জাভাস্ক্রিপ্টে **স্ট্রিং** হলো একটি প্রিমিটিভ ডেটা টাইপ যা টেক্সট ডেটা সংরক্ষণ করে। জাভাস্ক্রিপ্ট স্ট্রিং ম্যানিপুলেশনের জন্য অনেকগুলি বিল্ট-ইন মেথড প্রদান করে। নিচে জাভাস্ক্রিপ্টের সমস্ত গুরুত্বপূর্ণ স্ট্রিং মেথডগুলির বাংলায় ব্যাখ্যা দেওয়া হলো:

**১. স্ট্রিং লেংথ (Length):**

length প্রপার্টি ব্যবহার করে স্ট্রিং এর দৈর্ঘ্য (ক্যারেক্টারের সংখ্যা) জানা যায়।

**উদাহরণ:**

javascript

Copy

let str = "Hello";

console.log(str.length); // আউটপুট: 5

**২. স্ট্রিং ইনডেক্সিং:**

স্ট্রিং এর প্রতিটি ক্যারেক্টারকে ইনডেক্স ব্যবহার করে অ্যাক্সেস করা যায়। ইনডেক্স 0 থেকে শুরু হয়।

**উদাহরণ:**

javascript

Copy

let str = "Hello";

console.log(str[0]); // আউটপুট: H

console.log(str[1]); // আউটপুট: e

**৩. charAt():**

charAt() মেথড ব্যবহার করে নির্দিষ্ট ইনডেক্সে অবস্থিত ক্যারেক্টার রিটার্ন করা যায়।

**উদাহরণ:**

javascript

Copy

let str = "Hello";

console.log(str.charAt(1)); // আউটপুট: e

**৪. concat():**

concat() মেথড ব্যবহার করে একাধিক স্ট্রিং যুক্ত করা যায়।

**উদাহরণ:**

javascript

Copy

let str1 = "Hello";

let str2 = "World";

console.log(str1.concat(" ", str2)); // আউটপুট: Hello World

**৫. includes():**

includes() মেথড ব্যবহার করে চেক করা যায় যে স্ট্রিং এর মধ্যে নির্দিষ্ট সাবস্ট্রিং আছে কিনা।

**উদাহরণ:**

javascript

Copy

let str = "Hello World";

console.log(str.includes("World")); // আউটপুট: true

**৬. indexOf():**

indexOf() মেথড ব্যবহার করে নির্দিষ্ট সাবস্ট্রিং এর প্রথম ইনডেক্স রিটার্ন করা যায়। যদি সাবস্ট্রিং না পাওয়া যায়, তাহলে -1 রিটার্ন করে।

**উদাহরণ:**

javascript

Copy

let str = "Hello World";

console.log(str.indexOf("World")); // আউটপুট: 6

**৭. lastIndexOf():**

lastIndexOf() মেথড ব্যবহার করে নির্দিষ্ট সাবস্ট্রিং এর শেষ ইনডেক্স রিটার্ন করা যায়।

**উদাহরণ:**

javascript

Copy

let str = "Hello World World";

console.log(str.lastIndexOf("World")); // আউটপুট: 12

**৮. slice():**

slice() মেথড ব্যবহার করে স্ট্রিং এর একটি অংশ কেটে নেওয়া যায়।

**উদাহরণ:**

javascript

Copy

let str = "Hello World";

console.log(str.slice(0, 5)); // আউটপুট: Hello

**৯. substring():**

substring() মেথড ব্যবহার করে স্ট্রিং এর একটি অংশ কেটে নেওয়া যায়। এটি slice() এর মতো, কিন্তু নেগেটিভ ইনডেক্স সাপোর্ট করে না।

**উদাহরণ:**

javascript

Copy

let str = "Hello World";

console.log(str.substring(6, 11)); // আউটপুট: World

**১০. substr():**

substr() মেথড ব্যবহার করে স্ট্রিং এর একটি অংশ কেটে নেওয়া যায়। এটি শুরু ইনডেক্স এবং দৈর্ঘ্য নেয়।

**উদাহরণ:**

javascript

Copy

let str = "Hello World";

console.log(str.substr(6, 5)); // আউটপুট: World

**১১. replace():**

replace() মেথড ব্যবহার করে স্ট্রিং এর একটি অংশকে অন্য স্ট্রিং দিয়ে প্রতিস্থাপন করা যায়।

**উদাহরণ:**

javascript

Copy

let str = "Hello World";

console.log(str.replace("World", "JavaScript")); // আউটপুট: Hello JavaScript

**১২. toUpperCase():**

toUpperCase() মেথড ব্যবহার করে স্ট্রিং এর সমস্ত ক্যারেক্টারকে আপার কেসে রূপান্তর করা যায়।

**উদাহরণ:**

javascript

Copy

let str = "Hello";

console.log(str.toUpperCase()); // আউটপুট: HELLO

**১৩. toLowerCase():**

toLowerCase() মেথড ব্যবহার করে স্ট্রিং এর সমস্ত ক্যারেক্টারকে লোয়ার কেসে রূপান্তর করা যায়।

**উদাহরণ:**

javascript

Copy

let str = "Hello";

console.log(str.toLowerCase()); // আউটপুট: hello

**১৪. trim():**

trim() মেথড ব্যবহার করে স্ট্রিং এর শুরু এবং শেষের স্পেস (হোয়াইটস্পেস) রিমুভ করা যায়।

**উদাহরণ:**

javascript

Copy

let str = " Hello World ";

console.log(str.trim()); // আউটপুট: Hello World

**১৫. split():**

split() মেথড ব্যবহার করে স্ট্রিং কে একটি নির্দিষ্ট সেপারেটর অনুযায়ী ভাগ করে অ্যারে রিটার্ন করা যায়।

**উদাহরণ:**

javascript

Copy

let str = "Hello,World,JavaScript";

console.log(str.split(",")); // আউটপুট: ["Hello", "World", "JavaScript"]

**১৬. startsWith():**

startsWith() মেথড ব্যবহার করে চেক করা যায় যে স্ট্রিং নির্দিষ্ট সাবস্ট্রিং দিয়ে শুরু হয় কিনা।

**উদাহরণ:**

javascript

Copy

let str = "Hello World";

console.log(str.startsWith("Hello")); // আউটপুট: true

**১৭. endsWith():**

endsWith() মেথড ব্যবহার করে চেক করা যায় যে স্ট্রিং নির্দিষ্ট সাবস্ট্রিং দিয়ে শেষ হয় কিনা।

**উদাহরণ:**

javascript

Copy

let str = "Hello World";

console.log(str.endsWith("World")); // আউটপুট: true

**১৮. repeat():**

repeat() মেথড ব্যবহার করে স্ট্রিং কে নির্দিষ্ট সংখ্যক বার পুনরাবৃত্তি করা যায়।

**উদাহরণ:**

javascript

Copy

let str = "Hello";

console.log(str.repeat(3)); // আউটপুট: HelloHelloHello

**১৯. padStart():**

padStart() মেথড ব্যবহার করে স্ট্রিং এর শুরুতে নির্দিষ্ট ক্যারেক্টার যোগ করে স্ট্রিং কে একটি নির্দিষ্ট দৈর্ঘ্যে প্যাড করা যায়।

**উদাহরণ:**

javascript

Copy

let str = "5";

console.log(str.padStart(3, "0")); // আউটপুট: 005

**২০. padEnd():**

padEnd() মেথড ব্যবহার করে স্ট্রিং এর শেষে নির্দিষ্ট ক্যারেক্টার যোগ করে স্ট্রিং কে একটি নির্দিষ্ট দৈর্ঘ্যে প্যাড করা যায়।

**উদাহরণ:**

javascript

Copy

let str = "5";

console.log(str.padEnd(3, "0")); // আউটপুট: 500

**সারসংক্ষেপ:**

| **মেথড** | **বর্ণনা** | **উদাহরণ** |
| --- | --- | --- |
| length | স্ট্রিং এর দৈর্ঘ্য রিটার্ন করে | str.length |
| charAt() | নির্দিষ্ট ইনডেক্সে ক্যারেক্টার রিটার্ন করে | str.charAt(1) |
| concat() | একাধিক স্ট্রিং যুক্ত করে | str1.concat(str2) |
| includes() | সাবস্ট্রিং আছে কিনা চেক করে | str.includes("World") |
| indexOf() | সাবস্ট্রিং এর প্রথম ইনডেক্স রিটার্ন করে | str.indexOf("World") |
| lastIndexOf() | সাবস্ট্রিং এর শেষ ইনডেক্স রিটার্ন করে | str.lastIndexOf("World") |
| slice() | স্ট্রিং এর একটি অংশ কেটে নেয় | str.slice(0, 5) |
| substring() | স্ট্রিং এর একটি অংশ কেটে নেয় | str.substring(6, 11) |
| substr() | স্ট্রিং এর একটি অংশ কেটে নেয় | str.substr(6, 5) |
| replace() | স্ট্রিং এর অংশ প্রতিস্থাপন করে | str.replace("World", "JS") |
| toUpperCase() | স্ট্রিং কে আপার কেসে রূপান্তর করে | str.toUpperCase() |
| toLowerCase() | স্ট্রিং কে লোয়ার কেসে রূপান্তর করে | str.toLowerCase() |
| trim() | স্ট্রিং এর শুরু এবং শেষের স্পেস রিমুভ করে | str.trim() |
| split() | স্ট্রিং কে অ্যারেতে ভাগ করে | str.split(",") |
| startsWith() | স্ট্রিং নির্দিষ্ট সাবস্ট্রিং দিয়ে শুরু হয় কিনা চেক করে | str.startsWith("Hello") |
| endsWith() | স্ট্রিং নির্দিষ্ট সাবস্ট্রিং দিয়ে শেষ হয় কিনা চেক করে | str.endsWith("World") |
| repeat() | স্ট্রিং কে পুনরাবৃত্তি করে | str.repeat(3) |
| padStart() | স্ট্রিং এর শুরুতে প্যাড যোগ করে | str.padStart(3, "0") |
| padEnd() | স্ট্রিং এর শেষে প্যাড যোগ করে | str.padEnd(3, "0") |

এই মেথডগুলি ব্যবহার করে আপনি জাভাস্ক্রিপ্টে স্ট্রিং ডেটা ম্যানিপুলেট করতে পারবেন। 😊

**matchAll()**

জাভাস্ক্রিপ্টে **matchAll()** মেথডটি ব্যবহার করা হয় যখন আপনি একটি স্ট্রিং এর মধ্যে একটি **রেগুলার এক্সপ্রেশন (Regular Expression)** এর সাথে মিলে যাওয়া সমস্ত ম্যাচ (matches) একসাথে পেতে চান। এটি বিশেষভাবে উপকারী যখন আপনি একটি স্ট্রিং এর মধ্যে একাধিক প্যাটার্ন ম্যাচ করতে চান এবং প্রতিটি ম্যাচের ডিটেইলস (যেমন গ্রুপ ক্যাপচার) অ্যাক্সেস করতে চান।

**matchAll() মেথডের ব্যবহার:**

matchAll() মেথডটি একটি রেগুলার এক্সপ্রেশন এর সাথে মিলে যাওয়া সমস্ত ম্যাচের জন্য একটি **ইটারেটর (Iterator)** রিটার্ন করে। প্রতিটি ম্যাচ একটি অ্যারে হিসেবে রিটার্ন হয়, যেখানে ম্যাচের ডিটেইলস (গ্রুপ ক্যাপচার সহ) থাকে।

**সিনট্যাক্স:**

javascript

Copy

let matches = string.matchAll(regexp);

* string: যে স্ট্রিং এর মধ্যে ম্যাচ খুঁজতে হবে।
* regexp: একটি রেগুলার এক্সপ্রেশন অবজেক্ট। এটি অবশ্যই **গ্লোবাল ফ্ল্যাগ (g)** সহ থাকতে হবে, নাহলে matchAll() একটি এরর থ্রো করবে।

**matchAll() এর সুবিধা:**

1. **সমস্ত ম্যাচ একসাথে পাওয়া**:
   * matchAll() সমস্ত ম্যাচ একসাথে রিটার্ন করে, যা match() মেথডের চেয়ে বেশি সুবিধাজনক যখন একাধিক ম্যাচ থাকে।
2. **গ্রুপ ক্যাপচার সহ ডিটেইলস**:
   * প্রতিটি ম্যাচের জন্য গ্রুপ ক্যাপচার সহ সম্পূর্ণ ডিটেইলস পাওয়া যায়।
3. **ইটারেটর**:
   * এটি একটি ইটারেটর রিটার্ন করে, যা for...of লুপ বা Array.from() এর মাধ্যমে সহজে ব্যবহার করা যায়।

**উদাহরণ:**

**১. সাধারণ ম্যাচ:**

javascript

Copy

let str = "Hello World, Hello JavaScript";

let regexp = /Hello/g;

let matches = str.matchAll(regexp);

for (let match of matches) {

console.log(match);

}

**আউটপুট**:

Copy

["Hello", index: 0, input: "Hello World, Hello JavaScript", groups: undefined]

["Hello", index: 13, input: "Hello World, Hello JavaScript", groups: undefined]

**২. গ্রুপ ক্যাপচার সহ ম্যাচ:**

javascript

Copy

let str = "John: 25, Alice: 30";

let regexp = /(\w+): (\d+)/g;

let matches = str.matchAll(regexp);

for (let match of matches) {

console.log(`Name: ${match[1]}, Age: ${match[2]}`);

}

**আউটপুট**:

Copy

Name: John, Age: 25

Name: Alice, Age: 30

**matchAll() vs match():**

| **বৈশিষ্ট্য** | **matchAll()** | **match()** |
| --- | --- | --- |
| **রিটার্ন টাইপ** | ইটারেটর (Iterator) | অ্যারে (Array) |
| **গ্লোবাল ফ্ল্যাগ** | অবশ্যই g ফ্ল্যাগ প্রয়োজন | g ফ্ল্যাগ ছাড়াও কাজ করে |
| **গ্রুপ ক্যাপচার** | প্রতিটি ম্যাচের গ্রুপ ক্যাপচার সহ ডিটেইলস | শুধুমাত্র প্রথম ম্যাচের গ্রুপ ক্যাপচার |
| **ইটারেশন** | for...of লুপ বা Array.from() ব্যবহার | সরাসরি অ্যারে হিসেবে ব্যবহার |

**কখন matchAll() ব্যবহার করবেন?:**

1. যখন আপনি একটি স্ট্রিং এর মধ্যে একাধিক ম্যাচ খুঁজতে চান।
2. যখন আপনি প্রতিটি ম্যাচের গ্রুপ ক্যাপচার সহ সম্পূর্ণ ডিটেইলস চান।
3. যখন আপনি রেগুলার এক্সপ্রেশন এর সাথে মিলে যাওয়া সমস্ত ম্যাচ একসাথে প্রসেস করতে চান।

**সারসংক্ষেপ:**

* matchAll() মেথডটি একটি স্ট্রিং এর মধ্যে রেগুলার এক্সপ্রেশন এর সাথে মিলে যাওয়া সমস্ত ম্যাচের জন্য একটি ইটারেটর রিটার্ন করে।
* এটি গ্রুপ ক্যাপচার সহ প্রতিটি ম্যাচের সম্পূর্ণ ডিটেইলস প্রদান করে।
* এটি বিশেষভাবে উপকারী যখন আপনি একাধিক ম্যাচ এবং তাদের গ্রুপ ক্যাপচার একসাথে প্রসেস করতে চান।

এই মেথডটি ব্যবহার করে আপনি জাভাস্ক্রিপ্টে স্ট্রিং ডেটা ম্যানিপুলেশন আরও শক্তিশালী এবং নিয়ন্ত্রিত করতে পারবেন। 😊

String Template

জাভাস্ক্রিপ্টে **স্ট্রিং টেমপ্লেট** (String Template) বা **টেমপ্লেট লিটারাল** (Template Literal) হলো একটি শক্তিশালী ফিচার যা ES6 (ECMAScript 2015) এ প্রবর্তিত হয়। এটি স্ট্রিং তৈরি এবং ম্যানিপুলেট করার একটি সহজ এবং আরও পাঠযোগ্য উপায় প্রদান করে। স্ট্রিং টেমপ্লেট ব্যবহার করে আপনি সহজেই ভেরিয়েবল, এক্সপ্রেশন, এবং মাল্টি-লাইন স্ট্রিং যোগ করতে পারেন।

**স্ট্রিং টেমপ্লেটের বৈশিষ্ট্য:**

1. **ব্যাকটিক (``)** ব্যবহার:
   * স্ট্রিং টেমপ্লেট তৈরি করতে ব্যাকটিক (``) ব্যবহার করা হয়।
   * উদাহরণ:

javascript

Copy

let name = "John";

let message = `Hello, ${name}!`;

console.log(message); // আউটপুট: Hello, John!

1. **ভেরিয়েবল ইন্টারপোলেশন**:
   * ${} সিনট্যাক্স ব্যবহার করে স্ট্রিং এর মধ্যে ভেরিয়েবল বা এক্সপ্রেশন যোগ করা যায়।
   * উদাহরণ:

javascript

Copy

let a = 5;

let b = 10;

console.log(`The sum is ${a + b}.`); // আউটপুট: The sum is 15.

1. **মাল্টি-লাইন স্ট্রিং**:
   * স্ট্রিং টেমপ্লেট ব্যবহার করে সহজেই মাল্টি-লাইন স্ট্রিং তৈরি করা যায়।
   * উদাহরণ:

javascript

Copy

let multiLine = `

This is line 1.

This is line 2.

This is line 3.

`;

console.log(multiLine);

1. **এক্সপ্রেশন ইন্টারপোলেশন**:
   * ${} এর মধ্যে যেকোনো জাভাস্ক্রিপ্ট এক্সপ্রেশন ব্যবহার করা যায়।
   * উদাহরণ:

javascript

Copy

let price = 10;

let quantity = 5;

console.log(`Total cost: ${price \* quantity}`); // আউটপুট: Total cost: 50

1. **ট্যাগড টেমপ্লেট**:
   * ট্যাগড টেমপ্লেট ব্যবহার করে আপনি স্ট্রিং টেমপ্লেট কাস্টমাইজ করতে পারেন। এটি একটি ফাংশনকে কল করে এবং স্ট্রিং এবং ভেরিয়েবল গুলি আর্গুমেন্ট হিসেবে পাস করে।
   * উদাহরণ:

javascript

Copy

function tag(strings, ...values) {

console.log(strings); // ["Hello, ", "!", ""]

console.log(values); // ["John"]

return "Processed String";

}

let name = "John";

let result = tag`Hello, ${name}!`;

console.log(result); // আউটপুট: Processed String

**স্ট্রিং টেমপ্লেটের সুবিধা:**

1. **পাঠযোগ্যতা**:
   * স্ট্রিং টেমপ্লেট কোডকে আরও পাঠযোগ্য এবং পরিচ্ছন্ন করে তোলে।
2. **ডাইনামিক স্ট্রিং**:
   * ভেরিয়েবল এবং এক্সপ্রেশন সহজেই স্ট্রিং এর মধ্যে যোগ করা যায়।
3. **মাল্টি-লাইন স্ট্রিং**:
   * মাল্টি-লাইন স্ট্রিং তৈরি করা সহজ হয়।
4. **ট্যাগড টেমপ্লেট**:
   * ট্যাগড টেমপ্লেট ব্যবহার করে স্ট্রিং প্রসেসিং কাস্টমাইজ করা যায়।

**স্ট্রিং টেমপ্লেট vs সাধারণ স্ট্রিং:**

| **বৈশিষ্ট্য** | **স্ট্রিং টেমপ্লেট** | **সাধারণ স্ট্রিং** |
| --- | --- | --- |
| **সিনট্যাক্স** | ব্যাকটিক (``) ব্যবহার | সিঙ্গেল কোট (') বা ডাবল কোট (") ব্যবহার |
| **ভেরিয়েবল যোগ** | ${} ব্যবহার করে সহজেই যোগ করা যায় | কনক্যাটেনেশন (+) ব্যবহার করা হয় |
| **মাল্টি-লাইন স্ট্রিং** | সহজেই মাল্টি-লাইন স্ট্রিং তৈরি করা যায় | \n ব্যবহার করে মাল্টি-লাইন তৈরি করা হয় |
| **ট্যাগড টেমপ্লেট** | সমর্থিত | সমর্থিত নয় |

**উদাহরণ:**

**১. ভেরিয়েবল ইন্টারপোলেশন:**

javascript

Copy

let name = "John";

let age = 30;

console.log(`My name is ${name} and I am ${age} years old.`);

// আউটপুট: My name is John and I am 30 years old.

**২. মাল্টি-লাইন স্ট্রিং:**

javascript

Copy

let multiLine = `

This is line 1.

This is line 2.

This is line 3.

`;

console.log(multiLine);

// আউটপুট:

// This is line 1.

// This is line 2.

// This is line 3.

**৩. এক্সপ্রেশন ইন্টারপোলেশন:**

javascript

Copy

let a = 5;

let b = 10;

console.log(`The sum of ${a} and ${b} is ${a + b}.`);

// আউটপুট: The sum of 5 and 10 is 15.

**৪. ট্যাগড টেমপ্লেট:**

javascript

Copy

function highlight(strings, ...values) {

let result = "";

strings.forEach((str, i) => {

result += str;

if (values[i]) {

result += `<strong>${values[i]}</strong>`;

}

});

return result;

}

let name = "John";

let age = 30;

let message = highlight`Hello, my name is ${name} and I am ${age} years old.`;

console.log(message);

// আউটপুট: Hello, my name is <strong>John</strong> and I am <strong>30</strong> years old.

**সারসংক্ষেপ:**

* স্ট্রিং টেমপ্লেট ব্যাকটিক (``) ব্যবহার করে তৈরি করা হয়।
* ${} সিনট্যাক্স ব্যবহার করে ভেরিয়েবল এবং এক্সপ্রেশন সহজেই স্ট্রিং এর মধ্যে যোগ করা যায়।
* মাল্টি-লাইন স্ট্রিং তৈরি করা সহজ।
* ট্যাগড টেমপ্লেট ব্যবহার করে স্ট্রিং প্রসেসিং কাস্টমাইজ করা যায়।

স্ট্রিং টেমপ্লেট ব্যবহার করে আপনি জাভাস্ক্রিপ্টে স্ট্রিং ম্যানিপুলেশন আরও সহজ এবং শক্তিশালী করতে পারবেন। 😊