# 3. Advance JavaScript

1) String
2) Number
3) Let & Const
4) Array
5) Date
6) Loop For In, For Of
7) Type Convert
8) Regular Expression
9) Try Catch
10) Arrow Function
11) Promise / then & catch
12) Async/Await
13) JS DOM(Event Listeners)
14) Window Object  Methods
15) Additional

**1) String**

■ charAt()

String ရဲ့ index ကို ပေးပြီးအဲ့ဒီ index မှာရှိနေတဲ့ character ကို return ပြန်ပေးတာဖြစ်ပါတယ်။

■ charCodeAt()

String ရဲ့ index ကို ပေးပြီးအဲ့ဒီ index မှာရှိနေတဲ့ character (UTF-16) ကို return ပြန်ပေးတာဖြစ်ပါတယ်။

■ concat()

String နှစ်ခုကို join ပေးတာ ဖြစ်ပါတယ်။ရေးရတဲ့ ပုံစံကတော့ ဘယ်ကbaseဖြစ်ပြီးတော့ ညာဘက်က ဆက်ချင်တဲ့ string ဖြစ်ပါတယ်။

■ toUpperCase()

စာသားတွေကို (capital letter)အကြီးပြောင်းချင်တဲ့အခါသုံးပါတယ်။

■ toLowerCase()

စာသားတွေကို (small letter) အသေးပြောင်းချင်တဲ့အခါသုံးပါတယ်။

■ trim()

Textတွေမှာ whitespaceတွေကို ဖယ်ချင်တဲ့အခါ သုံးပါတယ်။

■ slice(s,e)

String ကိုအစနဲ့အဆုံး position သတ်မှတ်ပြီးဖြတ်ယူလိုက်တာဖြစ်ပါတယ်။

First parameter ကို စချင်တဲ့ position ကိုရေးပေးရပါတယ်။

Second parameter ကို အဆုံးposition ကိုရေးပေးရပါတယ်။

Negative position တွေပေးမယ်ဆိုရင် string ရဲ့ အဆုံးကနေစတင်တွက်ချက်ပါတယ်။

■ substring(s,e)

Slice နဲ့အားလုံးနီးပါး တူပါတယ်။မတူတဲ့ အချက်ကတော့ negative position ကို လက်မခံပါဘူး။

■ substr(s,l)

Sliceနဲ့တူပေမယ့် သူကတော့ start position ပေးပြီး ရောက်တဲ့နေရာကတော့ စတင်ရေတွက်ပါတယ်။

First parameter ကတော့ start position ဖြစ်ပြီးတော့

Second parameter ကတော့ length ဖြစ်ပါတယ်။

■ split(char)

String ကို char တခုပေးပြီး တလုံးချင်းစီခွဲထုတ်တာဖြစ်ပါတယ်။

သတိထားရမှာတော့ကိုယ်ဖြတ်ချင်တဲ့ string မှာ separator တခုတော့ အနည်းဆုံးပါရပါတယ်။နောက်တခုကတော့ split ဖြတ်ပြီးတော့ ရလာတဲ့တန်ဖိုး Array တန်ဖိုးနဲ့ရလာမှာပါ။

※ Array ကို တော့ နောက်အခန်းတွေမှာရှင်းပြပေးသွားပါမယ်။

■ includes(char)

String ထဲမှာ လိုချင်တဲ့ text ပါလားမပါလားကိုစစ်ပေးတာဖြစ်ပါတယ်။

True နဲ့ falseကို return ပြန်ပေးတာဖြစ်ပါတယ်။

■ indexOf(char)

String ထဲက ကိုယ်ချင်တဲ့ text ရဲ့position ကို ရှာဖွေတာဖြစ်ပါတယ်။

အကယ်၍လိုချင်တဲ့ textဟာ နှစ်နေရာ ပါခဲ့မယ်ဆိုရင်တော့ ပထမဦးဆုံးတွေ့တဲ့ position ကိုပဲreturn ပြန်ပေးမှာဖြစ်ပါတယ်။

■ lastIndexOf(char)

နောက်ဆုံးတွေ့တဲ့position ကို return ပြန်ပေးမှာဖြစ်ပါတယ်။

■ startsWith(char)

String မှာအစစကားလုံးမှာ ကိုယ်လိုချင်တဲ့textဟုတ်ရဲ့လားဆိုတာ စစ်ပေးတာဖြစ်ပါတယ်။ position တွေ return ပြန်ပေးမှာ မဟုတ်ပဲ တွေ့တဲ့ဆို true မတွေ့ဘူးဆို false ပဲပြန်ပေးမှာပါ။

■ endsWith(char)

String မှာအဆုံးစကားလုံးမှာ ကိုယ်လိုချင်တဲ့textဟုတ်ရဲ့လားဆိုတာ စစ်ပေးတာဖြစ်ပါတယ်။ position တွေ return ပြန်ပေးမှာ မဟုတ်ပဲ တွေ့တဲ့ဆို true မတွေ့ဘူးဆို false ပဲပြန်ပေးမှာပါ။

**2) Number**

■ toString()

Number ကိုString အနေနဲ့ပြောင်းပေးတာဖြစ်ပါတယ်။
Dateကိုလည်း Numberပြောင်းလို့ရပါတယ်။

■ Number()

String ကို  Number အဖြစ်ပြောင်းပေးတာဖြစ်ပါတယ်။

■ parseInt()

String ကို Numberအနေနဲ့ပြောင်းပေးတာဖြစ်ပြီးတော့ String မှာ “10.5” ပုံစံတွေပါလာခဲ့ရင်တော့ 10 ပဲreturnပြန်ပေးမှာပါ။
ဒီနေရာမှာ Number() သာဆိုရင် 10.5ကို return ပြန်ပေးမှာဖြစ်ပါတယ်။

■ parseFloat()

String မှာ ဒသမကိန်းတွေပါလာတဲ့အခါ ဒသမကိန်းတွေ မပျောက်ပဲ ပြန်ရချင်တဲ့အခါ သုံးပါတယ်။

**3) Let & Const**

**Let**

Variableတွေကို ကြေညာတဲ့အခါ var အပြင် Let ကိုကြေညာနိုင်ပါတယ်။

Let declaration မှာတော့ တကြိမ်ပဲကြေညာခွင့်ရှိပါတယ်။

Block Scope ဖြစ်ပါတယ်။

အပေါ်ကအချက်နှစ်ချက်ကို အသေးစိတ်ရှင်းလင်းပါမယ်။

1. Let declaration မှာတော့ တကြိမ်ပဲကြေညာခွင့်ရှိပါတယ်။

ဥပမာ

```
let a = 10;
let a = 0;
```

ဒီလိုသာရေးမယ်ဆို error တတ်ပါလိမ့်မယ်။ဘာကြောင့်လည်းဆိုတော့ Let ဟာ တခါပဲကြေညာလို့ရပါတယ်။

Var ကတော့အကြိမ်ကြိမ်ပြန်လည်ကြေညာလို့ရပါတယ်။

2. Block Scope ဖြစ်ပါတယ်။

ဥပမာ

```
{
  let x = 10;
}
```

{}ရဲ့အပြင်ဘက်မှာ x ကို ခေါ်သုံးလို့မရပါဘူး။var သာဆို ခေါ်သုံးလို့ရပါတယ်။ အချက်ဒမှာ ပြောခဲ့ ပြန်ကြေညာလို့ မရဘူးဆိုတဲ့အချက်ကို block scope နဲ့ပြန်လည်ဖြေရှင်းထားပါတယ်။ဆိုလိုတာကတော့

```
let a = 10;
let a = 0;
```

ဒီလိုသာဆို error တတ်ပါမယ်။သို့ပေမယ့်

```
let a = 10;
{
```

```
        let a = 0;
}
```

သာဆို error တတ်တော့ပါဘူး။block scope ထဲ ရောက်သွားတဲ့အခါ သက်သက်စီ ဖြစ်သွားတဲ့အတွက်ဖြစ်ပါတယ်။

**Const**

const declaration မှာတော့ တကြိမ်ပဲကြေညာခွင့်ရှိပါတယ်။

const declaration မှာတော့ value ကို ပြန်လည်Assign ထည့်လို့မရပါဘူး။

Block Scope ဖြစ်ပါတယ်။

Let နဲ့မတူညီတဲ့ အချက်ကတော့

const declaration မှာတော့ value ကို ပြန်လည်Assign ထည့်လို့မရပါဘူး။

```
const greeting = "say Hi";
greeting = "say Hello instead";
```

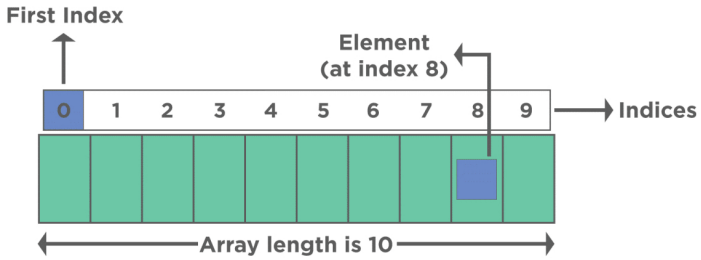
ဒီလိုမျိုး value ကို ပြန်လည်ကြေညာလို့ မရနိုင်ပါဘူး။ var နဲ့ Let ကတော့ ပြန်လည် Assign ထည့်လို့ရပါတယ်။

Const ကို

1. Array
2. Object အသစ်တွေကို Assign ထည့်တဲ့ အခါ သုံးကြပါတယ်။

**4) Array**

Array ကတော့ ရထားတွဲတတွဲနဲ့တူပါတယ်။အတွဲတတွဲမှာ data တန်ဖိုးတခုစီရှိကြပါတယ်။အတွဲတတွဲစီကို , နဲ့ပိုင်းဖြတ်ပါတယ်။




Syntax

```
const name_ = [data1, data2, ...];
```

Example

```
const friend   = [ “John”,”Mary”,”David”];
const age = [23,30,55];
```

နောက်တမျိုးကတော့Array အလွတ်တခုကိုစကြေညာပြီးနောက်မှ တန်ဖိုးကို အစားထည့်တာဖြစ်ပါတယ်။

Syntax

```
const home = [];
home[0] = “bedroom”;   //index အခန်း ၀
home[1] = “living room”;
home[2] = “kitchen”;
```

* Array ရဲ့အခန်းနံပါတ်တွေကို 0က စပြီးရေတွက်ပါတယ်။
အခန်းတွေကို index လို့ခေါ်ပါတယ်။

## Method (အသုံးများတဲ့Methodတချို့)

■ toString()

Array ကို String ပြောင်းချင်တဲ့အခါ အသုံးပြုပါတယ်။

```
const array1 = [1, 2, 'a', '1a'];
array1.toString())
// "1,2,a,1a"
```

■ concat()

Array တွေကို ဆက်ချင်တဲ့အခါသုံးပါတယ်။

```
const array1 = ['a', 'b', 'c'];
const array2 = ['d', 'e', 'f'];
const array3 = array1.concat(array2);
//"a", "b", "c", "d", "e", "f"
```

■ forEach()

Looping ပတ်တဲ့အခါသုံးပါတယ်။ရေးသားပုံရေးသားနည်း များစွာရှိပါတယ်။ လွယ်ကူတဲ့ပုံစံနဲ့ပြပါမယ်။

```
const avengers = ['thor', 'captain america', 'hulk'];
avengers.forEach((item, index)=>{
        console.log(index, item)
})

// 0 “thor”
// 1 “captain america”
// 2 “hulk”
```

■ includes()

Array ထဲ ကိုယ်လိုချင်တဲ့ data ရှိမရှိစစ်တဲ့အခါ သုံးပါတယ်။

```
const pets = ['cat', 'dog', 'bat'];

console.log(pets.includes('cat'));
// true
```

■ indexOf()

Includesနဲ့တူပေမယ့်သူကတော့အရင်ဆုံးတွေ့တဲ့ index numberကိုreturn ပြန်ပေးတာဖြစ်ပါတယ်။မတွေ့ရင်တော့ -1 ပြန်ပေးပါတယ်။Index 0 ကနေစစစ်ပါတယ်။

```
const array = [2, 9, 9];
array.indexOf(2);     // 0
array.indexOf(7);     // -1
```

■ lastIndexOf()

IndexOf() နဲ့တူပါတယ်သူကတော့ နောက်ဆုံးတွေ့တဲ့index number ကို Return ပြန်ပေးတာဖြစ်ပါတယ်။

```
const numbers = [2, 5, 9, 2];
numbers.lastIndexOf(2);     // 3
numbers.lastIndexOf(7);     // -1
```

■ isArray()

Array ဟုတ်လား မဟုတ်လား စစ်တဲ့အခါ သုံးပါတယ်။

```
Array.isArray([1, 2, 3]);  // true
Array.isArray({foo: 123}); // false
Array.isArray('foobar');   // false
Array.isArray(undefined);  // false
```

■ length

Array ရဲ့Length ကိုလိုချင်တဲ့အခါ သုံးပါတယ်။

```
const clothing = ['shoes', 'shirts', 'socks', 'sweaters'];
clothing.length;
//4
```

■ pop()

နောက်ဆုံးdataကိုဖြုတ်ချင်တဲ့အခါသုံးပါတယ်။

```
const plants = ['broccoli', 'cauliflower', 'cabbage', 'kale', 'tomato'];
console.log(plants.pop());
// "tomato"

console.log(plants);
```

```
// ["broccoli", "cauliflower", "cabbage", "kale"]
```

■ push()

Array ကိုနောက်ကနေထပ်ပေါင်းထည့်ချင်တဲ့အခါသုံးပါတယ်။

```
const animals = ['pigs', 'goats', 'sheep'];

animals.push('cows');
// ["pigs", "goats", "sheep", "cows"]
```

■ reverse()

Array ကိုပြောင်းပြန်ပုံစံပြောင်းချင်တဲ့အခါ သုံးပါတယ်။

```
const array1 = ['one', 'two', 'three'];

array1.reverse();
// ["three", "two", "one"]
```

■ shift()

ပထမဆုံးအခန်းက data ကို remove လုပ်ပေးတာဖြစ်ပါတယ်။

```
const array1 = [1, 2, 3];

array1.shift();
// [2,3]
```

■ slice(s,e)

Array ထဲdataတွေကို ဖြတ်ယူချင်တဲ့အခါသုံးပါတယ်။

```
const animals = ['ant', 'bison', 'camel', 'duck', 'elephant'];

const sliceArray = animals.slice(2, 4);
// ["camel", "duck"]
```

■ sort()

Arrayထဲက dataတွေကို Alphabet အတိုင်းစီပေးတာဖြစ်ပါတယ်။

```
const months = ['March', 'Jan', 'Feb', 'Dec'];

months.sort();
// ["Dec", "Feb", "Jan", "March"]
```

**Looping Array**

**1)**

```
const array = ['Item 1', 'Item 2', 'Item 3'];

for (let index = 0; index < array.length; index++) {
  console.log(array[index]);
}
```

**2)**

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let fLen = fruits.length;

text = "<ul>";
for (let i = 0; i < fLen; i++) {
  text += "<li>" + fruits[i] + "</li>";
}

text += "</ul>";
```

**5) Date**

လက်ရှိအချိန်၊နေ့ရက်တွေကို ရယူချင်တဲ့အခါ Date object ကိုအသုံးပြုကြပါတယ်။
Computer ရဲ့setting ထားတဲ့ အချိန်ကို ရယူတာဖြစ်ပါတယ်။

အသုံးပြုပုံကတော့

```
const date = new Date();
```

နောက်တမျိုးကတော့

```
const date = new Date(year, month, day, hour, minute, second, millisecond);
```

**Date.parse(string)**

String တွေကို date object ပြောင်းချင်တဲ့အခါ သုံးပါတယ်။
Valid ဖြစ်တဲ့string တွေကိုပဲ ပြောင်းလို့ရနိုင်ပါတယ်။

Valid

```
Var st_date = Date.parse("May 11, 2001");
const date = new Date(st_date);
```

Invalid

```
Date.parse("text 110”);
```

**Date Get Methods**

| | |
|---|---|
| getFullYear() | (yyyy) |
| getMonth() | (0-11) |
| getDate() | (1-31) |
| getHours() | (0-23) |
| getMinutes() | (0-59) |
| getSeconds() | (0-59) |
| getMilliseconds() | (0-999) |
| getTime() | milliseconds |
| getDay() | (0-6) |

**Date Set Methods**

```
const d = new Date();
setDate()             (1-31)
        d.setDate(15);
setFullYear()        (yyyy)
        d.setFullYear(2020);
setHours()            (0-23)
        d.setHours(22);
setMilliseconds()   (0-999)
        d.setMilliseconds(1628793509104);
setMinutes()         (0-59)
        d.setMinutes(30);
setMonth()            (0-11)
        d.setMonth(11);
setSeconds()          (0-59)
        d.setSeconds(30);
```

## 6) Loop For in, For Of

### For in

```
for (key in object) {
 //something
}
```

For in ကတော့ object တွေကို loop ပတ်တဲ့အခါ မှာ သုံးကြပါတယ်။

Example

```
Let person = {
   "name": "John ",
   "gender": "male",
   "age": 16
};
for (const key in person) {
   console.log(key + "=" + person[key]);
}
```

For in နဲ့ loop ပတ်တဲ့အခါ ရတာက key ဖြစ်ပါတယ်။ဒါကြောင့် object တွေမှာသုံးတဲ့အခါObject ရဲ့key ကိုရပြီတော့ valueကိုထုတ်ချင်တဲ့အခါ object[key] နဲ့ထုတ်ရပါတယ်။

### For Of

```
for (variable of iterable) {
  //something
}
```

Array တို့ string တို့ iteration လုပ်လို့ရတာတွေမှာအသုံးပြုပါတယ်။Object လို key တွေထုတ်ဖို့ မလိုတဲ့အခါ for of နဲ့သုံးရတာပိုလွယ်ကူစေပါတယ်။

```
const iterable = [10, 20, 30];

for (const value of iterable) {
  console.log(value);
}
// 10
// 20
// 30
```

**For In Vs For of**

```
let list = [4, 5, 6];

for (let i in list) {
   console.log(i); // "0", "1", "2",
}

for (let i of list) {
   console.log(i); // "4", "5", "6"
}
```

## 7) Type Convert

### Implicit

JavaScript ကအလိုလို type တခုကို auto convert လုပ်ပေးတာကို ဆိုလိုတာဖြစ်ပါတယ်။ဥပမာ ဒီလိုမျိုး အခြေအနေတွေပါ။

■ Implicit other datatype to String

```
let result;

result = '3' + 2;
console.log(result) // "32"

result = '3' + true;
console.log(result); // "3true"
```

■ Implicit String => Number

```
let result;

result = '4' - '2';
console.log(result); // 2

result = '4' - 2;
console.log(result); // 2
```

■ Implicit Boolean => Number

```
let result;

result = '4' - true;
console.log(result); // 3

result = 4 + true;
console.log(result); // 5

result = 4 + false;
console.log(result); // 4
```

■ Implicit Null => Number

```
let result;

result = 4 + null;
console.log(result);  // 4

result = 4 - null;
console.log(result);  // 4
```

■ Implicit undefined with other datatype

```
let result;

result = '4' + undefined;
console.log(result);  // 4undefined

result = 4 - undefined;
console.log(result);  // NaN

result = true + undefined;
console.log(result);  // NaN

result = null + undefined;
console.log(result);  // NaN
```

**Explicit Method**

Manually type Convert လုပ်ချင်တဲ့အခါသုံးတဲ့ method တွေကိုခေါ်ပါတယ်။

■ String()

```
//number to string
result = String(324);
console.log(result);  // "324"

result = String(2 + 4);
console.log(result); // "6"
```

```
//other data types to string
result = String(null);
console.log(result); // "null"

result = String(undefined);
console.log(result); // "undefined"

result = String(NaN);
console.log(result); // "NaN"

result = String(true);
console.log(result); // "true"

result = String(false);
console.log(result); // "false"
```

■ toString()

```
// using toString()
result = (324).toString();
console.log(result); // "324"

result = true.toString();
console.log(result); // "true"
```

■ Number()

```
// string to number
result = Number('324');
console.log(result); // 324

result = Number('324e-1')
console.log(result); // 32.4

// boolean to number
result = Number(true);
console.log(result); // 1
```

```
result = Number(false);
console.log(result); // 0

// null to number
result = Number(null);
console.log(result);  // 0

// '' to number
 result = Number(' ')
console.log(result);  // 0

// undefined to number
result = Number(undefined);
console.log(result);  // NaN
```

■ Boolean()

```
// other type to boolean
result = Boolean('');
console.log(result); // false

result = Boolean(0);
console.log(result); // false

result = Boolean(undefined);
console.log(result); // false

result = Boolean(null);
console.log(result); // false

result = Boolean(NaN);
console.log(result); // false
```

■ Date to Number()

```
const d1 =Number(new Date());
console.log(d1);
```

■ Date.toDateString()

```
const d1 = date.toDateString();
console.log(d1);
```

■ Date.toLocalTimeString()

```
const time = date.toLocaleTimeString();
console.log(time); // 1:13:12 PM
```

**8) Regular Expression**

Character တွေရဲ့ pattern တွေကိုသတ်မှတ်ပြီး၊ကိုယ်စစ်ချင်တဲ့ variable ကိုက်ညီလား မညီလား။စစ်တဲ့အခါ မှာ regular expression တွေကို အသုံးပြုပါတယ်။ တည်ဆောက်တဲ့ပုံနှစ်မျိုးရှိပါတယ်။

■ let re = /hi/;

Forward slash နဲ့သတ်မှတ်တဲ့ပုံစံ

■ let re = new RegExp('hi');

Constructor နဲ့ သတ်မှတ်တဲ့ပုံစံ

**Method**

1) reg.test(‘*_sometext*’);

Regular expression ကိုတည်ပြီးစစ်ချင်တဲ့ textကို ထည့်ပြီးစစ်ပေးရပါတယ်။

Return ကတော့ true / false ပြန်ပေးပါတယ်။

Example:

```
let re = /hi/i;
let result = re.test('Hi John');

console.log(result); // true
```

2) Str.match(reg);

စစ်ချင်တဲ့ string ကိုတည်ပြီးpattern ထည့်ပြီးစစ်ပေးရပါတယ်။

Return ကတော့ ကိုက်ညီတဲ့resultတွေကို array အနေနဲ့ ပြန်ပေးပါတယ်။

Example :

```
let str = "Are you Ok? Yes, I'm OK";
let result = str.match(/OK/gi);

console.log(result);

// ["Ok", "OK"]
```

**How to write Reg**

Syntax

/pattern/modifiers;

**Modifiers**

g : global match search until the end
i : ignore case-sensitive
m : multi line match

**Pattern**

| | |
|---|---|
| \d | Any digit character |
| \w | An word character (a-z, A-Z, 0-9, _) |
| \s | Any whitespace character (space, tab, newline, and similar) |
| \D | A character that is not a digit |
| \W | A nonalphanumeric character |
| \S | A nonwhitespace character |
| . | Any character except for newline |

**Useful Pattern**

[abcde..] - Any character between the brackets
[A-Z] - Any character from uppercase A to uppercase Z
[a-z] - Any character from lowercase a to lowercase z
[A-z ]- Any character from uppercase A to lowercase z
[^0-9] Any character that is NOT a digit
[^abc] Any character NOT a lowercase a b c.

**9) Try Catch**

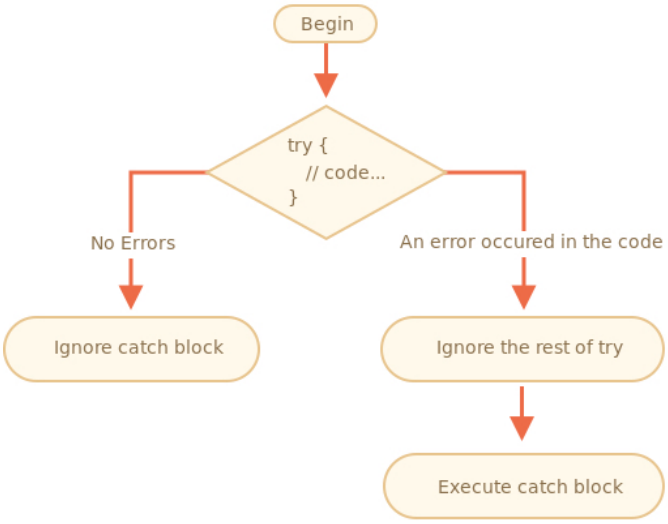
Syntax

```
try {

 // code...

} catch (err) {

 // error handling

} finally{

//execute ignore try catch Result

}
```

Try : ရဲ့သဘောကတော့ block ထဲမှာရေးလိုက်တဲ့ codeတွေကို run တယ်ပြီးတော့ အဲ့ဒီblockထဲမှာရေးထားတဲ့ code တွေထဲမှာ errorတွေတတ်ခဲ့ပြီဆိုရင် catch ထဲရောက်ပါတယ်။Error တတ်ခဲ့တဲ့ line အောက်က code တွေလည်း အလုပ်မလုပ်တော့ပါဘူး။

Catch : try ထဲရေးထားတဲ့ code တွေထဲerror or unexpected result တွေဖြစ်ပေါ်လာတဲ့အခါသူထဲကိုရောက်ပါတယ်။

Finally : error တတ်သည်ဖြစ်စေမတတ်သည်ဖြစ်စေအမြဲ သူ့block ထဲမှာရေးထားတဲ့ code တွေကို run မှာဖြစ်ပါတယ်။




- **No Error State**

```
try {
   console.log(1);
   console.log(2);
} catch (err) {
   console.log('errors');
}
finally {
   console.log("finish")
}
```

Result :
// 1
// 2
// finish

- **Error State**

```
try {
   console.log(1);
   addcodetoerroroccurs
   console.log(2);
} catch (err) {
   console.log('errors');
}
finally {
   console.log("finish")
}
```

```
// 1
// errors
// finish
```

**Error object catching**

```
try {
        ……
} catch (err) {
        console.log(err);
        console.log(err.name);
        console.log(err.message);
        console.log(err.stack);
   }
finally {
        ……
   }
```

Error တွေတတ်လာတဲ့အခါဘာကြောင့် ဒီerror ဖြစ်လာတဲ့ဆိုတာ trace လုပ်နိုင်ဖို့အတွက်Catch မှာ error object ထည့်ပေးထားပါတယ်။

Err: overall Reference Error Exception တွေကိုပြပေးပါတယ်။

Err.Name : တတ်နေတဲ့error name

Err.Message : error ရဲ့ဖြစ်နေတဲ့ အကြောင်းအရင်း

Err.stack : ဘယ်lineဘယ်နေရာမှာတတ်နေတယ်ဆိုတာပြပေးပါတယ်။

**10) Arrow Function**

Regular function တွေကို compact ဖြစ်အောင် ရိုးရိုးရှင်းရှင်းရေးနိုင်ဖို့ရာ arrow function တွေကို သုံးကြပါတယ်။Regular နဲ့ arrowမှာ this ကိုအသုံးပြုတဲ့အခါ ကွာခြားချက်တွေကတော့ regular မှာဆိုရင် this ကို redefine လုပ်ပါတယ်။ Arrow မှာကတော့ redefine မလုပ်ပါဘူး။

Syntax

```
() => expression

param => expression

(param1,…) => expression
```

■ Regular function

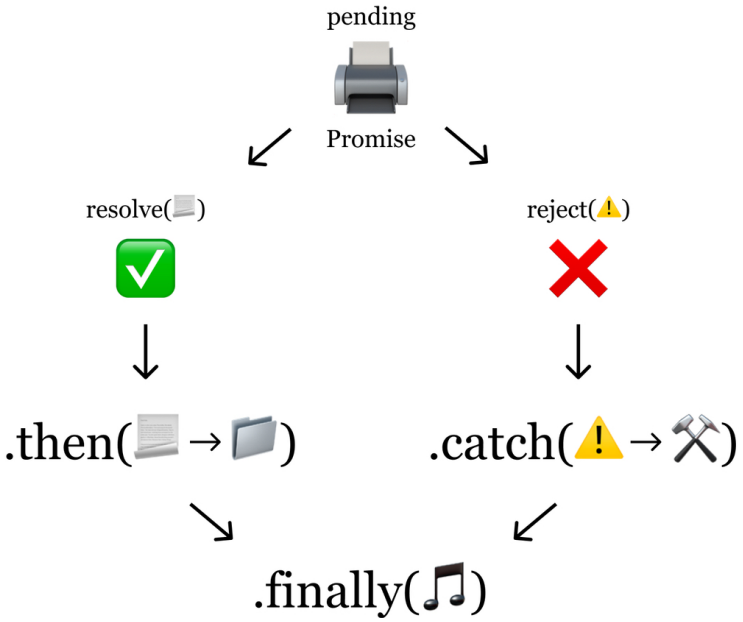
```
function sum(x,y){
        return x+y;
}

function sum2(x){
      return x+100;
}

function sum3(){
      return 10;
}
```

■ Arrow function

```
let sum = (x+y) => x+y;

let sum2 = x => x+100;

let sum3 = () => 10;
```

## 11) Promise / then & catch

Promise ဆိုတာကတော့ object တခုပါ။သူကတော့ Developer က resolve or reject State တွေကို promise ထဲမှာ Manage လုပ်ပြီး ပြန်ခေါ်သုံးပါတယ်။ Promise Code run နေတဲ့ အချိန်ဟာ pending ဖြစ်နေမှာပါ။




**Syntax**

```
let p = new Promise((resolve, reject) => {
   // resolve
   resolve(result);
   // reject
   reject(result);
})

p().then((param) => {
   // code for success
}).catch((param) => {
   // code for failure
}).finally(() => {
   // code for final
})
```

**Promise object**

Resolve : success ဖြစ်တဲ့ statement တွေအတွက် returnပြန်မယ်ဆိုရင် ခေါ်သုံးပါတယ်။

Reject : reject ဖြစ်တဲ့ အခါ return ပြန်ဖို့ ခေါ်သုံးပါတယ်။

**Using Promise**

.then : resolve() return ပြန်လာတဲ့အခါ

.catch : reject() return ပြန်လာတဲ့အခါ

.finally : result က ဘာပဲဖြစ်ဖြစ်နောက်ဆုံးလုပ်မယ့်အလုပ်

Example code:

```
function exam() {
   return new Promise((resolve, reject) => {
      let mark = 100;
      var result;
      if (mark == 100) {
         result = { price: "$100", level: "2" }
         resolve(result)
      } else {
         result = { fail: "Try Again!" }
         reject(result)
      }
   })

}

exam().then((message) => {
   console.log("resolve:" + message.price)
}).catch((error) => {
   console.log("reject:" + error.fail)
}).finally(() => {
   console.log("Exam Finish.")
})
```

**12) Asyn/ await**

API တွေ pending state (promise-based APIs)တွေနဲ့ရေးတဲ့အခါ ရိုးရှင်းလွယ်ကူစေရန်အသုံးပြုပါတယ်။

Syntax

```
async function name([param[, param[, ...param]]]) {
        await   function
}
```

■ **Not using Async and await**

```
function flow() {
   return new Promise((resolve, reject) => {
      let i = 0;
      resolve("work2");
   })
}

console.log("work1")
flow().then((message) => console.log(message));
console.log("work3")
```

```
//work1
//work3
//work2
```

■ **Using Async and await**

```
function flow() {
   return new Promise((resolve, reject) => {
      let i = 0;
      resolve("work2");
   })
}

async function running() {
   console.log("work1")
   await flow().then((message) => console.log(message));
   console.log("work3")
}

running()
```

```
//work1
//work2
//work3
```

## 13) JS DOM ( Event Listeners)

Events တွေကို Html tag တွေမှာ ထည့်မရေးပဲ JavaScript ဘက်က listener နဲ့ယူရေးတာဖြစ်ပါတယ်။နောက်ပိုင်းဒီပုံစံနဲ့ပေးဖို့ recommended ပေးပါတယ်။

Syntax

```
element.addEventListener(event, function);
```

Elementကတော့ target ထားတဲ့ selectorဖြစ်ပါတယ်။

Event ကတော့ target ပေါ်ကကိုယ်လိုချင်တဲ့ event ကို ဖမ်းပေးရတာဖြစ်ပါတယ်။

Onclick,onmouseout,onload..,

For All Events

https://www.w3schools.com/jsref/dom_obj_event.asp

Function ကတော့ actin ဖြစ်ပေါ်လာတဲ့အခါ အလုပ်လုပ်မယ့် code တွေကို ရေးပေးရမှာဖြစ်ပါတယ်။

Example:

```
Var p = document.getElementById(“#block”);

P.addEventListener("click", function(){
        alert("Hello World!");
});
```

## 14) Window Object Methods

1. alert()
2. atob()
3. blur()
4. btoa()
5. clearInterval()
6. clearTimeout()
7. close()
8. confirm()
9. focus()
10. getComputedStyle()
11. getSelection()
12. matchMedia()
13. moveBy()
14. moveTo()
15. open()
16. print()
17. prompt()
18. requestAnimationFrame()
19. resizeBy()
20. resizeTo()
21. scroll()
22. scrollBy()
23. scrollTo()
24. setInterval()
25. setTimeout()
26. stop()

## 15) Additional

### ■ Array

#### 1. Map

JavaScript Array or Array [object] တွေမှာ value တွေကိုလိုသလိုCombine လုပ်တာ၊Mapping လုပ်တာ စတဲ့ customization တွေလုပ်ပြီး Array အသစ်ဖန်တီးတဲ့အခါမှာအသုံးပြုပါတယ်။

#### 2. Filter

JavaScript Array or Array [object] ထဲမှာလိုချင်တာတွေကို လိုချင်တဲ့ condition ဖြစ်တဲ့ value or object တွေကိုFilter လုပ်ပြီး ဖြတ်ယူချင်တဲ့အခါအသုံးပြုပါတယ်။

#### 3. Sort

JavaScript Array or Array [object] တွေကို customize sorting စီတဲ့အခါမှာ အသုံးပြုပါတယ်။

### ■ Operator

#### 1. Logical OR (||)

Syntax

```
Exp1 || Exp2
```

Logical OR မှာဆိုရင်တော့ exp1 က true ဖြစ်ရင် exp1 ကို return ပြန်ပါတယ်။ exp1 က false ဖြစ်ရင် exp2ကို return ပြန်ပါတယ်။

```
o1 = true  || true      // t || t returns true
o2 = false || true      // f || t returns true
o3 = true  || false     // t || f returns true
o4 = false || (3 == 4)  // f || f returns false
o5 = 'Cat' || 'Dog'     // t || t returns "Cat"
o6 = false || 'Cat'     // f || t returns "Cat"
o7 = 'Cat' || false     // t || f returns "Cat"
o8 = ''    || false     // f || f returns false
o9 = false || ''        // f || f returns ""
o10 = false || varObject // f || object returns varObject
```

False ဖြစ်စေတဲ့အချက်တွေကတော့

- null;
- NaN;
- 0;
- empty string ("" or '' or ``);
- undefined

2. **Nullish ??**

Syntax

```
Exp1 ?? Exp2
```

Logical || နဲ့ပုံစံတူပါတယ်။ဒါပေမယ့် false ဖြစ်စေတဲ့ အချက်တွေကတော့

- Null
- Undefined နှစ်ခုပဲfalseဖြစ်ပါတယ်။ကျန်တာtrueဖြစ်ပါတယ်။

## ■ value Vs Reference

Javascript မှာ

1. Number
2. String
3. Null
4. Undefined
5. Boolean တွေဟာ value pass ဖြစ်ပါတယ်။ကျန်တာတွေက Reference pass ဖြစ်ပါတယ်။

## ■ Object. Freeze({})

Object တွေကို ပြင်လို့မရအောင် freeze လုပ်ချင်တဲ့အခါ အသုံးပြုပါတယ်။

**Useful Js Libraries**

- **Chart.js**
- **Apexcharts.js**
- **w2uijs**
- **animejs**
- **splidejs**
- **AOS Library**