

Java Script

Print:

document.write():

```
<script>
  document.write("Hello World" + "<br>");
</script>
```

Output:

Hello World

console.log():

Result will be shown on browser's console. Browser's inspect -> console:

```
<script>
  console.log("Hello World");
</script>
```

Output:

Hello World

Variable declaration:

```
<script>
  let a = 10;
  let b = 20;
  var c = 30;
  const d = 40;
</script>
```

Operators:

Assignment:

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Arithmetic:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

```
<script>
  let a = 10;
  let b = 20;

  document.write("Hello World" + "<br>");
  document.write(a + "<br>");
  document.write(b + "<br>");
  document.write(a + b + "<br>");
  document.write(a - b + "<br>");
  document.write(a * b + "<br>");
  document.write(a / b + "<br>");
  document.write((a % b) + "<br>");
</script>
```

Comparison:

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Logical:

Operator	Description
&&	logical and
	logical or
!	logical not

Type Operators:

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

Bitwise Operators:

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
	OR	5 1	0101 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2

Comments:

Single Line:

```
// document.write("Hello World" + "<br>");
```

Multiple Line:

```
/*  
document.write(a + "<br>");  
document.write(b + "<br>");  
document.write(a + b + "<br>");  
document.write(a - b + "<br>");  
document.write(a * b + "<br>");  
document.write(a / b + "<br>");  
document.write((a % b) + "<br>");  
*/
```

Data Types:

1. String
2. Number
3. Boolean
4. Null
5. Array (Non Primitive)
6. Float
7. Object (Non Primitive)
8. Undefined
9. RegEx (Regular Expression) (Non Primitive)

```
<script>
```

```
let a = "John Doe"; // String  
let b = 20; // Number  
let c = true; // Boolean  
let d = 10.1; // float  
let e = null; // null  
let f = [1, 2, 3]; // Array  
let g = ["John", "Doe"]; // Array  
let h = {  
  name: "John Doe",
```

```

    city: "New York",
    age: 30,
  }; // Object
  let i; // undefined

  // to show on browser
  document.write(a + "<br>");
  document.write(b + "<br>");
  document.write(c + "<br>");
  document.write(d + "<br>");
  document.write(e + "<br>");
  document.write(f + "<br>");
  document.write(g + "<br>");
  document.write(h + "<br>");
  document.write(i + "<br>");

  // to show on console
  console.log(a);
  console.log(b);
  console.log(c);
  console.log(d);
  console.log(e);
  console.log(f);
  console.log(g);
  console.log(h);
  console.log(h["name"]); // picking specific value
  console.log(h.city); // picking specific value
  console.log(i);

</script>

```

Output:

on browser:

```

John Doe
20
true
10.1
null
1,2,3
John,Doe
[object Object]
undefined

```

on console:

John Doe

```
03. data-types.html:37 20
03. data-types.html:38 true
03. data-types.html:39 10.1
03. data-types.html:40 null
03. data-types.html:41 (3) [1, 2, 3]
03. data-types.html:42 (2) ['John', 'Doe']
03. data-types.html:43 {name: 'John Doe', city: 'New York', age: 30}
03. data-types.html:44 John Doe
03. data-types.html:45 New York
03. data-types.html:46 undefined
```

If-Else:

```
<script>

  const marks = 80;

  if (marks >= 80 && marks <= 100) {
    console.log("A");
  } else if (marks >= 60 && marks < 80) {
    console.log("B");
  } else if (marks >= 40 && marks < 60) {
    console.log("C");
  } else {
    console.log("D");
  }

</script>
```

Loop:

1. For Loop
2. For In Loop
3. While Loop
4. Do While Loop

for loop:

```
<script>

  let i;
```

```
    for (i = 0; i < 10; i++) {  
        console.log(i);  
    }  
  
    for (i = 0; i < 10; i++) {  
        document.write("<button>Click</button><br/>");  
    }  
  
</script>
```

while loop:

```
<script>  
  
    let i = 0;  
  
    while (i < 10) {  
        console.log(i);  
        i++;  
    }  
  
</script>
```

do while loop:

```
<script>  
  
    let i = 0;  
  
    do {  
        console.log(i);  
        i++;  
    } while (i < 10);  
  
</script>
```

```
<script>  
  
    let i = 0;  
  
    do {  
        document.write("<button>Click</button><br/>");  
        i++;  
    } while (i < 10);
```

</script>

Function:

JavaScript ফাংশন (Function) কি?

সাধারণ ভাষায় বুঝি 🤔

ফাংশন হল একটি "মজিক বাক্স" যা কিছু নির্দিষ্ট কাজ করে। আপনি যখন ফাংশনকে "কল" করেন, সেটি তার ভিতরের কোড গুলো সম্পাদন করে।

কোডের বিস্তারিত ব্যাখ্যা 🧐

ফাংশন ডিক্লেয়ারেশন

```
function add() {  
  let a = 10;  
  let b = 20;  
  let c = a + b;  
  console.log(c);  
}
```

ফাংশনের প্রধান অংশগুলি:

1. `function` কীওয়ার্ড
2. ফাংশনের নাম (`add`)
3. `()` বন্ধনী
4. `{ }` ব্লক যার মধ্যে কোড থাকে

ফাংশন কল করা

```
add(); // এটি ফাংশনকে কল করে
```

ফাংশনের বিভিন্ন ধাপ

1. ভ্যারিয়েবল ডিক্লেয়ারেশন

```
let a = 10; // প্রথম সংখ্যা  
let b = 20; // দ্বিতীয় সংখ্যা
```


2. গণনা

```
let c = a + b; // a এবং b যোগ করে c-তে সংরক্ষণ
```

3. আউটপুট

```
console.log(c); // c-এর মান প্রিন্ট করবে
```

সহজ উদাহরণ:

মনে করুন, আপনি একটি ফাংশন বানাচ্ছেন যা আপনার নাম প্রিন্ট করে:

```
function sayHello() {  
    console.log("হ্যালো, আমি রাহুল!");  
}  
sayHello(); // এটি কল করলে "হ্যালো, আমি রাহুল!" প্রিন্ট হবে
```

গুরুত্বপূর্ণ নিয়ম

1. ফাংশন নাম সাধারণত camelCase হয়
2. ফাংশন নাম দিতে হবে অর্থবহ
3. ফাংশনের কাজ সুস্পষ্ট হওয়া উচিত

সাবধানতা

- ফাংশন কল না করলে তার ভিতরের কোড চলবে না
- নাম ঠিকভাবে লিখতে হবে

মজার তথ্য

JavaScript-এ ফাংশন হল "first-class citizen" - মানে আপনি ফাংশনকে ভ্যারিয়েবল হিসাবেও ব্যবহার করতে পারেন!

সাধারণ ফাংশন সিনট্যাক্স

```
function functionName() {  
    // কোড ব্লক  
    // এখানে আপনার কাজগুলো লিখবেন  
}
```

কোড ব্যাখ্যা

1. function কীওয়ার্ড

- ফাংশন তৈরির জন্য ব্যবহৃত হয়
- JavaScript বুঝতে পারে যে এখন একটি ফাংশন শুরু হচ্ছে

2. functionName

- আপনার ফাংশনের নাম
- সাধারণত camelCase ব্যবহার করুন
- অর্থবহ নাম দিন যেন বুঝা যায় ফাংশন কী করে

3. ()

- খালি বন্ধনী
- পরবর্তীতে এখানে আর্গুমেন্ট পাস করা যাবে

4. { }

- কোড ব্লক
- ফাংশন যে কাজগুলো করবে সেগুলো এখানে লিখবেন

ফাংশন কল করা 📞

```
functionName(); // ফাংশন কল করা হয় এইভাবে
```

সম্পূর্ণ উদাহরণ

```
function add() {  
  let a = 10;    // প্রথম সংখ্যা  
  let b = 20;    // দ্বিতীয় সংখ্যা  
  let c = a + b; // যোগফল  
  console.log(c); // ফলাফল দেখান  
}  
  
add(); // ফাংশন কল করা হল
```

গুরুত্বপূর্ণ বিষয়সমূহ ✅

- ফাংশন নাম ইংরেজিতে লিখতে হয়
- নাম কোনো সংখ্যা দিয়ে শুরু হতে পারবে না

Arrow Function: মজার মজার ভাবে বুঝি 🚀

কল্পনা করুন...

মনে করুন আপনি একটি "মহাকাশের যান" পেয়েছেন যা খুব দ্রুত কাজ করতে পারে! এই মহাকাশ যানটিই হচ্ছে Arrow Function! 🚀

সাধারণ ফাংশন vs Arrow Function 🔍

পুরানো পদ্ধতি (Normal Function)

```
function add(a, b) {  
  return a + b;  
}
```

নতুন পদ্ধতি (Arrow Function)

```
const add = (a, b) => a + b;
```

কেন এটা মজার? 😊

1. দ্রুত কাজ
 - কম লাইন কোড
 - তাৎক্ষণিক ফলাফল
 - খুব সহজ লেখার নিয়ম
2. সোজা বোঝা
 - `=>` চিহ্ন মানে "যাও" বা "কর"
 - খুব সহজ সিনট্যাক্স

কখন ব্যবহার করবেন? 💡

- ✅ যখন কাজটা খুব সহজ
- ✅ দ্রুত ফলাফল চাই
- ✅ কম কোড লিখতে চাই

সাবধানে থাকুন ⚠️

- সব জায়গায় ব্যবহার করা যায় না
- জটিল কাজের জন্য সাধারণ ফাংশন ভাল

মজার তথ্য 🍕

Arrow Function হল JavaScript-এর "স্মার্ট শর্টকাট"। আপনি কম কোডে বেশি কাজ করতে পারবেন!

Functions with Parameters: সহজ ভাষায় ব্যাখ্যা



ফাংশন কী? 😞

ফাংশন হল কাজ করার জন্য তৈরি একটি ছোট মেশিন। এটি প্যারামিটার নামক "ইনপুট" নেয়, কাজ করে এবং কখনো "আউটপুট" দেয়।

এখন আসুন এই উদাহরণগুলো সহজভাবে বুঝি। 😊

১. দুইটি সংখ্যার যোগফল বের করা

কোড:

```
function add(a, b) {  
  let c = a + b; // a এবং b যোগ করে c তে জমা করা হচ্ছে।  
  console.log(c); // c-এর মান দেখানো হচ্ছে।  
}  
  
add(10, 20); // ফাংশনটি ১০ এবং ২০ যোগ করবে।
```

কীভাবে কাজ করে?

- `add(10, 20)` ডাক: এখানে `a = 10` এবং `b = 20`।
- গণনা: `c = 10 + 20 = 30`।
- ফলাফল দেখানো: `console.log` ব্যবহার করে ৩০ দেখাবে।

ফলাফল:

30

২. একটি সংখ্যার বর্গ (Square) বের করা

```
function square(x) {  
  const result = x * x; // x এর বর্গফল বের করে result-এ রাখছি।  
  return result; // ফলাফল ফেরত দিচ্ছি।  
}  
  
console.log(square(10)); // ১০ এর বর্গফল দেখাবে।
```

কীভাবে কাজ করে?

- `square(10)` ডাক: এখানে `x = 10`।
- গণনা: `result = 10 * 10 = 100`।
- ফলাফল ফেরত দেওয়া: `return` দিয়ে 100 পাঠানো হয়েছে।

ফলাফল:

```
100
```

বর্গ বের করার আরও সহজ পদ্ধতি (Arrow Function):

```
const square = (x) => x * x; // একই কাজ কম কোডে।  
console.log(square(10)); // ১০ এর বর্গফল দেখাবে।
```

৩. একটি বার্তা এবং নাম দেখানো

```
function sayHello(greetings, name) {  
  console.log(greetings + ' ' + name); // সাধারণ যোগ।  
  console.log(`${greetings} ${name}`); // স্ট্রিং লিটারাল দিয়ে।  
}  
sayHello("Hello", "John"); // বার্তা এবং নাম দেখাবে।
```

কীভাবে কাজ করে?

- `sayHello("Hello", "John")` ডাক: এখানে `greetings = "Hello"` এবং `name = "John"`।
- প্রথম লাইন: `Hello` এবং `John` একত্রে দেখাবে: `"Hello John"`।
- দ্বিতীয় লাইন (স্ট্রিং লিটারাল): একই ফলাফল আরও সহজ পদ্ধতিতে দেখাবে।

ফলাফল:


```
Hello John  
Hello John
```

উপসংহার:

ফাংশন আপনার কোডকে পুনরায় ব্যবহারযোগ্য করে তোলে। আপনি:

1. গণনা করতে পারবেন।
2. বার্তা তৈরি করতে পারবেন।
3. এবং আরও অনেক কাজ সহজে করতে পারবেন! 🚀

ডিফল্ট প্যারামিটার (Default Parameters)

কল্পনা করুন... 

মনে করুন, আপনি একটি যাদুকরী বাক্সে কিছু জিনিস রাখতে চান। এই বাক্সে যদি আপনি কিছু না রাখেন, তবে এটি স্বয়ংক্রিয়ভাবে একটি নির্দিষ্ট জিনিস রাখবে। এই বাক্সটি হচ্ছে "ডিফল্ট প্যারামিটার"! 📦 ✨

কোডের উদাহরণ 🔍

১. স্বাগতম বলা (sayHello)

```
function sayHello(greetings, name = "John") {  
    console.log(`${greetings} ${name}`); // স্ট্রিং লিটারেল 14  
}  
  
sayHello("Hello"); // দ্বিতীয় প্যারামিটারের জন্য কিছু না দেওয়া হলে, এটি স্বয়ংক্রিয়ভাবে  
"John" নেবে
```

ব্যাখ্যা:

- এখানে sayHello একটি ফাংশন যা দুটি প্যারামিটার নেয়: greetings এবং name।
- name প্যারামিটারের জন্য একটি ডিফল্ট মান দেওয়া হয়েছে, যা হলো "John"।
- যখন আপনি sayHello("Hello") কল করেন, তখন name প্যারামিটারটি কিছু না দেওয়া হলে, এটি স্বয়ংক্রিয়ভাবে "John" নেবে।

২. নাম পরিবর্তন করা

```
function sayHello(greetings, name = "John") {  
    console.log(`${greetings} ${name}`); // স্ট্রিং লিটারেল  
}  
  
sayHello("Hello", "Jane"); // ডিফল্ট প্যারামিটার সেট করা থাকলেও, নতুন নাম "Jane"  
পাস করলে সেটি গ্রহণ করবে
```

ব্যাখ্যা:

- এখানে আবার sayHello ফাংশনটি ব্যবহার করা হয়েছে, কিন্তু এবার আমরা name প্যারামিটারে "Jane" পাস করেছি।
- যেহেতু আমরা একটি নতুন নাম পাস করেছি, তাই ডিফল্ট "John" এর পরিবর্তে "Jane" ব্যবহার করা হবে।

কেন এটি মজার? 😊

- সহজ ব্যবহার: আপনি যদি দ্বিতীয় প্যারামিটার না দেন, তবে এটি স্বয়ংক্রিয়ভাবে একটি মান নেয়।
- নাম পরিবর্তন: আপনি চাইলে ডিফল্ট নাম পরিবর্তন করতে পারেন, এবং এটি খুব সহজ!

কখন ব্যবহার করবেন? 💡

- ✓ যখন আপনি চান যে কিছু প্যারামিটার স্বয়ংক্রিয়ভাবে একটি মান নিক।
- ✓ যখন আপনি ফাংশনকে আরও নমনীয় করতে চান।

immediate invoked function (IIF)/Anonymous function:

```
(()=>{  
    let a = 10;  
    let b = 20;  
    let c = a + b;  
  
    console.log(c);  
})();
```

ব্যাখ্যা:

- এটি একটি বিশেষ ধরনের ফাংশন যা স্বয়ংক্রিয়ভাবে কাজ করে।
- এখানে `a` এবং `b` নামের দুটি সংখ্যা (১০ এবং ২০) তৈরি করা হয়েছে।
- `c` নামের একটি নতুন সংখ্যা তৈরি হয়েছে যা `a` এবং `b` এর যোগফল।
- `console.log(c)` ব্যবহার করে আমরা `c` এর মান (৩০) দেখতে পাচ্ছি।

ভেরিয়েবল ঘোষণা সহ (অ্যারো সহ):

```
<script>  
  
let add = (a, b) => {  
    let c = a + b;  
  
    console.log(c);  
};  
  
add(10, 20);  
  
</script>
```

ব্যাখ্যা:

- এখানে একটি ফাংশন `add` তৈরি করা হয়েছে যা দুটি সংখ্যা (যেমন `a` এবং `b`) গ্রহণ করে।
- `c` নামের একটি নতুন সংখ্যা তৈরি হয় যা `a` এবং `b` এর যোগফল।
- `console.log(c)` ব্যবহার করে আমরা `c` এর মান দেখতে পাই।
- যখন আমরা `add(10, 20)` লিখি, তখন এটি ১০ এবং ২০ এর যোগফল (৩০) দেখায়।

অন্য পদ্ধতি (অ্যারো ছাড়া):

```
const square = function (a) {  
    return a * a;  
};  
  
console.log(square(10));
```

ব্যাখ্যা:

- এখানে একটি সাধারণ ফাংশন `square` তৈরি করা হয়েছে যা একটি সংখ্যা (যেমন `a`) গ্রহণ করে।
- এটি `a` এর বর্গফল (যেমন 10×10) প্রদান করে।
- `console.log(square(10))` ব্যবহার করে আমরা `10` এর বর্গফল (`100`) দেখতে পাই।

Unlimited Parameter Passing: সহজ ভাষায় ব্যাখ্যা ✨

কীভাবে কাজ করে? 😞

কখনো কখনো আপনি জানেন না কতগুলো ইনপুট প্যারামিটার লাগবে। `...args` (Rest Parameter) ব্যবহার করে আপনি অসীম সংখ্যক প্যারামিটার নিতে পারবেন এবং এগুলোকে একটি অ্যারেতে পেতে পারবেন।

১. প্যারামিটারগুলোকে একটি অ্যারেতে দেখানো

```
<script>  
  
    numbers = (...args) => {  
        console.log(args); // args হলো সব প্যারামিটারের একটি অ্যারে।  
    };  
  
    numbers(10, 20, 30, 40, 50, 60, 70, 80);  
    // Output: [10, 20, 30, 40, 50, 60, 70, 80] (একটি অ্যারে)  
  
</script>
```

কীভাবে কাজ করে?

- `...args`: এটি সকল প্যারামিটারকে একটি অ্যারেতে রূপান্তরিত করে।
- `console.log(args)`: প্যারামিটারগুলোর অ্যারে দেখাবে।

ফলাফল:

```
[10, 20, 30, 40, 50, 60, 70, 80]
```


২. অসীম সংখ্যক প্যারামিটারের যোগফল বের করা

```
<script>

add = (...args) => {
  let sum = 0; // যোগফলের জন্য প্রাথমিক মান
  for (let i = 0; i < args.length; i++) { // প্রতিটি প্যারামিটার ধরে যোগ
    sum += args[i];
  }

  console.log(sum); // যোগফল দেখানো।
};

add(10, 20, 30, 40, 50); // Output: 150

</script>
```

কীভাবে কাজ করে?

- `...args` : প্যারামিটারগুলোকে একটি অ্যারেতে রূপান্তরিত করেছে।
- **লুপ**: প্রতিটি মান ধরে `sum`-এ যোগ করেছে।
- **ফলাফল দেখানো**: `console.log(sum)` ব্যবহার করে যোগফল দেখানো হয়েছে।

ফলাফল:

150

উপসংহার:

`...args` ব্যবহার করলে আপনি যত ইচ্ছা প্যারামিটার পাঠাতে পারবেন। 🚀

JavaScript অবজেক্ট (Object) কি?

সাধারণ ভাষায় বুঝি 😞

অবজেক্ট হচ্ছে একটি "বাক্স" যার মধ্যে আমরা বিভিন্ন তথ্য সংরক্ষণ করতে পারি। এটা একটি বিশেষ জিনিস যেখানে আমরা একই সাথে একাধিক তথ্য রাখতে পারি।

কোডের বিস্তারিত ব্যাখ্যা 🤖

অবজেক্ট ডিক্লেয়ারেশন

```
const person = {
  firstName: "John",
  lastName: "Doe",
```

```
    age: 50,  
    eyeColor: "blue"  
};
```

এখানে `person` হচ্ছে আমাদের অবজেক্টের নাম। এর মধ্যে আমরা রেখেছি:

- `firstName`: ব্যক্তির প্রথম নাম
- `lastName`: ব্যক্তির শেষ নাম
- `age`: ব্যক্তির বয়স
- `eyeColor`: চোখের রঙ

অবজেক্ট থেকে তথ্য বের করা

```
person.firstName // "John"  
person.lastName  // "Doe"  
person.age       // 50  
person.eyeColor  // "blue"
```

টেমপ্লেট লিটারাল ব্যবহার ☑

```
document.write(`This person's name is ${person.firstName} ${person.lastName}, his  
age is ${person.age} and his eye color is ${person.eyeColor}.`);
```

সহজ উদাহরণ

মনে করুন, আপনি এক বাক্সে (অবজেক্ট) আপনার তথ্য রাখলেন:

```
const student = {  
  name: "রাহুল",  
  age: 12,  
  className: "৬ষ্ঠ শ্রেণী"  
};
```

গুরুত্বপূর্ণ নিয়ম ✅

1. অবজেক্ট ডিক্লেয়ার করার সময় `const` ব্যবহার করুন
2. `key` এবং `value` এর মাঝে `:` ব্যবহার করুন
3. `key` গুলো হয় `camelCase` বা `snake_case` হয়

সাবধানতা ⚠

- `key` এর নাম ইংরেজিতে হবে
- `value` যে কোনো ধরনের হতে পারে (string, number, boolean)

মজার তথ্য 🎉

JavaScript-এ অবজেক্ট হচ্ছে "key-value pair" এর একটি সংগ্রহ, যেখানে প্রতিটি `key` হল নাম এবং `value` হল সেই নামের মান!

Array Declaration & Usage: সহজ ভাষায় ব্যাখ্যা ✨

Array কী?

Array একটি বিশেষ "ডাটা স্টোরেজ" যেখানে আপনি একাধিক মান (value) রাখতে পারেন। এটি একটি বাক্সের মতো যেখানে প্রতিটি মান একটি নির্দিষ্ট "স্থান" (index) অনুযায়ী রাখা হয়।

Array ডিক্লারেশন:

```
const variableName = [  
  value1, // প্রথম মান  
  value2, // দ্বিতীয় মান  
  value3  // তৃতীয় মান  
];
```

উদাহরণ: ফলের তালিকা

```
const fruits = ["apple", "banana", "orange", 10];
```

- `"apple"`: প্রথম মান (index 0)
- `"banana"`: দ্বিতীয় মান (index 1)
- `"orange"`: তৃতীয় মান (index 2)
- `10`: চতুর্থ মান (index 3)

ডেটা অ্যাক্সেস করা:

```
document.write(`Please bring ${fruits[0]}, ${fruits[1]} and ${fruits[2]}.  
${fruits[3]}kgs each.`);
```

কীভাবে কাজ করে?

- `fruits[0]`: apple নেয়।
- `fruits[1]`: banana নেয়।
- `fruits[2]`: orange নেয়।
- `fruits[3]`: সংখ্যা 10 নেয়।

আউটপুট:

Please bring apple, banana and orange. 10kgs each.

উপসংহার:

Array ব্যবহার করে অনেকগুলো মান সহজেই সংরক্ষণ এবং অ্যাক্সেস করা যায়।