

Modern Javascript

Fat arrow functions

```
let number() => {
    return 10;
}
```

Output: 10

$\boxed{() \Rightarrow \{ \} }$

Parameter
Area.

⇒ যদি একটাই statement থাকে ১০,

```
let number() => return 10;
```

যদি এই ক্ষেত্রে কিন্তু Return করে ৬৫লে,

let number() => 10 অহঁৰs Return নৰ লিখলেও
চলব. return বাছ অন্ত কিন্তু মাঝে দেখলে লিখলে
মাপুৰ নৰ.

⇒ Parameter থাকলে,

```
let number(n) =>
```

```
let number = (n) => {
    return n
}
```

বিষয় Parameter একটাৰ থাবলৈ Bracket omit কৰা আপুনি

let number = n => n

বিষয় একটা one-line arithmetic calculation & return কৰা আপুনি,

১ let number = (a, b) => a + b

বিষয় Fat arrow function আসত্বে main Reason হ'ল
this keyword গো confusion কৰে easy কৰা.

var Javascript = {

name: "Javascript"

libraries: ["React", "Angular", "Vue"]

printLibraries: function () {

var self = this;

this.libraries. \mapsto forEach (function (a) {

console.log(`\${this.name} loves \${a}`)

});

};

};

এবং function কে call করলে তা পুরো নিম্নোক্ত দেখাবে

undefined loves React

undefined loves the Angular

undefined loves Vue.

এখন undefined দেখাবে কাণ্ডা, this কে আরি call করার
পর, call করবে forEach: সেইজে this কে করে call করে
বিদ্বানের বাইপুর্স this এর,

এটো একটো probable solve হচ্ছে,

printLibraries: function (a) {

 var self = this;

 this.libraries.forEach(function (a) {

 console.log(` \${self.name} loves \${a}`);

 });

};

};

This is not convenient.

এটো যদি বাঁচাব তবে অস্থির arrow function use করি,

arrow function কে বাইপুর্স this কে কৈ value,

বিদ্বানের this কে কৈ value.

4 Event Listener w/ C#P Arrow function.

~~diff~~ after Event Listener w/ C#P Normal

const searchInput = Document.querySelector(".search")
("search")

const result = Document.querySelector(".result")

const thanks = Document.querySelector(".thanks")

function show() {

display.innerHTML = this.value;

var self = this

setTimeout(function() {

thanks.innerHTML = `You have typed:

\$self.value

}, 1000);

}

Event Arrow function use self or value auto at writing,

const show = () => {

display.innerHTML = this.value;

} care or self or

এবং Arrow function এর this টি হলু বাইকেও this, কিন্তু Global window.

এছা Javascript এর অন্যান্য function ও constructor function.

যদি arrow function এ convert করলে constructor function এর property একই রূপ থাকা

Truthy / Falsy Method

Truthy → • Explicitly false এর মতো, • Explicitly 0 এর মতো,
• Explicitly null এর মতো, • Explicitly undefined এর
মতো, • Explicitly empty string এর মতো, • Not a
number এর মতো,

Otherwise false.

~~let myVar = " "~~
~~if (myVar) {~~
~~if (myVar)~~

```
let myVar = ""  
if (myVar) {  
    console.log ("I am Truthy")  
} else {  
    console.log ("I am falsy")  
}
```

Output = I am falsy

Ternary Operator

`var age = 18;`

var type;

```
if (age >= 18) {  
    type = "adult"  
} else {  
    type = "child"  
}
```

એટો ચાર્ટેજું Ternary Operator નિયમ અને લિખી મુજબ
થાણું.

```
var type = (age >= 18) ? something : else something
```

Nesting Ⓛ କୁଟୁମ୍ବ ଧ୍ୟାନ

```
var type = (age >= 18) ? "adult" : (age <= 10) ? "child" : "young"
```

- * Age 18 ପୁଣ୍ଡ ତାରିଖ କୁଟୁମ୍ବ ରିଟର୍ନ ଏବଂ ଅଧିକାରୀ,

Age 18 " " 飛凡: ~~2nd~~ 2nd Age

↳ अगि age 10 वर्ष ६२८० रुपये,
सिल.

યું હન young.

Tips: 1 Level zip Maximum 2 Level zip Ternary operator
ব্যাপ্তিটা কৃত আছে.

Example: [1, 2, 3, 4] = odd or even

B | Array find() method

var numbers = [1, 2, 3, 4, 5, 6, 10]

numbers.find(function(o) { currentvalue }) {

return currentValue > 4

Output → 5, এটা ব্যাপ্তি আছে

function function নি মাধ্যমে true return করে করে iterate
ব্যক্তি দ্বারা আছে,

এখন find দ্বারা second parameter রিভেন্যু this কে আপ্তি আছে.

this কে second parameter রিভেন্যু নি আপ্তি আছেতে this
কে আপ্তি আর না, তেওঁর দ্বারা arrow function কে
convert করে দাবি করে this দ্বারা আপ্তি আছে.

findIndex()

```
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
var result = numbers.findIndex((currentValue, index, arr) => {  
    return !(currentValue % 2)  
})
```

Current Value

কাহি স্বত্ত্বা $\boxed{\text{currentValue} \% 2}$ ফিল্টার কোর্স,

এবং Iteration @, @ $1 \% 2 = \textcircled{1} \rightarrow$ কেবল ১। True এবং Go
function যেখানে কেবল প্রথম মেটি, কিন্তু অস্থায়, $!(\text{currentValue} \% 2)$ ফিল্টার কোর্স। $2 \% 2 = \textcircled{0} \rightarrow$ কেবল ০। false কেবল $!(2 \% 2)$ ।

কেবল true কেবল কেবল কেবল কেবল কেবল কেবল কেবল
কেবল currentvalue / Index কেবল কেবল কেবল কেবল কেবল

filter()

```
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
var result = numbers.filter((currentValue, index, arr) => {  
    return currentValue > 4;  
})
```

Output = [5, 6, 7, 8, 9, 10]

■ main array change 20% npt.

slice()

Var numbers1 = [1, 2, 3, 4, 5]

var result = numbers.slice(1, 3);

■ main array change 20% npt.

concat()

■ main array change 20% npt.

Var numbers1 = [0, 1, 2, 3, 4]

var numbers2 = [5, 6, 7, 8, 9, 10]

var result = numbers1.concat(numbers2);

■ main array change 20% npt.

multiple array থাকলে,

```
var result = numbers1.concat(numbers2, numbers3, numbers4)
```

push()

এখন Element add করে নোঃ

এখন Array কে change করে।

map() ** very much important for React.

for loop @ এখন আবশ্যিক Map use করো।

```
var numbers = [1, 2, 3, 4, 5, 6]
```

```
numbers.map((num) => {
```

```
    return 2 * num
```

```
})
```

প্রতিটা Element কে 2 দিলে তা কুণ্ডলীর main array
change হচ্ছে না।

reduce()

এখনকাণ্ডা complex কাটু রেডিচ নিষ্পত্তি পাবো।

মাইন অরেজেন্স কে চেঞ্চ কৰো আৰু

```
var numbers = [1, 2, 3, 4, 5, 6]
```

```
var result = numbers.reduce(function (preValue, currentValue) {
    return preValue + currentValue;
})
```

কেন্দ্ৰীয় ভোগ্যৰ value কে **পৰিৱেচন কৰা হৈছে** এবং ভোগ্যৰ value কে
মাইন current value মোড় কৰো **return** কৰা হৈ।

কিন্তু বিশ্বালো মাধ্যন কৰ **preCurrentValue** হিমোটো আৰু **কেণ্দ্ৰীয়**
কিন্তু তাৰ ফোলো **previous value** হৈ। এই **এক** sum টাৰে ২০
আৰু **দুই** কাঠলো গোৱাকৰ **2nd parameter** **newValue**
input **preValue** **input** হিমোটো হৈ।

এখনকাণ্ডা **parameter** **hিমোটো** **currentIndex** ও **main array** কে নিতো
পাবো।

এখনকাণ্ডা **array** কে **reduce** কৰো **বিকল্প** **single value** কৰ
convert কৰো আৰু।

for loop, for in, for of.

⇒ `for (let i=0; i<=5; i++) {
 console.log(i)
}`

`console.log(i)` ⇒ output হিতে এট কোন \neq let এবং
একটি \neq block scope, block এতে এইটি সব কাগ করে
না।

~~for (let i=0; i<=5; i++) {
 console.log(i)
}~~

`console.log(i)`

Output

0
1
2
3
4
5

→ ⑥ ⇒ এখন শেষ এক টুকু আছে, কিন্তু এখন
যে কোনো Loop এলো নাই।

Important object topics

```
var myObj = {  
    name: "Javascript",  
    founder: "Brendan Eich",  
    estd: "1995",  
    ranking:  
    ranking: 1  
}
```

Key അല്ലെങ്കിൽ

```
var keys = Object.keys(myObj) output [name, founder, estd, ranking]
```

values അല്ലെങ്കിൽ

```
var values = Object.values(myObj) output [Javascript, Brendan Eich, 1995, 1]
```

key value pair അല്ലെങ്കിൽ

```
var entries = Object.entries(myObj) output [  
    [name, 'Javascript'],  
    [founder, 'Brendan Eich'],  
    [estd, '1995'],  
    [ranking, 1]
```

Professional life હજુ most of the things ગુરુ ૧૦/૩૫
નિર્વાહ હતું ૨૦૨૫ વર્ષ,

Object Shorthand:

* ~~var newObj =~~

x = 25;

y = 40;

var newObj = {

x, // સર્ટ એક કેન્દ્રીય વિષય

y, // સર્ટ, એક કેન્દ્રીય વિષય

function default

function myFunc(x){

return x;

}

આમણું આપનું એવી ફંક્શન હોય કે તેણું પરમેટર નથી અને,
જો કેંદ્રીય Default value હિન્દું

```
function Myfunc(x=10){  
    return x  
}
```

- **कोला Parameter** ए अप्लॉट चर 10 फॉर.
- ⇒ यादि यही अप्लॉट undefined अप्लॉट तेंवर चर 10 फॉर. काठा
असे undefined मार्ग निकाळ आपासीला आ.
- ⇒ null अप्लॉट चर null value return करता.

spread Operator

इति Extensively Operator use करता.

इति शुभेच्छा द्वारा

```
var numbers = [1, 2, 3]
```

```
var newNumbers = [numbers[0], numbers[1], numbers[2], 4, 5,
```

]

अशीजू नंबर्स अर्रेय एचे एलेमेंट ओप्पाटे नंबर्स नंबर्स
ची शुभेच्छा द्वारा शक्ता. निकाळ देते द्वारा शक्ता spread oper

operators ହିଁଲେ କଣ୍ଠ ମେତା.

newNumbers = [...numbers, 4, 5, 6]

↳ numbers array କୁ element
ଅନ୍ୟାନ୍ୟ ଦୃଢ଼ିତ୍ରୀ ଥିଲା ନାହିଁ.

Important

ଏ Immutability କୁ କିମ୍ବା ~~number Array~~ spread operator
use କଣ୍ଠ ଥିଲା. Like,

var Numbers = [1, 2, 3, 4];

var a = Numbers;

• ଏଥିଲା ପ୍ରଥମ ଯାହିଁ Numbers change କାହିଁ ହେବାରେ a ଓ
କିମ୍ବା bକୁ କିମ୍ବା କିମ୍ବା ମାତ୍ର କିମ୍ବା problem ଯିବା ଥାଏନ୍ତି ଏହି
spread operator ଆବଶ୍ୟକ ହେବାରେ କାହାରେ କାହାରେ ଆବଶ୍ୟକ ଥାଏନ୍ତି.

• var Number = [5, 6, 7, 8]

var b = [...Number] ଏହିଟି Number Array change
କରାଯାଇବାକୁ b change କରାଯାଇବାକୁ

React ৰে state update কৃত্যাংশ কথন ও spread operator
কৈলাস খেমা প্রয়োগ কৰি দেই,

ৰে Spread operator কৰাবলো কৈলাস Array concat কৰি দেই.

`var numbers1 = [1, 2, 3]`

`var numbers2 = [4, 5, 6]`

`var numbers = [...numbers1, ...numbers2]`

Object ৰে spread operator কৰাবলো কৰি দেই.

```
var myObj = {  
    x: 1,  
    y: 2  
}
```

```
var myObj2 = {  
    a: 1,  
    b: 2  
}
```

`var newObj = {...myObj1, ...myObj2}`

Rest Operator

- এখানে spread operator দ্বাৰা মূলৰ (.) কিন্তু আপোস্ট
ব্যবহৃত হচ্ছে আৰু ভিত্তিমূলৰ কেবলু উচ্চৈৱ different.
- function দ্বাৰা Parameter দ্বাৰা (.) ব্যৱহৃত
কৰলে অটা Rest operator দ্বাৰা মূলৰ কাৰণ কৰাৰ,

```
function myFunc (a, b) {  
    console.log(a, b)  
}
```

myFunc(2, 5) \Rightarrow output \Rightarrow (2, 5)

এখানে ব্যৱহৃত কৰি আৰি, কৃষ্ণী Parameter কৰিবলৈ নিবৰ্তন,
কিন্তু বিশেষ scenario পেজাতে আৰু স্থানীয় এখানে
আপোস্ট আৰি এই Function দ্বাৰা (.) দ্বাৰা Parameter
use কৰাৰ Rest operator আপোস্ট ব্যৱহৃত কৰাৰ
arguments কিন্তু কৰা হৈতো.

```
function myFunc() {  
    console.log(arguments)
```

myFunc(4, 2, 3) $\xrightarrow{\text{out}}$ ~~{'0': 4, '1': 2, '2': 3, '3': 7, '4': 9}~~
~~{'0': 1, '1': 2, '2': 3}~~

ဒြို့ ပွဲ ဒြို့ object ဒြို့ ပေး for in နှင့် iterate အတွက်
ထုတ် value အားလုံး ပေါ်မှု.

Rest operator ပျော်ဆုံး ပွဲ အန္တိတိ နည် ၂၀၁၇ ခုနှစ်၊ ၃၀ ဧပြီ

function myFunc (...params) {

 console.log (params);

}

myFunc (4,5,6,7,8,9); → [4,5,6,7,8,9]

function myFunc (a, ...params) {

 console.log (params);

}

myFunc (6, 7, 8, 9)

 ↑ a ↓ params.

console.log

Destructuring

এই এসি গুরুত্বপূর্ণ Beautiful feature.

Object Destructuring:

কোনো Object থেকে, তেওঁর কোনো কোটি পুরো ফার্ম করে
নিয়ে আসা।

```
const user = {  
    id: 339,  
    name: 'Sakib'  
    age: 36  
};
```

-আমি টাক্সি object থেকে Name (যে কোন এক) variable
assign করবে আর অন্য কোনো কোটি কৃতি নাহি।

Old method:

```
var new name = user["name"];  
console.log(name)
```

Destructuring:

Object destructuring

1.

= user;



এখন একটি object যাকে বাস্তবাণীতে object
নামে আসুন,

2.

{ } = user;

↓ ↓
object object

এখন একটি object name কে প্রেরণ করে নিতে আবশ্যিক

3. const {name} = user;



name এর একটি constant হিসেবে user
object এর name প্রেরণ করে দিনোরা.

4. যদি আমি অন্য ফোলো বাস্তবাণী variable হিসেবে assign
করতে চাই?

const {name: title} = user

↳ title এর একটি constant হিসেবে
assign করা যাবে.

Nested Objects

```
const user = {
    id: 333
    name: 'sakib'
    age: 35
    education: {
        degree: "Masters"
    }
};
```

এই আর্গেমেন্ট করে কী?

```
const {education: {degree}} = user;
```

↳ Last level এর অবস্থা object এর মধ্যে।

```
const {education: {degree: x}} = user
```

কুনি আর্গেমেন্ট করে json format কী

Data প্রেসের ক্ষেত্রে অবশ্যই education ক্ষেত্রের মধ্যে object
কী education নাই। তখন ফেল ক্ষেত্রে Error সিগন,

- କେବଳ ଡିଫ୍‌ଲୁଟ ଡେଫ୍‌ଲୁଟ ନାହିଁ ଏହିରେ

const { education : { degree } = {} } use = user;

console.log (degree);

→ education ନାହିଁ ଏହିରେ error return
ନାହିଁ କଣ୍ଠେ undefined return କରିବ.

■ Array Destructuring:

var numbers = [1, 2, 3, 4, 5]

-ଆମିର ବିଶ୍ୱ ତ କେବଳ ଏକଟି variable କି assign କରିବୁ, ୨ ଥିଲା
ଅଛି ଏକଟି variable କି Assign କରିବୁ

var [a, b] = numbers;

numbers ଏହି ଏକ ମୂଳ୍ୟ ନିମ୍ନେ a ଓ b କି assign କରୁଣିବୁ;

-ଆମିର ଏହି ୨ ଫାର୍ମାଟ କେବଳ assign କରିବୁ କିମ୍ବା

var [
 ↑
 1 a b
 ↓
 skip 2 in a skip 4 in b,

a → 2
b → 4

~~Var~~

```
var numbers = [1, 2, [3, 100, 500], 4, 6]
```

```
var [, [, a, b], ] = numbers;
```

1 skip	500 → b
2 skip	
3 skip	
100 → a	

```
console.log(a, b); → 100 500
```

Value swapping

```
var a = 1
```

```
var b = 2
```

$[b, a] = [a, b]$

Value swap swap 2(2) 2(1)

Module Import/Export

```
const pi = 3.1416
```

```
const a = 2.9
```

```
export pi = 3.1416
```

```
export a = 2.9
```

External.js

```
{
  "type": "module"
}
```

এটি একটি স্ট্রিঙের মধ্যে অন্তর্ভুক্ত
import/Export করা হয়।

Package.json

```
import {pi, a} from
  './external.js';
```

named import
 React.js
 নির্দেশ করা হয়।

main.js

একটি বিশেষ করে নাম দিয়ে বিনিয়োগ করা হয়।

Library use করার Package.json file নির্মাণ,

মাধ্যমিক বিকল্প উন্নতি করা

```
import * from "./external.js";
```

```
import * as test from "./external.js";
alias.
```

```
console.log(test.pi) → 3.1416
```

* \Rightarrow import {a as varA, pi as varPi} from "./external.js";
varA গো মুক্তি a এ varpi গো মুক্তি pi রেখা করো,

বিন্দু Default Import & Export:

বিন্দু একটা file by default কিম্বা export রেখা;

* \Rightarrow import external from './external.js'
↳ default import রেখা is external.js
file রেখা by default রেখা export রেখা
মানে রেখা রেখা 'external'.

ওয়ান্স external.js এ কিম্বা এ চেণ্ট করো

বিন্দু রেখা;

export const pi = 3.1415;

const a = 2.9;

export default a;

↳ a ~~একটা~~ a রেখা by default export রেখা

import default external, {pi, ~~varpi~~} from "./external.js";

main.js

Default (go on) { } will be default.

Function export

```
export function myFunc() { // named export.  
    console.log ("I am myFunc")  
}
```

now

Import external, {myFunc, pi} from "et external.js";

↓
main.js

default export is func,

```
export default function myFunc()
```

test as I say func.

test as it is

}

Default export is func or var.

Template Literals

```
var a = 5;
```

```
var b = 6;
```

```
console.log(`I am 'a' and I am 'b'`)
```

```
console.log(`I am ${a} and I am ${b}`)
```

```
console.log(`I am ${a+b}`)
```

```
console.log("I am test")
```

```
this is test"
```

↓
error siror,

```
console.log(`I am test  
this is test`)
```

→ no error.

Output → I am test
this is test

multi line go on template literal ega useful.

sets & weak set

▣ Unique Data store করার জন্য set ব্যবহার করো।

~~যথৈতু~~

থাইল গোষ্ঠী Array Declare করি,

```
let myArray = [] ; // literal syntax
```

```
let myArray = new Array() // constructor syntax
```

▷ Set Declare করার জন্য Literal syntax ব্যবহার করে
মাত্র। constructor syntax ব্যবহার করো।

```
let mySet = new Set();
```

Set Methods

Data Insert:

```
mySet.add(5);  
              ↓  
              value
```

```
mySet.add('Bangladesh')
```

Delete:

```
mySet.delete('Bangladesh')
```

Has or not:

```
mySet.has(5) → True
```

কারূণ্য 5 set এ আছে।

▣ Set কর্তৃ মেথড কর্তৃপক্ষ কর্তৃপক্ষ প্রক্রিয়া set কর
কর্তৃ মডিফাইড set কর্তৃ কর্তৃ কর্তৃ কর্তৃ।

Sub.: _____

SAT SUN MON TUE WED THU FRI
O O O O O O O

DATE: / /

mySet.add(5).add(6).add('Bangladesh')

↓
return {5}.add(6)

↓
return {5, 6}.add('Bangladesh')

↓
return {5, 6, 'Bangladesh'}

clear; mySet.clear()

↳ এটির প্রস্তুতি clear API কর্তৃপক্ষ মনে

size; mySet.size.

SET from Array:

let myArray = [1, 2, 3]

let mySet = new Set(myArray);

Set গো এটি পৃথিবীর Array Pass করা থাকে এবং এমন না

set গো এটি ইটেমের Iterable Pass করা থাকে,

এবং Object ফর্মের ক্ষেত্রে এখানে object Iterable এর

let mySet = new Set ('Bangladesh')

{ "B", "a", "n", "g", "l", "d", "e", "s", "h" }

Set নিম্নোক্ত Iterable,

for (let value of mySet) {
 log (value);

}

Set এর কোনো Order নেই।

Set Array from set.

let mySet = new Set (myArray) } (for use ১১৫)

[... mySet] → array

or

Array. from (mySet)

Set Usage:

⇒ Unique Element নিম্নোক্তগুলি

let myArray = [1, 2, 3];

let mySet = new Set (myArray);

mySet. add (4);

log (mySet); $\Rightarrow \{1, 2, 3, 4\}$

পৰ্যন্ত 3 add কৰলেও 2nd আবিৰণ 3 already set

-যোগী,

~~let mySet = {~~

~~mySet.add~~

mySet.add ~~object~~ (

a: 5

b: 2,

)

mySet.add (

a: 5

b: 2

)

ত্ৰিভেনি mySet পৰ্যন্ত object কৰিব। আবিৰণ কৰে

object কৰে শৰকিলু same অস্তি' গতি Reference

যোগীতা:

Q পৰ্যন্ত 2nd,

let object = {

a: 5,

b: 2,

)

mySet.add (object)

mySet.add (object)

-କେବଳ କ୍ରମିକ object ଥାକିଲା.

Ans

କୁ set ଗୁଡ଼ union = new set ([a

କୁ set ଗୁଡ଼ union:

let a = new set ([1, 2, 3])

let

let a = new set ([1, 2, 3])

let b = new set ([4, 3, 2])

let union = new set ([...a, ...b]);

e

କୁ set ଗୁଡ଼ Intersection:

let a = intersection = new set ([...a]. filter (x => b.has(x)));

b set କେବଳ x ଟାଙ୍କେ ଗଲାଟୁ value true ଗୁଡ଼

ଫିଲ୍ଟର a set filter କିମ୍ବା 2ଟି

କୁ set ଗୁଡ଼ difference:

let difference = new set ([...a]. filter (x => !b.has(x))

ସେଇବେ a set କେବଳ element B set କେବଳ ନାହିଁ
କେବଳ filter.

Weak Set

```
const ws = new WeakSet();
```

```
log(ws);
```

adding:

```
ws.add({})
```

* weakset ദേശവ്യാപക ഓബ്ജക്ട് add ഫൂട്ട് ആണ്
• number, string add ചെയ്യാൻ അനുഭവിക്കാം

• അന്തര്,

```
ws const ws = new WeakSet([{}], {})
```

* weakset iterable എന്നതും ഒരു iterable

എന്ന്

WeakSet Usage:

```
class const ws = new WeakSet();
```

```
class someClass {  
    constructor() {
```

```
}
```

```
method() {
```

?

எழுநிர் class @ object access க்கு படியும் என,

```
const a = new SomeClass();
```

a.method

தொடர் access க்குடி இருக்கும்,

உங்கள்,

```
log (SomeClass.prototype.method());
```

அதில் ~~function~~ வெளியிட சால்தர் Method க்கு directly access நிலை ஏற்கு அதே போன்று,

```
☞ const ws = new WeakSet()
```

```
class SomeClass {
    constructor() {
        ws.add(this)
    }
}
```

```
method() {
    if (!ws.has(this)) {
        throw new Error('You cannot access
        this method directly!');
    } else {
        return 'i am method';
    }
}
```

এখন direct Q10 access করা হচ্ছে এই,

Tagged Template Literals:

```
var player1 = "Sakib";
```

```
var player2 = "Tamim"
```

```
console.log(`We have ${player1} and ${player2} in  
our cricket team`)
```

এখন এই template literals রে কেবল value দেওয়া
করার পাশে আর কোনো কোড দেওয়া না হলো,
কেবল কোথাও কোড নেই।

```
function no modifier(strings, ...values) {
```

}

```
console.log(modifier `We have ${player1} and ${player2}  
in our cricket team`)
```

Output →

strings → ['we have' , 'and' , 'in our cricket team'
] → value র মধ্যে ফিল্ট.

values ['sakib' , 'tamim']

ବିଭିନ୍ନ ଶାଖାରେ କମିଟି ପାଇଁ ଏହାରେ କମିଟି ପାଇଁ

function modifier (strings, . . . values) {

const = strings. reduce ((prev, current) ⇒ {

return prev + current + values.length ? value="Mr."
+ values.shift() : ""

}, "")

return m.

}

Q: prev 2nd, " " Go; current 2nd "We have".

Q: shift, 1st 2nd Mr. + values. shift()
↳ values array ↳ 1st element.

Q: 1st prev 2nd, "We have Mr. Sakib" ↳ 3rd

current 2nd "and", Q: shift (2nd) 2nd

Mr. + values. shift(), ↳ 4th

Q: 2nd

prev 2nd, " we have Mr. sakib and Mr.
tamim" Q: current 2nd "in our team"

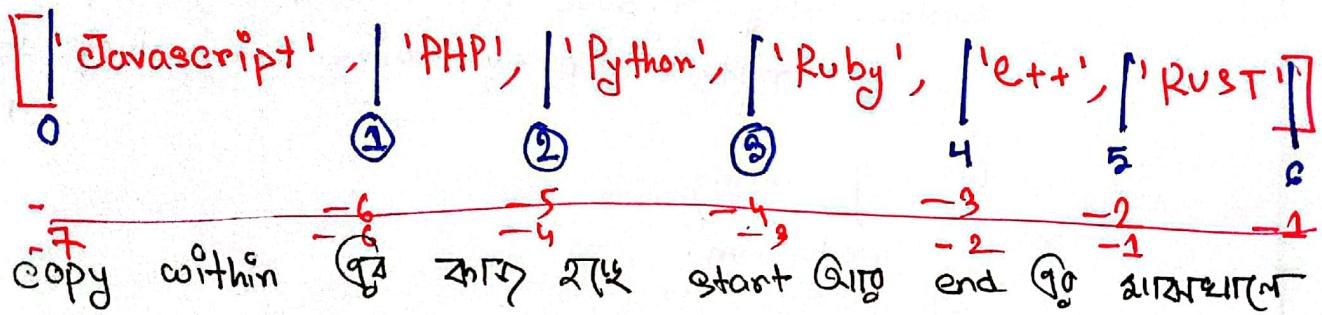
so, ultimate output

" we have Mr. sakib and Mr. tamim in
our team."

Array. copyWithin()

• `copyWithin(target, start, end)`

বিঃ: 3 1 2



যা পেরু গিল্ডেক কপি করে target @ স্থানে ফেলো।

অন্যান্য বিধানে Ruby এর সাথে বিঃ (1) & (2) কে মানে

PHP অন্য (3) Ruby অন্যান্য target এর সাথে মানে

বাস্তু সাথে

কেন্দ্রে • position (>) কিন্তু count করে আছে,

Q

মুঠে copyWithin() কি কৈবল্যে?

মুঠে Target is 'Required'

মুঠে start কে কৈবল্যে end optional.

মুঠে 'start' defaults to 0.

মুঠে end defaults to array.length.

- never changes array length.
- Overwrites the original array.
- Return ~~to~~ no modified array.