



JavaScript

Written By

আবদুল্লাহ আল ফারুক

Published By : WordPress Group

<http://ebooks.WpBangla.com>

জাভাস্ক্রিপ্ট বাংলা ই-বুক

JavaScript Bangla

E-book

(For Web Developers)

মোঃ আবদুল্লাহ্ আল-ফারুক

<https://www.facebook.com/faruk.ice09>

<http://www.WebTechnologyBlog.com>

প্রথম প্রকাশ
২১ শে ফেব্রুয়ারী ২০১৩

প্রকাশক
ওয়ার্ডপ্রেস গ্রুপ বাংলাদেশ
(<https://www.facebook.com/groups/Wordpress2Smashing>)

ওয়ার্ডপ্রেস গ্রুপ ওয়েবসাইট
(<http://www.wpbangla.com>)

লেখক
মোঃ আবদুল্লাহ আল-ফারুক
(<https://www.facebook.com/faruk.ice09>)
(<http://www.WebTechnologyBlog.com>)

প্রচ্ছদ
জামিল হোসেন সিজান
(<https://www.facebook.com/zamil.hossainsezan32>)

কপিরাইট
আবদুল্লাহ আল-ফারুক ও ওয়ার্ডপ্রেস গ্রুপ বাংলাদেশ

সর্তকতা
এই বইটি বিক্রয়ের জন্য নয়
বইটি বিনামূল্যে বিতরণযোগ্য

JavaScript Bangla E-book is written by MD.Abdullah Al Faruk & Published by Wordpress Group Bangladesh. Copyright by MD.Abdullah Al Faruk & Wordpress Group Bangladesh. Caution: **This book is not for sell. It's free to distribute.**

উৎসর্গ

সকল ভাষা শহীদদের প্রতি.....

আমার সম্পর্কে-

আমি মোঃ আবদুল্লাহ আল-ফারুক(<https://www.facebook.com/faruk.ice09>)। পড়ালেখা করছি কুষ্টিয়া ইসলামী বিশ্ববিদ্যালয়ের ইনফরমেশন এন্ড কমিউনিকেশন ইঞ্জিনিয়ারিং (ICE) বিভাগের তৃতীয় বর্ষে। আমি যখন জাভাস্ক্রিপ্ট শেখা শুরু করি তখন বাংলাতে জাভাস্ক্রিপ্টের তেমন কোন রিসোর্স ছিল না এবং এখন পর্যন্ত বাংলায় জাভাস্ক্রিপ্টের উপর আমার জানা মতে তেমন কোন ভালো বই নাই। আমার খুব ইচ্ছা ছিল জাভাস্ক্রিপ্টের উপর একটা স্বয়ংসম্পূর্ণ বই পাবলিশ করা কিন্তু তেমন কাউকে পাইনি যে এ ব্যাপারে সাহায্য করতে পারে। যাহোক শেষ পর্যন্ত ওয়ার্ডপ্রেস গ্রুপের সৌজন্যে ছোটখাটো একটা জাভাস্ক্রিপ্ট বই লিখেই ফেললাম। এর আগে আমি এইচটিএমএল ও সিএসএস নিয়ে আরও দুটি বই লিখেছি যেখানে যথেষ্ট সাড়া পেয়েছি। জাভাস্ক্রিপ্ট বাংলা ইবুকটি ও আশা করি আপনাদের ওয়েব ডেভলপমেন্ট শেখার কাজে সামান্য হলেও কাজে দিবে। আরেকটি কথা বইটিতে জাভাস্ক্রিপ্টের প্রায়োগিক দিক নিয়ে তেমন বেশি কিছু আলোচনা করা হয় নি এখানে জাভাস্ক্রিপ্ট শেখানোর প্রতি গুরুত্ব দেওয়া হয়েছে। বইটি দ্রুত লিখে শেষ করার কারণে কিছু ভুলত্রুটি থাকতে পারে আশা করি ক্ষমা সুন্দর দৃষ্টিতে দেখবেন। আর এই বইটি যদি আপনাদের শেখার কাজে সামান্য উপকারে লাগে তবে নিজেকে ধন্য ও আমার পরিশ্রম সার্থক হয়েছে বলে মনে করবো। আমি বিশেষভাবে ধন্যবাদ জানাই জামিল হোসেন সিজান

(<https://www.facebook.com/zamil.hossainsezan32>) ও ওয়ার্ডপ্রেস গ্রুপ বাংলাদেশ (<https://www.facebook.com/groups/Wordpress2Smashing/>) কে।

এই বইটির স্বব্বস্বত্ব আমার। অনুগ্রহ করে অনুমতি ছাড়া এই বইটির আংশিক বা সম্পূর্ণ কপি বা বিকৃত বা নিজের নামে চালিয়ে দেয়ার চেষ্টা করবেন না। আপনাদের নিজ নিজ ব্লগের মাধ্যমে বইটি সবার মাঝে ছড়িয়ে দিন। শেয়ার করুন সবার সাথে।

-আবদুল্লাহ আল-ফারুক

-:সূচিপত্র:-

অধ্যায়ঃ এক- সাধারণ আলোচনা

- জাভাস্ক্রিপ্ট কী?
- ইতিহাস
- প্রোগ্রামিং ল্যাঙ্গুয়েজ ও স্ক্রিপ্ট ল্যাঙ্গুয়েজের মধ্যে পার্থক্য
- জাভাস্ক্রিপ্ট বনাম জাভা
- **JAVA** এবং **JavaScript** কি এক ?
- জাভাস্ক্রিপ্ট কেন প্রয়োজন
- জাভাস্ক্রিপ্ট এর সাহায্যে নিচের কাজগুলো এর যায়।
- জাভাস্ক্রিপ্ট আরও যে কাজগুলো করতে পারে
- জাভাস্ক্রিপ্ট সক্রিয় করা
- জাভাস্ক্রিপ্ট কে ইন্টারনেট এক্সপ্লোরার- এ সচল করারপদ্ধতি
- জাভাস্ক্রিপ্ট কে ফায়ারফক্স- এ সচল করারপদ্ধতি
- জাভাস্ক্রিপ্ট কে অপেরা - তে সচল করারপদ্ধতি
- জাভাস্ক্রিপ্ট কোথায় লিখতে হয়

অধ্যায়ঃ দুই- জাভাস্ক্রিপ্ট শুরু করা

- জাভাস্ক্রিপ্ট সিনট্যাক্স
- প্রথম জাভাস্ক্রিপ্ট কোড লেখা
- জাভাস্ক্রিপ্ট স্টেটমেন্ট
- জাভাস্ক্রিপ্ট ব্লক
- জাভাস্ক্রিপ্ট কোথায় থাকবে?
 - `<head>` ট্যাগের মাঝে জাভাস্ক্রিপ্টের ব্যবহার-
 - `<body>` ট্যাগের মাঝে জাভাস্ক্রিপ্টের ব্যবহার-
 - `<body>` ও `<head>` উভয় ট্যাগের মাঝে জাভাস্ক্রিপ্টের ব্যবহার-
- এক্সটারনাল জাভাস্ক্রিপ্টের ব্যবহার
- জাভাস্ক্রিপ্ট কমেন্টস
- যে কাজটি জাভাস্ক্রিপ্ট দিয়ে করতে পারবেন না।
- সার্ভারের রিসোর্স আপনি জাভাস্ক্রিপ্ট দিয়ে একসেস করতে পারবেন না

অধ্যায়ঃ তিন- জাভাস্ক্রিপ্ট ভেরিয়েবল

- জাভাস্ক্রিপ্ট ভেরিয়েবল কী?
- জাভাস্ক্রিপ্টে ভেরিয়েবল ডিকলার করা
- জাভাস্ক্রিপ্টে ভেরিয়েবল ডিকলারের বিভিন্ন পদ্ধতি
- ভেরিয়েবল এর উদাহরণ

- জাভাস্ক্রিপ্ট ভেরিয়েবল নামের নিয়মনীতি
- ভেরিয়েবলের কার্যএলাকা
- লোকাল ভেরিয়েবল (Local Variable)
- গ্লোবাল ভেরিয়েবল (Global Variable)
- জাভাস্ক্রিপ্ট Loosely-typed Language...!!!!

অধ্যায়ঃ চার- জাভাস্ক্রিপ্ট ডাটা টাইপ

- জাভাস্ক্রিপ্ট ডাটা টাইপ
 - ১.নাম্বার ডাটা টাইপ (Number Data Type)
 - ২. লজিক্যাল ডাটা টাইপ (Logical/Boolean Data Type)
 - ৩.স্ট্রিং ডাটা টাইপ (String Data Type)
 - ৪.নাল ডাটা টাইপ (Null Data Type)
 - ৫.আনডিফাইন্ড ডাটা টাইপ (Undefined Data Type)

অধ্যায়ঃ পাঁচ- জাভাস্ক্রিপ্ট কনস্ট্যান্ট ও রিজার্ভড ওয়ার্ড

- জাভাস্ক্রিপ্ট কনস্ট্যান্ট (JavaScript : Constants)
- জাভাস্ক্রিপ্ট রিজার্ভড ওয়ার্ড
- Table of JavaScript Reserved Words
- Java Keywords (Reserved by JavaScript)
- ECMAScript Reserved Words

- Other JavaScript Keywords

অধ্যায়ঃ ছয়- জাভাস্ক্রিপ্ট অপারেটর

- জাভাস্ক্রিপ্ট অপারেটর কি?
- অপারেটরের প্রকারভেদ
 - 1.string Operators
 - 2.comparison Operators
 - 3.arithmetic Operators
 - 4. assignment Operators
 - 5. logical(or Relational) Operators
 - 6. Conditional (or ternary) Operators
- জাভাস্ক্রিপ্ট অপারেটরের ভেরিয়েবল সহ উদাহরণ
- স্ট্রিং ও নাম্বার যোগ করা

অধ্যায়ঃ সাত- জাভাস্ক্রিপ্ট ব্যবহার নির্দেশিকা

- ১.জাভাস্ক্রিপ্ট কেস সেন্সেটিভ
- ২.হোয়াইট স্পেস
- ৩.কোডলাইনকে ব্রেক করা
- ৫.সেমিকোলন
- ৪.জাভাস্ক্রিপ্ট স্পেসাল ক্যারেক্টার

অধ্যায়ঃ আট- জাভাস্ক্রিপ্ট কন্ডিশনাল (শর্তবাচক) স্টেটমেন্ট

- জাভাস্ক্রিপ্ট কন্ডিশনাল (শর্তবাচক) স্টেটমেন্ট কি?
- কন্ডিশনাল স্টেটমেন্টের প্রকারভেদ
 - ১.if statement
 - ২.if...else statement
 - ৩.if...else if...else statement
 - ৪.switch statement

অধ্যায়ঃ নয়- জাভাস্ক্রিপ্ট লুপ/ পুনঃরাবৃত্তি স্টেটমেন্ট

- লুপ (পুনঃরাবৃত্তি) স্টেটমেন্ট কি?
- লুপ (পুনঃরাবৃত্তি) স্টেটমেন্টের প্রকারভেদঃ
- While loop
- Do.....while loop
- For loop
- For.....in loop
- জাভাস্ক্রিপ্ট লুপ কন্ট্রোলঃ
- ব্রেক স্টেটমেন্ট (break Statement)
- কন্টিনিউ (continue Statement)

অধ্যায়ঃ দশ- জাভাস্ক্রিপ্ট অ্যারে

- জাভাস্ক্রিপ্ট অ্যারে কি?
- জাভাস্ক্রিপ্ট অ্যারে তৈরি করা
- জাভাস্ক্রিপ্ট অ্যারে একসেস করা
- অ্যাসোসিয়েটিভ অ্যারে কি?

অধ্যায়ঃ এগার-জাভাস্ক্রিপ্ট ফাংশন

- জাভাস্ক্রিপ্ট ফাংশান কি?
- জাভাস্ক্রিপ্ট ফাংশানের প্রকারভেদ
- ১.বিল্ট-ইন ফাংশান
- জাভাস্ক্রিপ্ট অ্যারে ফাংশান (**JavaScript Array Function**)
- জাভাস্ক্রিপ্ট বুলিয়ান ফাংশান (**JavaScript Boolean Function**)
- জাভাস্ক্রিপ্ট ম্যাথ ফাংশান (**JavaScript Math Function**)
- জাভাস্ক্রিপ্ট ডেট ফাংশান(**JavaScript Date Function**)
- জাভাস্ক্রিপ্ট নাম্বার ফাংশান (**JavaScript Number Function**)
- জাভাস্ক্রিপ্ট স্ট্রিং ফাংশান (**JavaScript String Function**)
- জাভাস্ক্রিপ্ট রেগুলার এক্সপ্রেসান ফাংশান (**JavaScript RegExp Function**)
- ২.ইউজার ডিফাইন ফাংশান
- ইউজার ডিফাইন ফাংশান তৈরি করা

- ফাংশানের উদাহরণ
- ফাংশন কল করা
- ফাংশনে প্যারামিটার ব্যবহার করা
ফাংশান রিটার্ন স্টেটমেন্ট

অধ্যায়ঃ বার -জাভাস্ক্রিপ্ট ইভেন্ট

- জাভাস্ক্রিপ্ট ইভেন্ট কি?
- জাভাস্ক্রিপ্ট ইভেন্টের উদাহরণ
- জাভাস্ক্রিপ্ট ইভেন্ট হ্যান্ডেলার
- জাভাস্ক্রিপ্ট ইভেন্ট অবজেক্ট
- জাভাস্ক্রিপ্ট ইভেন্ট এট্রিবিউট
- মাউস/কীবোর্ড এট্রিবিউট
- অন্যান্য ইভেন্ট এট্রিবিউট

অধ্যায়ঃ তের- ইউজারের সাথে যোগাযোগ

- Alert Box
- Confirm Box
- Prompt Box

অধ্যায়ঃ চৌদ- অবজেক্ট ওরিয়েন্টেড জাভাস্ক্রিপ্ট

- অবজেক্ট ওরিয়েন্টেড জাভাস্ক্রিপ্ট কি?
- অবজেক্ট
- প্রপার্টি
- মেথড
- অবজেক্টের প্রকারভেদ
- ১.বিল্ট-ইন অবজেক্ট
- ২. ইউজার ডিফাইন অবজেক্ট
- জাভাস্ক্রিপ্টে অবজেক্ট তৈরি করা
- **1.new** অপারেটর ব্যবহার করে সরাসরি অবজেক্ট তৈরি করা
- **2. অবজেক্ট ইনিশিয়ালাইজার (initializer) / Constructor function** ব্যবহার করে অবজেক্ট তৈরি

অধ্যায়ঃ পনের - জাভাস্ক্রিপ্ট কুকি

- জাভাস্ক্রিপ্ট কুকি কি?
- কুকির উদাহরণ
- কুকি তৈরি ও জমা করা

অধ্যায়ঃ ষোল-জাভাস্ক্রিপ্ট ফর্ম ভেলিডেশন

- জাভাস্ক্রিপ্ট ফর্ম ভেলিডেশন কি?
- ফর্ম ভেলিডেশন দিয়ে যে কাজগুলো করা যায়
- ইনপুট ফিল্ড ফাঁকা কিনা তা চেক করা
- ইমেইল ভেলিডেশন
- জাভাস্ক্রিপ্ট গেটএলিমেন্টবাইআইডি কি?
- গেটএলিমেন্টবাইআইডি বিষয়ে যা মনে রাখা দরকার

অধ্যায়ঃ সতের- একনজরে জাভাস্ক্রিপ্ট

অধ্যায়ঃ এক- সাধারণ আলোচনা

জাভাস্ক্রিপ্ট কী?

জাভাস্ক্রিপ্ট হল ফ্রন্ট-এন্ড অবজেক্ট ওরিয়েন্টেড স্ক্রিপ্টিং ল্যাঙ্গুয়েজ। জাভাস্ক্রিপ্টের একটি বড় সুবিধা হল একটি ছোট প্রোগ্রামিং-এর সাহায্যে অনেক বড় কাজ করা যায়। জাভাস্ক্রিপ্ট হল একটি ইন্টারপ্রিটেড ল্যাঙ্গুয়েজ (যার অর্থ হল এটার পূর্ববর্তী কোন কম্পাইলেশনের প্রয়োজন হয় না। জাভাস্ক্রিপ্ট হল একটি ক্লাইন্ট সাইড স্ক্রিপ্টিং ল্যাঙ্গুয়েজ বা ব্রাউজার স্ক্রিপ্টিং। ক্লাইন্ট সাইড স্ক্রিপ্টিং ল্যাঙ্গুয়েজ এর অর্থ হচ্ছে যে ওয়েব ব্রাউজ করবে তার ব্রাউজার এই স্ক্রিপ্টগুলোকে **run/execute** করবে। স্ক্রিপ্টিং ল্যাঙ্গুয়েজ হল প্রোগ্রামিং ল্যাঙ্গুয়েজের সহজ ও সংক্ষিপ্ত রূপ। ওয়েব পেজে প্রোগ্রামিং-এর ছোঁয়া দিতেই স্ক্রিপ্টের উদ্ভাবন। আপনি যদি ওয়েব অ্যাপ্লিকেশন তৈরি করতে চান তবে আপনাকে অবশ্যই স্ক্রিপ্টিং ল্যাঙ্গুয়েজ ভালভাবে জানতে হবে। জাভাস্ক্রিপ্টের জন্ম সি/সি++ ও জাভা থেকে। এর সিনট্যাক্স সি/সি++ ও জাভার মত হলেও সি/সি++ ও জাভার অনেক জটিলতাই এখানে নেই। তাই এর সিনট্যাক্স অনেকটা ঐ সব ল্যাঙ্গুয়েজের মত। আরেকটি কথা জাভাস্ক্রিপ্ট ওয়েবের জন্য ছোট স্ক্রিপ্ট লিখতে সাহায্য করে যা HTML ডকুমেন্টের ইন্টারঅ্যাক্টিভিটি বাড়ায়। ক্লাইন্ট সাইড এর বিপরীত হল সার্ভার সাইড, সার্ভার সাইড ল্যাঙ্গুয়েজ গুলোর কোড ওয়েব সার্ভার এর মাধ্যমে **execute/run** হয়। জাভাস্ক্রিপ্ট এর প্রধান সুবিধা হল এর মাধ্যমে ভিজিটরকে সাইটের এর প্রতি আকর্ষণ সৃষ্টি করা যায়। জাভাস্ক্রিপ্টে নিজস্ব ফাংশন তৈরি করতে পারবেন, ইচ্ছামত ভেরিয়েবল ব্যবহার করতে পারবেন এমনকি ভেরিয়েবল টাইপ ডিকলারেশনেরও দরকার হবে না। জাভাস্ক্রিপ্ট হল ইন্টারপ্রিটেড ল্যাঙ্গুয়েজ অর্থাৎ এটির প্রতিটি লাইন ইন্টারপ্রেট (ব্রাউজার) দ্বারা পালিত হয়। আরেকটি কথা ওয়েবসাইট সাধারণত দুই ধরনের হয়ে থাকে, ডাইনামিক এবং স্ট্যাটিক। স্ট্যাটিক ওয়েবসাইট হল তাইই যার ডাটা পরিবর্তনশীল নয়, অর্থাৎ সহজ কথায় স্থির। ডাইনামিক হল যার ডাটা পরিবর্তনশীল। জাভাস্ক্রিপ্ট ওয়েবসাইটকে ডাইনামিক রূপ দেয়ার ক্ষেত্রে এক অপরিহার্য ভূমিকা পালন করে।

ইতিহাসঃ

জাভাস্ক্রিপ্ট উদ্ভাবন করেছে নেটস্কেপ কমুনিকেশন কর্পোরেশনের প্রোগ্রামার **Brendan Eich**। ১৯৯৫ সালের সেপ্টেম্বরে **LiveScript** নামে প্রথম জাভাস্ক্রিপ্ট আত্মপ্রকাশ করে। এরপর ডিসেম্বর ৪, ১৯৯৫ সালে এর নাম পরিবর্তন করে জাভাস্ক্রিপ্ট রাখা হয়, যদিও তখনও জাভাস্ক্রিপ্টের অফিসিয়াল নাম ছিল **EcmaScript**। **ECMAScript** ডেভলপ ও নিয়ন্ত্রণ করত **ECMA (European Computer Manufacturer's Association)** নামক একটি আন্তর্জাতিক সংগঠন।

প্রোগ্রামিং ল্যাঙ্গুয়েজ ও স্ক্রিপ্ট ল্যাঙ্গুয়েজের মধ্যে পার্থক্যঃ

১. স্বয়ংসম্পূর্ণ অ্যাপ্লিকেশন তৈরি করা যায় প্রোগ্রামিং ল্যাঙ্গুয়েজ ব্যবহার করে কিন্তু স্ক্রিপ্ট ল্যাঙ্গুয়েজের দ্বারা তৈরিকৃত অ্যাপ্লিকেশন চালানোর জন্য আলাদা অ্যাপ্লিকেশন/ল্যাঙ্গুয়েজ প্রয়োজন হয়।
২. স্ক্রিপ্ট ল্যাঙ্গুয়েজের কোডকে কম্পাইল করার দরকার হয় না।

জাভাস্ক্রিপ্ট বনাম জাভাঃ

JAVA এবং Javascript কি এক ?

কখনোই না। দুটির উদ্দেশ্য, নিয়ম সবই পুরোপুরি আলাদা। জাভা হল একটি জটিল ল্যাংগুয়েজ যা কোনো ওয়েবপ্রোগ্রাম নয়। এর মাধ্যমে আপনি আপনার মোবাইল বা ডেস্কটপের জন্য প্রোগ্রাম বা যেকোনো গেম তৈরি করতে পারেন। অন্যদিকে জাভাস্ক্রিপ্ট হল একটি সহজবোধ্য প্রোগ্রামিং ল্যাংগুয়েজ। তাছাড়া এদের ব্যবহার, লেখার নিয়মও আলাদা।

কিছু কিছু ক্ষেত্রে জাভাস্ক্রিপ্ট ও জাভা এক হলেও মৌলিক দিক থেকে দুটি আলাদা আলাদা ল্যাঙ্গুয়েজ। জাভা হল প্রোগ্রামিং ল্যাঙ্গুয়েজ যা Sun Microsystems নামক কোম্পানি ডেভলপ করেছে। অন্যদিকে জাভাস্ক্রিপ্ট উদ্ভাবন করেছে নেটস্কেপ কমুনিকেশন কর্পোরেশনের প্রোগ্রামার Brendan Eich। জাভাকে কম্পাইল করতে হয় এবং এটা যে কোন প্লাটফর্মে রান করা যেতে পারে, অন্যদিকে জাভাস্ক্রিপ্ট HTML কোডের মাঝে রাখা হয় এবং সরাসরি ব্রাউজার এটাকে ইন্টারপ্রেট করে। সিনট্যাক্স, reserved-words- জাভা ও জাভাস্ক্রিপ্টে আলাদা।

জাভাস্ক্রিপ্ট কেন প্রয়োজনঃ

ডাইনামিক ওয়েব পেজের চালিকাশক্তি হল স্ক্রিপ্ট(script)। স্ক্রিপ্ট হল কতগুলো এক্সিকিউটেবল স্টেটমেন্টের সমষ্টি (যেমন- macro or batch file) যা স্ক্রিপ্টিং ল্যাঙ্গুয়েজ দ্বারা তৈরি। HTML ব্যবহার করে বিভিন্ন ব্রাউজারে প্রদর্শনযোগ্য ওয়েব পেজে তৈরি করা হয় এবং সেই ওয়েব পেজকে আর্কিটেকচার ফরম্যাট দেওয়া হয় CSS ব্যবহার করে। কিন্তু কেবল HTML ও CSS ব্যবহার করে গতিময় ও ইন্টারঅ্যাক্টিভ ওয়েব পেজ তৈরি করা সম্ভব না। ওয়েব পেজে ইন্টারঅ্যাক্টিভিটি আনতে প্রয়োজন প্রোগ্রামিং। আর ওয়েব পেজে এই প্রোগ্রামিং এর কাজ করা হয় স্ক্রিপ্ট ল্যাঙ্গুয়েজ ব্যবহার করে।

জাভাস্ক্রিপ্ট এর সাহায্যে নিচের কাজগুলো এর যায়।

- ওয়েব সাইটে ঘড়ি তৈরি।
- Mouse Trailers (site ব্রাউজ এর সময় মাউস এ স্ট্রি এনিমেশন)
- ওয়েব পেজে পাসওয়ার্ড প্রদান।
- ব্রাউজার নাম, ভার্শন, আইএসপি জেনে তা ইউজারকে জানানো।
- সময়ের সাথে অভিবাদন জানানো।
- সময়ের সাথে ডিজাইন পরিবর্তন।
- ওয়েবে প্রবেশের কিছু সময় পর অন্য লকেসানে নিয়ে যাওয়া।
- বিভিন্ন কন্ডিশানের উপর ভিত্তি করে অন্য পেজে প্রবেশের সুযোগ দেওয়া
- গেম তৈরি।
- ইনপুট ও এনভায়রনমেন্ট অনুযায়ী সাড়া দেওয়া।
- ডায়নামিক ড্রপডাউন মেনু
- Alert মেসেজ
- পপআপ উইন্ডো
- ফর্ম ভেলিডেশন
- স্লাইড শো
- চলন্ত খবর

- আরও অনেক...

জাভাস্ক্রিপ্ট আরও যে কাজগুলো করতে পারেঃ

- এইচটিএমএল ডিজাইনারদের জন্য জাভাস্ক্রিপ্ট একটি টুল হিসেবে কাজ করে।
- এইচটিএমএল কোডের মাঝে ডায়নামিক টেক্সট ইনপুট করে।
- কোন ঘটনা ঘটলেই জাভাস্ক্রিপ্ট সাড়া দিতে পারে যেমন- কোন পেজ লোড হবার পর বা কোন বাটনে প্রেস করলেই জাভাস্ক্রিপ্ট কাজ করবে।
- জাভাস্ক্রিপ্ট HTML এলিমেন্টকে রিড,রাইট ও পরিবর্তন করতে পারে।
- সার্ভারে ডাটা পাঠানোর আগেই জাভাস্ক্রিপ্ট তা চেক করতে পারে। এর ফলে সার্ভারে অতিরিক্ত প্রসেসিং-এর দরকার হয় না।
- ব্রাউজার ডিটেক্ট করে নির্ধারণ করে ঐ ব্রাউজারের জন্য কোন পেজ প্রদর্শন করতে হবে।

জাভাস্ক্রিপ্ট ব্যবহার করে আপনি আপনার ওয়েব পেজে এনিমেশন ইফেক্ট যোগ করতে পারেন কোন প্রকার এক্সটারনাল ফ্লাশ প্লাগইন ছাড়া যেমন- HTML5 এর canvas (ওয়েব পেজে সরাসরি ড্রয়িং-এর সুবিধা দেয়) এলিমেন্টের ব্যবহার। এছাড়াও ড্রাগ ড্রপের সুবিধা, আপনার ওয়েব সাইটকে এক্সটারনাল ওয়েব সাইটের (যেমন- Facebook, Twitter, etc.)সাথে ইন্টিগ্রেট করার সুবিধা দেয়।

জাভাস্ক্রিপ্ট সক্রিয় করা

জাভাস্ক্রিপ্ট কে ইন্টারনেট এক্সপ্লোরার- এ সচল করারপদ্ধতি:

Internet Explorer এ আপনি security setting এ গিয়ে check করতে পারেন যে আপনার জাভাস্ক্রিপ্ট কি সচল রয়েছে কিনা। নিচে জাভাস্ক্রিপ্ট সচল করার উপায় দেয়া হলো।

১ প্রথমে **Tools** menu তে **Click** করতে হবে

২ তারপর menu হতে **Internet Options** নির্বাচন করতে হবে

৩ **Internet Options** এর **Security** tab এ **Click** করতে হবে

৪ তারপর **Custom Level** বাটনে **Click** করে **security settings** এ প্রবেশ করতে হবে

৫ **Scroll** করে **Scripting** section এ যেতে হবে

৬ **script** সচল করা জন্য **Enable** বাটন **Select** করতে হবে

৭ প্রক্রিয়াটি সম্পন্ন করতে **OK** বাটনে **Click** করতে হবে

৮ করার জন্য **Yes** বাটনে **Click** করতে হবে

জাভাস্ক্রিপ্ট কে ফায়ারফক্স- এ সচল করারপদ্ধতি:

Firefox এ আপনি **Options** এর **Content setting** এ গিয়ে **check** করতে পারেন যে আপনার জাভাস্ক্রিপ্টটি কি সচল রয়েছে কিনা। নিচে জাভাস্ক্রিপ্ট সচল করার উপায় দেয়া হলো।

১ প্রথমে **Tools menu** তে **Click** করতে হবে

২ তারপর **menu** হতে **Options** নির্বাচন করতে হবে

৩ **Options** এর **Content tab** এ **Click** করতে হবে

৪ নিশ্চিত করুন যে **Enable JavaScript check box** এ টিক দেয়া আছে কিনা

৫ প্রক্রিয়াটি সম্পন্ন করতে **OK** বাটনে **Click** করতে হবে

জাভাস্ক্রিপ্ট কে অপেরা - তে সচল করারপদ্ধতি:

Opera তে আপনি **Preferences** এর **Content setting** এ গিয়ে **check** করতে পারেন যে আপনার জাভাস্ক্রিপ্টটি কি সচল রয়েছে কিনা। নিচে জাভাস্ক্রিপ্ট সচল করার উপায় দেয়া হলো।

১ প্রথমে **Tools menu** তে **Click** করতে হবে

২ তারপর **menu** হতে **Preferences** নির্বাচন করতে হবে

৩ **Preferences** এর **Advanced tab** এ **Click** করতে হবে

৪ বাম পাশের লিস্ট **item** হতে **Content** নির্বাচন করতে হবে

৫ নিশ্চিত করুন যে **Enable JavaScript check box** এ টিক দেয়া আছে কিনা

৬ প্রক্রিয়াটি সম্পন্ন করতে **OK** বাটনে **Click** করতে হবে

জাভাস্ক্রিপ্ট কোথায় লিখতে হয়:

HTML-এর মত জাভাস্ক্রিপ্ট কোড লিখতে আলাদা কোন সফটওয়্যারের প্রয়োজন হবে না। টেক্সট এডিটর নোটপ্যাডই কোড লিখা যাবে। [Notepad++](#) (for Windows users) and [TextWrangler](#) (for Mac users)

অধ্যায়ঃ দুই- জাভাস্ক্রিপ্ট শুরু করা

জাভাস্ক্রিপ্ট সিনট্যাক্সঃ

জাভাস্ক্রিপ্ট সিনট্যাক্স হল কতগুলো নিয়মের (a set of rules) সমষ্টি যা নির্ণয় করে কিভাবে একজন প্রোগ্রামার ল্যান্ডমার্ক লিখবে এবং কোডটিকে কিভাবে ব্রাউজার ইন্টারপ্রেট করবে। জাভাস্ক্রিপ্ট হল কতগুলো স্টেটমেন্টের সমষ্টি যা এইচটিএমএল-এর `<script>... </script>` ট্যাগের মাঝে থাকে। এইচটিএমএল কোডের যে কোন স্থানে `<script>` ট্যাগ স্থাপন করা যায় তবে হেড সেকশনে রাখা উত্তম। `<script>` ট্যাগ ব্রাউজার প্রোগ্রামকে `<script>... </script>` ট্যাগের মাঝের স্ক্রিপ্টকে ইন্টারপ্রেট করার নির্দেশ দেয়।

সাধারণ জাভাস্ক্রিপ্ট সিনট্যাক্স হল-

```
<script type="text/javascript">
<!--
এখানে অপারেটর, ফাংশন, ভেরিয়েবল, ডাটা(এককথায় প্রোগ্রামের সকলকিছু এখানে থাকবে)
-->
</script>
```

স্ক্রিপ্ট ট্যাগের দুটি গুরুত্বপূর্ণ এট্রিবিউট আছে-

- **language:** ল্যান্ডমার্ক এট্রিবিউট নির্দেশ করে স্ক্রিপ্টিং ল্যান্ডমার্ক হিসেবে কোন ল্যান্ডমার্ক ব্যবহার করা হচ্ছে। HTML ও XHTML-এর সাম্প্রতিক ভার্সনে এটা ব্যবহারে নিষেধ করা হয়েছে।

```
<script language = "JavaScript">
```

- **type:** এই এট্রিবিউট স্ক্রিপ্টিং ল্যান্ডমার্ক নির্দেশ করে। স্ক্রিপ্ট ল্যান্ডমার্ককে সাধারণত কনটেন্ট টাইপ (যেমন- "text/javascript") হিসেবে নির্দিষ্ট করা হয়।

```
<script type="text/javascript">
```

প্রথম জাভাস্ক্রিপ্ট কোড লেখাঃ

"Hello World" লেখাটি আমরা ব্রাউজারে দেখাতে চাচ্ছি। নিচের মত করে কোড লিখুন-

```
<html>
<body>
<script type="text/javascript">
<!--
  document.write("Hello World!")
//-->
</script>
</body>
</html>
```

আমাদের প্রথম ধাপ হচ্ছে `<script>` ট্যাগ ব্যবহারের মাধ্যমে ব্রাউজার কে বোঝাতে হবে যে আমরা জাভাস্ক্রিপ্ট ব্যবহার করছি। `script type` হিসাবে "text/JavaScript" সেট করতে হবে। এখানে আমরা `document.write` নামে একটি স্টেটমেন্ট ব্যবহার করেছি যার কাজ হল টেক্সট প্রদর্শন করা। আমরা নিচের মত আউটপুট দেখতে পাব-

Hello World!

জাভাস্ক্রিপ্ট স্টেটমেন্ট

জাভাস্ক্রিপ্ট হল কতগুলো পর্যায়েক্রমিক স্টেটমেন্টের সমষ্টি যা ব্রাউজার দ্বারা এক্সিকিউট হয়। প্রতিটা স্টেটমেন্ট ব্রাউজারকে একটি নির্দেশ প্রদান করে। জাভাস্ক্রিপ্ট স্টেটমেন্ট মূলত ব্রাউজারকে কোন নির্দেশ প্রদান করে। আর এই নির্দেশের উদ্দেশ্য হল ব্রাউজারকে কি করতে হবে তা বলে দেওয়া। যেমনঃ আমরা যদি **Hello Dolly** লেখাটি ওয়েব পেজে প্রদর্শন করতে চাই তবে ব্রাউজারকে নিচের মত নির্দেশ প্রদান করতে হবে-

```
document.write("Hello Dolly");
```

স্টেটমেন্ট দিয়ে ব্রাউজারকে **Hello Dolly** লিখতে বলে। স্টেটমেন্টের শেষে সেমিকোলন (;) দিয়ে শেষ করতে হয়। এটা অপশনাল। প্রতিটা **Statement** এক লাইনে লিখতে হবে (এক লাইনের **Statement** হলে তাদের শেষে সেমিকোলন না দিলেও হবে) আর একাধিক **Statement** এক লাইনে লিখলে তাদেরকে সেমিকোলন দ্বারা পৃথক করতে হবে। জাভাস্ক্রিপ্ট কেস সেন্সেটিভ তাই স্টেটমেন্ট, ভেরিয়েবল, অবজেক্ট, ফাংশান তৈরির সময় সঠিক থাকতে হবে।

স্টেটমেন্টের উদাহরণ-

নিচের কোডটি ওয়েব পেজে একটি হেডিং ও দুটি প্যারাগ্রাফ প্রদর্শন করবে-

```
<html>
```

```
<body>
```

```
<script type="text/javascript">
```

```
document.write("<h1>This is a heading</h1>");
```

```
document.write("<p>This is a paragraph.</p>");
```

```
document.write("<p>This is another paragraph.</p>");
```

```
</script>
```

```
</body>
```

```
</html>
```

জাভাস্ক্রিপ্ট ব্লকঃ

জাভাস্ক্রিপ্ট স্টেটমেন্টগুলো গ্রুপ আকারে একটা ব্লকের মাঝে থাকে। ব্লক শুরু হয় একটা বাম বাঁকানো ব্যাকেট”{” দিয়ে এবং শেষ হয় ডান বাঁকানো ব্যাকেট”}” দিয়ে। এই ব্লক তৈরির উদ্দেশ্য হল পর্যায়েক্রমিক কতগুলো স্টেটমেন্টকে একসাথে একত্রিত করা। নিম্নে ব্লকের একটি উদাহরণ দেওয়া হল-

```
<html>
```

```
<body>
```

```
<script type="text/javascript">
```

```
{
```

```
document.write("<h1>This is a heading</h1>");  
  
document.write("<p>This is a paragraph.</p>");  
  
document.write("<p>This is another paragraph.</p>");  
  
}
```

```
</script>
```

```
</body>
```

```
</html>
```

কোড বিশ্লেষণঃ এখানে যেভাবে ব্লকের মাঝে কোড লেখা হয়েছে আসলে সচারচর এভাবে লেখা হয় না। কতগুলো স্টেটমেন্টসহ একটা ফাংশান বা কন্ডিসানের উপর ভিত্তি করে ব্লক তৈরি করা হয়। পরবর্তীতে এসব নিয়ে বিস্তারিত আলোচনা করা হবে।

জাভাস্ক্রিপ্ট কোথায় থাকবে?

নিচের উল্লেখিত যে কোন স্থানে আপনি জাভাস্ক্রিপ্ট রাখতে পারেন-

- এইচটিএমএল-এর **<head>** ট্যাগের মাঝে।
- এইচটিএমএল-এর **<body>** ট্যাগের মাঝে।
- **<body>** ও **<head>** উভয় সেকশনেই রাখা যাবে।
- এক্সটারনাল ফাইল হিসেবে (যা এইচটিএমএল-এর সাথে লিঙ্ক করে দিতে হবে)

<head>ট্যাগের মাঝে জাভাস্ক্রিপ্টের ব্যবহার-

ফাংশন ও গুরুত্বপূর্ণ স্ক্রিপ্ট সমূহ, যার উপর অন্য স্ক্রিপ্ট নির্ভর করে তা হেড সেকশনে রাখা ভাল। যদি আপনি চান জাভাস্ক্রিপ্ট কে কিছু ইভেন্ট (যেমন যখন কোন ব্যবহারকারী কোন বাটনে ক্লিক করবে) এর উপর রান করাবেন সেক্ষেত্রে আপনি জাভাস্ক্রিপ্ট কে হেড ট্যাগে রাখতে পারেন। যেমন- নিচের উদাহরণে ব্যবহারকারী যখন বাটনে ক্লিক করবে তখন **alert box** দেখাবে।

```
<html>
```

```
<head>
<script type="text/javascript">
<!--
function sayHello() {
  alert("Hello World")
}
//-->
</script>
</head>
<body>
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

আমরা একটা ফাংশন তৈরী করেছি যার নাম **sayHello** এবং এটাকে এইচটিএমএল ডকুমেন্ট এর **head** ট্যাগ এ রেখেছি । এখন আমরা যতবারই বাটনে ক্লিক করবো ততবারই "Hello World!" নামের **alert box** দেখাবে।

<body>ট্যাগের মাঝে জাভাস্ক্রিপ্টের ব্যবহার-

কোন পেজ লোড হওয়া মাত্রই যদি আপনি জাভাস্ক্রিপ্ট কে রান করতে চান সেক্ষেত্রে আপনি জাভাস্ক্রিপ্ট কে বডি ট্যাগের মাঝে রাখতে পারেন। এক্ষেত্রে কোন ফাংশান ডিকলারেশনের দরকার নেই।

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
<!--
document.write("Hello World")
//-->
</script>
<p>This is web page body </p>
</body>
</html>
```

নিচের মত আউটপুট দেখা যাবে-

Hello World

This is web page body

<body> ও <head> উভয় ট্যাগের মাঝে জাভাস্ক্রিপ্টের ব্যবহার-

একই সাথে হেড ও বডি সেকশনে জাভাস্ক্রিপ্ট কোড লেখা যায়। যেমন-

```
<html>
<head>
<script type="text/javascript">
<!--
function sayHello() {
  alert("Hello World")
}
//-->
</script>
</head>
<body>
<script type="text/javascript">
<!--
document.write("Hello World")
//-->
</script>
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

নিচের মত আউটপুট দেখা যাবে-

Hello World

এক্সটারনাল জাভাস্ক্রিপ্টের ব্যবহারঃ

একই জাভাস্ক্রিপ্ট কোড বিভিন্ন পেজে ব্যবহার করতে বা কোডের পরিমাণ বেশি হলে এক্সটারনাল জাভাস্ক্রিপ্ট ব্যবহার করা হয়। হেড ও বডি উভয় সেকশানেই এক্সটারনাল জাভাস্ক্রিপ্ট ব্যবহার করা যায়। প্রথমত জাভাস্ক্রিপ্ট কোডগুলো একটি এডিটরে লিখতে হবে যেখানে `<script>.....</script>` ট্যাগ থাকবে না, শুধুমাত্র জাভাস্ক্রিপ্ট কোড থাকবে। যেমন-

```
function sayHello() {  
  alert("Hello World")  
}
```

এরপর ফাইলটিকে ".js" extension দিয়ে সেভ করতে হবে। মনে করি ফাইলটিকে filename.js নাম দিয়ে সেভ করা হয়েছে। এরপর একটি HTML কোড লিখতে হবে যেখানে `<script>` ট্যাগের src এট্রিবিউটের মাধ্যমে ঐ filename.js ফাইলের লোকেশান জানিয়ে দিতে হবে। যেমন-

```
<html>  
<head>  
<script type="text/javascript" src="filename.js" ></script>  
</head>  
<body>  
.....  
</body>  
</html>
```

এরপর কোডটিকে "something.html" নাম দিয়ে সেভ করতে হবে। এক্ষেত্রে HTML file এবং জাভাস্ক্রিপ্ট file একই directory তে আছে বলে মনে করা হচ্ছে।

জাভাস্ক্রিপ্ট কমেন্টসঃ

আপনি যদি আগের কোন প্রগ্রামিং ল্যাঙ্গুয়েজ শিখে থাকেন তাহলে কমেন্টস এর প্রয়োজনীয়তা বুঝতে পারেন। প্রোগ্রামের মধ্যে যা প্রগ্রামের অংশ না কিন্তু প্রগ্রামারের সুবিধার জন্য নোট আকারে কমেন্টস লেখা হয়। জাভাস্ক্রিপ্টে কমেন্ট লেখার ক্ষেত্রে // চিহ্ন দিয়ে কমেন্ট লিখুন। যেমন-

```
<script type="text/javascript">  
// এটি কমেন্ট, যা প্রোগ্রামের অংশ নয়।  
document.write("<h1>This is a heading</h1>");  
// এক লাইনের কমেন্ট এভাবে লিখুন  
document.write("<p>This is a paragraph.</p>");  
document.write("<p>This is another paragraph.</p>");  
</script>
```

একাধিক লাইন হলে

```
<script type="text/javascript">
/*
উপরের চিহ্ন দিয়ে কমেণ্ট লেখা শুরু করুন
একাধিক লাইনের কমেণ্ট এভাবে লিখতে হয়।
*/
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

```
<script language="javascript" type="text/javascript">
<!--

// সিঙ্গেল লাইন কমেণ্টের জন্য।

/*
একাধিক লাইনের কমেণ্টের জন্য।
*/
//-->
</script>
```

যে কাজটি জাভাস্ক্রিপ্ট দিয়ে করতে পারবেন না।

আপনি জাভাস্ক্রিপ্টকে একটি ব্রাউজারে রান করতে বাধ্য করতে পারেন না। কারণ, আমরা জানি জাভাস্ক্রিপ্ট ক্লাইন্ট সাইড স্ক্রিপ্টিং ল্যান্গুয়েজ যা ব্রাউজারে এক্সিকিউট বা রান হয়। আপনি যদি পুরাতন ভার্শনের ব্রাউজার ব্যবহার করেন বা আপনার ব্রাউজারে যদি জাভাস্ক্রিপ্ট ডিজেবল করা থাকে তবে ব্রাউজারে জাভাস্ক্রিপ্ট কাজ করবে না।

সার্ভারের রিসোর্স আপনি জাভাস্ক্রিপ্ট দিয়ে একসেস করতে পারবেন নাঃ

আমরা জানি জাভাস্ক্রিপ্ট একটি ক্লাইন্ট সাইড ল্যান্গুয়েজ যা কেবল মাত্র ব্রাউজার এনভায়রনমেন্টে কাজ করে। জাভাস্ক্রিপ্ট সার্ভারের রিসোর্স (যেমন-ডাটাবেজ) একসেস করতে পারে না।

অধ্যায়ঃতিন- জাভাস্ক্রিপ্ট ভেরিয়েবল

জাভাস্ক্রিপ্ট ভেরিয়েবল

ভেরিয়েবল হচ্ছে একটা পাত্রের মত (Container) যেখানে আমরা অনেক তথ্য রাখতে পারি। যেমন-একটা টেক্সট স্ট্রিং- "Hello Bangladesh" অথবা একটা Integer value 100। কোন একটা ভেরিয়েবলে একবার তথ্য রেখে (কোন Variable এ কিছু রাখা এটাকে বলে Variable declare বা ঘোষণা) সেটা পুরো কোডজুড়ে বারবার ব্যবহার করতে পারেন।

ভেরিয়েবল হল তথ্য জমা রাখার একটি পাত্রের মত, যা ভেলু ($x=5$) বা কোন এক্সপ্রেশন($z=x+y$) ধারণ করতে পারে।

ভেরিয়েবল হল একটা প্রতীকি (symbolic) নাম যা আপনার নির্দিষ্ট করে দেয়া ভেলুকে উপস্থাপন করে। ভেরিয়েবল এর উদ্দেশ্য হচ্ছে আপনার তথ্যকে জমা রাখা যাতে আপনি পরবর্তিতে তা ব্যবহার করতে। ভেরিয়েবল নাম (name) দ্বারা ডাটা মোড়ানো থাকে যাতে আপনি সহজে এটাকে move করতে পারেন।

জাভাস্ক্রিপ্টে ভেরিয়েবল ডিকলার করাঃ

জাভাস্ক্রিপ্ট প্রোগ্রামে ভেরিয়েবল ব্যবহারের পূর্বে অবশ্যই তাকে ডিকলার করে নিতে হবে। সম্পূর্ণ প্রোগ্রামে ভেরিয়েবল একবার ডিকলার বা ইনিশিয়ালাইজ করলে দ্বিতীয়বার প্রোগ্রামে ডিকলার বা ইনিশিয়ালাইজ করার দরকার হবে না। জাভাস্ক্রিপ্টে ভ্যারিয়েবলের ডাটা টাইপ উল্লেখ করার প্রয়োজন নাই। জাভাস্ক্রিপ্টে Variable ঘোষণা করা হয় Var দিয়ে। এরপর Variable নাম ও এর একটা মান(Value) সেট করে দিতে হয়। যেমনঃ

```
Var Variable_Name="Value";
```

জাভাস্ক্রিপ্টে ভ্যারিয়েবল ডিকলারেশনের জন্য Var ব্যবহার না করলেও চলে। যেমনঃ

```
Var x=12; এর পরিবর্তে শুধু x=12; লিখলেও চলবে।
```

উল্লেখ্য, উপরের প্রতিটা লাইনও একেকটা স্টেটমেন্ট তাই, প্রতিটা স্টেটমেন্ট এর পরে সেমিকোলন (;) দেয়া হল।

জাভাস্ক্রিপ্টে ভেরিয়েবল ডিকলারের বিভিন্ন পদ্ধতিঃ

```
// একটি জাভাস্ক্রিপ্ট ভেরিয়েবল ডিকলার
```

```
var firstName;
```

```
// একাধিক জাভাস্ক্রিপ্ট ভেরিয়েবল ডিকলার  
var firstName, lastName;
```


```
// একটি জাভাস্ক্রিপ্ট ভেরিয়েবল ডিকলার ও তার মান এসাইন করে দেওয়া।  
var firstName = 'Homer';
```

```
// একাধিক জাভাস্ক্রিপ্ট ভেরিয়েবল ডিকলার ও তাদের মান এসাইন করে দেওয়া।  
var firstName = 'Homer', lastName = 'Simpson';
```

নোটঃ ভেরিয়েবলের ভেলু হিসেবে টেক্সট ডিকলার করলে তাকে অবশ্যই ডাবল কোটেশানের মধ্যে রাখতে হবে। আপনি যদি ভেরিয়েবলকে পুনরায় ডিকলার করেন তবে এটা তার মান হারাতে না।

ভেরিয়েবল এর উদাহরন:

প্রথমবার ভেরিয়েবল ব্যবহারের ক্ষেত্রে ভেরিয়েবল নামের পুরে "var" লেখা জরুরী নয় তবে ভাল programming practice এর জন্য প্রথমবার ভেরিয়েবল নামের পুরে "var" লেখা উচিত। নিচে উদাহরনের মাধ্যমে বিষয়টি পরিস্কার করা হল।

```
<body>  
<script type="text/JavaScript">  
<!--  
var linebreak = "<br />"  
var my_var = "Hello World!"  
  
document.write(my_var)  
document.write(linebreak)  
  
my_var = "I am learning JavaScript!"  
document.write(my_var)   
document.write(linebreak)  
  
my_var = "Script is Finishing up..."  
document.write(my_var)  
//-->  
</script>  
</body>
```

প্রদর্শন:

Hello World!
I am learning JavaScript!
Script is Finishing up...

উপরের উদাহরণে আমরা দুটি ভেরিয়েবল তৈরী করেছি যার একটি লাইন ব্রেকের জন্য HTML কে ধরেছে অপরটি হচ্ছে ডাইনামিক ভেরিয়েবল যার উপরের script এর মধ্যে তিনটি ভিন্ন ধরনের ভেলু রয়েছে। ভেরিয়েবল এর ভেলু নির্দিষ্ট করতে সমান চিহ্ন (=) ব্যবহার করতে হবে। যেখানে বাম পাশে থাকবে ভেরিয়েবল এবং ডান পাশে থাকবে ভেরিয়েবল এর ভেলু। যেমন my_var = "Hello World!" এর মানে হচ্ছে my_var সমান "Hello World!" । ভেরিয়েবল এবং এর ভেলু বসানোর order ঠিক রাখতে হবে অর্থাৎ প্রথমে ভেরিয়েবল এর নাম পরে ভেরিয়েবল এর ভেলু তা না হলে script ঠিকমত কাজ করবে না।

আরেকটি উদাহরণ দেখুনঃ

স্ক্রিপ্ট এক্সিকিউট হবার সময়ও ভেরিয়েবলের মান পরিবর্তন করে দেওয়া যায়। আপনি একটি ভেরিয়েবলকে তার নাম দিয়ে রেফার করতে পারেন ভেরিয়েবলের মান ডিসপ্লে বা পরিবর্তনের জন্য।

```
1 <html>
2 <body>
3
4 <script type="text/javascript">
5 var firstname;
6 firstname="faruk";
7 document.write(firstname);
8 document.write("<br />");
9 firstname="himu";
10 document.write(firstname);
11 </script>
12
13 <p>The script above declares a variable,
14 assigns a value to it, displays the value,
15 changes the value,
16 and displays the value again.</p>
17
18 </body>
19 </html>
```

জাভাস্ক্রিপ্ট ভেরিয়েবল নামের নিয়মনীতিঃ

- ভেরিয়েবলের নাম কেস সেন্সিটিভ(y এবং Y are দুটি আলাদা ভেরিয়েবল)
- ভেরিয়েবলের নাম অবশ্যই বর্ণ (A,B,c) বা আন্ডারস্কোর(_) দিয়ে শুরু হবে।
- ভেরিয়েবলের নামে যে কোন বর্ণ বা বর্ণমালা সংখ্যা (0-9) বা আন্ডারস্কোর থাকতে পারে।
- ভেরিয়েবলের নামের মাঝে কোন স্পেস থাকতে পারবে না।
- ভেরিয়েবলের নামের মাঝে বিভিন্ন চিহ্ন যেমন কমা, ফুলস্টপ ব্যবহার করা যাবে না।

- প্রথম ক্যারেক্টার ডিজিট(0-9) হতে পারবে না।
- ভেরিয়েবলের নাম হিসেবে "জাভাস্ক্রিপ্ট রিজার্ভ ওয়ার্ড" ব্যবহার করা যাবে না। যেমন-*break* বা *boolean* ভেরিয়েবলের নাম হিসেবে ব্যবহার করা যাবে না। জাভাস্ক্রিপ্ট রিজার্ভ ওয়ার্ডের তালিকা নিম্নে দেওয়া হল।

নোটঃ জাভাস্ক্রিপ্ট কেস সেন্সেটিভ তাই ভেরিয়েবলের নাম ও কেস সেন্সেটিভ।

ভেরিয়েবলের কার্যএলাকাঃ

ভেরিয়েবল ডিকলারেশনের স্থান অনুসারে এটির কার্য এলাকা **Local** ও **Global** দুধরনের হতে পারে।

লোকাল ভেরিয়েবল (Local Variable): যদি কোন ফাংশনের অধিনে ভেরিয়েবল ডিক্লেয়ার করেন তবে তা হবে **Local** ভেরিয়েবল। কারণ ঐ ভেরিয়েবল কেবল ঐ ফাংশানের মাঝেই কাজ করবে। যখন ফাংশানের কাজ শেষ হবে তখন ঐ ভেরিয়েবলও ভ্যানিশ হয়ে যাবে। বিভিন্ন ফাংশানে একই নামে ভেরিয়েবল ডিকলার করা যাবে। **Local** ভেরিয়েবল থাকবে বন্ধনী ({}) মধ্য। যেমন-

```
<script type="text/javascript">
function checkscope( )
{
  var myVar = "local"; // Declare a local variable
  document.write(myVar);
}
</script>
```

গ্লোবাল ভেরিয়েবল (Global Variable) : কোন ফাংশনের (Function) বাইরে ভেরিয়েবল ডিক্লেয়ার করলে তা **Global Variable** রূপে কাজ করবে। ঐ ভেরিয়েবলকে প্রোগ্রামের সকল ফাংশান ব্যবহার করতে পারবে। সুতরং কোন **Variable** কে একাধিক ফাংশানে ব্যবহার করতে চাইলে অবশ্যই তাকে **Global Variable** হিসেবে ঘোষণা করবেন। প্রোগ্রামের শুরু থেকে প্রোগ্রাম ক্লোজ না করা পর্যন্ত **Global Variable** কাজ করে। একই নামে প্রোগ্রামে একটি লোকাল ও গ্লোবাল ভেরিয়েবল থাকলে লোকাল ভেরিয়েবল প্রাধান্য পাবে। এক্ষেত্রে গ্লোবাল ভেরিয়েবল হাইড থাকবে। যেমন-

```
<script type="text/javascript">
<!--
var myVar = "global"; // Declare a global variable
function checkscope( ) {
  var myVar = "local"; // Declare a local variable
  document.write(myVar);
}
```

```
//-->  
</script>
```

আউটপুটঃ

```
local
```

নোটঃ "var" কীওয়ার্ড ছাড়া ভেরিয়েবল ডিকলার করলে তা **Global Variable** হিসেবে কাজ করবে। এক্ষেত্রে সাধারণত ভেরিয়েবলের মান বসিয়ে দেওয়া হয়। যেমন-

```
x=5;  
carname="Volvo";
```

জাভাস্ক্রিপ্ট ভেরিয়েবলের সাহায্যে এরিথমেটিক অপারেশন করা যায়। যেমন-

```
y=x-5;  
z=y+5;
```

জাভাস্ক্রিপ্ট Loosely-typed Language...!!!!

জাভাস্ক্রিপ্টকে বলা হয় "loosely-typed language" বা **untyped language**। কারণ ভেরিয়েবল ডিকলার করতে ডাটা টাইপ উল্লেখ করতে হয় না। এটার অর্থ হল একটি ভেরিয়েবল বিভিন্ন সময় বিভিন্ন টাইপের ডাটা ধারণ করতে পারে। যেমন-
var Age = 34; হল ইন্টিজার কিন্তু **var strAge = "34";** হল ক্যারেক্টার কারণ ভেলুতে ডাবল কোটেসান দেওয়া আছে। প্রোগ্রাম এক্সিকিউট হবার সময় ভেরিয়েবলের ভেলু টাইপ স্বয়ংক্রিয়ভাবে জাভাস্ক্রিপ্ট দ্বারা পরিবর্তিত হতে পারে।

যদি আপনি এটা ম্যাথ ফাংশান দিয়ে করতে চান (যেমন- ভেলুকে চার দিয়ে গুন করা) তবে ইন্টিজার টাইপের ভেলু পাওয়া যাবে। যদিও এটা সুবিধাজনক, তথাপি এটা ব্যবহারে সতর্ক থাকতে হবে কারণ অনাকাঙ্ক্ষিত ফলাফল আসতে পারে।

অধ্যায়ঃ চার- জাভাস্ক্রিপ্ট ডাটা টাইপ

জাভাস্ক্রিপ্টঃ ডাটা টাইপ-

জাভাস্ক্রিপ্ট বিভিন্ন ধরনের ডাটা টাইপ আছে যা আমরা ভেরিয়েবলের মান হিসেবে ব্যবহার করে থাকি। ভেলু হল একটি ইনফরমেশান যা **Number, String, Boolean, Null** ইত্যাদি হতে পারে। জাভাস্ক্রিপ্ট আপনাকে তিনটি প্রিমিটিভ (**primitive**) ডাটা টাইপ ব্যবহারের সুযোগ দেয়। যেমন-

১.সংখ্যা (Number) যেমন:1,2,55,.3641 ইত্যাদি.

২.যৌক্তিক (Boolean) মান অর্থাৎ TRUE অথবা FALSE.

৩.শব্দগুচ্ছ (Strings of text) যেমন:islamic university,cse,ice ইত্যাদি

এছাড়াও **null** ও **undefined** নামে দুটি **trivial** ডাটা টাইপ আছে যারা শুধুমাত্র একটি ভেলু ডিফাইন করে। **object** নামে জাভাস্ক্রিপ্টে একটি **composite** ডাটা টাইপ আছে।

নিম্নে বিভিন্ন ধরনের ডাটা টাইপের বর্ণনা দেওয়া হল-

১.নাম্বার ডাটা টাইপ (Number Data Type)

নাম্বার ডাটা টাইপ দুই ধরনের নিউমেরিক ভেলু উপস্থাপন করে- **32-bit** ইন্টিজার এবং **64-bit** ফ্লটিং পয়েন্ট নাম্বার।

নাম্বার ভেলু বিভিন্ন ফরম্যাটের হতে পারে। যেমন-

```
var age = 25; // simple, decimal, integer
var price = 45.95; // floating point
var permissions = 0775; // integer in octal, 509 in decimal
// (note the leading zero)
var flags = 0x1c; // integer in hexadecimal, 28 in decimal
// (note the 0x prefix)
var measurement = 5.397e-9; // floating point in
// scientific notation
```

নোটঃ জাভাস্ক্রিপ্ট ইন্টিজার ও ফ্লটিং পয়েন্ট ভেলুর মাঝে পার্থক্য করে না। জাভাস্ক্রিপ্টে সকল সংখ্যাকে ফ্লটিং পয়েন্ট হিসেবে ধরে নেওয়া হয়। জাভাস্ক্রিপ্ট সংখ্যা প্রকাশের ক্ষেত্রে **64-bit** ফ্লটিং পয়েন্ট ফরম্যাট ব্যবহার করে যা **IEEE 754** স্ট্যান্ডার্ড দ্বারা স্বীকৃত।

২. লজিক্যাল ডাটা টাইপ (Logical/Boolean Data Type)

Boolean সকল লাস্ফুয়েজের কমন ডাটা টাইপ। এটার দুটি মাত্র ভেলু আছে- **true** এবং **false**। এই দুটি ভেলু জাভাস্ক্রিপ্টের রিজার্ভ ওয়ার্ড, যাদের নিজস্ব কাজ রয়েছে। কন্ডিশনাল স্টেটমেন্টে বুলিয়ান ডাটা টাইপ ব্যবহার করা হয়।

```
var enabled = true;
var disabled = false;
```

৩. স্ট্রিং ডাটা টাইপ (String Data Type)

স্ট্রিং অনেক জনপ্রিয় একটা ডাটা টাইপ যা টেক্সট উপস্থাপনের জন্য ব্যবহার করা হয়। স্ট্রিং হল কতগুলো ক্যারেক্টারের সমষ্টি যা সিঙ্গেল বা ডাবল কোটেশান দ্বারা আবদ্ধ থাকে।

```
var name = 'Homer', lastName = "Simpson";
var host = 'Conan O'Brien';
var path = 'c:\\temp\\dir\\myfile.txt';
var tabDelimited = "COL1\tCOL2\tCOL3\nVAL1\tVAL2\tVAL3";
```

৪. নাল ডাটা টাইপ (Null Data Type)

null একটি বিশেষ ধরনের কীওয়ার্ড যা **null** ভেলু (কোন মান থাকবে না) প্রকাশ করে। অন্যভাবে বলা যায় **null** এমন একটি ডাটা টাইপ যার শুধুমাত্র একটি ভেলু- **null**। **null** ভেলু তখনই আমরা ব্যবহার করব যখন ভেরিয়েবলের মান আমাদের অজানা। আমরা জানি ভেরিয়েবলের নাম কেস সেন্সিটিভ তাই **null** কীওয়ার্ড **Null**, **NULL** অথবা অন্যকোন কনভিনেশন এক নয়।

```
var name = "Homer";
var ssn = null;
```

৫. আনডিফাইন্ড ডাটা টাইপ (Undefined Data Type)

Undefined ডাটা টাইপের শুধুমাত্র একটি ভেলু রয়েছে- **undefined**। এটা **null**-এর মত কিন্তু আবার সঠিকভাবে এক জিনিস না। **Undefined** মূলত কোন ভেরিয়েবলের ডিফল্ট মান প্রকাশ করে যা ইনিশিয়ালাইজের প্রয়োজন হয় না, এটাকে জাভাস্ক্রিপ্ট **Constant** ও বলা যায়। যেমন-

```
var name = "Homer";
var ssn;
```

উপরের উদাহরণে `ssn` হল `undefined` ভেরিয়েবল যা `null` বা অন্যকোন ভেলু দ্বারা ইনিশিয়ালাইজ করা হয় নি।

অধ্যায়ঃ পাঁচ- জাভাস্ক্রিপ্ট কনস্ট্যান্ট ও রিজার্ভড ওয়ার্ড

জাভাস্ক্রিপ্ট কনস্ট্যান্ট(JavaScript : Constants)

"`const`" কীওয়ার্ড ব্যবহার করে জাভাস্ক্রিপ্টে কনস্ট্যান্ট ডিকলার করা হয়। জাভাস্ক্রিপ্টে ফাংশান ডিকলারের সময় কনস্ট্যান্ট লোকাল বা গ্লোবাল উভয় ধরণের হতে পারে। কনস্ট্যান্ট হল `read-only` অর্থাৎ একবার ডিকলার করা হলে এটা আর পরিবর্তন করা যায় না। কনস্ট্যান্ট নামের নিয়ম নীতি ভেরিয়েবল নামের নিয়ম নীতির অনুরূপ, শুধু একটি বিষয়ের পাঠ্য যে কনস্ট্যান্ট ডিকলারের ক্ষেত্রে সবুদা "`const`" কীওয়ার্ড ব্যবহার করতে হবে। যদি "`const`" কীওয়ার্ড ব্যবহার করা না হয় তবে এটা ভেরিয়েবল হিসেবে ধরে নেওয়া হবে।

উদাহরণঃ

```
const country = 'Bangladesh';
```

//ফাংশান ডিকলারের সময় কনস্ট্যান্ট ও ফাংশানের নাম একই হওয়া যাবে না। একই ফাংশানের মাঝে কনস্ট্যান্ট ও ভেরিয়েবলের নাম একই হওয়া যাবে না। নিচের স্টেটমেন্টগুলিতে ইরর দেখাবে।

```
function abc()
{
const abc = 55;
}
function abc()
{
const x = 15;
var x;
}
```

জাভাস্ক্রিপ্ট রিজার্ভড ওয়ার্ডঃ

নিম্নে জাভাস্ক্রিপ্ট রিজার্ভ ওয়ার্ডের তালিকা দেওয়া হল। জাভাস্ক্রিপ্টে ফাংশন, ভেরিয়েবল, মেথড, লুপ লেবেল এবং যে কোন অবজেক্টের নাম ডিকলার করতে এইসকল রিজার্ভ ওয়ার্ড ব্যবহার করা থেকে বিরত থাকুন কারণ এদের নিজস্ব ব্যবহার আছে। এদের মাঝে কিছু জাভাস্ক্রিপ্ট কীওয়ার্ড রয়েছে।

Table of JavaScript Reserved Words							
break	continue	do	for	import	new	this	void
case	default	else	function	in	return	typeof	while
comment	delete	export	if	label	switch	var	with

Java Keywords (Reserved by JavaScript)		
abstract	implements	protected
boolean	instanceOf	public
byte	int	short
char	interface	static
double	long	synchronized
false	native	throws
final	null	transient
float	package	true
goto	private	

ECMAScript Reserved Words		
catch	enum	throw
class	extends	try
const	finally	
debugger	super	

Other JavaScript Keywords

alert	eval	Link	outerHeight	scrollTo
Anchor	FileUpload	location	outerWidth	Select
Area	find	Location	Packages	self
arguments	focus	locationbar	pageXoffset	setInterval
Array	Form	Math	pageYoffset	setTimeout
assign	Frame	menubar	parent	status
blur	frames	MimeType	parseFloat	statusbar
Boolean	Function	moveBy	parseInt	stop
Button	getClass	moveTo	Password	String
callee	Hidden	name	personalbar	Submit
caller	history	NaN	Plugin	sun
captureEvents	History	navigate	print	taint
Checkbox	home	navigator	prompt	Text
clearInterval	Image	Navigator	prototype	Textarea
clearTimeout	Infinity	netscape	Radio	toolbar
close	innerHeight	Number	ref	top
closed	innerWidth	Object	RegExp	toString
confirm	isFinite	onBlur	releaseEvents	unescape
constructor	isNaN	onError	Reset	untaint
Date	java	onFocus	resizeBy	unwatch
defaultStatus	JavaArray	onLoad	resizeTo	valueOf
document	JavaClass	onUnload	routeEvent	watch
Document	JavaObject	open	scroll	window
Element	JavaPackage	opener	scrollbars	Window
escape	length	Option	scrollBy	

অধ্যায়ঃ ছয়- জাভাস্ক্রিপ্ট অপারেটর

জাভাস্ক্রিপ্ট অপারেটর কি?

অপারেটর হল, বিভিন্ন ম্যাথমেটিক্যাল অপারেশন করার জন্য ব্যবহৃত চিহ্ন। যেমনঃ যোগ, বিয়োগ, গুন, ভাগ ইত্যাদি হল ম্যাথমেটিক্যাল অপারেশন, আর এ সকল অপারেশন সম্পন্ন করা হয় যথাক্রমে $+$, $-$, $*$ ও $/$ চিহ্নের মাধ্যমে। এসকল চিহ্নই হল অপারেটর।

জাভাস্ক্রিপ্টে সিদ্ধান্ত গ্রহণের ক্ষেত্রে মূল ভূমিকা পালন করে থাকে অপারেটর। অন্যান্য প্রোগ্রামিং ল্যাংগুয়েজ এর অপারেটর এর সাথে জাভাস্ক্রিপ্ট অপারেটর এর মিল আছে। অপারেটর হচ্ছে এমন একটি **symbol**(প্রতীক) যা কোন গাণিতিক কাজ করতে ব্যবহৃত হয়। বেশির ভাগ ক্ষেত্রে কাজগুলো হচ্ছে পাটিগণিতীয়(**arithmetic**) যেমন যোগ, বিয়োগ ইত্যাদি তবে সবক্ষেত্রে নয়। **Variable** ও অন্যান্য অবজেক্টের পারস্পরিক গাণিতিক ও যৌক্তিক সম্পর্ক বোঝানোর জন্য অপারেটর ব্যবহার করা হয়। কাজের ধরন অনুসারে জাভাস্ক্রিপ্টে পাঁচ ধরনের অপারেটর ব্যবহার করা হয়। যথাঃ

- 1.string Operators
- 2.comparison Operators
- 3.arithmetic Operators
4. assignment Operators
5. logical(or Relational) Operators
6. Conditional (or ternary) Operators

1.string operator:

জাভাস্ক্রিপ্টে দুটি string operator আছে-

+	দুই বা ততোধিক উপাদানকে যুক্ত করে
+=	একটি string এর সাথে আরেকটি string যুক্ত করে।

+ অপারেটরঃ

স্ট্রিং ভেরিয়েবলকে(দুই বা ততোধিক) একত্র করতে বা টেম্প্লেট ভেলুকে একত্র করতে + অপারেটর ব্যবহার করা হয়। যেমনঃ নিচের উদাহরণে কতগুলো স্ট্রিংকে একত্র করা হয়েছে-

উদাহরণঃ

"i"+"am"+myName

এখানে I ও am দুটো string এবং myName একটি variable। যদি myName variable এর মান faruk হয় তবে উপরের স্টেটমেন্ট হবে-

"I am faruk"

+= অপারেটরঃ

Book+="about JavaScript" হয়

যদি book variable-এর মান "This book is" হয়, তবে উপরের স্টেটমেন্ট হবে-

"This book is about JavaScript"

মেসেজ, পরামর্শ বা তথ্য প্রদর্শন করার জন্য অপারেটর অপরিহার্য। নিচের উদাহরণে ইউজারের নাম জেনে তা মেসেজ হিসেবে দেখাবে-

```
1 <HTML>
2 <HEAD>
3 <TITLE>Welcome to my Webpage</TITLE>
4 <BODY>
5 <SCRIPT LANGUAGE="JavaScript">
6   fullName = prompt ("What is your name, please?","")
7   document.write ("Welcome to my Webpage" + fullName)
8 </SCRIPT>
9 </BODY>
10 </HTML>
```

2. Comparison অপারেটরঃ

দুটো মানের মাঝে তুলনা করার জন্য Comparison অপারেটর ব্যবহার করা হয়। অন্যভাবে বলা যায়-ভেরিয়েবল বা ভেলুস এর মধ্যে সম্পর্ক নির্ণয় করতে Comparison অপারেটর ব্যবহৃত হয়। Comparison অপারেটর সাধারণত conditional স্টেটমেন্টগুলিতে বিভিন্ন ভেলুর মধ্য তুলনা করতে এবং ফলাফলের উপর ভিত্তি করে সিদ্ধান্ত নিতে সাহায্য করে। যেমন-

```
if (age<18) document.write("Too young");
```

Comparison অপারেটরসমূহ-

x=5 এর জন্য নিচের টেবিলে comparison অপারেটর ব্যাখ্যা করা হল

অপারেটর	বর্ণনা	উদাহরণ
==	is equal to	x==8 is false x==5 is true
===	is exactly equal to (value and type)	x===5 is true x=== "5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

```

2 <body>
3 <script type="text/javascript">
4   var a = 10;
5   var b = 20;
6   var linebreak = "<br />";
7
8   document.write("(a == b) => ");
9   result = (a == b);
10  document.write(result);
11  document.write(linebreak);
12
13  document.write("(a < b) => ");
14  result = (a < b);
15  document.write(result);
16  document.write(linebreak);
17
18  document.write("(a > b) => ");
19  result = (a > b);
20  document.write(result);
21  document.write(linebreak);
22
23  document.write("(a != b) => ");
24  result = (a != b);
25  document.write(result);
26  document.write(linebreak);
27
28  document.write("(a >= b) => ");
29  result = (a >= b);
30  document.write(result);
31  document.write(linebreak);
32
33  document.write("(a <= b) => ");
34  result = (a <= b);
35  document.write(result);
36  document.write(linebreak);
37 </script>
38 </body>

```

একটা **equal** চিহ্ন যে ভেলু সেট করে আর **double equal** চিহ্ন (**==**) দুটি ভেলুর মধ্যের তুলনা করে **Comparison** অপারেটর যা **conditional statement** এর ভিতরে ব্যবহৃত হয় এবং সত্য, মিথ্যা নির্ণয় করে। সমান(**==**) এবং সমান নয়(**!=**) অপারেটর ব্যবহার করা হলে স্ক্রিপ্ট ইঞ্জিন সেই স্টেটমেন্টের সত্যতা যাচাইয়ের আগে ডাটা টাইপ রূপান্তর (কনভার্সন) করে না। যেমন- **"10"==10** কে সমান ধারা হবে। কিন্তু অবশ্যই সমান(**==**) এবং অবশ্যই সমান নয়(**!=**) অপারেটর ব্যবহার করা হলে সেই স্টেটমেন্ট যাচাইয়ের আগে স্ক্রিপ্ট ইঞ্জিন ডাটা টাইপ কনভার্সনের কাজ সেরে নেবে। এক্ষেত্রে **"10"==10** সমান হবে

না। কারণ "10" হল স্ট্রিং আর 10 হল সংখ্যা। Conditional loop তৈরিতে Comparison অপারেটর সর্ধকভাবে ব্যবহৃত হয়।

3.arithmetic operator:

বিভিন্ন ভেরিয়েবলের মাঝে Arithmetic operators করতেই arithmetic অপারেটর ব্যবহার করা হয়।

y=5 এর জন্য arithmetic operators নিচে ব্যাখ্যা করা হল-

অপারেটর	বর্ণনা	উদাহরণ	ফলাফল	
+	Addition	$x=y+2$	x=7	y=5
-	Subtraction	$x=y-2$	x=3	y=5
*	Multiplication	$x=y*2$	x=10	y=5
/	Division	$x=y/2$	x=2.5	y=5
%	Modulus (division remainder)	$x=y\%2$	x=1	y=5
++	Increment	$x=++y$	x=6	y=6
		$x=y++$	x=5	y=6
--	Decrement	$x=--y$	x=4	y=4
		$x=y--$	x=5	y=4

```
4 <script type="text/javascript">
5 var a = 33;
6 var b = 10;
7 var c = "Test";
8 var linebreak = "<br />";
9
10 document.write("a + b = ");
11 result = a + b;
12 document.write(result);
13 document.write(linebreak);
14
15 document.write("a - b = ");
16 result = a - b;
17 document.write(result);
18 document.write(linebreak);
19
20 document.write("a / b = ");
21 result = a / b;
22 document.write(result);
23 document.write(linebreak);
24
25 document.write("a % b = ");
26 result = a % b;
27 document.write(result);
28 document.write(linebreak);
29
30 document.write("a + b + c = ");
31 result = a + b + c;
32 document.write(result);
33 document.write(linebreak);
34
35 a = a++;
36 document.write("a++ = ");
37 result = a++;
38 document.write(result);
39 document.write(linebreak);
40
41 b = b--;
42 document.write("b-- = ");
43 result = b--;
44 document.write(result);
45 document.write(linebreak);
46 </script>
```

4. assignment operator:

জাভাস্ক্রিপ্ট ভেরিয়েবলে মান এসাইন করতে **Assignment operators** ব্যবহার করা হয়।

x=10 এবং **y=5** এর জন্য নিম্নে assignment operators ব্যাখ্যা করা হল-

অপারেটর	উদাহরণ	Same As	ফলাফল
=	$x=y$		$x=5$
+=	$x+=y$	$x=x+y$	$x=15$
-=	$x-=y$	$x=x-y$	$x=5$
=	$x=y$	$x=x*y$	$x=50$
/=	$x/=y$	$x=x/y$	$x=2$
%=	$x%=y$	$x=x\%y$	$x=0$

```
<script type="text/javascript">
<!--
var a = 33;
var b = 10;
var linebreak = "<br />";

document.write("Value of a => (a = b) => ");
result = (a = b);
document.write(result);
document.write(linebreak);

document.write("Value of a => (a += b) => ");
result = (a += b);
document.write(result);
document.write(linebreak);

document.write("Value of a => (a -= b) => ");
result = (a -= b);
document.write(result);
document.write(linebreak);

document.write("Value of a => (a *= b) => ");
result = (a *= b);
document.write(result);
document.write(linebreak);

document.write("Value of a => (a /= b) => ");
result = (a /= b);
document.write(result);
document.write(linebreak);

document.write("Value of a => (a %= b) => ");
result = (a %= b);
document.write(result);
document.write(linebreak);
//-->
</script>
```

5. logical operator:

variables এবং values মধ্যে লজিক ডিটারমাইন করতে Logical operators ব্যবহার করা হয়

Comparison এবং Logical operators ব্যবহার করে সত্য মিথ্যা নির্ণয় করা হয়।

Assume variable A holds 10 and variable B holds 20 then:

অপারেটর	বর্ণনা	উদাহরণ
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is true.
	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Given that $x=6$ and $y=3$, the table below explains the logical operators:

Operator	Description	Example
&&	and	($x < 10$ && $y > 1$) is true
	or	($x==5$ $y==5$) is false
!	not	!($x==y$) is true

```

2 <body>
3 <script type="text/javascript">
4   var a = true;
5   var b = false;
6   var linebreak = "<br />";
7
8   document.write("(a && b) => ");
9   result = (a && b);
10  document.write(result);
11  document.write(linebreak);
12
13  document.write("(a || b) => ");
14  result = (a || b);
15  document.write(result);
16  document.write(linebreak);
17
18  document.write("(!(a && b) => ");
19  result = (!(a && b));
20  document.write(result);
21  document.write(linebreak);
22 </script>

```

6. Conditional Operator:

conditional operator কিছু কন্ডিশানের উপর ভিত্তি করে ভেরিয়েবলে ভেলু এসাইন করে।

সিনট্যাক্স:

`variablename=(condition)?value1:value2 ;`

উদাহরণ:

`greeting=(visitor=="PRES")?"Dear President ":"Dear ";`

যদি **visitor** ভেরিয়েবলের মান "PRES" হয় তবে, **greeting** ভেরিয়েবল "Dear President " মান এসাইন করবে। অন্যথায় "Dear " এসাইন করবে।

জাভাস্ক্রিপ্ট অপারেটরের ভেরিয়েবল সহ উদাহরণ:

নিচে একটা সাধারণ উদাহরণ দেয়া হলো যেখানে মৌলিক পাটিগনিতীয় কার্য সম্পন্ন হয়েছে।

```
2 <body>
3 <script type="text/JavaScript">
4 <!--
5 var two = 2
6 var ten = 10
7 var linebreak = "<br />"
8   document.write("two plus ten = ")
9   var result = two + ten
10  document.write(result)
11  document.write(linebreak)
12
13
14  document.write("ten * ten = ")
15  result = ten * ten
16  document.write(result)
17  document.write(linebreak)
18  document.write("ten / two = ")
19  result = ten / two
20  document.write(result)
21  //-->
22 </script>
23 </body>
```

প্রদর্শন:

two plus ten = 12

ten * ten = 100

ten / two = 5

স্ট্রিং ও নাম্বার যোগ করাঃ

যদি আপনি নাম্বার ও স্ট্রিং এক সাথে যোগ করেন তবে আউটপুট স্ট্রিং হবে। যেমন-

x=5+5;

document.write(x); // আউটপুট 10 হবে

x="5"+"5";

```
document.write(x);    //  
  
x=5+"5";  
document.write(x);    // আউটপুট 55 হবে  
x="5"+5;  
document.write(x);    // আউটপুট 55 হবে
```

অধ্যায়ঃ সাত- জাভাস্ক্রিপ্ট ব্যবহার নির্দেশিকা

জাভাস্ক্রিপ্ট ব্যবহার নির্দেশিকাঃ

যখন জাভাস্ক্রিপ্ট কোডিং করবেন তখন কিছু বিষয় খেয়াল রাখবেন-

১. জাভাস্ক্রিপ্ট কেস সেন্সেটিভঃ

জাভাস্ক্রিপ্ট কেস সেন্সেটিভ তাই ভেরিয়েবল, অবজেক্ট, ফাংশান তৈরির সময় সতর্ক থাকুন। যেমন- "myfunction" ও "myFunction" দুটি ফাংশান আলাদা অর্থ বহন করে। তেমনি "myVar" ও "myvar" দুটি ভেরিয়েবলও আলাদা অর্থ বহন করে।

২. হোয়াইট স্পেসঃ

জাভাস্ক্রিপ্ট হোয়াইট স্পেস, ট্যাব ও নুতন লাইনকে উপেক্ষা করে। আপনি ইচ্ছামত হোয়াইট স্পেস ব্যবহার করতে পারেন কোডিং-এ কোন ভুল হবে না। যেমন-

```
var name="Hege";  
var name = "Hege";
```

এদের মাঝে কোন পার্থক্য নেই।

৩. কোডলাইনকে ব্রেক করাঃ

কোড লিখতে যদি এক লাইনে না ধরে তবে একটা ব্যাকস্লাশ("\") দিয়ে পরের লাইন থেকে লিখা শুরু করা যেতে পারে। যেমন-

```
document.write("Hello \  
World!");
```


আউটপুট এক লাইনেই দেখা যাবে।

তবে নিচের মত করে লাইন ব্রেক করতে পারবেন না-

```
document.write \  
("Hello World!");
```

৫.সেমিকোলনঃ

সাধারণ স্টেটমেন্টগুলো জাভাস্ক্রিপ্টে সেমিকোলন দ্বারা আলাদা করা হয়। কিন্তু আপনি যদি স্টেটমেন্টগুলো আলাদা আলাদা লাইনে লিখেন তবে সেমিকোলন দরকার হবে না। যেমন-

```
<script language="javascript" type="text/javascript">  
<!--  
  var1 = 10  
  var2 = 20  
  //-->  
</script>
```

কিন্তু স্টেটমেন্টগুলো একই লাইনে লিখলে সেমিকোলন দরকার হবে। যেমন-

```
<script language="javascript" type="text/javascript">  
<!--  
  var1 = 10; var2 = 20;  
  //-->  
</script>
```

নোটঃ ভালো প্রোগ্রামিং-এর জন্য সেমিকোলন ব্যবহার করা দরকার।

৪. জাভাস্ক্রিপ্টে স্পেসাল ক্যারেক্টার

জাভাস্ক্রিপ্টে ব্যাকস্লাশ (\) ব্যবহার করে স্পেসাল ক্যারেক্টার যেমন-*apostrophes, new lines, quotes*, এবং অন্যান্য স্পেসাল ক্যারেক্টার টেক্সট স্ট্রিং এর মাঝে ইনসার্ট করা হয়। জাভাস্ক্রিপ্টে স্ট্রিং সিঙ্গেল বা ডাবল কোটেশান দিয়ে শুরু বা শেষ হয়। যেমন-

```
var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);
```

আউটপুটঃ

We are the so-called "Vikings" from the north.

এই ডিকলারেশনকে আমরা যদি নিচের মত লিখি তবে আউটপুট দেখা যাবে না-

```
var txt="We are the so-called "Vikings" from the north.";
document.write(txt);
```

তাহলে আমরা বুঝতে পারলাম কোটেশান দিতে হলে তার আগে একটি ব্যাকস্লাস সাইন দিতে হবে।

ব্যাকস্লাশ দিয়ে আর যে কাজগুলো করা যায়-

কোড	আউটপুট
\'	একক কোটেশান
\"	ডাবল কোটেশান
\\	ব্যাকস্লাস
\n	নুতন লাইন
\r	ক্যারেজ রিটান
\t	ট্যাব
\b	ব্যাকস্পেস
\f	ফর্ম ফিড

অধ্যায়ঃ আট- জাভাস্ক্রিপ্ট কন্ডিশনাল (শর্তবাচক) স্টেটমেন্ট

জাভাস্ক্রিপ্ট কন্ডিশনাল (শর্তবাচক) স্টেটমেন্টঃ

বিভিন্ন কন্ডিশানের উপর ভিত্তি করে বিভিন্ন কাজ সম্পাদন করতে কন্ডিশনাল স্টেটমেন্ট ব্যবহার করা হয়। কন্ডিশনাল স্টেটমেন্টে কোন শর্ত পূরণ হলেই কেবল সেই স্টেটমেন্টে প্রদত্ত নির্দেশ পালিত হয়। যেমন- কোন ব্লগ সাইটের রেজিস্ট্রেশনকৃত ইউজাররাই ব্লগ লিখতে ও মন্তব্য করতে পারবে। এখানে কন্ডিশান **Satisfy** করলেই কেবল এই কাজ করা সম্ভব। ওয়েব সাইটকে প্রানবন্ত,আর্কষণীয় এককথায় ডায়নামিক করতে অবশ্যই কন্ডিশনাল স্টেটমেন্ট সম্বন্ধে ভালো জ্ঞান থাকতে হবে।

জাভাস্ক্রিপ্টে নিম্নলিখিত কন্ডিশনাল স্টেটমেন্টগুলো বিদ্যমানঃ

১.if statement - একটি মাত্র শর্তযুক্ত কোন ব্লক এক্সিকিউট করতে এই স্টেটমেন্ট ব্যবহার করা হয়। এই স্টেটমেন্ট ব্যবহার করা হয় যদি শুধুমাত্র একটি কন্ডিশান সত্য হয়।

২.if...else statement - কন্ডিশান সত্য হলে কিছু কোড এক্সিকিউট হয় এবং কন্ডিশান মিথ্যা হলে অন্য কিছু কোড এক্সিকিউট হবে। এক্ষেত্রে এই স্টেটমেন্ট ব্যবহার করা হয়।

৩.if...else if....else statement - এই স্টেটমেন্ট ব্যবহার করা হয় যদি একাধিক ব্লকের কোড এক্সিকিউট করতে হয় যেখানে প্রতিটা ব্লকে আলাদা আলাদা শর্ত দেওয়া থাকে।

৪.switch statement - এই স্টেটমেন্টের কাজ **if...else if....else statement** এর মতোই।

নিচে প্রতিটি স্টেটমেন্ট নিয়ে বিস্তারিত আলোচনা করা হল-

১.If Statement:

If স্টেটমেন্ট তখনই ব্যবহার করা হয় যখন একটি শর্ত পূরণ হলেই কেবল কিছু কোড এক্সিকিউট হবে। ভেরিয়েবল এবং কিছু প্রকারের ডাটার উপর ভিত্তি করে "**If Statement**" এর সাহায্যে সিদ্ধান্ত নেয়া হয়। যেমন- আপনার আপনার বয়স যদি ১৮ হয় তবেই আপনি ভোট দিতে পারবেন। এখানে শুধুমাত্র একটি কন্ডিশান সত্য হলেই আউটপুট পাওয়া যাবে।

সিনট্যাক্সঃ

```
if (কন্ডিশান)
{
    কন্ডিশান সত্য হলে এই ব্লকের মাঝের স্টেটমেন্টগুলো এক্সিকিউট হবে।
}
```

If Statement এর প্রধান দুটি অংশ রয়েছে তার একটি কন্ডিশনাল স্টেটমেন্ট এবং অপরটি হল নির্দিষ্ট কোড যা কার্যে পরিনত হবে। কন্ডিশনাল স্টেটমেন্ট হল এমন একটি স্টেটমেন্ট যা সত্য, মিথ্যা যাচাই করে। বেশির ভাগক্ষেত্রে কন্ডিশনাল স্টেটমেন্ট দিয়ে কোন কিছু চেক করতে ব্যবহৃত হয়। যেমন-

```
<script type="text/javascript">
var myColor = "Blue";

if (myColor == "Blue") {
    document.write("Just like the sky!");
}
</script>
```

কোড বিশ্লেষণঃ

প্রথমে আমরা "myColor" নামে একটি ভেরিয়েবল ডিকলার করেছি এবং তার মান দিয়েছি "Blue"। এরপর আমরা **If Statement** ব্যবহার করে চেক করেছি ভেরিয়েবলের মান "Blue" কিনা? এরপর আমরা **document.write** নামে একটি ফাংশান ব্যবহার করেছি এবং তার ভেলু দিয়েছি ("Just like the sky!")। এক্ষেত্রে যদি কন্ডিশন সত্য হয় তবে আউটপুটে **Just like the sky!** লেখাটি দেখা যাবে। কন্ডিশন সত্য তাই **Just like the sky!** লেখাটি আউটপুটে দেখা যাবে।

২. If...else Statement:

If স্টেটমেন্টের উদাহরণে আমরা দেখলাম যে, কেবল কন্ডিশন সত্য হলেই **If** স্টেটমেন্টের ব্লকটি এক্সিকিউট হয় এবং আউটপুট পাওয়া যায়। কিন্তু আমরা যদি চাই কন্ডিশন মিথ্যা হলেও আউটপুটে কোন কিছু দেখাবে তবে **IfElse** স্টেটমেন্ট ব্যবহার করতে হবে। **IfElse** স্টেটমেন্ট হল **If** স্টেটমেন্টের বর্ধিত অংশ যাকে **Else clause** বলা হয়। **Else clause** টি কাজ করে যখন কন্ডিশনাল স্টেটমেন্ট টি মিথ্যা হয়। এককথায় বলা যায় **if...else** স্টেটমেন্টে একটি ব্লক এক্সিকিউট হবে যদি কন্ডিশন সত্য হয় এবং কন্ডিশন মিথ্যা হলে অন্য ব্লক এক্সিকিউট হবে।

সিনট্যাক্সঃ

```
if (কন্ডিশন)
{
    কন্ডিশন সত্য হলে এই ব্লকের মাবের স্টেটমেন্টগুলো এক্সিকিউট হবে।
}
Else
{
    আর যদি কন্ডিশন মিথ্যা হয় তবে এই ব্লকের মাবের স্টেটমেন্টগুলো এক্সিকিউট হবে।
}
```

উদাহরণঃ

```
<script type="text/javascript">
var myColor = "Red";

if (myColor == "Blue") {
    document.write("Just like the sky!");
}
else {
    document.write("Didn't pick blue color?");
}
</script>
```

উপরের উদাহরণে "myColor" ভেরিয়েবলের মান যদি "Blue" হয় তবে Just like the sky! লেখাটি দেখাবে আর যদি "Blue" না হয় তবে Didn't pick blue color? লেখাটি দেখাবে।

৩. If...else if...else Statement

উপরের দুটি স্টেটমেন্টের চেয়ে If ...Else If ... Else স্টেটমেন্ট শক্তিশালী। কারণ এই স্টেটমেন্ট বিভিন্ন কন্ডিশানের উপর ভিত্তি করে বিভিন্ন আউটপুট দিতে পারে। এই স্টেটমেন্ট ব্যবহার করা হয় যদি একাধিক ব্লকের কোড এক্সিকিউট করতে হয় যেখানে প্রতিটা ব্লকে আলাদা আলাদা শর্ত দেওয়া থাকে।

সিনট্যাক্সঃ

```
if (কন্ডিশান-১)
{
    কন্ডিশান-১ সত্য হলে এই ব্লকের কোড এক্সিকিউট হবে।
}
else if (কন্ডিশান-২)
{
    কন্ডিশান-২ সত্য হলে এই ব্লকের কোড এক্সিকিউট হবে।
}
else if (কন্ডিশান-৩)
{
    কন্ডিশান-৩ সত্য হলে এই ব্লকের কোড এক্সিকিউট হবে।
}
Else
{
    উপরের কোন কন্ডিশানই সত্য না হলে এই ব্লকের কোড এক্সিকিউট হবে।
}
}
```

উদাহরণঃ

```
<script type="text/javascript">
var myColor = "Red";

if (myColor == "Blue") {
    document.write("Just like the sky!");
}
else if (myColor = "Red") {
    document.write("Just like the sun!");
}
else if (myColor = "Green")
{
    document.write("Just like the tree");
}
}
```

```
else {
    document.write("Suit yourself then...");
}
</script>
```

8. JavaScript Switch Statement

if...else if statements ব্যবহার করে আমরা একাধিক কন্ডিশান চেক করতে পারি এবং বিভিন্ন কন্ডিশানের উপর ভিত্তি করে বিভিন্ন আউটপুট পেতে পারি। যেমন- উপরের *if...else if* স্টেটমেন্টে ভেরিয়েবল "myColor"-এর মান যদি "Blue" হয় তবে একধরনের আউটপুট পাওয়া যাবে আবার যদি ভেরিয়েবলের মান "Red" হয় তবে অন্য আউটপুট পাওয়া যাবে। এই একই ধরনের কাজ করার জন্য জাভাস্ক্রিপ্টে আরেকটি স্টেটমেন্ট ব্যবহার করে হয়, যার নাম **Switch statement**। প্রোগ্রামে *if...else if statement* এবং **Switch statement**-এর কাজ করার ধরন একরকম হলেও, *if...else if statement*-এর কন্ডিশান হিসেবে সাধারণত লজিক্যাল বা রিলেশনাল এক্সপ্রেশন ব্যবহার করা হয়। কিন্তু **Switch statement**-এ কোন **conditional expression** ব্যবহার করা হয় না, বরং একটা ভেরিয়েবল ব্যবহার করা হয়, যার মানের উপর নির্ভর করে কোন **case statement**-টা কাজ করবে। বড় প্রোগ্রামের ক্ষেত্রে যখন কন্ডিশানের সংখ্যা অনেক বেশি হয় তখন **Switch statement** ব্যবহার করা সুবিধাজনক।

সিনট্যাক্সঃ

```
switch(expression)
```

```
{
```

```
case constant 1:
```

```
এই ব্লকের স্টেটমেন্টগুলো এক্সিকিউট হবে।
```

```
break;
```

```
case constant 2:
```

```
এই ব্লকের স্টেটমেন্টগুলো এক্সিকিউট হবে।
```

```
break;
```

```
.....
```

```
case constant n:
```

```
এই ব্লকের স্টেটমেন্টগুলো এক্সিকিউট হবে।
```

```
break;
default:
  case 1 এবং case 2 থেকে যদি n ভিন্ন হয় তবে এই ব্লক এক্সিকিউট হবে।
```

বিশ্লেষণঃ

১. প্রথমে **switch** কীওয়ার্ড ব্যবহার করা হয়েছে।

২. এরপর প্রথম বন্ধনীর মাঝে একটি একক এক্সপ্রেশান ব্যবহার করা হয়েছে যা সাধারণত একটি ভেরিয়েবল হয়ে থাকে, যার মানের উপর নির্ভর করে কোন **case statement**-টা কাজ করবে।

৩. তারপর **switch**-এর **expression**-এ ব্যবহৃত ভেরিয়েবলের মানের সাথে **case**-এর **constant**-এর মানের তুলনা করা হয় এবং যার সাথে মিল পাওয়া যাবে প্রোগ্রামে সেই **case** সংশ্লিষ্ট স্টেটমেন্ট কাজ করবে।

৪. **case**-এর সাথে যেসব **simple** বা **compound statement** ব্যবহার করা হয়, সেসব স্টেটমেন্টের পর **break** কে শেষ **statement** হিসেবে ব্যবহার করা হয়। এখানে **break**; ব্যবহার করার অর্থ হল থামুন। অর্থাৎ শর্ত পূরণ হলে আর কন্ডিশান চেক করতে হবে না তখন ঐ ব্লকটিকেই এক্সিকিউট করে আউটপুট দেখবে।

৫. আর যদি কোন শর্তই পূরণ না হয় তবে **default**: সংশ্লিষ্ট স্টেটমেন্ট কাজ করবে। **default**: আসলে **if...else if statement**-এর **else**-এর মত কাজ করে।

উদাহরণঃ **if...else if statement**-এর উদাহরণটি আমরা এখানে **switch statement** ব্যবহার করে লিখেছি।

```
<script type="text/javascript">
```

```
<!--
```

```
var myColor = "Red";
```

```
switch (myColor)
```



```
{
case "Blue":
    document.write("Just like the sky!");
    break
case "Red":
    document.write("Just like shiraz!");
    break
default:
    document.write("Suit yourself then...");
}
//-->
</script>
```

আউটপুটঃ

Just like shiraz!

if...else if statements ব্যবহার করে আমরা একাধিক কন্ডিশান চেক করতে পারি কিন্তু সব সময় এটা সঠিক সমাধান না, বিশেষ করে যখন প্রতিটা ব্লক একটি একক ভেরিয়েবলের উপর নির্ভর করে। এক্ষেত্রে সঠিক সমাধান হল **switch statement**

অধ্যায়ঃ নয়- জাভাস্ক্রিপ্ট লুপ/ পুনঃরাবৃত্তি স্টেটমেন্ট

লুপ (পুনঃরাবৃত্তি) স্টেটমেন্টঃ

প্রোগ্রাম লেখার সময় কখনও কখনও একই কাজ করার জন্য একই ব্লক বার বার লিখতে হয়। একই ধরনের লাইনগুলো এভাবে বার বার না লিখে লুপের মাধ্যমে এদের সহজভাবে লেখা যায়। যেমন- আপনি চাচ্ছেন ১ থেকে ১০০ পর্যন্ত শুধুমাত্র জোড় সংখ্যাগুলো খুঁজে বের করতে চান, এক্ষেত্রে বার বার ব্লক না লিখে একটা কন্ডিশান দেওয়া যেতে পারে যাতে করে ১ থেকে ১০০ পর্যন্ত জোড় সংখ্যাগুলো পাওয়া যায়। অর্থাৎ ঘুরে ফিরে একই কাজ একটা নির্দিষ্ট সময় ধরে করতে চাইলে জাভাস্ক্রিপ্ট লুপ স্টেটমেন্ট ব্যবহার করা হয়। এই পদ্ধতিতে প্রোগ্রাম লিখলে প্রোগ্রামে লাইন সংখ্যা কমে যায় এবং প্রোগ্রামের দক্ষতা বৃদ্ধি পায়। যতক্ষণ পর্যন্ত কন্ডিশান সত্য থাকে ততক্ষণ পর্যন্ত লুপ চলতে থাকে এবং নির্দিষ্ট ব্লককে এক্সিকিউট করে। জাভাস্ক্রিপ্ট চার ধরনের লুপ বিদ্যমান-

1. While loop
2. Do.....while loop
3. For loop
4. For.....in loop

১. While Loop

while লুপের মাধ্যমে কোন কাজ বারবার করতে পারেন যখন আপনার কন্ডিশনাল স্টেটমেন্টটি সত্য হবে। কন্ডিশন সত্য হলেই কেবল **while** লুপ একটি ব্লককে এক্সিকিউট করে।

সিনট্যাক্স:

```
while (condition)
{
statement;
}
```

বিশ্লেষণ:

১ *while loop* এর কোড কার্যে পরিনত হওয়ার জন্য অবশ্যই কন্ডিশনাল স্টেটমেন্ট সত্য হতে হবে।

২ *while loop* এর কোড বাকানো ব্রাকেট "{ }" ধারণ করে যা কাজে পরিনত হবে যদি কন্ডিশনাল স্টেটমেন্টটি সত্য হয়।

৩. যখন *while* লুপের কাজ শুরু হয়, তখন জাভাস্ক্রিপ্ট চেক করে দেখে যে **conditional** স্টেটমেন্টটি সত্য কিনা। যদি সত্য হয় তবে বাকানো ব্রাকেট "{ }" এর মধ্যের কোডগুলো এক্সিকিউট হয়।

৪. এরপর প্রোগ্রাম আবার **conditional** স্টেটমেন্টে ফিরে গিয়ে কন্ডিশনাল চেক করে, যদি কন্ডিশনাল সত্য হয় তবে বাকানো ব্রাকেট "{ }" এর মধ্যের কোডগুলো আবার এক্সিকিউট হয়।

৫. এভাবে লুপটি চলতে থাকবে যতক্ষণ পর্যন্ত না কন্ডিশনাল মিথ্যা হবে। কন্ডিশনাল মিথ্যা হলেই প্রোগ্রাম লুপ থেকে বের হয়ে আসবে। কিন্তু যদি **condition statement** টি সবসময় সত্য হয় তবে আপনি কখনও *while loop* হতে বের হয়ে আসতে পারবেন না। তাই *while loop* ব্যবহারের সময় সতর্ক হওয়া উচিত।

উদাহরণ:

1. `<script type="text/javascript">`
2. `var count = 0;`
3. `document.write("Starting Loop" + "
");`
4. `while (count < 10){`
5. `document.write("Current Count : " + count + "
");`
6. `count++;`
7. `}`
8. `document.write("While loop is finished!");`
9. `</script>`

আউটপুটঃ

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
While loop is finished!
```

কোড বিশ্লেষণঃ

১. **count** নামে একটি ভেরিয়েবল ডিকলার করা হয়েছে যার মান ০.

২. **while** লুপের মাঝে প্রথমে কন্ডিশান চেক করা হবে। এক্ষেত্রে প্রথমে **count** এর মান ০ যা, ১০ থেকে ছোট। সুতরাং কন্ডিশান সত্য। কন্ডিশান সত্য বলে ৫ নং লাইনে এসে তার ভেলু প্রিন্ট হবে।

৩. ৬ নং লাইনে এসে **count** ভেরিয়েবলের মান ১ বৃদ্ধি হবে। প্রোগ্রাম আবার ৪ নং লাইনে এসে কন্ডিশান চেক করবে। কন্ডিশান সত্য। তাই আবার ১ প্রিন্ট হবে।

৪. আবার ৬ নং লাইনে এসে **count** ভেরিয়েবলের মান ১ বৃদ্ধি হয়ে ২ হবে। প্রোগ্রাম আবার ৪ নং লাইনে এসে কন্ডিশান চেক করবে। কন্ডিশান সত্য। তাই আবার ২ প্রিন্ট হবে। এভাবে কন্ডিশান যতক্ষণ পর্যন্ত সঠিক হবে ততক্ষণ লুপ চলতে থাকবে। কন্ডিশান মিথ্যা হলে প্রোগ্রাম লুপ থেকে বের হয়ে আসবে।

২. do...while Loop

do...while লুপ **while** লুপের মতোই, শুধু পার্থক্য হল **while** লুপে প্রথমে কন্ডিশান চেক করা হয় আর **do...while** লুপে একদম লুপের শেষে কন্ডিশান চেক করা হয়। অর্থাৎ **do...while** লুপে প্রথমে কোন কন্ডিশান চেক না করেই একবার কোড এক্সিকিউট হবে এবং আউটপুট দেখাবে, এমনকি কন্ডিশান মিথ্যা হলেও।

সিনট্যাক্সঃ

```
do
{
statement;
}
while (condition);
```

নোটঃ do...while লুপের শেষে সেমিকোলন(";") ব্যবহার করা হয়।

উদাহরণঃ

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

আউটপুটঃ

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Loop stopped!
```

৩. for Loop

জাভাস্ক্রিপ্ট ফর লুপ অন্যান্য প্রোগ্রামিং ল্যাংগুয়েজের ফর লুপের মত। প্রোগ্রামে এক বা একাধিক স্টেটমেন্ট একটা নির্দিষ্ট বার পর্যন্ত কাজ করতে for loop ব্যবহার করা হয়। for loop তিনটি অংশ নিয়ে গঠিত-

সিনট্যাক্সঃ

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

ব্যাখ্যা:

- **Initialization** অংশে প্রথমে ভেরিয়েবলের (**counter**) মান ইনিশিয়ালাইজ করে দিতে হবে। লুপ শুরু হবার আগেই **initialization statement** এক্সিকিউট হবে।
- কন্ডিশান সত্য না মিথ্যা **test statement** তা চেক করবে। যদি কন্ডিশান সত্য হয় তবে লুপের মাঝের কোড এক্সিকিউট হবে, অন্যথায় প্রোগ্রাম লুপ থেকে বের হয়ে আসবে।
- **iteration statement** অংশে আপনি **counter** অর্থাৎ ভেরিয়েবলের মান **increase** অথবা **decrease** করতে পারবেন।

নোটঃ for লুপের এই তিনটি অংশ পরস্পর সেমিকোলন দ্বারা পৃথক থাকবে।

উদাহরণঃ

```
<script type="text/javascript">
var count;
document.write("Starting Loop" + "<br />");
for(count = 0; count < 10; count++)
{
    document.write("Current Count : " + count );
    document.write("<br />");
}
document.write("Loop stopped!");
</script>
```

আউটপুটঃ

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
```

```
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
```

8. for...in loop

জাভাস্ক্রিপ্টে **for...in** নামে আরেকটি লুপ রয়েছে। জাভাস্ক্রিপ্টে অবজেক্ট প্রপার্টিতে এই লুপ ব্যবহার করা হয়।

সিনট্যাক্সঃ

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

নোটঃ লুপের মাঝে যে ব্লক আছে সেখানে কোডগুলো প্রতিটা প্রপার্টির জন্য একবার করে কাউন্ট হয়। প্রতিটা পুনরাবৃত্তিতে অবজেক্ট থেকে একটা করে প্রপার্টি *variablename* —এ এসাইন হবে এবং এই লুপটা চলতে থাকবে যতক্ষণ পর্যন্ত না অবজেক্টের প্রপার্টি শেষ হবে।

উদাহরণঃ

```
<html>
<body>
<script type="text/javascript">

var person={fname:"John",lname:"Doe",age:25};

for (x in person)
{
document.write(person[x] + " ");
}
```

```
</script>
</body>
</html>
```

আউটপুটঃ

```
John Doe 25
```

জাভাস্ক্রিপ্ট লুপ কন্ট্রোলঃ

জাভাস্ক্রিপ্ট লুপ কন্ট্রোল করতে **break** ও **continue** স্টেটমেন্ট ব্যবহার করা হয়। যদি এমন হয় যে কোন প্রোগ্রামে লুপের শেষ পর্যন্ত না পৌঁছে লুপের মাঝ থেকেই বের হয়ে আসার প্রয়োজন হয় বা লুপের কোন একটা ব্লককে বাদ দিয়ে পরবর্তী ব্লক থেকে কাজ শুরু করতে হয় এমন হলে এই স্টেটমেন্ট দুটি ব্যবহারের প্রয়োজন হয়।

ব্রেক স্টেটমেন্ট (break Statement)

switch নিয়ে আলোচনা করার সময় আমরা break স্টেটমেন্ট ব্যবহার করেছিলাম। তবে break স্টেটমেন্ট শুধু switch স্টেটমেন্টেই নয় for, while, do...while লুপেও এদের ব্যবহার করা যায়। মূলত কোন লুপের কন্ডিশন এর মান ০ হওয়ার পূর্বেই লুপ থেকে বের হওয়ার জন্য লুপের মাঝে break স্টেটমেন্ট ব্যবহার করা হয়।

উদাহরণঃ

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
{
if (i==3)
{
break;
}
}
```

```
document.write("The number is " + i);
document.write("<br />");
}
</script>
<p>Explanation: The loop will break when i=3.</p>
</body>
</html>
```

আউটপুটঃ

```
The number is 0
The number is 1
The number is 2
```

Explanation: The loop will break when i=3.

কোড বিশ্লেষণঃ

উপরের প্রোগ্রামে for লুপ ব্যবহার করা হয়েছে ১ থেকে ১০ পর্যন্ত প্রিন্ট করার জন্য। কিন্তু মাঝখানে একটি কন্ডিশন দেওয়া হয়েছে [if (i==3)] এবং break স্টেটমেন্ট ব্যবহার করা হয়েছে। তাই যখন ভেরিয়েবলের মান ৩ হবে তখন লুপ আর কাজ করবে না।

কনটিনিউ (continue Statement):

এই স্টেটমেন্টের কাজ হল যে কন্ডিশনের জন্য **continue statement** ব্যবহার করা হয় সেই ব্লককে স্কেপ করে পরবর্তী কন্ডিশনের জন্য লুপের কাজ করা। অন্যভাবে বলা যায় **continue statement** বর্তমান লুপকে ব্রেক করে পরবর্তী ভেলুগুলোর জন্য লুপকে সচল রাখে। **continue statement** হল **break Statement**-এর বিপরীত অর্থাৎ এটি কোন কাজ চালু রাখতে বলে।

উদাহরণঃ

```
<html>

<body>

<script type="text/javascript">

var i=0;

for (i=0;i<=10;i++)
```



```
{  
if (i==3)  
  
  {  
    continue;  
  }  
  
document.write("The number is " + i);  
  
document.write("<br />");  
  
}  
  
</script>
```

<p>Explanation: The loop will break the current loop and continue with the next value when i=3.</p>

</body>

</html>

আউটপুটঃ

The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10

Explanation: The loop will break the current loop and continue with the next value when i=3.

অধ্যায়ঃ দশ- জাভাস্ক্রিপ্ট অ্যারে

জাভাস্ক্রিপ্ট অ্যারে কি?

অ্যারে হল বিশেষ ধরনের ভেরিয়েবল যা একই কাজে ব্যবহৃত একই ধরনের ডাটাকে একটি সিঙ্গেল ভেরিয়েবলের মাধ্যমে ধারণ করতে পারে। অ্যারে অবজেক্ট গুলোকে তাদের সাবস্ক্রিপ্টের মাধ্যমে এসেস করা যায়। অ্যারের প্রথম এলিমেন্টের পজিশান জিরো (০), দ্বিতীয় এলিমেন্টের পজিশান (১), তৃতীয় এলিমেন্টের পজিশান (২), এভাবে অন্যগুলো হবে। যেমন- আপনি কতগুলো প্রোডাক্টের নাম সিঙ্গেল ভেরিয়েবল হিসেবে নিচের মত ডিকলার করতে পারেন-

```
var product1="pen";  
var product2="book";  
var product3="marker";
```

কিন্তু আপনার প্রোডাক্ট সংখ্যা যদি ৪০০ হয় তাহলে আপনি কি এভাবে ৪০০ভেরিয়েবল ডিকলার করবেন? না, আপনাকে এ কাজটি করতে হবে না। এর সঠিক সমাধান হল অ্যারে। অথবা যদি এমন হয় ৪০০ প্রোডাক্ট থেকে নির্দিষ্ট কোন প্রোডাক্ট বাছাই করতে হবে এক্ষেত্রে আমরা অ্যারে ব্যবহার করতে পারি। নিম্নে উদাহরণের সাহায্যে ব্যাখ্যা করা হল কিভাবে অ্যারে তৈরি করা হয়-

জাভাস্ক্রিপ্ট অ্যারে তৈরি করাঃ

প্রায় সকল ল্যাঙ্গুয়েজের অ্যারে তৈরির সিনট্যাক্স একই। জাভাস্ক্রিপ্টের অ্যারে তৈরির সিনট্যাক্স নিম্নরূপ-

সিনট্যাক্সঃ

```
var array_name = new Array(number_of_elements) //অ্যারে তৈরিতে new কীওয়ার্ড ব্যবহার করতে হয়।
```

এরপর অ্যারেতে ভেলু এসাইন করে দিতে হবে।

```
array_name[0] = "Array element"
```

এবার আমরা উপরের উদাহরণটি থেকে অ্যারে তৈরি করব।

তিনটি ধাপে অ্যারে তৈরি করা হয়-

জাভাস্ক্রিপ্ট অ্যারে তৈরির সময় প্রথমে অ্যারে অবজেক্টকে ভেরিয়েবলের নাম হিসেবে এসাইন করে দিতে হয়।

১. নিচের কোড myProduct নামে একটি অ্যারে তৈরি করবে-

```
var myProduct = new Array(400); // regular array (add an optional integer
myProduct [0]="pen"; // argument to control array's size)
myProduct [1]="book";
myProduct [2]="marker";
```

```
২. var myProduct=new Array("pen","book","marker"); // condensed array
```

```
৩. var myCars=["pen","book","marker"]; // literal array
```

নোটঃ যদি আপনি অ্যারের মাঝে **numbers** বা **true/false** ভেলু ইনসার্ট করেন তবে ভেরিয়েবল টাইপ নাম্বার অথবা বুলিয়ান হবে, স্ট্রিং হবে না।

উদাহরণঃ

```
<!DOCTYPE html>

<html>

<body>

<script>

var i;

var mycars = new Array();

mycars[0] = "Saab";

mycars[1] = "Volvo";

mycars[2] = "BMW";

for (i=0;i<mycars.length;i++)

{

document.write(mycars[i] + "<br>");

}

</script>
```

</body>

</html>

উদাহরণঃ

```
<script type="text/javascript">
var myArray = new Array();
  myArray[0] = "Football";
myArray[1] = "Baseball";
myArray[2] = "Cricket";
document.write(myArray[0] + myArray[1] + myArray[2]);
</script>
```

প্রদর্শন:

FootballBaseballCricket

লক্ষ্য আপনি ব্রাকেটে তেলুর **position** ঠিক করে দেয়ার ফলে আপনার ইচ্ছা অনুযায়ী ভেলু বের করতে পেরেছেন।

জাভাস্ক্রিপ্টে অ্যারে একসেস করাঃ

অ্যারে থেকে নির্দিষ্ট যে কোন একটি এলিমেন্ট এসেস করতে প্রথমে অ্যারের নাম ও তারপর তার ইনডেক্স নাম্বার (জাভাস্ক্রিপ্টে ইনডেক্স নাম্বার ০ থেকে শুরু হয়) দিতে হবে। যেমন- উপরের উদাহরণের **book** লেখাটি যদি দেখাতে চাই তবে নিচের মত লিখতে হবে-

```
document.write(myProduct [1]);
```

অ্যাসোসিয়েটিভ অ্যারেঃ

সাধারণ অ্যারেতে ইনডেক্স হয় সংখ্যাগতভাবে কিন্তু অ্যাসোসিয়েটিভ অ্যারেতে ইনডেক্স করা হয় "নাম(name)" কে "কী" হিসেবে ধরে। এর সুবিধা হল "কী" হল অর্থপূর্ণ, যাকে সহজে অ্যারে এলিমেন্ট রেফারেন্স হিসেবে আনা যায়। নিম্নে দেখানো হয়েছে কীভাবে অ্যাসোসিয়েটিভ অ্যারে তৈরি করা হয়-

ধরুন `student` নামে একটা অবজেক্ট রয়েছে যার তিনটা প্রপার্টি রয়েছে `properties name, class, rollno` । এদের নিচের মত করে ডিফাইন করা যায়-

```
student.name = "David Rayy"  
student.class = "V"  
student.rollno = 1
```

জাভাস্ক্রিপ্টে প্রপার্টি ও অ্যারে প্রায় একই ধরণের। আসলে তাদের ইন্টারফেস আলাদা কিন্তু ডাটা স্ট্রাকচার একই। `student` অবজেক্টের প্রপার্টিগুলো নিচের মত করে এক্সেস করা যায়-

```
student["name"] = "David Rayy"  
student["class"] = "V"  
student["rollno"] = 1
```

উপরে লক্ষ্য করুন আমরা প্রপার্টিকে কী হিসেবে ধরেছি অর্থাৎ এখানে এখন ইনডেক্স নাম্বারটা স্ট্রিং। আর এটাই হল অ্যাসোসিয়েটিভ অ্যারে।

নিচের উদাহরণে `object name and properties` গুলো আরগুমেন্ট হিসেবে `show_obj_property` ফাংশানে পাসড (`passed`) করা হয়েছে যেটা, `student` অবজেক্টের প্রপার্টিগুলো দেখাচ্ছে।

```
function show_obj_property(obj, obj_name)  
{  
  var output = ""  
  for (var i in obj)  
    result += obj_name + "." + i + " = " + obj[i] + "\n";  
  return output;  
}
```

আউটপুটঃ

```
student.name = David Rayy  
student.class = V  
student.rollno = 1
```

[একটা উদাহরণ দেখুন-](#)

```
<html>  
<head>  
<title>JavaScript Arrays</title>  
<script type="text/javascript">
```

```
var BEATLES = [];  
BEATLES["singer1"] = "Paul";  
BEATLES["singer2"] = "John";  
BEATLES["guitarist"] = "George";  
BEATLES["drummer"] = "Ringo";  
</script>  
</head>  
<body>  
<p align="center">  
<script type="text/javascript">  
  document.write(BEATLES["singer1"]);  
  document.write(BEATLES["singer2"]);  
  document.write(BEATLES["guitarist"]);  
  document.write(BEATLES["drummer"]);  
</script>  
</p>  
</body>  
</html>
```

অধ্যায়ঃ এগার-জাভাস্ক্রিপ্ট ফাংশন

যেকোন প্রোগ্রামিং ল্যাংগুয়েজ শিখতে যান ফাংশন হচ্ছে তার মূল জিনিসগুলির মধ্যে একটা। সব ল্যাংগুয়েজেই ফাংশন আছে আর সবখানেই ফাংশনের মূল কনসেপ্ট টা একই।

জাভাস্ক্রিপ্ট ফাংশান কি?

ফাংশন আর কিছুই না শুধু একটা কোডব্লক কে নাম দেয়া। পরে কোডের যেকোন জায়গায় সেই নাম ধরে ডাকলে কোডব্লকটি এক্সিকিউট হবে। যখন ওয়েব পেজ লোড হয় তখন ব্রাউজার দ্বারা স্ক্রিপ্ট এক্সিকিউট করতে স্ক্রিপ্টকে ফাংশানের মধ্য রাখতে হবে। ফাংশানের কোডগুলো ইভেন্ট দ্বারা এক্সিকিউট হয় এবং ফাংশানকে কল করা যায়। আপনি কোডের যে কোন জায়গা থেকে ফাংশানকে কল করতে পারেন(অথবা অন্য কোন পেজে ফাংশান কল করতে পারেন যদি ফাংশান এম্বেডেড করা থাকে একটি এক্সটারনাল .js ফাইল হিসেবে)। ফাংশানকে হেড বা বডি উভয় সেকশানেই ডিফাইন করা যেতে পারে। যদি ফাংশানকে কল করার আগেই স্ক্রিপ্ট ব্রাউজার দ্বারা লোড হয় তবে ফাংশানকে <head> হেড সেকশানে রাখা উত্তম। আমরা ফাংশানকে কল করতে পারি, ফাংশানকে একটি ভেরিয়েবলে স্টোর করা যায়, ফাংশানকে মডিফাই করা যায় ইত্যাদি কাজ ফাংশান দিয়ে করা যায়। **You can divide your big programme in a number of small and manageable functions.**

জাভাস্ক্রিপ্টে দুই ধরনের ফাংশান রয়েছে-

১. বিল্ট-ইন ফাংশান

- জাভাস্ক্রিপ্ট অ্যারে ফাংশান (JavaScript Array Function)
- জাভাস্ক্রিপ্ট বুলিয়ান ফাংশান (JavaScript Boolean Function)

- জাভাস্ক্রিপ্ট ম্যাথ ফাংশান (JavaScript Math Function)
- জাভাস্ক্রিপ্ট ডেট ফাংশান (JavaScript Date Function)
- জাভাস্ক্রিপ্ট নম্বর ফাংশান (JavaScript Number Function)
- জাভাস্ক্রিপ্ট স্ট্রিং ফাংশান (JavaScript String Function)
- জাভাস্ক্রিপ্ট রেগুলার এক্সপ্রেসান ফাংশান (JavaScript RegExp Function)

২. ইউজার ডিফাইন ফাংশান

ইউজার ডিফাইন ফাংশানঃ

জাভাস্ক্রিপ্টের একটি বড় সুবিধা হল নিজের মত করে ফাংশান তৈরি করা এবং প্রয়োজনে তা নিজের মত করে ব্যবহার করা। আপনি ফাংশান তৈরিতে যত দক্ষ হবেন জাভাস্ক্রিপ্টের সুবিধাও তত বেশি পাবেন। যেমন- একটি অফিসে বিভিন্ন কর্মকর্তা বিভিন্ন কাজ করেন যাদেরকে একেকটি জাভাস্ক্রিপ্ট ফাংশানের সাথে তুলনা করা যায় যারা ভিন্ন ভিন্ন কাজ করে থাকে।

ফাংশান তৈরি করাঃ

সিনট্যাক্সঃ

```
<script type="text/javascript">
function functionname(var1,var2,...,varX)
{
  statements
}
</script>
```

ফাংশান ডিক্লার করতে প্রথমেই **function** কীওয়ার্ড লিখে স্পেস দিয়ে তার পর ফাংশান নাম দিতে হবে। শুরুতে **Function** শব্দটি নির্দেশ করে যে এর পরবর্তী **statement** গুলো একটি ফাংশানের অন্তর্গত। এরপর **functionname** এ বলে দেওয়া হয় এ ফাংশানটিকে আমরা কোন নামে ডাকব। জাভাস্ক্রিপ্ট **Reserved words** ছাড়া যে কোন নাম দেওয়া যেতে পারে **functionname** হিসেবে। তবে সব সময় সহজবোধ্য নাম দেওয়া উচিত। নামের পর থাকে প্রথম বন্ধনী যার মধ্য **arguments** বা প্যারামিটার উল্লেখ করা হয়। কোন ফাংশানের প্যারামিটারগুলো (**var1, var2** ইত্যাদি হল ভেরিয়েবল বা ভেলু যা ফাংশানে ইনপুট হবে)কমা দিয়ে প্রকাশ করতে হবে। এরপর থাকে দ্বিতীয় বন্ধনীর শুরু। তারপরের লাইনে কি কি কাজ করতে হবে তা **statement**-এ উল্লেখ করা হয়। শেষে দ্বিতীয় বন্ধনীর শেষ করা হয়। যে সব জাভাস্ক্রিপ্ট কোড ব্যবহার করা হবে তা অবশ্যই **{ }** এর মধ্যে দিতে হবে।

নোটঃ যদি ফাংশানে কোন প্যারামিটার না থাকে তবে অবশ্যই ফাংশানের নামের পরে প্যারান্থিসিস **()** দিতে হবে। যেমন-

```
function functionname()
{
```

এখানে কিছু কোড বা জাভাস্ক্রিপ্ট স্টেটমেন্ট থাকবে।

```
}
```

নোটঃ ফাংশনের নাম অবশ্যই লোয়ার কেস (**lowercase letter**) লেটারে লিখতে হবে, নতুবা জাভাস্ক্রিপ্ট ইরর দেখাবে। আপনাকে অবশ্যই ফাংশান কল করার সময় সর্তক থাকতে কারন যে নামে ফাংশান ডিকলার করেছেন সেই নামেই ফাংশান কল করতে হবে, যার অর্থ হল লেটারের কোন পরিবর্তন হওয়া যাবে না।

ফাংশানের উদাহরণঃ

ব্রাউজার জাভাস্ক্রিপ্টকে লাইনের পর লাইন ইন্টারপ্রেট করে। তাই কাজের সুবিধার জন্য ফাংশনকে আগেই লোড করে নিতে ফাংশনকে `<head>` ট্যাগের মধ্য রাখা উচিত। যেমন-

```
<head>
<script type="text/javascript">
function sayHello()
{
  alert("Hello there");
}
</script>
</head>
<body>
<input type="button" value="Click me!" onclick="sayHello()">
</body>
```

কোড বিশ্লেষণঃ

এখানে `sayHello()` নামে একটি ইউজার ডিফাইন ফাংশান তৈরি করা হয়েছে। এই ফাংশানের মাঝে `alert()` নামে আরেকটি বিল্ট-ইন ফাংশান ব্যবহার করা হয়েছে। `<body>` ট্যাগের মাঝে একটি বাটন তৈরি করা হয়েছে এবং তাতে `onclick` ইভেন্ট যুক্ত করা হয়েছে। ইউজার যখন ঐ বাটনে ক্লিক করবে তখন `sayHello()` ফাংশানকে কল করা হবে। এবার `sayHello()` ফাংশান স্টেটমেন্টগুলোকে এক্সিকিউট করবে।

ফাংশন কল করাঃ

নিচে একটা ছোট ফাংশন লিখেছি আর নাম দিয়েছি `popup()` এবং ইনপুট ট্যাগের ভিতর এই নাম ধরে ডাক দিয়েছি (এটাকে বলে ফাংশন কল করা)।


```

<html>
<head>
<script type="text/javascript">
function popup() {
    alert("Hello Webcoachbd")
}
</script>
</head>
<body>
<input type="button" onclick="popup()"
value="popup">
</body>
</html>

```

এখন বাটনে ক্লিক করলেই **popup()** ফাংশনটি কল হবে এবং এর ভিতর সেকেন্ড ব্রাকেটের মধ্যে থাকা কোডটুকু এক্সিকিউট হবে। **onclick** হচ্ছে ইভেন্ট।

প্রদর্শন:



যাইহোক ফাংশন লেখার সময় প্রথমে **function** এই শব্দটি এরপর ফাংশনের যেকোন নাম যেমন আমি দিয়েছি **popup()**। ফাংশনের নাম দেয়ার সময় আপনি ইচ্ছেমত যেকোন নাম দিতে পারেন। আপনি ইচ্ছে করলেই করতে পারেন তার মানে এই নয় যে আপনার এমনই করা উচিত। বরং ফাংশনের নাম দেয়ার সময় প্রাসঙ্গিক নাম দেয়া ভাল। যেমন ধরুন দুটি সংখ্যার যোগফল এর মান বের করার জন্য একটা ফাংশন লিখলেন এটার নাম হতে পারে **getAddition()**। জাভাস্ক্রিপ্টের কিছু সংরক্ষিত নাম আছে এসব ফাংশনের নাম হিসেবে ব্যবহার করা যাবেনা। যেমন **with(),while()**

ফাংশনে প্যারামিটার ব্যবহার করাঃ

আপনি যখন ফাংশন লিখবেন তখন এখানে প্যারামিটার ব্যবহার করতে পারেন। এই প্যারামিটার প্রথম ব্রাকেটের ভিতর রাখতে হবে, এগুলি একধরনের ভেরিয়েবল। যদি কোন প্যারামিটার না থাকে তাহলে প্রথম ব্রাকেটের ভিতর কিছু থাকবেনা। যেমন **popup()** ফাংশনটি দেখুন এখানে কোন প্যারামিটার নেই। প্যারামিটার সহ একটি ফাংশন

```

<html>
<head>
<script type="text/javascript">
function getAddition(firstNumber,secondNumber){
var result;
result = firstNumber + secondNumber;
return result;

```

```
}  
var myResult = getAddition(10,20);  
alert(myResult);  
</script>  
</head>  
<body>  
</body>  
</html>
```

ব্যাখ্যা:

এখানে ফাংশনটিতে দুটি প্যারামিটার আছে **firstNumber** এবং **secondNumber**. এরপর একটা ভেরিয়েবল ডিক্লেয়ার করেছি যার নাম **result** এবং এই ভেরিয়েবলে প্যারামিটার দুটি যোগ করেছি। সবশেষে **result** রিটার্ন করেছি। এটা ফাংশনের একটা গুরুত্বপূর্ণ বৈশিষ্ট্য যে আপনি শুধু একটা মান ফেরৎ (রিটার্ন) পাঠাতে পারেন। **return** স্টেটমেন্ট ব্যবহার করে এটা করা যায়।

একটা জিনিস মনে রাখতে হবে যে যখন **return** স্টেটমেন্ট ব্যবহার করবেন তখন এই স্টেটমেন্টের পর আর কোন কোড কাজ করবেনা। একটা ফাংশন **return** স্টেটমেন্ট দেখলেই সে সংশ্লিষ্ট মান টি রিটার্ন করে কোড পড়া বন্ধ করে দেয়।

যাইহোক এরপর ১৭ নম্বর লাইনে দেখুন ফাংশনটিকে কিভাবে কল করেছি। **return** স্টেটমেন্ট দিয়ে পাঠানো মান এভাবে একটা ভেরিয়েবল ডিক্লেয়ার করে ধরতে হয়। যেমন আমি **var myResult** দিয়ে করেছি। এরপর **alert()** ফাংশন দিয়ে আউটপুট এনেছি। যদি **alert(result)** দেন তাহলে হবেনা। কারণ তো বললামই যে রিটার্নকৃত মান ভেরিয়েবল দিয়ে ধরতে হয়।

সবশেষে **getAddition** এ দুটি আর্গুমেন্ট পাঠিয়েছি ১০ এবং ২০ কারণ প্যারামিটার দুটি আছে। যতগুলি প্যারামিটার আছে ফাংশনটি কল করার সময় ততগুলি আর্গুমেন্ট পাঠাতে হবে।

*ফাংশনের ভিতরে কোন প্যারামিটার বা ভেরিয়েবল ব্যবহার করলে সেই ভেরিয়েবলের প্রভাব বাইরে থাকবেনা। এমনকি একই নামের একটা ভেরিয়েবল যদি ফাংশনের বাইরে থাকে তারপরেও ভেরিয়েবল দুটি সম্পূর্ণ আলাদা।

ফাংশান রিটার্ন স্টেটমেন্টঃ

জাভাস্ক্রিপ্ট ফাংশানে একটা অপশনাল রিটার্ন স্টেটমেন্ট রয়েছে। এটা তখনই প্রয়োজন যখন আপনি চান ফাংশন থেকে ভেলু রিটার্ন করতে। এই স্টেটমেন্টটা হল ফাংশানের সব্বশেষ স্টেটমেন্ট।

ফাংশান রিটার্ন স্টেটমেন্ট তৈরির সাধারন রূপটি হল-

```
function functionname (arguments)
```

```
{
```

```
Script statement(s)
```

```
return[variables]
```

```
}
```

উদাহরণঃ

The return statement is used to specify the value that is returned from the function. So, functions that are going to return a value must use the return statement. The example below returns the product of two numbers (a and b):

আমরা ফাংশনের মাধ্যমে একটি ছোট যোগ অংক করবো। এখানে দুইটা নাম্বারের গুনফল রিটার্ন করবে। প্রোজেক্টটি দেখুন:

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>

<body>
<script type="text/javascript">
document.write(product(4,3));
</script>

</body>
</html>
```

আউটপুটঃ

যতক্ষণ জাভাস্ক্রিপ্ট ফাংশনটি চলমান থাকে ততক্ষণ ভ্যারিয়াবলটি কাজ করবে। ফাংশনটি বন্ধ হওয়ার সাথে সাথে তা সয়ংক্রিয়ভাবে মুছে যায়।

অধ্যায়ঃ বার -জাভাস্ক্রিপ্ট ইভেন্ট

জাভাস্ক্রিপ্ট ইভেন্ট কি?

আর্কমণীয় ওয়েব পেজ তৈরীর ভিত্তি হচ্ছে জাভাস্ক্রিপ্ট ইভেন্ট। ইভেন্টের সাধারণ অর্থ ঘটনা। ডায়নামিক ওয়েব সাইটে বিভিন্ন ঘটনা ঘটে থাকে যা ইভেন্ট নামে পরিচিত। ঘটনাগুলো হতে পারে কোন বাটনে ক্লিক করা বা কোন লিঙ্কের উপর মাউস ওভার করা ইত্যাদি। আমরা জানি এইচটিএমএল পেজে জাভাস্ক্রিপ্ট তখনই এক্সিকিউট হয় যখন পেজটি লোড হয়। কিন্তু এটা সব সময় ঘটে না। আমরা মাঝেমাঝে চাই যে জাভাস্ক্রিপ্ট তখনই এক্সিকিউট হোক যখন একটি ইভেন্ট সংগঠিত হবে। যেমন-যখন কোন ইউজার একটি বাটনে ক্লিক করবে, তখনই কেবল জাভাস্ক্রিপ্ট কোড এক্সিকিউট হবে। এক্ষেত্রে আমরা স্ক্রিপ্ট ফাংশানের মাঝে স্থাপন করব। আসলে ইভেন্টকে ফাংশানের সাথে কম্বিনেশন করা হয় (যেমন- যখন কোন ইভেন্ট সংগঠিত হবে তখন ফাংশানকে কল করা হবে)। প্রতিটা অবজেক্টেরই কিছু নিজস্ব ইভেন্ট বা ঘটনা আছে। ইভেন্ট সাধারণত ওয়েব সার্ভার, ওয়েব ব্রাউজার ও ওয়েব ইউজারের **interactions** এর মাধ্যমে ঘটে থাকে। ইভেন্ট **Document Object Model (DOM)** -এর একটি অংশ। ওয়েব পেজের প্রতিটি এলিমেন্টের কিছু না কিছু ইভেন্ট রয়েছে যা জাভাস্ক্রিপ্ট কোডকে হ্যান্ডেল করে। উদাহরণস্বরূপ- আমরা একটি বাটনে **onClick** ইভেন্ট ব্যবহার করি যাতেকরে ইউজার ঐ বাটনে ক্লিক করলেই কেবল নির্দিষ্ট কোন ফাংশান কাজ করে। ইভেন্টকে **HTML** ট্যাগের মাঝে ডিফাইন করতে হবে।

জাভাস্ক্রিপ্ট ইভেন্টের উদাহরণ-

- ১ একটি মাউস ক্লিক
- ২ ওয়েব পেজ লোড হওয়া
- ৩ চিহ্নিত স্থানের উপর মাউস রাখা যাকে আমরা **hover** নামে জানি।
- ৪ এইচটিএমএল ফর্ম এ ইনপুট বক্স কে সিলেক্ট করা

৫ একটি **keystroke**

নিচের উদাহরণে "onclick" ইভেন্ট যোগ করা হলো-

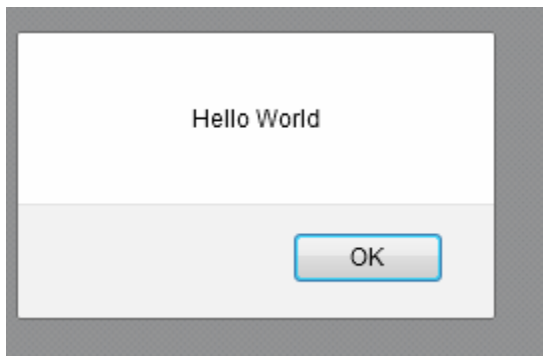
```
<html>
<head>
<script type="text/javascript">
```

```
<!--  
function popup() {  
alert("Hello World")  
}  
/-->  
</script>  
</head>  
<body>  
<input type="button" value="Click Me!" onclick="popup()">  
</body>  
</html>
```

আউটপুটঃ



বাটনে ক্লিক করলে নিচের মত পপআপ মেসেজ দেখাবে।



জাভাস্ক্রিপ্ট ইভেন্ট হ্যান্ডেলারঃ

ইউজার একটি ওয়েব সাইট ভিজিট করে কোন লিঙ্কে ক্লিক করা বা কোন লিঙ্কের উপর মাউস ওভার করা, ফর্ম সাবমিট করা ইত্যাদি এই ধরনের বিভিন্ন কাজ করে থাকে যা জাভাস্ক্রিপ্টে ইভেন্ট নামে পরিচিত। জাভাস্ক্রিপ্ট ইভেন্ট হ্যান্ডেলার ব্যবহার করে ইভেন্টকে রেসপন্স করা হয়। আপনি **HTML** এলিমেন্টে একটি ইভেন্ট হ্যান্ডেলার যোগ করে দিতে পারেন যাতে করে কোন ইভেন্ট সংগঠিত হলেই কেবল রেসপন্স পাওয়া যাবে। উদাহরণস্বরূপ- আপনি যখন জাভাস্ক্রিপ্ট **onMouseover** ইভেন্ট হ্যান্ডেলার কোন বাটনে যুক্ত করবেন এবং কিছু জাভাস্ক্রিপ্ট কোড নির্দিষ্ট করে দেবেন যা কেবল মাত্র ইভেন্ট সংগঠিত হলেই(এক্ষেত্রে ঐ বাটনের উপর মাউস ওভার করা) কোডটি রান হবে।

জাভাস্ক্রিপ্ট ইভেন্ট অবজেক্টঃ

ইভেন্ট হল কোন কাজ যা জাভাস্ক্রিপ্ট দিয়ে করা হয়ে থাকে আর ইভেন্ট অবজেক্ট সংগঠিত ইভেন্ট সম্বন্ধে তথ্য প্রদান করে।

। আমরা মাঝেমাঝে চাই যে জাভাস্ক্রিপ্ট তখনই এক্সিকিউট হোক যখন একটি ইভেন্ট সংগঠিত হবে। যেমন-যখন কোন ইউজার একটি বাটনে ক্লিক করবে, তখনই কেবল জাভাস্ক্রিপ্ট কোড এক্সিকিউট হবে।

জাভাস্ক্রিপ্ট ইভেন্ট এট্রিবিউটঃ

নিম্নে ইভেন্ট এট্রিবিউটের তালিকা দেওয়া হল যেগুলো বিভিন্ন এইচটিএমএল এলিমেন্টের মাঝে স্থাপন করা হয়। এদের কাজ হল ইভেন্টের কাজের ধরণ বর্ণনা করা।

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C Standard.

এট্রিবিউট	ঘটনা ঘটে যখন.....	IE	F	O	W3C
onblur	An element loses focus	3	1	9	Yes
onchange	The content of a field changes	3	1	9	Yes
onclick	Mouse clicks an object	3	1	9	Yes
ondblclick	Mouse double-clicks an object	4	1	9	Yes
onerror	An error occurs when loading a document or an image	4	1	9	Yes
onfocus	An element gets focus	3	1	9	Yes
onkeydown	A keyboard key is pressed	3	1	No	Yes
onkeypress	A keyboard key is pressed or held down	3	1	9	Yes
onkeyup	A keyboard key is released	3	1	9	Yes
onload	A page or image is finished loading	3	1	9	Yes
onmousedown	A mouse button is pressed	4	1	9	Yes
onmousemove	The mouse is moved	3	1	9	Yes
onmouseout	The mouse is moved off an element	4	1	9	Yes
onmouseover	The mouse is moved over an element	3	1	9	Yes
onmouseup	A mouse button is released	4	1	9	Yes
onresize	A window or frame is resized	4	1	9	Yes
onselect	Text is selected	3	1	9	Yes
onunload	The user exits the page	3	1	9	Yes

মাউস/কীবোর্ড এন্ট্রিবিউট

Property	Description	IE	F	O	W3C
altKey	Returns whether or not the "ALT" key was pressed when an event was triggered	6	1	9	Yes
button	Returns which mouse button was clicked when an event was triggered	6	1	9	Yes
clientX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
clientY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
ctrlKey	Returns whether or not the "CTRL" key was pressed when an event was triggered	6	1	9	Yes
metaKey	Returns whether or not the "meta" key was pressed when an event was triggered	6	1	9	Yes
relatedTarget	Returns the element related to the element that triggered the event	No	1	9	Yes
screenX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
screenY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
shiftKey	Returns whether or not the "SHIFT" key was pressed when an event was triggered	6	1	9	Yes

অন্যান্য ইভেন্ট এন্ট্রিবিউট

Property	Description	IE	F	O	W3C
bubbles	Returns a Boolean value that indicates whether or not an event is a bubbling event	No	1	9	Yes
cancelable	Returns a Boolean value that indicates whether or not an event can have its default action prevented	No	1	9	Yes
currentTarget	Returns the element whose event listeners triggered the event	No	1	9	Yes
target	Returns the element that triggered the event	No	1	9	Yes

timeStamp	Returns the time stamp, in milliseconds, from the epoch (system start or event trigger)	No	1	9	Yes
type	Returns the name of the event	6	1	9	Yes

অধ্যায়ঃ তের- ইউজারের সাথে যোগাযোগ

জাভাস্ক্রিপ্ট পপআপ বক্স (ইউজারের সাথে যোগাযোগ)

ইউজারের সাথে যোগাযোগ গড়ে তোলা যেতে পারে উইন্ডো অবজেক্টের তিনটি মেথড **alert**, **confirm** ও **prompt**-এর মাধ্যমে। **alert()** মেথডের মাধ্যমে গুরুত্বপূর্ণ কোন তথ্য ইউজারকে অবহিত করতে পারেন। এক্ষেত্রে ইউজার সেই তথ্য জেনে কেবল **OK** বাটনে ক্লিক করবে। এখানে অন্য কোন অপশান থাকবে না। অন্যদিকে কোন তথ্য দিয়ে ইউজারের কাছ থেকে ইনপুট নিতে চাইলে ব্যবহার করতে পারেন **confirm()** মেথড। এক্ষেত্রে ইউজার সেটি গ্রহন বা বর্জন করতে পারে **OK** বা **Cancel** বাটন প্রেস করে। অন্যদিকে ইউজারের কাছ থেকে কোন ইনপুট নিয়ে সেটিকে ডকুমেন্টে ব্যবহারের জন্য **prompt()** মেথড ব্যবহার করতে পারেন। এতে ইউজারের সামনে একটি ইনপুট ফিল্ড দেয়া হবে যাতে সে ইনপুট দিতে পারে। প্রতিটি মেথডকে আবার পপআপ বক্স হিসেবে গন্য করা হয়। যথা- **alert box**, **confirm box**, **prompt box**.

নিম্নে এদের বর্ণনা দেওয়া হল-

Alert Box

alert() মেথডের বা **alert** বক্সের মাধ্যমে গুরুত্বপূর্ণ কোন তথ্য ইউজারকে অবহিত করতে পারেন। এক্ষেত্রে ইউজার সেই তথ্য জেনে কেবল **OK** বাটনে ক্লিক করবে। এখানে অন্য কোন অপশান থাকবে না।

সিনট্যাক্সঃ

alert("sometext");

উদাহরণঃ

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
```

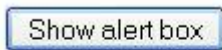


```
alert("Hello! I am an alert box!");
}
</script>
</head>
<body>

<input type="button" onclick="show_alert()" value="Show alert box" />

</body>
</html>
```

উইন্ডোতে নিচের মোট একটি বাটন দেখাবে-



যাটা ক্লিক করলে নিচের মত বক্স দেখাবে-



যদি আপনার ওয়েব ব্রাউজার এ জাভাস্ক্রিপ্ট এন্টিভ করা না থাকে তাহলে আপনি **Alert** দেখতে পাবেন না। জাভাস্ক্রিপ্ট এলটি হচ্ছে একটা ডায়ালগ বক্স যা **pops up** এবং চলতি ব্রাউজার উইন্ডো হতে দৃষ্টি অর্কমণ করে। ওয়েব ব্রাউজার কে **Alert** মেসেজটি পড়তে বাধ্য করে।

Confirm Box

কোন তথ্য দিয়ে ইউজারের কাছ থেকে ইনপুট নিতে চাইলে ব্যবহার করতে পারেন **confirm()** মেথড। এক্ষেত্রে ইউজার সেটি গ্রহন বা বর্জন করতে পারে **OK** বা **Cancel** বাটন প্রেস করে। যদি "OK" ক্লিক করা হয় তবে **true** ভেলু রিটার্ন করবে আর যদি "Cancel" ক্লিক করা হয় তবে **false** ভেলু রিটার্ন করবে।

সিনট্যাক্সঃ

confirm("sometext");


উদাহরণঃ

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button!");
if (r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_confirm()" value="Show a confirm box" />

</body>
</html>
```

উইন্ডোতে নিচের মত একটি বাটন দেখাবে-



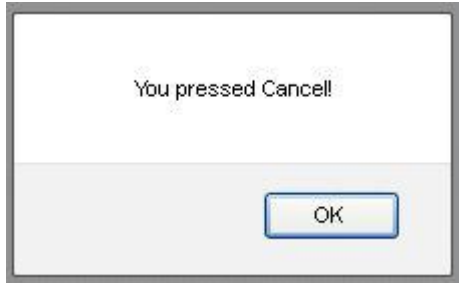
যাতে ক্লিক করলে নিচের মত একটা বক্স দেখাবে-



Ok ক্লিক করলে নিচের মত মেসেজ দেখা যাবে-



Cancel ক্লিক করলে মত মেসেজ দেখা যাবে-



JavaScript *confirm* ফাংশন এবং JavaScript *alert* ফাংশন প্রায় একই রকম। এটা একটা ছোট ডায়ালগ বক্স যা ওয়েব পেজ এর সামনে সরাসরি দৃষ্টি পাত করার জন্য প্রদর্শিত হয়। *confirm box* যা *alert box* হতে ভিন্ন। এটা ব্যবহারকারীকে দুটি অপশন দেয়:

১ পপআপ মেসেজটি যদি তারা *confirm* করতে চায় তবে **OK** প্রেস করতে হবে।

২ বা পপআপ মেসেজটির সাথে একমত না হয় তবে **cancel** প্রেস করতে হবে।

কোন কিছু নিশ্চিত করার জন্য *confirm* ফাংশন টি প্রায়ই ব্যবহার হয়ে থাকে। এই গুরুত্বপূর্ণ কাজটি **website** এর মাধ্যমে সম্পন্ন করা হয়। ব্যবহারকারীগণ কোন কিছু **submission** এবং ব্যবহারকারীকে বিজ্ঞপ্তি প্রদান করে যে তারা যে লিংকে ক্লিক করেছে তার মাধ্যমে অন্য **website** এ যেত ইচ্ছুক কিনা।

Prompt Box

ইউজারের কাছ থেকে কোন ইনপুট নিয়ে সেটিকে ডকুমেন্টে ব্যবহারের জন্য **prompt()** মেথড ব্যবহার করতে পারেন। এতে ইউজারের সামনে একটি ইনপুট ফিল্ড দেয়া হবে যাতে সে ইনপুট দিতে পারে। যদি "OK" ক্লিক করা হয় তবে **input value** ভেলু রিটার্ন করবে আর যদি "Cancel" ক্লিক করা হয় তবে **null** ভেলু রিটার্ন করবে।

সিনট্যাক্সঃ

```
prompt("sometext", "defaultvalue");
```

উদাহরণঃ

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
{
document.write("Hello " + name + "! How are you today?");
}
}
</script>
</head>
<body>

<input type="button" onclick="show_prompt()" value="Show prompt box" />

</body>
</html>
```

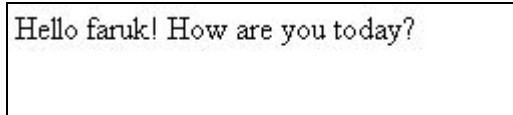
উইন্ডোতে নিচের মোট একটি বাটন দেখাবে-

Show prompt box

যাতে ক্লিক করলে



নাম ইনপুট করে ওকে ক্লিক



জাভাস্ক্রিপ্ট প্রম্পট এর ব্যবহার বর্তমানে কম দেখা যায়। জাভাস্ক্রিপ্ট প্রম্পট এর উদ্দেশ্য হচ্ছে ব্যবহারকারী(user) হতে তথ্য সংগ্রহ করা যাতে এই তথ্যগুলোকে এ ব্যবহার করা যায় যারফলে ব্যবহারকারীর(user) ব্যক্তিগত অনুভূতি সৃষ্টি হবে।

জাভাস্ক্রিপ্ট প্রম্পট খুব কাজে লাগে না এটা অনেকের কাছে বিরক্তি কর। তবে শেখার জন্য এখানে শিখবো।

তিনটি মেথডের সমন্বয়ে একটি উদাহরণ দেখুন-

```
<html>
<head>
<title>Open and close method demonstration</title>
</head>
<body>
<h3>alert, confirm and prompt method demonstration</h3>
<form>
<input type=button onClick="javascript:confirm('Do you really want to quit?')"
value="Quit">
<input type=button onClick="javascript:prompt('What is your name, please?')"
value="Your name">
<input type=button onClick="javascript:alert('You are going to enter a dangerous
zone')" value="Warning!">
</body>
```

</form>
</html>

অধ্যায়ঃ চৌদ্দ- অবজেক্ট ওরিয়েন্টেড জাভাস্ক্রিপ্ট

অবজেক্ট ওরিয়েন্টেড জাভাস্ক্রিপ্ট কি?

জাভাস্ক্রিপ্ট হল অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং ল্যাঙ্গুয়েজ (অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং কাজ করে কোড এবং তার সাথে সংশ্লিষ্ট ডাটা নিয়ে)। অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং ল্যাঙ্গুয়েজ আপনাকে নিজের মত অবজেক্ট (কোন বস্তু কিংবা ঘটনাই হল অবজেক্ট) তৈরি করতে দেবে এবং আপনি নিজস্ব ভেরিয়েবল টাইপ তৈরি করতে পারবেন। জাভাস্ক্রিপ্টকে জাভার মত স্বয়ংসম্পূর্ণ অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং (OOP) ল্যাঙ্গুয়েজ বলা যায় না। এটা ডিজাইন করা হয়েছে শুধুমাত্র সিম্পল অবজেক্ট বেজড মডেল হিসেবে। জাভাস্ক্রিপ্টে নিজস্ব বিল্ট-ইন অবজেক্ট রয়েছে, এছাড়াও জাভাস্ক্রিপ্ট কাস্টম অবজেক্ট তৈরি করার সুযোগ আছে। আবার অবজেক্টের বৈশিষ্ট্যই হল তার প্রপার্টি। অবজেক্টের সাথে রিলেটেড ফাংশান থাকতে পারে যা অবজেক্ট মেথড নামে পরিচিত। (বিঃ দ্রঃ অবজেক্ট, প্রপার্টি ও মেথড নিয়ে নিচে বিস্তারিত আলোচনা করা হয়েছে)। একটি প্রোগ্রামিং ল্যাঙ্গুয়েজকে অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং (OOP) ল্যাঙ্গুয়েজ বলা যাবে যদি তার নিচের চারটি বৈশিষ্ট্য থাকে-

- এনক্যাপসুলেশন (Encapsulation): একই ধরনের তথ্যকে(যা ডাটা বা মেথড যেকোনোটি হতে পারে) একত্রে করে অবজেক্ট তৈরি করাই হল এনক্যাপসুলেশন। বিঃ দ্রঃ অবজেক্ট কি তা নিচে আলোচনা করা হয়েছে।
- এগ্রিগেশান(Aggregation) : একটি অবজেক্ট অন্য আরেকটি অবজেক্টের মাঝে স্টোর করাই হল এগ্রিগেশান।
- ইনহেরিটেন্স, (Inheritance): the capability of a class to rely upon another class (or number of classes) for some of its properties and methods
-
- পলিমরফিজম (Polymorphism): the capability to write one function or method that works in a variety of different ways

অবজেক্টঃ

অবজেক্ট এর সাধারণ অর্থ বস্তু। কোন বস্তু বা ঘটনার প্রোগ্রামিং সংস্করণ হল অবজেক্ট। অন্যভাবে বলা যায় বাস্তব জগতের সকল বস্তু (যেমন-বই, মানুষ, বল ইত্যাদি) বা ঘটনা হল এক একটি অবজেক্ট। আবার বলা যায় অবজেক্ট হল বিশেষ ধরনের ডাটা যার রয়েছে নিজস্ব প্রপার্টি এবং মেথড। **Objects are useful to organize information.**

প্রপার্টিঃ

প্রপার্টি যার অর্থ হল বৈশিষ্ট্য। এখানে বৈশিষ্ট্য বলতে অবজেক্টেরই বৈশিষ্ট্য বুঝানো হয়। প্রতিটি অবজেক্টেরই কিছু না কিছু বৈশিষ্ট্য থাকে। যেমন- আমরা জানি একজন ব্যক্তি একটি অবজেক্ট। আর প্রপার্টি হল অবজেক্টের ভেলু। ঐ ব্যক্তির নাম, উচ্চতা, ওজন, ভর, বয়স, চোখের রঙ ইত্যাদি হল তার প্রপার্টি। সকল লোকেরই এসকল প্রপার্টি রয়েছে কিন্তু তা ব্যক্তিভেদে আলাদা হয়।

অবজেক্টে প্রপার্টি একসেস করার সিনট্যাক্স হল-

```
objectName.objectProperty = propertyValue;
```

আপনি সরাসরি প্রপার্টির ভেলু এসাইন করে দিতে পারেন। যেমন-

```
<script type="text/javascript">
personObj.firstname="Abdullah";
personObj.lastname="Faruk";
personObj.age=20;
personObj.eyecolor="black";
document.write(personObj.firstname);
</script>
```

এখানে `personObj` হল অবজেক্ট এবং `firstname`, `lastname`, `age`, `eyecolor` হল ঐ অবজেক্টের প্রপার্টি। সমান চিহ্নের ডানপাশের সকল ভেলু হল ঐ প্রপার্টির ভেলু। উপরোক্ত কোডের আউটপুট হবে- `Abdullah`।

মেথডঃ

প্রপার্টিতে আমরা বলেছিলাম একজন ব্যক্তি একটি অবজেক্ট। প্রতিটি অবজেক্টেরই আবার রয়েছে এক বা একাধিক মেথড। মেথড হল ক্রিয়া বা কাজ (`actions`) যা অবজেক্টের দ্বারা সম্পাদন করা হয়। যেমন- ঐ ব্যক্তির মেথড হল `eat()`, `sleep()`, `work()`, `play()` ইত্যাদি। একেকটি কাজ করতে একেকটি মেথড ব্যবহার করতে হয়।

নিচের সিনট্যাক্স ব্যবহার করে একটা মেথডকে কল করতে পারেন-

```
objName.methodName()
```

উদাহরণ-

```
document.write("javaScript");
```

এখানে `Document` অবজেক্টের মেথড হল `write()`। এই মেথডের কাজ হল কোনো কিছু লেখা। `write()` এর ব্র্যাকেটের মাঝে যা কিছু লেখা হবে `write()` মেথডটি সেটাই আউটপুটে দেখাবে।

অবজেক্টের প্রকারভেদঃ

জাভাস্ক্রিপ্ট অবজেক্টকে দুই ভাগে ভাগ করা যায়। যথা-

১. বিল্ট-ইন অবজেক্ট
২. ইউজার ডিফাইন অবজেক্ট

১. জাভাস্ক্রিপ্ট বিল্ট-ইন অবজেক্টঃ

জাভাস্ক্রিপ্ট কতগুলো বিল্ট-ইন অবজেক্ট রয়েছে যাদের তালিকা নিম্নে দেওয়া হল-

১. অ্যারে অবজেক্ট (Array Object)
২. বুলিয়ান অবজেক্ট (Boolean Object)
৩. ডেট (সময়) অবজেক্ট (Date Object)
৪. ম্যাথ অবজেক্ট (Math Object)
৫. নাম্বার অবজেক্ট (Number Object)
৬. স্ট্রিং অবজেক্ট (String Object)
৭. রেগুলার এক্সপ্রেশান অবজেক্ট (RegExp Object)
৮. গ্লোবাল অবজেক্ট (Global Object)

২. ইউজার ডিফাইন অবজেক্ট

জাভাস্ক্রিপ্টে অবজেক্ট তৈরি করাঃ

ইতিপূর্বে আমরা দেখলাম জাভাস্ক্রিপ্টের কিছু বিল্ট-ইন অবজেক্ট রয়েছে। যেমন- **String, Date, Array** ইত্যাদি। বিল্ট-ইন অবজেক্ট ছাড়াও জাভাস্ক্রিপ্টে নিজস্ব অবজেক্ট তৈরি করা যায়। জাভাস্ক্রিপ্টে বিভিন্নভাবে অবজেক্ট তৈরি করা যায়। নিচের দুটি পদ্ধতির সাহায্যে অবজেক্ট তৈরি করা দেখানো হল-

১. **new** অপারেটর ব্যবহার করে সরাসরি অবজেক্ট তৈরি করা।

২. অবজেক্ট ইনিশিয়ালাইজার (**initializer**) / **Constructor function** ব্যবহার করে অবজেক্ট তৈরি।

1. **new** অপারেটর ব্যবহার করে সরাসরি অবজেক্ট তৈরি করাঃ

“**new** “ অপারেটর ব্যবহার করে জাভাস্ক্রিপ্টে অবজেক্ট তৈরি করা যায়। এক্ষেত্রে বিভিন্ন **constructor** মেথড যেমন- **Object()**, **Array()** বা **Date()** ব্যবহার করা হয়। এসকল **constructor** মেথড আসলে জাভাস্ক্রিপ্ট ফাংশান।

সিনট্যাক্সঃ

```
objectName = { property1 : value1, property2 : value2, ...,propertyN : valueN};
```

এখানে,

1. **objectName** : নতুন অবজেক্টের নাম।

2. **property_1, property_2,property_n** : এগুলো প্রপার্টির নাম যা- নাম, সংখ্যা বা স্ট্রিং হতে পারে।

3. **value1, value2, ...,valueN** : এগুলো প্রপার্টির মান বা এক্সপ্রেসন।

নিচের কোড একটি অবজেক্ট তৈরি করে যেখানে চারটি প্রপার্টি রয়েছে-

```
personObj=new Object();
personObj.firstname="John";
personObj.lastname="Doe";
personObj.age=50;
personObj.eyecolor="blue";
```

এভাবে না লিখে সরাসরি নিচের মত করে লেখা যায়-

```
personObj={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue"};
```

উদাহরণ-১

```
<html>
<body>
<script type="text/javascript">
personObj={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue"}
document.write(personObj.firstname + " is " + personObj.age + " years old.");
</script>
</body>
</html>
```

আউটপুটঃ

John is 50 years old.

উদাহরণ-২

```
<html>
<head>
<title>User-defined objects</title>
<script type="text/javascript">
var book = new Object(); // Create the object
    book.subject = "Perl"; // Assign properties to the object
    book.author = "Mohtashim";
</script>
```

```
</head>

<body>

<script type="text/javascript">

    document.write("Book name is : " + book.subject + "<br>");

    document.write("Book author is : " + book.author + "<br>");

</script>

</body>

</html>
```

আউটপুটঃ

Book name is : Perl
Book author is : Mohtashim

2. অবজেক্ট ইনিশিয়ালাইজার (initializer) / Constructor function ব্যবহার করে অবজেক্ট তৈরিঃ

constructor ফাংশান ব্যবহার করে অবজেক্ট তৈরি করতে নিচের নিয়মগুলো ভালভাবে মেনে চলতে হবে-

- **constructor** ফাংশানের নাম অবজেক্টের নামের মত হবে।
- **constructor** ফাংশানে "this" কীওয়ার্ড ব্যবহার করে অবজেক্টে মেম্বার যোগ করতে হবে।
- সমান চিহ্নের ("=") পরে প্রপার্টি/ মেথডের মান ডিফাইন করে দিতে হবে।
- **constructor** ফাংশানে কোন "return" স্টেটমেন্ট থাকতে পারবে না।

constructor ফাংশান ব্যবহার করে অবজেক্ট তৈরি করতে মোট তিনটি ধাপ অনুসরণ করতে হবে। উদাহরণের সাহায্যে বিষয়টি আলোচনা করা হল-

প্রথম ধাপঃ

নিম্নে একটি ফাংশান তৈরি করা হল যা অবজেক্ট গঠন করে-

```
function student(name, class, rollno)
```

```
{  
  this.name = name;  
  this.class = class  
  this.rollno = rollno;  
}
```

উপরোক্ত উদাহরণে **student** হল একটি অবজেক্ট, যার তিনটি প্যারামিটার আছে- **name**, **class** and **rollno**। অবজেক্টের মান নির্ভর করে ফাংশানে কোন প্যারামিটার পাস (**passed**) করা হয়েছে তার উপর। এখানে "this" কীওয়ার্ড ব্যবহার করে **constructor** ফাংশানে অবজেক্টে মেম্বার (প্রপার্টি) যোগ করা হয়েছে। **The reason for all the "this" stuff is that you're going to have more than one person at a time (which person you're dealing with must be clear).**

দ্বিতীয় ধাপঃ

একবার **object constructor** গঠন হয়ে গেলে, আপনি নতুন অবজেক্ট তৈরি করতে পারবেন। এই ধাপে আপনাকে নতুন একটি অবজেক্ট তৈরি করতে হবে। যেমন-

```
studentv = new student("John", "V", 10)
```

উপরোক্ত স্টেটমেন্টে "studentv" নামে একটি নতুন অবজেক্ট তৈরি করা হয়েছে এবং এই অবজেক্টের প্রপার্টিতে নির্দিষ্ট ভেলু এসাইন করে দেওয়া হয়েছে। এমতাবস্থায়-

- **studentv.name** → এর মান হবে স্ট্রিং "John"
- **studentv.class** → এর মান হবে স্ট্রিং "V"
- **studentv.rollno** → এর মান হবে ইন্টিজার 10

. We can create any number of student objects by calls to new.

তৃতীয় ধাপঃ

এবার **constructor** ফাংশানের বাইরে নতুন যে অবজেক্ট তৈরি করা হয়েছে তাকে কল করতে হবে। বিষয়টি ভালভাবে বোঝার জন্য একটি উদাহরণ দেওয়া হল-

```
<html>
```

```
<body>
```

```
<script type="text/javascript">
```

```
function student(name, class, rollno)
```

```
{  
  this.name = name;  
  this.class = class  
  this.rollno = rollno;  
}
```

```
studentv = new student("John", "V", 10)
```

```
document.write(studentv.name + " reading in class " + student.class + "and his roll  
no:" + student.rollno);
```

```
</script>
```

```
</body>
```

```
</html>
```

অধ্যায়ঃ পনের - জাভাস্ক্রিপ্ট কুকি

জাভাস্ক্রিপ্ট কুকি কি?

কুকি যা HTTP কুকি , web কুকি, অথবা browser কুকি নামেও পরিচিত। কুকি হল একটি ছোট টেক্সট ফাইল যা ওয়েব সাইট থেকে ইউজারকে পাঠানো হয় এবং তা ইউজারের ব্রাউজারে জমা হয় যখন ইউজার কোন ওয়েব সাইট ভিজিট করে। যখন ঐ ইউজার ভবিতসতে আবার সেই ওয়েব সাইট ভিজিট করে তখন কুকিতে স্টোরজীত ডাটা ঐ ওয়েব সাইট দ্বারা আবার

retrieved করা হয়। ইউজারের পূর্বের **activity** (ঐ ওয়েব সাইটে ইউজার পূর্বে কি কি কাজ করেছে। যেমন- কোন কোন বাটনে ক্লিক করেছে, লগ-ইন করেছে কিনা, কোন কোন পেজ ভিজিট করেছে, কোন নাম দিয়ে রেজিস্ট্রেশন করেছে,

remembering the items in your shopping cart from previous visits. এককথায় ইউজারকে আইডেন্টিফাই করতে ইত্যাদি)চিহ্নিত করা হয়। কুকি ইউজারের কম্পিউটারে ভাইরাস ছড়ায় না বা ম্যাকওয়্যার সেটআপ করে না।

বর্তমানে বিশেষ কিছু ধরণের কুকি রয়েছে যা আধুনিক ওয়েব সাইটগুলতে বিশেষ কাজ করে থাকে। যেমন-**authentication**

cookies মেথড যা নির্ণয় করে ইউজার লগ-ইন অবস্থায় আছে কি নেই, অথবা যদি লগ-ইন অবস্থায় থাকে তবে কোন একাউন্ট

দিয়ে লগ-ইন আছে। এ ধরণের ইনফরমেশান না জেনে ওয়েব সাইট যে কোন সেন্সেটিভ পেজ ইউজারকে প্রদর্শন করতে পারে না।

এটা ওয়েবের জন্য হুমকি হয়ে দাঁড়াবে। কুকি কেবল মাত্র সেই সার্ভারই রিড করতে পারে যে সার্ভার এটা জেনারেট করেছে। কুকি মূলত ইউজারের কাজকে সহজ করতে তৈরি করা হয়েছে। জাভাস্ক্রিপ্ট একই সাথে কুকি তৈরি ও পুনঃরুদ্ধার করতে পারে।

কুকির উদাহরণ-

- **Name cookie** - ইউজার প্রথমে কোন ওয়েব সাইটে তার ইউজার নাম দিয়ে প্রবেশ করলে সেটা কুকিতে জমা থাকে। সে আবার যখন ঐ সাইটে লগিন করে তখন তাকে আবার তার নাম দিয়ে একটি ওয়েলকাম মেসেজ দেওয়া যেতে পারে আর এই নামটি কুকিতে সংরক্ষিত থাকে।
- **Password cookie** - পাসওয়ার্ড-এর ব্যাপারটাও একই। দ্বিতীয় বার সাইটে লগিন করলে কুকি থেকে পাসওয়ার্ড পুনঃরুদ্ধার করা হয়।
- **Date cookie** - দ্বিতীয় বার কোন সাইটে লগিন করলে আপনাকে জানিয়ে দেওয়া যেতে পারে আপনি শেষ কবে ঐ সাইটে লগিন করেছিলেন। পূর্বের তারিখ কুকিতে জমা থাকে।

কুকি তৈরি ও জমা করাঃ

নিচের উদাহরণে আমরা একটি কুকি তৈরি করবো যা ভিজিটরের নাম স্টোর করবে। প্রথমে সাইটে ভিজিট করলে ইউজারকে তার নাম ইনপুট করতে বলা হবে। এই নামটা কুকিতে জমা থাকবে। পরবর্তীতে ঐ ইউজার ঐ সাইটে প্রবেশ করলে তাকে একটি ওয়েলকাম মেসেজ দেওয়া হবে। প্রথমে আমরা একটি ফাংশান তৈরি করবো যা ইউজারের নাম একটি কুকি ভেরিয়েবলে জমা করবে-

```
function setCookie(c_name,value,exdays)
{
var exdate=new Date();
exdate.setDate(exdate.getDate() + exdays);
var c_value=escape(value) + ((exdays==null) ? "" : ";
expires="+exdate.toUTCString());
document.cookie=c_name + "=" + c_value;
}
```

উপরের উদাহরণে ফাংশানটি কুকির নাম, কুকির ভেলু এবং কুকির এক্সপায়ার ডেট ধারণ করবে। এই ফাংশানে প্রথমে দিনকে একটি ভ্যালিড তারিখে কনভার্ট করা হয়েছে। তারপর কুকির এক্সপায়ার ডেট সেট করা হয়েছে। তারপর কুকির নাম, কুকির ভেলু এবং কুকির এক্সপায়ার ডেট অবজেক্টে `document.cookie` স্টোর করা হয়েছে।

তারপর নিচের মত করে অন্য আরেকটি ফাংশান তৈরি করা হয়েছে , যার কাজ হল নির্দিষ্ট কুকি রিটান করা –

```
function getCookie(c_name)
{
var i,x,y,ARRcookies=document.cookie.split(";");
for (i=0;i<ARRcookies.length;i++)
{
x=ARRcookies[i].substr(0,ARRcookies[i].indexOf("="));
y=ARRcookies[i].substr(ARRcookies[i].indexOf("=")+1);
x=x.replace(/^\s+|\s+$/g,"");
if (x==c_name)
{
return unescape(y);
}
```

```
}  
}  
}
```

উপরের ফাংশানটি কুকির নাম ও ভেলু উদ্ধার করতে একটি অ্যারে তৈরি করে। তারপর এটা চেক করে দেখে নিদিষ্ট কুকি পাওয়া যায় কিনা , যদি পাওয়া যায় তবে কুকির মান রিটান করে।

সবশেষে একটি ফাংশান তৈরি করা হয়েছে যা ইউজারকে ওয়েলকাম মেসেজ দেখাবে যদি কুকি পাওয়া যায়। আর যদি কুকি না পাওয়া যায় তবে একটি প্রমোট বক্স দেখাবে যেখানে ইউজার নেম ও পাসওয়ার্ড দিতে হবে।

```
function checkCookie()  
{  
var username=getCookie("username");  
if (username!=null && username!="")  
{  
alert("Welcome again " + username);  
}  
else  
{  
username=prompt("Please enter your name:","");  
if (username!=null && username!="")  
{  
setCookie("username",username,365);  
}  
}  
}
```

তাহলে টোটাল কোডটা হল-

```
<html>  
  
<head>  
  
<script type="text/javascript">  
  
function getCookie(c_name)  
{  
  
var i,x,y,ARRcookies=document.cookie.split(";");
```

```
for (i=0;i<ARRcookies.length;i++)  
  
  {  
  
    x=ARRcookies[i].substr(0,ARRcookies[i].indexOf("="));  
  
    y=ARRcookies[i].substr(ARRcookies[i].indexOf("=")+1);  
  
    x=x.replace(/^\\s+|\\s+$/g,"");  
  
    if (x==c_name)  
  
      {  
  
        return unescape(y);  
  
      }  
  
  }  
  
}
```

```
function setCookie(c_name,value,exdays)  
  
{  
  
  var exdate=new Date();  
  
  exdate.setDate(exdate.getDate() + exdays);  
  
  var c_value=escape(value) + ((exdays==null) ? "" : ";  
  expires="+exdate.toUTCString());  
  
  document.cookie=c_name + "=" + c_value;  
  
}
```



```
function checkCookie()
{
var username=getCookie("username");
if (username!=null && username!="")
{
alert("Welcome again " + username);
}
else
{
username=prompt("Please enter your name:","");
if (username!=null && username!="")
{
setCookie("username",username,365);
}
}
}
</script>
</head>
<body onload="checkCookie()">
</body>
</html>
```

অধ্যায়ঃ ষোল-জাভাস্ক্রিপ্ট ফর্ম ভেলিডেশন

ইউজারের সাথে যোগাযোগের জন্য প্রতিটা ওয়েব সাইটে ফর্ম একটি অত্যন্ত গুরুত্বপূর্ণ একটি বিষয়। যেমনঃ ওয়েব সাইটে নিবন্ধন করা, কাস্টমার থেকে অর্ডার গ্রহন করা সহ অসংখ্য কাজ ফর্মের মাধ্যমে করা হয়। ফর্ম পূরণ করে ডাটাগুলো সার্ভারে পাঠানোর আগেই যদি প্রতিটা ফিল্ড চেক করে দেখা হয় যে কথাও ভুল আছে কি না তবে অনেক সময় বাঁচে, আর এই কাজটি হল "ফর্ম ভেলিডেশন"। অন্যভাবে বলা যায় জাভাস্ক্রিপ্ট ফর্ম ভেলিডেশন এমন একটি কৌশল যার মাধ্যমে ব্যবহারকারীগন ডেটা সাবমিট করার আগে তার ফর্ম এর তথ্যকে যাচাই করে। জাভাস্ক্রিপ্ট আপনাকে সাহায্যকারী এলটি দেখাবে এবং সেটা ব্যবহারকারীগনকে জানাবে যে তার তথ্যটি ভুল বা অসম্পূর্ণ ছিল এবং আরও বলবে যে কিভাবে সে সমস্যাটি সমাধান করতে পারবে। ফর্ম ভেলিডেশন দিয়ে যে কাজগুলো করা যায়-

- ইনপুট ফিল্ড খালি আছে কি না চেক করা।
- ইনপুট ফিল্ডে সব সংখ্যা আছে কি না চেক করা।
- ইনপুট ফিল্ডে প্রয়জনের চেয়ে কম বা বেশি ক্যারেক্টার ইনপুট করা হয়েছে কি না তা না চেক করা।
- ইমেইল এড্রেস টি বৈধ কিনা
- সঠিক ফরম্যাটের ডেট ইনপুট করা হয়েছে কি না তা না চেক করা।

ইত্যাদি দেখার জন্য ফর্ম ভেলিডেশন ব্যবহৃত হয়।

ইনপুট ফিল্ড ফাঁকা কিনা তা চেক করার জন্যঃ

এটা একটা সাধারণ টাইপের ফর্ম ভেলিডেশন । আপনি নিশ্চিত হতে পারবেন যে ব্যবহারকারী এইচটিএমএল ফিল্ড এর মধ্যে ডেটা লিখেছে কিনা। এইচটিএমএল ইনপুট ফিল্ডটি যদি খালি থাকে তবে একটা এলাট মেসেজ দেখাবে এবং ডাটা সার্ভারে সাবমিট করবে না। নিচের উদাহরণটি দেখুন-

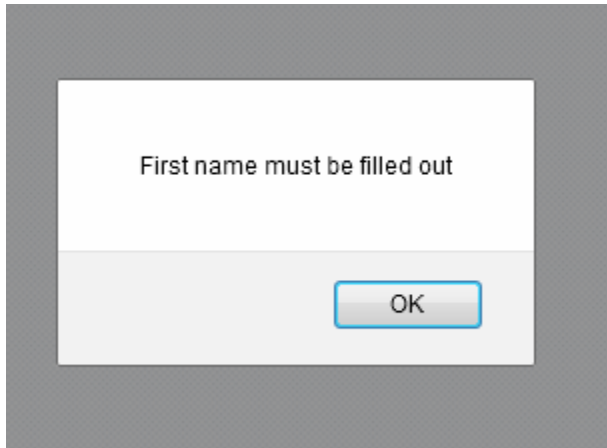
```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function validateForm()
{
var x=document.forms["myForm"]["fname"].value;
if (x==null || x=="")
{
alert("First name must be filled out");
return false;
}
}
```

```
}  
</script>  
</head>  
  
<body>  
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm()" method="post">  
First name: <input type="text" name="fname">  
<input type="submit" value="Submit">  
</form>  
</body>  
  
</html>
```

নিচের মত আউটপুট দেখাবে-

First name:

কোন ডাটা ইনপুট না করে সাবমিট করলে নিচের মত এলাট মেসেজ দেখাবে-



ইমেইল ভেলিডেশন:

এখন আপনাদের দেখাবো যে, ব্যবহারকারীর ইমেইলটি বৈধ কিনা তা কিভাবে আপনি চেক করতে পারবেন। একটি বৈধ ইমেইলে অবশ্যই একটি @ সাইন এবং ডট (.) থাকতে হবে যেখানে @ সাইনটি ইমেইল এর প্রথম ক্যারেক্টার হবে না এবং শেষের ডট (.) টি অবশ্যই @ সাইনের পরে হবে এবং অবশ্যই ডটের পরে অন্ততপক্ষে দুটি ক্যারেক্টারের হবে।

Valid উদাহরণ:

- bobby.jo@filltank.net

Invalid উদাহরণ:

- @deleted.net - @ sign এর আগে কোন characters নেই।
- free!dom@bravehe.art - invalid character !
- shoes@need_shining.com - domain name এ underscores ব্যবহৃত হয় না।

একটি উদাহরণ দেখুন-

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function validateForm()
{
var x=document.forms["myForm"]["email"].value;
var atpos=x.indexOf("@");
var dotpos=x.lastIndexOf(".");
if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
{
alert("Not a valid e-mail address");
return false;
}
}
</script>
</head>

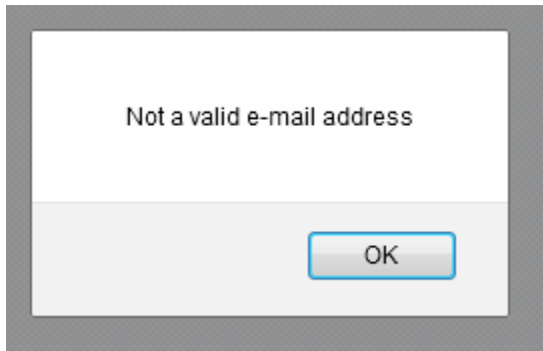
<body>
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm();"
method="post">
Email: <input type="text" name="email">
<input type="submit" value="Submit">
</form>
</body>
```

</html>

আউটপুটঃ

Email:

ফিল্ডে আপনি যদি ভ্যালিড ইমেইল না লেখেন তবে নিচের মত মেসেজ দেখাবে-



জাভাস্ক্রিপ্ট গেটএলিমেন্টবাইআইডি

আপনি কি কখনও জাভাস্ক্রিপ্ট ব্যবহার করে ফর্ম ভেলিডেশন করেছেন? টেক্সট ফিল্ডের কোন ভেলু যাচাই করতে কোন সমস্যায় পড়েছেন? একটা সহজ উপায়ের মাধ্যমে এইচটিএমএল এলিমেন্ট নিয়ন্ত্রন করতে পারেন। আইডি এট্রিবিউট এবং গেটএলিমেন্টবাইআইডি ফাংশনের মাধ্যমে তা করতে পারেন।

জাভাস্ক্রিপ্ট: ডকুমেন্ট.গেটএলিমেন্টবাইআইডি (document.getElementById)

```
<html>
```

```
<head>
```

```
<script type="text/javascript">
```

```
function notEmpty(){
```

```
var myTextField = document.getElementById('myText');
```

```
if(myTextField.value != "")
```

```
alert("You entered: " + myTextField.value)
```

else

```
alert("Would you please enter some text?")
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<input type='text' id='myText' />
```

```
<input type='button' onclick='notEmpty()' value='Form Checker' />
```

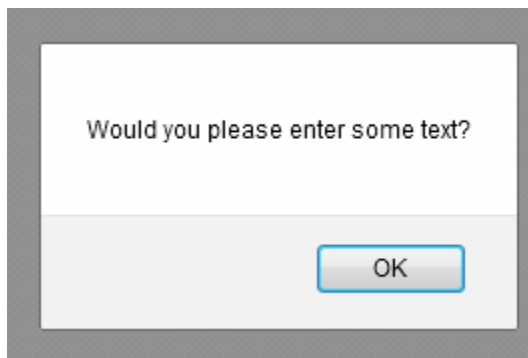
```
</body>
```

আউটপুটঃ

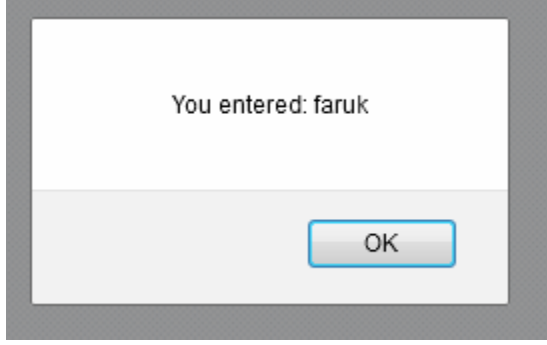


A screenshot of a web form. It features a single-line text input field on the left, which is currently empty. To the right of the input field is a button labeled "Form Checker".

কোন ইনপুট না দিয়ে বাটনে ক্লিক করলে নিচের মত মেসেজ দেখাবে-



আর ইনপুট দিয়ে বাটনে ক্লিক করলে নিচের মত মেসেজ দেখাবে-



`document.getElementById` টি মাইটেব্রট এইচটিএমএল এলিমেন্ট এর রেফারেন্সকে রিটার্ন করে। আমরা এই রেফারেন্সকে `myTextField` নামক ভেরিয়েবলে জমা করতে পারি।

গেটএলিমেন্টবাইআইডি বিষয়ে যা মনে রাখা দরকার:

যখন আপনি `getElementById` function ব্যবহার করবেন তখন আপনাকে কিছু বিষয় সম্বন্ধে নিশ্চিত হওয়া প্রয়োজন। আপনাকে অবশ্যই মনে রাখা প্রয়োজন যে গেটএলিমেন্টবাইআইডি হচ্ছে ডকুমেন্ট অবজেক্ট এর পদ্ধতি বা ফাংশন। আপনি এই গেটএলিমেন্টবাইআইডি ফাংশন ব্যবহার করতে চাইলে আপনাকে অবশ্যই এইচটিএমএল এলিমেন্ট এর আইডি এট্রিবিউট ব্যবহার করতে হবে।

অধ্যায়ঃ সতের- একনজরে জাভাস্ক্রিপ্ট

01. প্রথমে জাভাস্ক্রিপ্ট ডিফাইন করতে হবে।

```
<script type = "text/javascript">  
এখানে জাভাস্ক্রিপ্ট কোড লিখতে হবে।  
</script>
```

02. ভেরিয়েবল ডিফাইন করতে হবে। যেমনঃ

```
var variable_name
```

```
//JavaScript is loosely typed language
```

03. স্টেটমেন্ট লিখতে হবে যা সেমিকোলন দিয়ে শেষ হবে (:)

04. To PopUp a window:

```
alert("Hello world"); // একটি পপআপ উইন্ডো প্রদর্শন করবে।
```

```
// ডাবল বা সিঙ্গেল উভয় কোটেশান ব্যবহার করা যাবে।
```

05. ফাংশান ডিফাইন করতে হবে। যেমনঃ

```
function function_name ()  
{
```

```
Your code here  
}
```

06. স্ক্রীনে যথেষ্ট পরিমাণ প্রশস্ততা আছে কিনা তা পিছেলে পরিমাপ করবে।

```
screen.avaiWidth; //Returns the Screen Width in Pixel
```

07. Finding available Height of screen in pixel

```
screen.avaiHeight; //Returns the Screen Height in Pixel
```

08. Writing Something in the HTML

```
document.write("Hello World"); //Write Hello World to the document
```

09. Writing a Prompt box to take value.

```
prompt("label", "default value"); //Prompt to the user screen.
```

10. Getting the type of a variable

```
typeof(Variable Name); //Returns the type of a variable
```

11. To change data type into Number

```
Number(Variable Name); //Returns the number type value
```

```
parseInt(Variable Name); //Returns the Integer type value
```

```
parseFloat(Variable Name); //Returns the Float type value
```

12. Finding the value of a variable

```
variablename.length; //Returns the length of a variable.
```

13. Creating an object

//new keyword is used to create an object

```
var a = prompt("Enter a numbjer", ""); //Taking an String from the user
```

```
document.write(typeof(a)); // Writing the type of the String
```

```
var b = new String(a); //Creating an String Object Explicitly
```

using String Construction

```
document.write(typeof(b)); // Writing the type of variable b
```

14. The String Methods: indexOf() //Takes string as arguments and returns its index number

```
var a = prompt("Enter your Email Address", "");
```



```
var b = a.indexOf("@"); //Returns the index number of @, if not
found it returns -1
alert("The Index of @ is : " + b);
```

15. The String Methods: `substring()` //Takes two integer arguments as start and ends of the string

```
var a = "Hello World";
var b = a.substring(2,8); //second Parameter can be left off
```

16. **The Math Objects:** It can't be created explicitly. It doesn't store data

17. The **PI** Property: //Returns the value of PI

```
alert("The value of PI is " + Math.PI);
```

18. **round()** Methods: //Rounds a number when the decimal is .5 or up.

```
alert(Math.round(3.4)); //Alerts 3
alert(Math.round(3.6)); //Alerts 4
```

19. **ceil()** Methods: Always round a number up

```
alert(Math.ceil(2.15)); //Alerts 3
```

20. **floor()** Methods: Always round a number down

```
alert(Math.floor(2.15)); //Alerts 2
```

21. **random()** Methods: //Returns a random number between 0 & 1

```
alert(Math.random()); //Alerts a random number to the screen.
```

22. **Array()** constructor: It is used with **new** keywords to create array object.

```
var myarray = new Array(); //With no Predefined elements
var myarray = new Array(3); //with Predefined elements
var myarray = [1,2,3]; //Directly assign by value
var myarray = ["red","Bleu","Green"]; //Directly assign by value
var myarray["ruby"] = "Tiger"; //Assigning array index a name
```

23. **slice()** Method: //Slice and array and the new array begins with zero index

```
var myarray = [1,2,3,4,5,6];
var sarray = myarray.slice(1, 5); //1: First element's index, 5:last
Element's index+1
for(i=0; i<4; i++) //Total 4 as (5-1) = 4
document.write(sarray[i]+"<br />");
```

24. Array Method **concat()**: Joined array elements to form a new array, Property **length**: Returns the length of an object.

```
var arrayone = [1,2,3,4,5];
var arraytwo = [6,7,8];
var arraythree = [9,10,11,12];
var fullarray2 = arrayone.concat(arraythree); //Concatenate arrayone and
arraythree
alert(fullarray2.length); //Alerts the length: 9
var fullarray = arrayone.concat(arraytwo,arraythree);
//Concatenate arrayone arraytwo, and arraythree
var alength = fullarray.length; //Returns the length: 12
alert(alength);
document.write(arrayone.concat(arraytwo,arraythree)); //Writes the full array
elements
```

25. Converting an array to a string: the **join()** method. The methods use a string as a parameter. This parameter is used to separate the array elements.

```
var myarray = ["Abul","Babul","Rahul"];
var mystring = myarray.join("-");
//myarray turns into string seperated with hyphen (-) and store in
mystring
document.write(mystring);
```

26. **split()**: This method is used to turn an string into an array. A Parameter is used to separate one element from other.

```
var mystring = "Abul, Babul, Rahul";
var myarray = mystring.split(",");
var n = myarray.length;
for(i=0; i<n; i++)
document.write(myarray[i]+"<br />");
```

27. **sort()** and **reverse()** method. Short and Reverse an array

```
document.write("The array before sorting<br />");
var myarray = [3,2,7,4,9,5,6,8];
var n = myarray.length;
for(i=0; i<n; i++)
document.write(myarray[i]+"<br />");

document.write("The array after sorting <br />");
var shortedarray = myarray.sort(); //Shorts the array
```

```
for(i=0; i<n; i++)
document.write(shortedarray[i]+"<br />");

document.write("The array after Reversed <br />");
var reversedarray = shortedarray.reverse(); //Reverse the array
for(i=0;i<n; i++)
document.write(reversedarray[i] + "<br />");
```

28. **toUpperCase()** and **toLowerCase()** Methods

```
var strname = prompt("Enter your Name", "");
var name2 = strname.toUpperCase(); //Returns all Upper Case Letter.
document.write(name2 + "<br />");
var name3 = strname.toLowerCase(); //Returns all Lower Case Letter
document.write(name3);
```

29. **valueOf()** Method:// Return the value of the object.

```
var name1 = prompt("Enter your Name", "");
var obj1 = new String(name1);
document.write(obj1.valueOf());
```

30. Logical Operators: **AND(&&)**, **OR(II)**, **NOT(!)**

31. Conditional Statement: **if... else**

```
if (condition)
    { statement }
else
    {statement}
```

32. **break** and **continue** //used in the loop

33. **switch** statement

```
switch(expression)
{
    case somevalue:
        //Execute this code if expression = somevalue
        break;
    case value2:
        //Execute this code if expression = value2
        break;
    default:
```

```
        //Execute some code
        break;
    }
```

Example:

```
var number = Number(prompt("Enter a number among 1 to 4", ""));
switch(number)
{
    case 1:
        document.write("You have entered number 1");
        break;
    case 2:
        document.write("You have entered number 2");
        break;
    case 3:
        document.write("You have entered number 3");
        break;
    case 4:
        document.write("You have entered number 4");
        break;
    default:
        document.write("You have not entered number between 1 to 4");
        break;
}
```

34. the **for** loop: **for(initial-condition; loop-condition; alter-condition) {statement}**

```
var a = prompt("Enter a number", "");
for(i=0; i<=a; i++) {
    document.write(i + "<br />");
}
```

35. the **for... in** loop. //Only for Array

```
var myarray = [1,2,3,5,7,8];
var index;
for(index in myarray){
    document.write("The index of the array is "+index + "<br />");
    document.write("The value of the array is " + myarray[index] + "<br /><br />");
}
```

36. the **while** loop.

```
while(condition)
    { code to be executed; }
```

37. **do ... while** loop.

```
do {
    code to be executed
} while (condition)
```

38. Import in css: `<style type="text/css">@import 'style.css';</style>`

39. Including External JavaScript: `<script type="text/javascript" src="jscript.js"></script>`

40. **navigator** object. **appName** property and **appVersion** property.

```
document.write("Your Browser name is : "+ navigator.appName);           //
Returns browser Name
document.write("<br />Your Browser Version is : " + navigator.appVersion); //Returns
browser version
```

41. Reading out the browser name and version is commonly known as **browser sniffing**

42. **camelCase**: Lowercase first word's letter and Uppercase every first letter of the world.

43. Create a function:

```
function functionname()
{
    //Your Code here
}
```

44. `getElementByTagName()`: //Gets the elements by the tag name;

```
function countnum(){
    var listitems = document.getElementsByTagName('li');
    var nbr = listitems.length;
    alert("The Length number of Li is "+nbr);
}
window.onload = countnum;
```

45. Get the first Paragraph:

```
var firstParagraph = document.getElementsByTagName('p')[0];
```

46. **Node type** value 1 = element node, 3 = text node. (Totally 12).

47. **Node name** can be uppercase or lowercase. You should convert it to lowercase.

```
if(obj.nodeName.toLowerCase()=='li'){}
```

48. **Node Value** is the value of Node. If element than null. Else if text node that the text.

49. In case of **Text Node**, Node Value can be read and set.

50. YourElements.**firstChild** &

your Elements.**lastChild**.

```
yourElements.childNodes[0]
```

```
yourElements.childNodes[yourElements.childNodes.length-1]
```

51. To check the children --

```
hasChildNodes()
```

52. Find Parent Node:

```
var parent = document.getElementById('linked');
```

```
alert(parent.parentNode.nodeName);
```

53. **nextSibling** & **previousSibling**

54. **getAttribute()** & **setAttribute()** //get the attribute value and set the attribute value

55. document.

আল্লাহ হাফেজ
সমাপ্ত