class** ទីនេះ។
- Extensibility - អ្នកអាចពង្រីកក្នុងបច្ចុប្បន្នបានយ៉ាងងាយស្រួល និងកំណត់មុខងារថ្មី។
- Dynamic behaviors-
អ្នកអាចមានថ្មាក់ដាប្រើនៃលមាននិធីសាស្ត្រដូចជាដឹងមុខងារផ្លូវការក្នុងក្នុងក្នុង **class dynamically**

សូមអគ្គិសន !!!