

1

जावास्क्रिप्ट JAVASCRIPT

guru rana

वेब में उपयोगिता

वर्तमान समय में वेबसाइट को आकर्षक बनाने के लिए डायनमिक वेबसाइट का चलन है, इसके लिए छोटे-छोटे प्रोग्राम खण्डों या जावास्क्रिप्ट की आवश्यकता होती है।

परिचय (Introduction)

जावास्क्रिप्ट का विकास नेटस्केप कम्युनिकेशन्स कम्पनी के ब्रैंडन ईच (Brendan Eich) द्वारा मई, 1995 में मात्र 10 दिनों में किया गया था। जावास्क्रिप्ट की सहायता से वेबसाइट में मल्टीमीडिया एलिमेंटों को सरलता से जोड़ा जा सकता है। वेबसाइट को डायनमिक लुक देने के लिए यूजर का इन्ट्रैक्शन सरलता से हो पाता है।

1.1 एल्गोरिथम (Algorithm)

यह किसी कार्य को करने के लिए एक विशेष क्रम में लिखे गए आदेशों का समूह होता है। यह आदेश इस प्रकार लिखा जाता है कि कोई यूजर उसे समझकर उसी क्रम में सही तरीके से पालन करता जाए, तो कार्य पूरा हो जाता है। “अपनी भाषा में प्रोग्राम की गतिविधियों का प्रतिरूप तैयार करना एल्गोरिथम कहलाता है।” एल्गोरिथम में जोड़ने, घटाने, गुणा एवं भाग की क्रियाओं के लिए क्रमशः +, -, * एवं / के चिह्नों का प्रयोग किया जाता है। इसी प्रकार हम घात के चिह्न के रूप में “**” का प्रयोग करते हैं, जैसे X^2 को हम $X * X$ लिखेंगे।

उदाहरण, हमें किसी वृत्त की त्रिज्या ज्ञात होने पर उसकी परिधि तथा क्षेत्रफल ज्ञात करना है। हम जानते हैं कि अगर किसी वृत्त की त्रिज्या r हो, तो उसकी परिधि (P) $2\pi r$ एवं क्षेत्रफल (A) πr^2 होता है, जहाँ π का मान लगभग 3.1416 होता है। इसके लिए एल्गोरिथम को निम्न प्रकार से लिखा जा सकता है—

1. Read r
2. $P = 2 * 3.1416 * r$
3. $A = 3.1416 * r * r$
4. Print P, A
5. Stop

आप त्रिज्या r का कोई भी मान लेकर सत्यता की जाँच कर सकते हैं।

एल्गोरिथम का विश्लेषण (Analysis of Algorithm)

एल्गोरिथम विश्लेषण प्रोग्राम का निष्पादन (Execution) करने के लिए आवश्यक संसाधनों (जैसे समय और भंडारण) का निर्धारण (Determination) करता है। विशेष तौर पर, एक एल्गोरिथम की दक्षता या रनिंग समय (Running Time) उस एल्गोरिथम की क्रियाओं की संख्या तथा भंडारण स्थानों की संख्या (Storage Locations) पर निर्भर करती है। एल्गोरिथम का विश्लेषण उस एल्गोरिथम की टाइम तथा स्पेस कॉम्प्लैक्सिटी (Time and Space Complexity) को ज्ञात करके किया जाता है।

टाइम कॉम्प्लैक्सिटी (Time Complexity)

एक एल्गोरिथम की टाइम कॉम्प्लैक्सिटी, उस एल्गोरिथम द्वारा किसी दिए गए इनपुट के लिए निष्पादित (Execute) होने में लगे समय की मात्रा होती है।

स्पेस कॉम्प्लैक्सिटी (Space Complexity)

एक एल्गोरिथ्म की स्पेस कॉम्प्लैक्सिटी, उस एल्गोरिथ्म द्वारा किसी दिए गए इनपुट के लिए निष्पादित (Execute) होने में लगे कुल भंडारण स्थान (Total Storage Space) की मात्रा को कहते हैं।



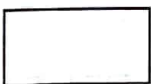
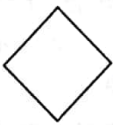
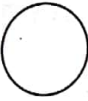
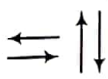
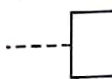
सामान्यतः दो प्रकार के टूल, एल्गोरिथ्म को डॉक्युमेंट के रूप में प्रदर्शित करने में सहायक होते हैं—

1. फ्लोचार्ट (Flowchart)
2. स्यूडोकोड (Pseudocode)

फ्लोचार्ट (Flowchart)

फ्लोचार्ट एल्गोरिथ्म लिखने की विधि होती है, जिसमें एल्गोरिथ्म के आदेशों (Commands) को विशेष प्रकार की आकृतियों के रूप में दर्शाया जाता है। इसमें हम उन प्रतीकों (Symbols) का प्रयोग करते हैं जो स्टार्ट, इनपुट, प्रोसेस, डिजीजन, कनेक्टर, आउटपुट, गति की दिशा एवं स्टॉप आदि को दर्शाते हैं। अलग-अलग कथनों के लिए अलग-अलग आकृतियाँ (Shapes) होती हैं तथा उन आकृतियों के अन्दर उस कथन को संक्षेप में लिखा जाता है। इन आकृतियों को उनके पालन के क्रम की दिशा में तीर (↓) के चिह्नों द्वारा जोड़ दिया जाता है।

“किसी भी समस्या का एल्गोरिथ्म तैयार करने के बाद उसका चित्रों के माध्यम द्वारा प्रदर्शन ही फ्लोचार्ट कहलाता है।” फ्लोचार्ट बनाने में प्रयोग की जाने वाली मुख्य आकृतियाँ निम्नलिखित हैं—

सिम्बल	नाम	कार्य
	टर्मिनल	फ्लोचार्ट का प्रारम्भ तथा अन्त दर्शाने के लिए
	इनपुट/आउटपुट	डेटा को इनपुट तथा आउटपुट कराने के लिए
	प्रोसेस	सभी प्रकार की गणनाओं के लिए
	डिजीजन	निर्णय लेने के लिए अथवा तुलना करने के लिए
	कनेक्टर	फ्लोचार्ट को जोड़ने के लिए
	फ्लो लाइन	कथन के पालन की दिशा दर्शाने के लिए
	कमेण्ट	कमेण्ट देने के लिए

फ्लोचार्ट के प्रकार (Types of Flowchart)

फ्लोचार्ट मुख्यतः दो प्रकार के होते हैं—

प्रोग्राम फ्लोचार्ट (Program Flowchart)

कम्प्यूटर प्रोग्राम लिखने के लिए प्रयोग किया जाने वाला फ्लोचार्ट प्रोग्राम फ्लोचार्ट कहलाता है। इसे इतने विस्तार से लिखा जाता है कि कम्प्यूटर प्रोग्राम लिखना सरल हो जाता है। इसमें गणितीय गणनाएँ, लॉजिकल ऑपरेशन, लूपिंग एवं ब्रान्चिंग का प्रयोग किया जाता है।

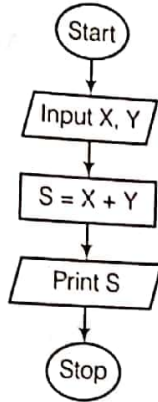
सिस्टम फ्लोचार्ट (System Flowchart)

इस प्रकार के फ्लोचार्ट का प्रयोग एनालिस्ट द्वारा किया जाता है जो यह दर्शाते हैं कि किस मशीन से डेटा आया तथा कहाँ प्रोसेस किया जाएगा एवं परिणाम कहाँ भेजा जाएगा।

फ्लोचार्ट का महत्त्व (Importance of Flowchart)

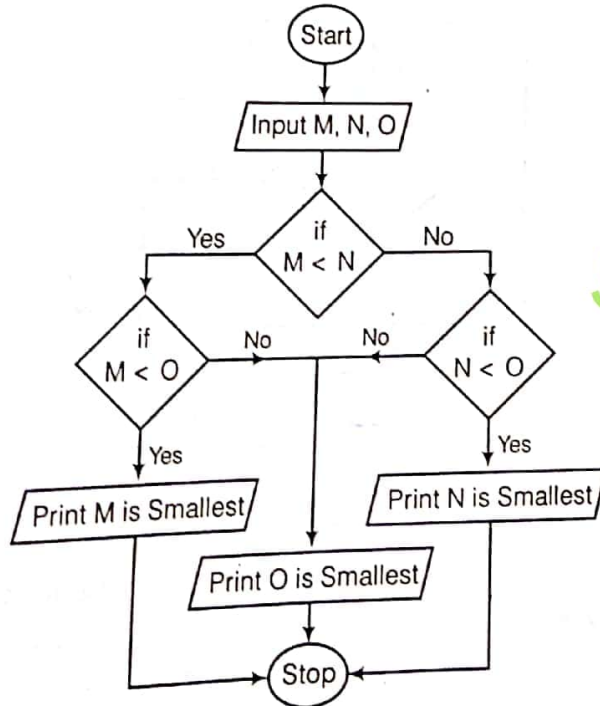
साधारणतः प्रोग्रामर द्वारा किसी प्रोग्राम को लिखने के लिए फ्लोचार्ट का प्रयोग किया जाता है। इसकी सहायता से कठिन-से-कठिन प्रोग्रामों को सरल बनाया जा सकता है। अंग्रेजी भाषा में लिखे जाने के कारण प्रोग्राम को समझना बहुत सरल होता है। इसमें, हल करने के तरीकों की क्रमबद्ध डिटेल्स होती हैं। इसमें प्रोग्राम के कण्ट्रोल का नियन्त्रण, लूपिंग आदि का ज्ञान भी सरल हो जाता है।

उदाहरण दो संख्याओं को इनपुट कीजिए एवं उनके योग को स्क्रीन पर दर्शाए।
हल इसके लिए हम निम्नलिखित फ्लोचार्ट रेखांकित करते हैं—



प्रत्येक फ्लोचार्ट का हम टर्मिनल के द्वारा प्रारम्भ एवं अन्त करते हैं। दिए गए फ्लोचार्ट की दूसरी आकृति समान्तर चतुर्भुज है, जिसमें X तथा Y वेरिएबल के इनपुट का कथन लिखा गया है। अगली आकृति आयत है, जिसमें X तथा Y को जोड़कर एक अन्य वेरिएबल S में स्टोर का कथन लिखा गया है। इससे अगली आकृति समान्तर चतुर्भुज में आउटपुट के लिए, Print S लिखा गया है। फ्लोचार्ट की सभी आकृतियों को फ्लो लाइन द्वारा जोड़ा गया है, जिससे पता चलता है कि कथनों के पालन का क्रम क्या है।

उदाहरण तीन संख्याओं को इनपुट कीजिए एवं उनमें से सबसे छोटी संख्या ज्ञात करने हेतु फ्लोचार्ट खींचिए।
हल

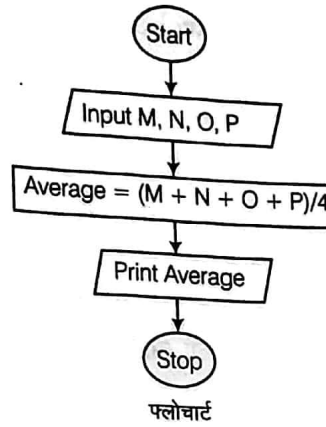


guru rana

स्यूडोकोड (Pseudocode)

स्यूडोकोड सिर्फ शब्द होते हैं इसलिए इसे किसी कम्प्यूटर फाइल में रखना फ्लोचार्ट की अपेक्षा सरल होता है। इसे वर्ड प्रोसेसिंग की सहायता से सरलतापूर्वक बदला भी जा सकता है। सूडोकोड प्रोग्राम लॉजिक को परिभाषित एवं उसका अनुसरण करने का एक उत्तम माध्यम है।

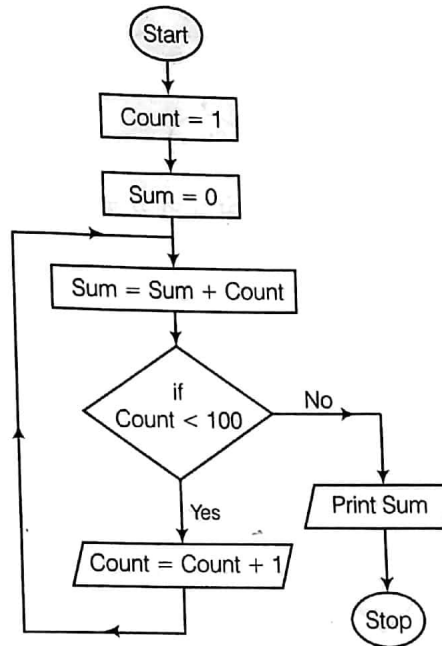
guru rana



फ्लोचार्ट एवं सूडोकोड Flowchart and Pseudocode प्रोग्राम के लॉजिक को चित्रानुसार रेखांकित करना फ्लोचार्ट (Flowchart) एवं लॉजिक के पदों को क्रमानुसार अपनी भाषा में लिखना सूडोकोड (Pseudocode) कहलाता है।

1. Start
2. Input M, N, O, P
3. Average = $(M + N + O + P)/4$
4. Print Average
5. Stop

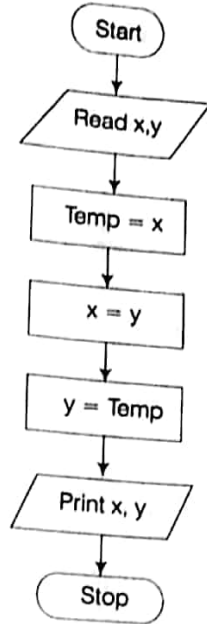
उदाहरण 1 से 100 तक की संख्याओं का योगफल स्क्रीन पर दर्शाएँ तथा इसका सूडोकोड भी लिखिए।
हल

**स्यूडोकोड**

1. Start
2. Count = 1
3. Sum = 0
4. Sum = Sum + Count
5. If Count < 100 then go to step 4
6. Count = Count + 1
7. Go to step 4
8. Print Sum
9. Stop

Output = 5050

उदाहरण एक समस्या के लिए एक फ्लोचार्ट का निर्माण कीजिए, जिसमें दो चर (variables) हों एवं उन दोनों के तत्वों का आपस में हस्तान्तरण भी करना हो।
हल फ्लोचार्ट द्वारा तत्वों का हस्तान्तरण



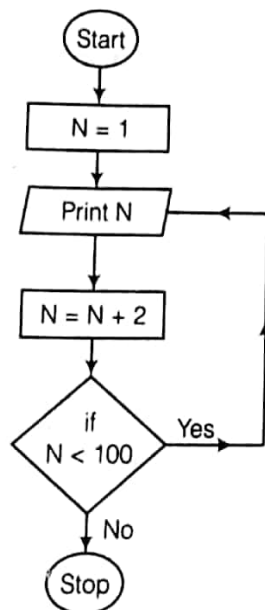
guru rana

उदाहरण 1 से 100 के मध्य की सभी विषम संख्याओं को दर्शाने के लिए एल्गोरिथ्म तथा फ्लोचार्ट बनाइए।
हल हम जानते हैं कि विषम संख्याएँ 1, 3, 5, ... आदि होती हैं। हम एक चर N=1 मान लेते हैं।

एल्गोरिथ्म

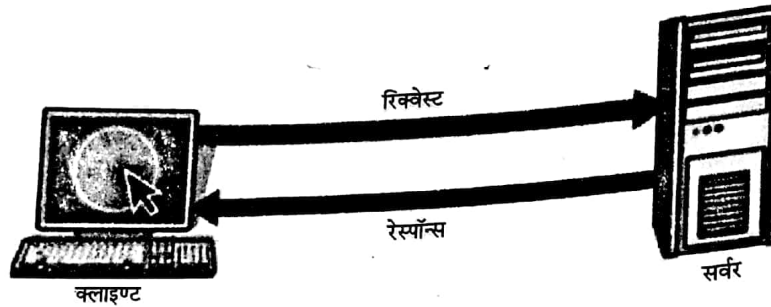
- Step 1. N=1
- Step 2. Print N
- Step 3. N = N + 2
- Step 4. If N<100, then go to step 2
- Step 5. Stop

फ्लोचार्ट



वेब सर्वर (Web Server) ✓

वेब सर्वर एक प्रोग्राम होता है जो क्लाइंट/सर्वर मॉडल और World Wide Web (WWW) के Hyper Text Transfer Protocol (HTTP) का प्रयोग कर वेब यूजर को वेबपेज की सेवा प्रदान करता है। क्लाइंट/सर्वर मॉडल में क्लाइंट एक प्रकार का प्रोग्राम होता है जिसके द्वारा वेब यूजर (वेब उपयोगकर्ता) सर्वर को वेबपेज (Webpage) एक्सेस करने की रिक्वेस्ट (Request) भेजता है। उदाहरण के लिए वेब ब्राउज़र (web browser) एक क्लाइंट प्रोग्राम है, जिसके द्वारा वेब यूजर सर्वर (Server) को वेबपेज को उपयोग करने की रिक्वेस्ट (Request) भेजता है। इसके पश्चात् वेब ब्राउज़र का मुख्य कार्य होता है कि वेब ब्राउज़र सर्वर के रिसपॉन्स (response) को वेब यूजर तक सही समय में पहुँचा दे। इस पूरी प्रक्रिया को निम्न चित्र द्वारा समझाया जा सकता है



इण्टरनेट (Internet) पर किसी कम्प्यूटर में यदि वेबसाइट स्टोर की गई है, तब उसमें वेब सर्वर (Web server) अवश्य होना चाहिए, क्योंकि बिना वेब सर्वर के इस वेबसाइट को कोई भी क्लाइंट प्रोग्राम या वेब यूजर एक्सेस नहीं कर सकता है। वेब सर्वर प्रायः इण्टरनेट के बड़े पैकेजों के साथ आते हैं, जिनमें E-mail, FTP (File Transfer Protocol) फाइल डाउनलोडिंग एवं वेबसाइट बनाने तथा पब्लिश करने के विकल्प भी विद्यमान होते हैं। वेब सर्वर का चयन करते समय यह ध्यान रखना चाहिए कि यह ऑपरेटिंग सिस्टम (Operating System) और अन्य सर्वरों के साथ किस प्रकार कार्य करता है। सर्वर साइड प्रोग्रामिंग को हैंडल करने की क्षमता कैसी है, इसकी सुरक्षा विशेषताएँ कितनी अच्छी हैं तथा यह सर्च इंजन को किस प्रकार सूचनाएँ प्रदान करने में सक्षम होता है। कई बार वेब सर्वर के साथ साइट-बिल्डिंग टूल (Tool) भी आते हैं। अतः उनकी गुणवत्ता (Quality) भी देखनी चाहिए। वेब सर्वर के प्रमुख उदाहरण निम्नलिखित हैं

सबसे व्यापक रूप से स्थापित (Widely-installed) किया गया वेब सर्वर अपाचे (Apache) है। इसके अतिरिक्त कुछ प्रमुख वेब सर्वर माइक्रोसॉफ्ट का आई.एस.एस. (ISS-Internet Information Server), नोवेल (Novell) का वेब सर्वर (NetwareOS के लिए) एवं आई.बी.एम. (IBM-International Business Machines) का लोटस डोमिनो सर्वर (Lotus Domino Server) आदि हैं।

वेब सर्वर की विशेषताएँ (Features of Web Server)

वेब सर्वर (webservers) को वेब साइट होस्टिंग के लिए बनाया गया है। अतः इसकी मुख्य विशेषता (Feature) वेबसाइट होस्टिंग एनवायरमेंट बनाना एवं इसे मेन्टेन करना है। अधिकतर वेब सर्वर कुछ प्रमुख विशेषताएँ (features) वेबसाइट होस्टिंग एनवायरमेंट बनाना एवं इसे मेन्टेन करना है। अधिकतर वेब सर्वर कुछ प्रमुख विशेषताएँ (features) रखते हैं जो निम्न हैं

1. वेबसाइट (Website) बनाने की सुविधा
2. लॉग फाइल (Log File) सेटिंग को कन्फिगर करने की सुविधा
3. वेब साइट या डायरेक्ट्री सुरक्षा (Security) की पहचान करना
4. FTP साइट बनाना (FTP साइट के द्वारा यूजर फाइल को साइट पर अपलोड कर सकते हैं एवं साइट से डाउनलोड भी कर सकते हैं।
5. एरर पेज (Error Page) की पहचान करना। इसके द्वारा User friendly error messages साइट पर डिस्प्ले किए जाते हैं, जैसे—यदि यूजर ऐसे पेज को एक्सेस करने की रिक्वेस्ट (Request) करता है, जो सर्वर के पास उपलब्ध नहीं होता है तो सर्वर (Server) "404 page not found error" पेज पर प्रदर्शित करता है।
6. डिफॉल्ट डॉक्यूमेंट को निर्धारित करना।

guru rana

1.2 कम्प्यूटर प्रोग्रामिंग (Computer Programming) ✓

कम्प्यूटर एक इलेक्ट्रॉनिक मशीन है जो कोई भी कार्य करने के लिए आदेशों के द्वारा बाध्य होता है। वह केवल उसको दिए गए आदेशों का उसी क्रम में एक-एक करके पालन करता है। हम जानते हैं कि कम्प्यूटर किसी सरल पदों को व्यक्त करने के लिए निर्देशों का प्रयोग करता है, जिन्हें प्रोग्रामिंग भाषा में लिखा जाता है। इन निर्देशों के समूह को ही प्रोग्राम (Program) कहा जाता है तथा किसी कम्प्यूटर के लिए प्रोग्राम लिखने की क्रिया को प्रोग्रामिंग कहते हैं। कम्प्यूटर को कम-से-कम आदेश देकर उससे सही-सही अधिक कार्य करा लेना एक उपयोगी कला है एवं उसके द्वारा बड़े-बड़े कार्य करा लेना, प्रोग्रामिंग की कला का ही चमत्कार है।

कम्प्यूटर की सहायता से किसी समस्या को हल करने के लिए क्रियाओं एवं पदों को क्रमवार करना ही **कम्प्यूटर प्रोग्रामिंग** कहलाता है। जो व्यक्ति कम्प्यूटर के लिए प्रोग्राम लिखते हैं या तैयार करते हैं, उन्हें **प्रोग्रामर (Programmer)** कहा जाता है। प्रोग्रामर कम्प्यूटर प्रोग्रामिंग के लिए कुछ विशेष भाषा का प्रयोग करते हैं जिसे **प्रोग्रामिंग भाषा (Programming Language)** कहा जाता है; उदाहरण, C, C++, JAVA इत्यादि।

प्रोग्रामिंग के चरण (Steps of Programming)

कम्प्यूटर के लिए प्रोग्राम लिखना एक कला है। किसी कार्य के लिए प्रोग्राम लिखने से पहले पूरी योजना बनानी पड़ती है। समस्या को सुनते ही प्रोग्राम लिखने लग जाना गलत है। यह गलती प्रायः नए प्रोग्रामर किया करते हैं, फलस्वरूप उनका अधिक समय उस प्रोग्राम की गलतियाँ ढूँढने एवं उन्हें ठीक करने में लग जाता है। इसके विपरीत सफल प्रोग्रामर वे होते हैं, जो प्रोग्राम की योजना बनाकर क्रमबद्ध तरीके से प्रोग्राम लिखते हैं और वे प्रायः काफी कम समय में सफल प्रोग्राम तैयार कर लेते हैं। अतः एक सही प्रोग्राम लिखने के लिए कुछ महत्वपूर्ण चरणों का क्रमशः पालन किया जाना चाहिए। ये महत्वपूर्ण चरण निम्नलिखित हैं—

(i) समस्या को समझना (Understanding the Problem)

जब तक किसी समस्या को पूर्ण रूप से समझ न लिया जाए, उसको हल करना या उसे हल करने की कोशिश करना सम्भव नहीं है। यह एक तथ्य है कि प्रोग्राम लिखने के लिए प्रायः जो समस्याएँ दी जाती हैं, वे कभी भी पूर्ण नहीं होतीं, कोई-न-कोई बात उनमें छूट ही जाती है। अनुभवी प्रोग्रामर, उस बात को शीघ्र समझकर या तो अतिरिक्त सूचनाएँ माँग लेते हैं या उसके स्थान पर अपने ज्ञान के अनुसार कल्पना कर लेते हैं। अतः समस्या को ठीक-ठीक समझना अनिवार्य है।

(ii) इनपुट डेटा का परीक्षण करना (Examining the Input Data)

हम जानते हैं कि प्रोग्राम इनपुट डेटा पर क्रिया करने के लिए लिखे जाते हैं इसलिए जब तक आपको यह पता नहीं होगा कि इनपुट किस रूप में होगा और किस प्रकार प्राप्त होगा, तब तक आप प्रोग्राम नहीं लिख सकते हैं इसलिए इस समस्या को हल करने के लिये सभी आवश्यक इनपुट का सही रूप में उपलब्ध होना अतिआवश्यक है।

(iii) आउटपुट की योजना बनाना (Planning of the Output)

अगले चरण में हमें आउटपुट की योजना बनानी पड़ती है। इससे यह स्पष्ट हो जाता है कि हमारा प्रोग्राम वास्तव में क्या करना चाहता है। आउटपुट की योजना तैयार करते समय यह भी सुनिश्चित हो जाता है कि हमने समस्या को ठीक से समझ लिया है और यह भी कि हमें जो इनपुट दिया गया है वह माँगे गए आउटपुट के लिए, पर्याप्त है या नहीं।

(iv) एल्गोरिथ्म बनाना (Preparing Algorithm)

यह प्रोग्राम लिखने का सबसे महत्वपूर्ण चरण है। इसमें हम समस्या के हल की क्रमबद्ध रूपरेखा देते हैं। इस चरण में दिए गए इनपुट से माँगे गए आउटपुट तक पहुँचने के सारे चरण क्रमानुसार लिखे जाते हैं। इस क्रमबद्ध लेखन को एल्गोरिथ्म (Algorithm) कहा जाता है। यह वास्तव में साधारण भाषा में लिखा गया प्रोग्राम होता है। आवश्यकतानुसार, प्रोग्राम के उद्देश्य को अनेक भागों में बाँट दिया जाता है और प्रत्येक भाग पर ध्यान केन्द्रित कर उसके लिए अलग-अलग एल्गोरिथ्म तैयार किए जाते हैं और अन्त में सभी एल्गोरिथ्म को मिलाकर एक पूर्ण एल्गोरिथ्म बना लिया जाता है।

(v) फ्लोचार्ट बनाना (Preparing Flowchart)

फ्लोचार्ट, एल्गोरिथ्म तैयार करने व दर्शाने की एक विधि है। इसमें एल्गोरिथ्म के सभी चरणों को विशेष प्रकार की आकृतियों द्वारा दर्शाया जाता है और उन आकृतियों के अन्दर आवश्यक सूचनाएँ लिखी जाती हैं।

(vi) वास्तविक प्रोग्राम लिखना (Writing of an Actual Program)

इस चरण में ऊपर बताए गए एल्गोरिथ्म/फ्लोचार्ट के अनुसार दिए गए प्रोग्राम को प्रोग्रामिंग भाषा में लिखा जाता है। इसके लिए हमें प्रोग्रामिंग भाषा के व्याकरण (Syntax) का ज्ञान होना आवश्यक होता है। प्रोग्राम लिखने के बाद कम्पाइलर (Compiler) द्वारा उसका कम्पाइलेशन कराया जाता है। इससे यदि प्रोग्राम में कोई व्याकरण की गलती रह जाती है, तो पता चल जाता है और उसे ठीक करके फिर से कम्पाइलेशन कराया जाता है। यह प्रक्रिया तब तक दोहराई जाती है जब तक कि प्रोग्राम की सारी गलतियाँ पूर्ण रूप से मुक्त नहीं हो जाती।

प्रोग्राम डॉक्युमेंटेशन

(Program Documentation)

यह एक प्रकार का दस्तावेज होता है जो प्रोग्राम के विषय में व्यापक प्रक्रियात्मक (Procedural) विवरण देता है। यह दर्शाता है कि प्रोग्राम कैसे लिखा गया है? प्रोग्राम डॉक्युमेंटेशन में प्रोग्राम के रख-रखाव तथा उसमें हुए परिवर्तन को बनाए रखने की क्षमता होती है। प्रोग्राम डॉक्युमेंटेशन प्रोग्राम के लिए आवश्यक इनपुट तथा उस इनपुट तथा होने वाली क्रियाओं का उल्लेख करके यह वर्णित करता है कि वास्तव में वह प्रोग्राम क्या करता है?

डि-बगिंग (De-Bugging)

डि-बगिंग किसी कम्प्यूटर प्रोग्राम में त्रुटियों का पता लगाने तथा उनको ठीक करने की प्रक्रिया है। किसी प्रोग्राम को डिबग (Debug) करने के लिए, पहले समस्या के स्रोत को पृथक् किया जाता है और फिर इसे ठीक किया जाता है।

(vii) प्रोग्राम परीक्षण (Testing the Program)

इस चरण में एक परीक्षण डेटा तैयार कर प्रोग्राम का परीक्षण किया जाता है। इसमें हम ऐसा इनपुट लेते हैं, जिसका आउटपुट हमें पहले से पता हो, जिससे हम देख सकें कि प्रोग्राम वही आउटपुट, देता है या नहीं। अगर दोनों में समानता होगी, तो प्रोग्राम सफल कहा जाएगा। परीक्षण डेटा तैयार कर लेने के बाद उस डेटा के लिए प्रोग्राम को चलाकर देखा जाता है और कोई गलती पाए जाने पर प्रोग्राम में आवश्यक सुधार किए जाते हैं। जब परीक्षण का आउटपुट सन्तोषजनक होता है, तो प्रोग्राम को वास्तविक डेटा के साथ प्रयोग करने के लिए दे दिया जाता है।

1.3 स्क्रिप्टिंग भाषा (Scripting Language)

ऐसी प्रोग्रामिंग भाषा को स्क्रिप्टिंग भाषा (Scripting Language) कहते हैं, जिसके द्वारा किसी दूसरे सॉफ्टवेयर अनुप्रयोग (जैसे मोज़िला फायरफॉक्स) पर नियन्त्रण किया जा सके और स्क्रिप्ट के सहारे, उस अनुप्रयोग से अधिक काम लिया जा सके। जावास्क्रिप्ट, पर्ल, पाइथन, रूबी, पीएचपी आदि कुछ प्रमुख स्क्रिप्टिंग भाषाएँ हैं।

उदाहरण, मोज़िला फायरफॉक्स एक ब्राउज़र है जो C/C++ प्रोग्रामिंग भाषा में लिखा हुआ है। इसके ऊपर जावास्क्रिप्ट में कुछ पंक्तियों का प्रोग्राम लिखकर बड़े-बड़े कार्य कराए जाते हैं।

स्क्रिप्टिंग भाषा की विशेषताएँ (Features of Scripting Language)

स्क्रिप्टिंग भाषाएँ अनेक मामले में भिन्न हैं, उनकी प्रमुख विशेषताएँ निम्न प्रकार हैं—

- ये प्रायः कम्पाइल नहीं की जातीं, बल्कि इण्टरप्रेट की जाती हैं।
- इनमें ऐसी विशेषताएँ या फीचर होते हैं, जिससे प्रोग्रामर की उत्पादकता में वृद्धि होती है। मुख्य रूप से इनमें स्वतः स्मृति प्रबन्धन (Automatic Memory Management) की व्यवस्था होती है तथा शक्तिशाली ऑपरेशन (बड़े-बड़े कार्य) करने की सुविधा प्रदान की जाती है (न कि लाइब्रेरी पर आश्रित रहा जाता है)।
- इनमें टाइपिंग के नियम सख्त नहीं होते।
- प्रायः ये किसी अधिक बड़े सॉफ्टवेयर एप्लिकेशन के एक भाग के रूप में प्रयोग के लिए डिज़ाइन की जाती हैं।
- स्क्रिप्टिंग भाषा सीखने में जितनी सरल होती है और उतनी ही सरलता से प्रयोग की जा सकती है।

1.4 जावास्क्रिप्ट से परिचय (Introduction to JavaScript)

जावास्क्रिप्ट (JavaScript) एक प्रोग्रामिंग लैंग्वेज है, जिसका प्रयोग इण्टरनेट (Internet) पर आधारित एप्लीकेशन्स (Applications) बनाने के लिए किया जाता है। यह लैंग्वेज HTML से अधिक सम्पन्न और समृद्ध है, साथ ही HTML की अपेक्षा अधिक सुविधाएँ प्रदान करती है।

जावास्क्रिप्ट एक क्लाइट साइड में रन होने वाली interpreter based scripting language है और interpreter based होने के कारण जावास्क्रिप्ट का अलग से कोई interpreter software नहीं होता, बल्कि जावास्क्रिप्ट प्रोग्राम्स जिस सॉफ्टवेयर में रन होते हैं, उसी सॉफ्टवेयर में ही जावास्क्रिप्ट के इंजन को build किया गया होता है। सामान्यतः वेब ब्राउज़र ही जावास्क्रिप्ट का होस्ट एनवायरनमेंट होता है, लेकिन इसका मतलब ये नहीं है कि जावास्क्रिप्ट के प्रोग्राम्स केवल वेब ब्राउज़र में ही रन हो सकते हैं। वास्तव में सच्चाई ये है कि जिस किसी भी सॉफ्टवेयर में JavaScript engine embedded होता है, हर उस सॉफ्टवेयर में जावास्क्रिप्ट के प्रोग्राम्स हो सकते हैं इसीलिए जावास्क्रिप्ट केवल वेब ब्राउज़र में ही प्रयोग नहीं की जाती, बल्कि JavaScript engine को किसी अन्य platforms पर भी embedded किया जा सकता है, जहाँ जावास्क्रिप्ट के प्रोग्राम रन हो सकते हैं।

उदाहरण, Adobe Flash एक प्रकार का Animation Software है, जहाँ प्रोग्रामिंग भाषा के रूप में ActionScript का प्रयोग किया जाता है। ये भी एक प्रकार की JavaScript Language ही है। इसी प्रकार से Adobe PDF Reader में भी जावास्क्रिप्ट सपोर्ट करती है।

वर्तमान समय में विभिन्न प्रकार के Web Development IDEs उपलब्ध हैं, जैसे Adobe Dream Weaver, Eclipse, NetBeans आदि, इनमें भी JavaScript Engine Embedded है, इसलिए ये भी जावास्क्रिप्ट के Host Environment हैं।

साथ ही वास्तविक प्रोग्रामिंग को प्रयोग करने की सुविधा प्रदान करती है।

जावास्क्रिप्ट लैंग्वेज की मूलभूत विशेषताएँ निम्नलिखित हैं—

1. जावास्क्रिप्ट, एक तीसरी पीढ़ी (Third Generation) की लैंग्वेज है। यह ऑब्जेक्ट ओरिएण्टेड लैंग्वेज, जावा (Java) से प्रेरित होकर विकसित की गई है इसलिए इस लैंग्वेज के सिन्टेक्स (Syntax) को सीखना अत्यन्त सरल है।
2. जावास्क्रिप्ट का कोई डेटा टाइप (Data Type) नहीं होता है।
3. जावास्क्रिप्ट में हम प्वाइण्टर्स को हैण्डल नहीं कर सकते हैं इसकी यह विशेषता डेटा क्षति (Data Loss) की आशंका (Fear) का त्याग (Avoid) करती है। यद्यपि हम किसी ऑब्जेक्ट और वैरिएबल के लिए एक रेफरेंस एसाइन कर सकते हैं।

- इसमें बनाए गए ऑब्जेक्ट की आवश्यकता न होने पर अर्थात् उसका उपयोग समाप्त हो जाने पर उसके द्वारा प्रयोग किए गए मैमोरी में स्थान को मुक्त करने के लिए इसे नष्ट (Destroy) करने की आवश्यकता नहीं होती, क्योंकि इसमें भी जावा (Java) लैंग्वेज के समान ही ऑब्जेक्ट द्वारा प्रयुक्त की गई मैमोरी को मुक्त करने के लिए garbage collection मैथड का ही प्रयोग होता है।
- यद्यपि यह लैंग्वेज ऑब्जेक्ट पर आधारित है, परन्तु यह C++ अथवा Java की भाँति ऑब्जेक्ट ओरिएण्टेड लैंग्वेज नहीं है। इसका अर्थ यह है कि जावास्क्रिप्ट हमें object classes का निर्माण करने की अनुमति प्रदान करती है, ताकि हम ऑब्जेक्ट्स को क्रिएट और हैण्डल कर सकें। साथ ही, हम इस लैंग्वेज द्वारा ऑफर किए गए existing ऑब्जेक्ट्स का प्रयोग कर सकते हैं।
- जावास्क्रिप्ट, इनहेरिटेन्स (Inheritance) और पॉलीमॉर्फिज़्म (Polymorphism) की अवधारणा का सीमित उपयोग ऑफर करती है। चूँकि जावास्क्रिप्ट एक स्क्रिप्टिंग लैंग्वेज है, इसलिए यह हमें अपने वेब पेज पर वास्तविक प्रोग्रामिंग जोड़ने की अनुमति प्रदान करती है। हम जावास्क्रिप्ट के साथ छोटे एप्लीकेशन के प्रोसेसेज़; जैसे – कैलकुलेटर, को क्रिएट कर सकते हैं।
- चूँकि जावास्क्रिप्ट HTML नहीं है, इसलिए हमें HTML पेज पर <SCRIPT> टैग का प्रयोग करना होता है, ताकि ब्राउज़र यह जान सके कि इसमें स्क्रिप्ट लैंग्वेज का प्रयोग किया गया है। <SCRIPT> टैग का ओपनिंग और क्लोजिंग टैग होता है। HTML डॉक्युमेन्ट में टैग में हमें यह घोषित करना होता है कि प्रयोग की जाने वाली स्क्रिप्ट लैंग्वेज कौन-सी है? यह घोषणा निम्नानुसार करनी होती है—

```
<SCRIPT LANGUAGE = "JavaScript">
```

इसके बाद स्क्रिप्ट फंक्शन्स को <!-- और //--> टैग्स में घोषित किया जाता है। HTML डॉक्युमेन्ट के <BODY> टैग में स्क्रिप्ट को घोषित तथा कॉल (Call) किया जाता है परन्तु स्क्रिप्ट को सही तरीके से लिखने के लिए HTML डॉक्युमेन्ट में स्क्रिप्ट कोड्स को <HEAD> टैग के अन्दर ही घोषित किया जाता है और <BODY> टैग्स में इन कोड्स को कॉल (Call) किया जाता है।

जावास्क्रिप्ट का प्रयोग विभिन्न प्रकार के अनुप्रयोगों (Application) को बनाने में किया जाता है।

1. डायनमिक वेबसाइट बनाने में (Create Dynamic Website)
2. पॉपअप बॉक्सज़ (Popup Boxes)
3. मॉउस रोलओवर प्रभाव (Mouse Rollover Effects)
4. मेन्यू (Menu) बनाने में
5. स्टेटस बार (Status Bar) दर्शाने में
6. लिंक के साथ ड्रॉप-डाउन मेन्यू बनाना

1.5 जावास्क्रिप्ट के बेसिक्स (Basics of JavaScript)

जावास्क्रिप्ट के बेसिक्स निम्नलिखित हैं—

जावास्क्रिप्ट में स्टेटमेन्ट्स और ब्लॉक्स Statements and Blocks in JavaScript

जावास्क्रिप्ट के एक स्टेटमेन्ट (Statement) में एक अथवा अधिक एलिमेंट्स (Elements) और सिम्बल्स (Symbols) हो सकते हैं। जावास्क्रिप्ट कोड्स की प्रत्येक नई लाइन, एक नया स्टेटमेन्ट होती है। जावास्क्रिप्ट का प्रत्येक स्टेटमेन्ट सेमीकॉलन (;) से पूर्ण होता है अर्थात् नई लाइन तभी शुरू होती है, जब इससे पहली लाइन सेमीकॉलन (;) से Terminate हुई हो।

```
...
<SCRIPT LANGUAGE = JavaScript>
  //implicit and of instruction
  var Firstname = "Manish";
  //instruction over several lines
  //the end of the instruction in indicated on the 2nd line
  var Surname = "Jain";
  //explicit end of instruction
  alert (Firstname + " " + Surname);
</SCRIPT>
...
```

जावास्क्रिप्ट में एक ब्लॉक (Block) अथवा एक स्टेटमेन्ट ब्लॉक (Statement Block), स्टेटमेन्ट्स का एक समूह (Group) होता है, जिसे Brackets ({}) में समाहित किया जाता है। फंक्शन और कण्ट्रोल स्ट्रक्चर्स (Control Structures) को घोषित करने के लिए स्टेटमेन्ट ब्लॉक्स का प्रयोग किया जाता है।


```

<SCRIPT LANGUAGE = JavaScript>
  function combine (Firstname, Surname)
  {
    alert (Firstname + " " + Surname);
  }
  for (i = 0; i<2; i++)
  {
    Combine ("Manish" "Jain")||
    Combine ("Hemant" "Goyal");
  }
</SCRIPT>

```

guru rana

जावास्क्रिप्ट में कमेण्ट्स (Comments in JavaScript)

किसी प्रोग्राम में कमेण्ट्स अत्यन्त महत्त्वपूर्ण होते हैं। इनके प्रयोग से कोड्स को समझने में सरलता होती है। यदि किसी फंक्शन अथवा प्रोसीजर (Procedure) के साथ “कमेण्ट्स” लिखा जाता है तो फंक्शन अथवा प्रोसीजर द्वारा किए जाने वाले कार्य को सरलता से समझा जा सकता है। समस्त प्रोग्रामिंग लैंग्वेज की भाँति जावास्क्रिप्ट (JavaScript) में भी कमेण्ट्स (Comments) निम्नलिखित दो प्रकार से लिखे जा सकते हैं—

1. **सिंगल लाइन कमेण्ट (Single Line Comment)** एक लाइन का कमेण्ट (Comment) लिखने के लिए लाइन के पहले दो Slashes (//) लगाए जाते हैं।
2. **मल्टीलाइन कमेण्ट (Multiline Comment)** एक से अधिक लाइन्स का कमेण्ट (Comment) लिखने के लिए पहली लाइन के प्रारम्भ में /* और अन्तिम लाइन के अन्त में */ चिह्न का प्रयोग किया जाता है।

जावास्क्रिप्ट में आइडेन्टिफायर्स (Identifiers in JavaScript)

आइडेन्टिफायर्स (Identifiers) का तात्पर्य उस नाम से है, जो किसी वेरिएबल या प्रोसीजर या एक फंक्शन या एक क्लास को दिया जाता है। आइडेन्टिफायर्स का निर्धारण करते समय हमें निम्नलिखित नियमों का अनुपालन करना चाहिए—

1. किसी भी आइडेन्टिफायर का पहला कैरेक्टर कोई अक्षर (Alphabet), अन्डरस्कोर (_) या डॉलर चिह्न (\$) होना चाहिए।
2. किसी भी आइडेन्टिफायर के अन्य कैरेक्टर्स अक्षर (Alphabet), अंक (Numbers), अन्डरस्कोर (_) अथवा डॉलर चिह्न (\$) हो सकते हैं।
3. किसी भी आइडेन्टिफायर में स्पेस (Space), टैब (Tab) अथवा न्यू लाइन कैरेक्टर्स (New Line Characters) का प्रयोग नहीं किया जाना चाहिए।
4. जावास्क्रिप्ट एक केस-सेन्सिटिव (Case-Sensitive) लैंग्वेज है, इसका अर्थ है कि यह lower case कैरेक्टर्स और upper case कैरेक्टर्स की पृथक्-पृथक् पहचान करती है।
5. जावास्क्रिप्ट के आरक्षित कीवर्ड्स (Reserved Keywords) का प्रयोग आइडेन्टिफायर की भाँति नहीं किया जा सकता है। कन्वेंशन के आधार पर हमें आइडेन्टिफायर्स का निर्धारण करते समय निम्नलिखित बिन्दुओं का ध्यान रखना चाहिए—
 - (i) आइडेन्टिफायर, अर्थपूर्ण होना चाहिए ताकि यह जिसके लिए प्रयोग किया जा रहा है, उसका विवरण अच्छे से दर्शा सके।
 - (ii) प्रत्येक कन्स्ट्रक्टर फंक्शन का प्रथम शब्द upper case कैरेक्टर से शुरू होना चाहिए।
 - (iii) प्रत्येक फंक्शन अथवा वेरिएबल का आइडेन्टिफायर lower case कैरेक्टर से शुरू होना चाहिए। यदि कोई आइडेन्टिफायर एक से अधिक शब्दों से बना है, तो उन्हें आपस में जोड़ देना चाहिए। इसमें पहले शब्द का पहला कैरेक्टर तो lower case में ही होना चाहिए, परन्तु अन्य शब्दों का पहला कैरेक्टर upper case में लिखा जा सकता है।

उदाहरण, three Word Function। यह आइडेन्टिफायर तीन शब्दों three, word और function से मिलकर बना है। इसमें पहले शब्द के कैरेक्टर t को lower case में और अन्य दोनों शब्दों के पहले कैरेक्टर्स W और F को upper case में लिखा गया है।

- (iv) प्रत्येक कॉन्स्टेन्ट का नाम upper case कैरेक्टर से शुरू होना चाहिए। यदि किसी कॉन्स्टेन्ट का नाम एक से अधिक शब्दों से बना है, तो उन्हें अन्डरस्कोर (_) का प्रयोग करके आपस में जोड़ देना चाहिए।

जावास्क्रिप्ट के आरक्षित कीवर्ड्स (Reserved Keywords of JavaScript)

प्रत्येक प्रोग्रामिंग लैंग्वेज की भाँति जावास्क्रिप्ट के भी कुछ आरक्षित कीवर्ड्स होते हैं, जिनका प्रयोग जावास्क्रिप्ट द्वारा आन्तरिक ऑपरेशन्स (Internal Operations) के लिए किया जाता है। इन शब्दों को तीन कैटेगरीज़ (categories) में बाँटा गया है—Reserved keywords, Future keywords और वे शब्द जिनका प्रयोग करने से हमें बचना चाहिए।

Reserved Keywords

break	continue	delete	else	for	function	if	in
new	return	this	typeof	var	void	while	with

Future Keywords

abstract	boolean	byte	case	.catch
char	class	const	debugger	default
do	double	enum	export	extends
final	finally	float	goto	implements
import	instanceof	int	interface	long
native	package	private	protected	public
short	static	super	switch	synchronized
throw	throws	transient	try	volatile

वे शब्द, जो वास्तविक जावास्क्रिप्ट ऑब्जेक्ट्स अथवा फंक्शन्स के नाम के अनुरूप हों, का प्रयोग करने से बचना चाहिए।

उदाहरण, इस कैटेगरी में String और parseInt शब्द भी आते हैं।

यदि हम पहली दो कैटेगरीज़ में दिए गए शब्दों में से किसी शब्द का प्रयोग करने का प्रयास करते हैं, तो पहली बार अपनी स्क्रिप्ट को लोड करने पर एक इण्टरप्रीटेशन एरर (Interpretation Error) उत्पन्न होती है। यदि हम तीसरी कैटेगरी से किसी शब्द का प्रयोग करते हैं, तो हमें अप्रत्याशित परिणाम प्राप्त होते हैं।

जावास्क्रिप्ट में वेरिएबल्स (Variables in JavaScript)

वेरिएबल वह शब्द है, जो एक मेमोरी लोकेशन का सन्दर्भ (Reference), जो एक सूचना ग्रहण करती है और जिसका मान स्क्रिप्ट की running के दौरान परिवर्तित भी किया जा सकता है। जावास्क्रिप्ट में किसी वेरिएबल को स्पष्ट रूप से (Explicitly) घोषित करने के लिए, वेरिएबल के नाम से पहले var कीवर्ड का प्रयोग करना होता है।

लोकल वेरिएबल्स की घोषणा करते समय, var कीवर्ड का प्रयोग करना आवश्यक होता है। ग्लोबल वेरिएबल्स की घोषणा के समय var कीवर्ड का प्रयोग करना आवश्यक नहीं होता, परन्तु किसी वेरिएबल को स्पष्ट रूप से घोषित करना उपयुक्त रहता है। वेरिएबल को घोषित करने के कुछ उदाहरण निम्नलिखित हैं—

```
var myFirstName = "MANISH"
var avar = 11 ;
var nvar = null ;
var n2var ;
```

हम वेरिएबल के साथ डेटा टाइप का निर्धारण नहीं कर सकते हैं। वास्तव में, वेरिएबल को दिए जाने वाले मान के डेटा टाइप को जावास्क्रिप्ट वेरिएबल को एसाइन कराती है। वेरिएबल का कोई निश्चित डेटा टाइप नहीं होता है अर्थात् हम किसी वेरिएबल को कोई भी मान प्रदान कर सकते हैं।

यदि एक वेरिएबल को किसी फंक्शन अथवा ब्लॉक अथवा कण्ट्रोल स्ट्रक्चर (Control Structure) के अन्दर घोषित (Declare) किया जाता है, तो इस वेरिएबल के पास लोकल स्कोप (Local Scope) होता है। इसका तात्पर्य यह है कि किसी ब्लॉक में घोषित किए गए वेरिएबल का स्कोप लोकल होता है और इसे इस ब्लॉक के बाहर एक्सेस (Access) नहीं किया जा सकता। यह नियम केवल तभी लागू होता है जब इस वेरिएबल को var कीवर्ड के साथ घोषित किया गया हो। यदि किसी वेरिएबल की घोषणा के समय var कीवर्ड का प्रयोग नहीं किया गया है, तो स्क्रिप्ट में इस वेरिएबल का स्कोप ग्लोबल (Global) होता है। यदि एक वेरिएबल को किसी फंक्शन अथवा ब्लॉक अथवा कण्ट्रोल स्ट्रक्चर के बाहर घोषित (Declare) किया जाता है, तो इस वेरिएबल के पास ग्लोबल स्कोप (Global Scope) होता है अर्थात् इस वेरिएबल को सम्पूर्ण जावास्क्रिप्ट कोड में किसी भी स्थान पर एक्सेस (Access) किया जा सकता है।

जावास्क्रिप्ट में ऐसा कोई कीवर्ड उपलब्ध नहीं है, जिससे किसी ऐसे वेरिएबल को घोषित किया जा सके, जिसका मान स्क्रिप्ट में अपरिवर्तनीय रहे। यद्यपि जावास्क्रिप्ट const कीवर्ड उपलब्ध कराती है, यह कीवर्ड इस लैंग्वेज के भविष्य में किए जाने वाले विस्तार के लिए सुरक्षित होता है।

स्थिरांक (Constants)

किसी भी कम्प्यूटर प्रोग्राम में हम विभिन्न प्रकार के मानों को कम्प्यूटर में स्टोर करते हैं, उन्हें मैनेज करते हैं, उन पर required processing apply करते हैं और उनके परिणाम को output में प्राप्त करते हैं। यदि हम रियल वर्ल्ड में देखें, तो दो प्रकार के मान होते हैं। एक मान वे होते हैं जिन्हें कभी बदला नहीं जाता है। जैसे साल में कुल 12 महीने होते हैं। इन महीनों की संख्या हमेशा निश्चित होती है। कभी भी किसी भी साल में 11 या 13 महीने नहीं हो सकते। इसी प्रकार से हर महीने का एक निश्चित नाम होता है।

हर सप्ताह में सात दिन होते हैं। हर दिन का एक निश्चित नाम होता है। इसी प्रकार से पाइ का मान 22/7 होता है। हम समझ सकते हैं कि ऐसी ही हजारों चीजें हैं, जिनके मान हमेशा निश्चित होते हैं।

जो मान हमेशा निश्चित होते हैं, उन मानों को होल्ड करने वाले आइडेन्टिफायर्स (Identifiers) को स्थिरांक कहा जाता है। इसी प्रकार से किसी कम्प्यूटर प्रोग्राम में डिक्लेयर (Declare) किया गया वह आइडेन्टिफायर, जो ऐसे ही किसी स्थिरांक मान को होल्ड करता है और पूरे प्रोग्राम में अपने डेटा को बदलने नहीं देता है, स्थिरांक (constant) कहलाता है।

जावास्क्रिप्ट में, स्थिरांक const कीवर्ड के साथ डिक्लेयर किए जाते हैं तथा डिक्लेरेशन के समय पर ही एसाइन (Assign) किए जाते हैं।

जावास्क्रिप्ट में डेटा टाइप (Data Type in JavaScript)

जावास्क्रिप्ट में छह डेटा टाइप होते हैं। कुछ मुख्य डेटा टाइप हैं—नम्बर (Number), स्ट्रिंग (String), ऑब्जेक्ट (Object) और बूलियन (Boolean)। अन्य दो डेटा टाइप Null और Undefined हैं। नम्बर (Number), स्ट्रिंग (String), ऑब्जेक्ट (Object) और बूलियन (Boolean) ऑब्जेक्ट्स होते भी हैं। इसलिए इनमें से प्रत्येक के लिए निश्चित मैथड्स और प्रॉपर्टीज होती हैं।

नम्बर डेटा टाइप (Number Data Type)

जावास्क्रिप्ट में नम्बर डेटा टाइप एक्सपोनेन्ट्स (Exponents) के साथ अथवा बिना एक्सपोनेन्ट्स के धनात्मक अथवा ऋणात्मक पूर्णांक/इन्टीजर (Integer) और धनात्मक अथवा ऋणात्मक फ्लोटिंग प्वाइंट (Floating Point) संख्या के साथ आते हैं। फ्लोटिंग प्वाइंट (Floating Point) संख्या, दो भागों में मिलकर बनती है। इसके दोनों भाग दशमलव द्वारा पृथक् होते हैं।

हम किसी Integer को Decimal फॉर्मेट (Base 10), Hexadecimal फॉर्मेट (Base 16) अथवा Octal फॉर्मेट (Base 8) में लिख सकते हैं। कुछ विशिष्ट नम्बर डेटा टाइप के मान निम्नलिखित हैं—

1. Nan यह Not A Number दर्शाता है।
2. MAX_VALUE यह मान जावास्क्रिप्ट द्वारा रिप्रजेन्ट किए जा सकने वाले अधिकतम नम्बर के सदृश होता है।
3. MIN_VALUE यह मान जावास्क्रिप्ट द्वारा रिप्रजेन्ट किए जा सकने वाले न्यूनतम नम्बर के सदृश होता है।
4. NEGATIVE_INFINITY यह मान जावास्क्रिप्ट द्वारा रिप्रजेन्ट किए जा सकने वाले अधिकतम ऋणात्मक नम्बर से छोटे नम्बर के सदृश होता है।
5. POSITIVE_INFINITY यह मान जावास्क्रिप्ट द्वारा रिप्रजेन्ट किए जा सकने वाले अधिकतम धनात्मक नम्बर से बड़े नम्बर के सदृश होता है।

स्ट्रिंग डेटा टाइप (String Data Type)

जावास्क्रिप्ट में कैरेक्टर स्ट्रिंग को सिंगल अथवा डबल कोटेशन चिह्न (Single or Double Quotation Marks) से सीमित किया जाता है। इसके साथ ही हम अपनी स्ट्रिंग्स में कुछ लिट्रल कैरेक्टर्स (Literal Characters) को भी सम्मिलित कर सकते हैं। ये कैरेक्टर्स कण्ट्रोल कैरेक्टर्स कहलाते हैं। ये कैरेक्टर्स प्रिन्टेबल (Printable) नहीं होते हैं, परन्तु ये विशेष कैरेक्टर्स को रिप्रजेन्ट करते हैं। कुछ कण्ट्रोल कैरेक्टर्स निम्नलिखित हैं—

कण्ट्रोल कैरेक्टर्स	अर्थ
\\	Back Slash
'	Single Quote
\"	Double Quote
\b	Back Space
\r	Carriage Return
\t	Tab
\f	Page Break
\n	New Line

ऑब्जेक्ट डेटा टाइप (Object Data Type)

समस्त जावास्क्रिप्ट ऑब्जेक्ट्स मूल रूप से ऑब्जेक्ट डेटा टाइप पर निर्भर करते हैं। डेटा का यह प्रकार समस्त जावास्क्रिप्ट ऑब्जेक्ट्स के लिए कैरेक्टर सिंटेक्स का समूह कोमन प्रोपर्टीज को परिभाषित करता है।

बूलियन डेटा टाइप (Boolean Data Type)

बूलियन डेटा टाइप दो तार्किक मानों— True और False को रिप्रेजेंट करता है। जावास्क्रिप्ट उन सभी एक्सप्रेशन, जिनमें एक मान शून्य है, उनका False के रूप में और जिनमें non-zero numeric मान होता है, उनका True के रूप में निरूपण (Consider) करती है।

नल और अनडिफाइण्ड डेटा टाइप (Null and Undefined Data Type)

नल मान अर्थात् कोई मान नहीं और अनडिफाइण्ड (Undefined) कीवर्ड यह दर्शाता है कि यह वेरिएबल का प्रकार तब तक अपरिभाषित (Undefined) है, जब तक कि इस वेरिएबल को कोई मान नहीं प्रदान किया जाता।

डेटा टाइप को निर्धारित और परिवर्तित करना (Determining and Converting Data Types)

जावास्क्रिप्ट में हम किसी वेरिएबल को घोषित करते समय, उसका डेटा टाइप निर्धारित नहीं कर सकते। जावास्क्रिप्ट में किसी एक्सप्रेशन का डेटा टाइप निर्धारित करने के लिए typeof ऑपरेटर का प्रयोग किया जाता है। यह ऑपरेटर छह विभिन्न स्ट्रिंग मानों— number, string, boolean, object, function और undefined में से कोई एक मान वापस रिटर्न (Return) करता है। उदाहरण, निम्न कोड्स का अध्ययन कीजिए—

```
var name = "Manish" ;
typeof(name) //returns "string"
var age = 42 ;
typeof (age) //returns "number"
var bDate = new Date (1965, 7, 17) ;
typeof (bDate) //returns "object"
typeof (null) //returns "string"
typeof (Date) //returns "function"
typeof (unknownVar) //returns "undefined"
typeof (false) //returns "boolean"
```

उपरोक्त कोड्स में प्रत्येक typeof ऑपरेटर के सामने सिंगल लाइन कमेंट के रूप में उसके द्वारा वापस (Return) किए जाने वाले डेटा टाइप को दर्शाया गया है। स्क्रिप्ट के कॉण्टैक्ट्स के अनुरूप जावास्क्रिप्ट implicitly डेटा टाइप को परिवर्तित कर देती है।

उदाहरण, यदि कॉण्टैक्ट्स को यह आवश्यकता होती है कि मान स्ट्रिंग टाइप का होना चाहिए, तो जावास्क्रिप्ट मान को स्ट्रिंग डेटा टाइप में परिवर्तित कर देगी।

इसके अतिरिक्त जावास्क्रिप्ट कुछ फंक्शन्स एवं मैथड्स उपलब्ध कराती है, जिनके द्वारा explicitly स्ट्रिंग डेटा टाइप को न्युमैरिक टाइप में और ऑब्जेक्ट्स को स्ट्रिंग्स में परिवर्तित किया जा सकता है।

ऐसे कुछ प्रमुख फंक्शन्स और मैथड निम्नलिखित हैं—

1. **parseInt () फंक्शन** यह फंक्शन एक कैरेक्टर स्ट्रिंग को आरग्युमेण्ट के रूप में लेता है और इन्टीजर के रूप में रिटर्न करता है। इस फंक्शन का प्रयोग निम्न सिनटेक्स के अनुसार किया जाता है—

```
parseInt (string [,base])
```

इसमें string वह कैरेक्टर स्ट्रिंग है, जिसे हम परिवर्तित करना चाहते हैं और base वह नम्बर बेस है, जिसके नम्बर में हमें कैरेक्टर स्ट्रिंग को परिवर्तित करना है। Decimal नम्बर के लिए नम्बर बेस 10, Octal नम्बर के लिए नम्बर बेस 8 और Hexadecimal नम्बर के लिए नम्बर बेस 16 देना होता है। इसमें base पैरामीटर वैकल्पिक है अर्थात् यदि हम न चाहें, तो नम्बर बेस का उल्लेख करना आवश्यक नहीं है। ऐसी परिस्थिति में कैरेक्टर स्ट्रिंग Decimal मान में परिवर्तित होती है। यदि हमारी कैरेक्टर स्ट्रिंग में कोई आंकिक मान नहीं है, तो यह NaN मान रिटर्न करता है।

उदाहरण, निम्न कोड्स का अध्ययन कीजिए—

```
parseInt ("atoz") //returns NaN
parseInt ("3027", 8) //returns 1559
parseInt ("ED", 16) //returns 237
```


2. **parseFloat ()** फंक्शन यह फंक्शन एक कैरेक्टर स्ट्रिंग को आरम्भिक के रूप में लेता है और फ्लोटिंग-प्वाइण्ट नम्बर के रूप में रिटर्न करता है। इस फंक्शन का प्रयोग निम्न सिन्टेक्स के अनुसार किया जाता है—

`parseFloat (string)`

इसमें `string` वह कैरेक्टर स्ट्रिंग है, जिसे हम परिवर्तित करना चाहते हैं। यदि हमारा कैरेक्टर स्ट्रिंग में कोई फ्लोटिंग-प्वाइण्ट मान नहीं है, तो वह `NaN` मान रिटर्न करता है।

उदाहरण के लिए, निम्न कोड्स का अध्ययन कीजिए—

```
parseFloat ("atoz") //returns NaN
```

```
parseFloat ("30.27atoz") //returns 30.27
```

3. **toString ()** मैथड यह एक कैरेक्टर स्ट्रिंग को रिटर्न करता है, जो एक ऑब्जेक्ट को रिप्रजेंट करती है। इस कमाण्ड का प्रयोग निम्न सिन्टेक्स के अनुसार किया जाता है—

`object.toString ([base])`

इसमें `object` एक ऑब्जेक्ट का नाम है और `base` वह नम्बर बेस है, जिसके नम्बर में हमें कैरेक्टर स्ट्रिंग को परिवर्तित करना है। इस मैथड के द्वारा होने वाला परिवर्तन इस पर निर्भर करता है कि इसे किस प्रकार के ऑब्जेक्ट के लिए प्रभावी किया गया है। यदि ऑब्जेक्ट `Array` है, तो `toString` ऐरे के एलिमेंट्स को परस्पर concatenate करके एक टेक्स्ट-स्ट्रिंग में परिवर्तित कर देता है।

यदि ऑब्जेक्ट `boolean` है, तो बूलियन मान के सत्य होने पर यह `True` मान रिटर्न करता है अन्यथा यह `False` मान रिटर्न करता है। यदि ऑब्जेक्ट एक फंक्शन है, तो `toString` एक स्ट्रिंग रिटर्न करता है, जिसमें इस मैथड के साथ कॉल किए गए फंक्शन का नाम होता है। यदि ऑब्जेक्ट स्ट्रिंग प्रकार का है, तो यह मैथड स्ट्रिंग ऑब्जेक्ट का मान रिटर्न करता है।

guru rana

1.6 जावास्क्रिप्ट में ऐरेज़ (Arrays in JavaScript)

किसी विशेष उद्देश्य के लिए मानों का समूह स्टोर करने के लिए कुछ परिस्थितियों में ऐरे का प्रयोग अत्यन्त उपयोगी होता है। जावास्क्रिप्ट, ऐरे को मैनेज करने के लिए सामान्य डेटा प्रकार का ऑप्शन नहीं करती है, फिर भी इस लैंग्वेज में `Array` नामक ऑब्जेक्ट को जोड़ा गया है। जावास्क्रिप्ट ऐरे को एक ऑब्जेक्ट डेटा प्रकार के रूप में समझता है।

अतः हमें कन्स्ट्रक्टर मैथड्स का प्रयोग करके ऐरे क्रिएट करने चाहिए और हम ऐरे के मैथड्स और उनकी प्रॉपर्टीज़ का प्रयोग करके ऐरे को हैंडल कर सकते हैं। ऐरे का डेटा टाइप कोई भी हो सकता है।

जावास्क्रिप्ट में ऐरेज़ को घोषित और हैंडल करना (Declaring and Handling Arrays in JavaScript)

जावास्क्रिप्ट में ऐरे क्रिएट करने के लिए हमें एक ऐरे क्लास का एक ऑब्जेक्ट बनाना होता है। ऐरे क्लास को बनाने के लिए `new` ऑपरेटर का प्रयोग किया जाता है, जो हमें यह निर्धारित करने की अनुमति प्रदान करता है कि हम अपना ऐरे किस प्रकार क्रिएट करना चाहते हैं।

ऐरे के प्रकार (Types of Array)

वन-डाइमेंशनल ऐरे (One-dimensional Array)

वन-डाइमेंशनल ऐरे एक वेरिएबल का टाइप होता है। इसे प्रयोग करने से पहले `declare` करना होता है जावास्क्रिप्ट में हम इसे तीन प्रकार से प्रयोग कर सकते हैं—

`new Array ()`

इससे हम एक ऐसा ऐरे क्रिएट कर सकते हैं, जिसका कोई आकार निर्धारित नहीं किया गया है।

बिना साइज़ के ऐरे का सिन्टेक्स निम्न प्रकार है—

```
var array_name = new Array [ ];
```

`new Array (size)`

इससे हम एक ऐसा ऐरे क्रिएट कर सकते हैं, जिसका कोई आकार निर्धारित किया गया है। इसमें `size`, ऐरे में सुरक्षित हो सकने वाले `elements` की संख्या से है अर्थात् यह ऐरे का आकार निर्धारित करता है।

साइज़ के साथ ऐरे को निम्न प्रकार से लिखा जाता है—

```
var array_name = new Array [array_size];
```

`new Array (element1, element2..... elementN)`
 इन्हें हम ऐरे लिस्ट करने के साथ-साथ उसके विभिन्न एलिमेंट्स का निर्धारण कर सकते हैं। ऐसे ऐरे को डेंस ऐरे के नाम से जाना जाता है इसमें
`element1, element2, elementN` ऐरे में सुरक्षित एलिमेंट्स होते हैं। इसमें ऐरे के एलिमेंट्स के साथ-साथ ऐरे का आकार भी परिभाषित हो
 जाता है। एक बार ऐरे को लिस्ट करने के बाद हम ऐरे को किसी भी डेटा टाइप के मानों से fill कर सकते हैं। यहाँ तक कि हम एक ही ऐरे में अनेक
 डेटा टाइप के मानों का प्रयोग कर सकते हैं। डेंस ऐरे का सिन्टेक्स निम्न प्रकार है—

`var array_name = new Array [value1, ... valueN];`

दू-डाइमेंशनल ऐरे (Two-dimensional Array)

जावास्क्रिप्ट मल्टी-डाइमेंशनल ऐरे को सपोर्ट नहीं करती है, लेकिन आप ऐरे ऑफ ऐरेज़ (Array of Arrays) बनाने के लिए दू-डाइमेंशनल
 ऐरे का प्रयोग कर सकते हैं। दू-डाइमेंशनल ऐरे का सिन्टेक्स निम्न प्रकार है—

`var array_name = new Array [row_size][column_size];`

or

`var array_name = new Array [row_size, column_size];`

guru rana

1.7 जावास्क्रिप्ट में ऑपरेटर्स और एक्सप्रेशन्स (Operators and Expressions in JavaScript)

जावास्क्रिप्ट में ऑपरेटर्स को दो वर्गों में विभाजित किया गया है—यूनरी ऑपरेटर (Unary Operator), जिसके लिए एक ही ऑपरेण्ड
 (Operand) और बाइनरी ऑपरेटर (Binary Operator), जिसके लिए दो ऑपरेण्ड्स (Operands) की आवश्यकता होती है। इनके अतिरिक्त
 एक Ternary ऑपरेटर भी होता है, जो एक सर्कल ऑपरेटर (Conditional Operator) होता है।

यूनरी ऑपरेटर (Unary Operator)

इस ऑपरेटर को एक ही ऑपरेण्ड की आवश्यकता होती है। इन ऑपरेटर्स को दो प्रकार के नोटेशन के साथ प्रयोग किया जा सकता है जो निम्न हैं—

1. **Prefix Notation** इस प्रकार के नोटेशन में ऑपरेटर, ऑपरेण्ड के पहले प्रयोग किया जाता है, जैसे (++i)।
2. **Postfix Notation** इस प्रकार के नोटेशन में ऑपरेटर, ऑपरेण्ड के बाद प्रयोग किया जाता है, जैसे (i++)।

प्रिफिक्स नोटेशन में जावास्क्रिप्ट ; को इसके इन्क्रिमेंट के बाद इवैल्यूएट करता है, जबकि इसके ठीक विपरीत पोस्टफिक्स नोटेशन में
 जावास्क्रिप्ट ; को इवैल्यूएट करने के बाद इसका इन्क्रिमेंट करता है।

जावास्क्रिप्ट में उपलब्ध यूनरी ऑपरेटर निम्नलिखित हैं—

ऑपरेटर	उपयोग
-	Negation
~	One's Complement
++	Increment
--	Decrement
!	Logical NOT

बाइनरी ऑपरेटर (Binary Operator)

इस ऑपरेटर को दो ऑपरेण्ड्स की आवश्यकता होती है, जावास्क्रिप्ट में निम्नलिखित बाइनरी ऑपरेटर्स उपलब्ध हैं—

1. **एसाइनमेंट ऑपरेटर्स (Assignment Operators)** जावास्क्रिप्ट में हम किसी वैरिएबल को एसाइन करने के लिए बराबर का चिह्न (=)
 प्रयोग कर सकते हैं। जावास्क्रिप्ट अर्थमेटिक, लॉजिकल और बिटवाइज़ ऑपरेटर्स एवं एक सिंगल ऑपरेटर का प्रयोग करने वाले
 एसाइनमेंट ऑपरेटर्स को पूर्ण करने के लिए अन्य एसाइनमेंट ऑपरेटर्स उपलब्ध कराती है। जावास्क्रिप्ट के एसाइनमेंट ऑपरेटर्स
 निम्नलिखित हैं—

ऑपरेटर	परिभाषा
=	एसाइनमेण्ट
+=	जोड़ और एसाइनमेण्ट
-=	घटा और एसाइनमेण्ट
*=	गुणा और एसाइनमेण्ट
/=	भाग और एसाइनमेण्ट
%=	मॉड्यूलस और एसाइनमेण्ट
&=	बाइनरी AND और एसाइनमेण्ट
	बाइनरी OR और एसाइनमेण्ट
^=	बाइनरी एक्सक्लूजिव OR और एसाइनमेण्ट
<<=	बिटवाइज लेफ्ट साइड शिफ्ट और एसाइनमेण्ट
>>=	बिटवाइज राइट साइड शिफ्ट और एसाइनमेण्ट

2. तुलनात्मक ऑपरेटर्स (Comparison Operators) इन ऑपरेटर्स की सहायता से ऑपरेण्ड्स में तार्किक (Logical) तुलना की जाती है। इन ऑपरेटर्स के साथ तुलना किए जाने वाले ऑपरेण्ड्स न्युमैरिक और कैरेक्टर स्ट्रिंग मानों वाले हो सकते हैं। जावास्क्रिप्ट स्ट्रिंग्स की तुलना उनके unicode मानों के अनुरूप करती है। जावास्क्रिप्ट के तुलनात्मक (एसाइनमेण्ट) ऑपरेटर्स निम्नलिखित हैं—

ऑपरेटर	परिभाषा
==	Equality
!=	Inequality
===	Identify
!==	Non-Identity
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

ऑपरेण्ड्स के डेटा टाइप के आधार पर ये ऑपरेटर्स भिन्न व्यवहार करते हैं—

- (i) <, >, <= और >= ऑपरेटर्स ये ऑपरेटर्स ऑपरेण्ड्स को नम्बर्स में परिवर्तित करने के लिए प्रयोग किए जाते हैं। यदि दोनों ऑपरेण्ड्स स्ट्रिंग्स हैं, तो ये ऑपरेटर्स, कैरेक्टर्स के unicode मानों के अनुरूप तुलना करते हैं। यदि तुलना किए जा रहे ऑपरेण्ड्स में से कोई एक ऑपरेण्ड NaN है, तो ये ऑपरेटर्स false मान रिटर्न करते हैं।
 - (ii) == और != ऑपरेटर्स ये ऑपरेण्ड्स भिन्न डेटा प्रकार के हैं, तो ये ऑपरेटर्स ऑपरेण्ड्स को या तो स्ट्रिंग या नम्बर्स या बूलियन में परिवर्तित करते हैं।
 - (iii) === और !== ऑपरेटर्स ये ऑपरेटर्स == और != के समान व्यवहार करते हैं, इसके अतिरिक्त ये दो ऑपरेण्ड्स के डेटा टाइप को परिवर्तित नहीं करते हैं।
3. अर्थमेटिक ऑपरेटर्स (Arithmetic Operators) जावास्क्रिप्ट गणितीय गणना करने के लिए ऑपरेटर उपलब्ध कराती है। इन ऑपरेटर्स का प्रयोग इन्टीजर (Integer) और फ्लोटिंग-प्वाइंट (Floating Point) नम्बर्स के साथ किया जा सकता है। जावास्क्रिप्ट में उपलब्ध विभिन्न अंकगणितीय ऑपरेटर्स निम्नांकित हैं—

ऑपरेटर	परिभाषा
+	जोड़
-	घटा
*	गुणा
/	भाग
%	शेष (भाग का शेषफल)

4. **बिटवाइज ऑपरेटर्स (Bitwise Operators)** ये ऑपरेटर्स डेसीमल अथवा हैक्साडेसीमल अथवा ऑक्टल फॉर्मेट के इन्टीजर्स के बाइनरी रिप्रेजेंटेशन्स के लिए प्रयोग किए जाते हैं। ये परिणाम को रिटर्न करने से पहले integers में परिवर्तित कर देते हैं। जावास्क्रिप्ट में उपलब्ध विभिन्न बिटवाइज ऑपरेटर्स निम्नांकित हैं—

ऑपरेटर	परिभाषा
&	Binary AND
	Binary OR
^	Binary Exclusive OR
~	Bitwise NOT (one's complement)
<<	Bitwise left shift
>>	Bitwise right shift
>>>	Bitwise right shift with zero

guru rana

- बाइनरी AND ऑपरेटर 1 (True) मान तभी रिटर्न करता है, जब दोनों ऑपरेण्ड्स 1 (True) हों। बाइनरी OR ऑपरेटर 1 (True) मान तभी रिटर्न करता है, जब दोनों ऑपरेण्ड्स में से कम-से-कम ऑपरेण्ड 1 (True) हो। बाइनरी एक्सक्लूजिव OR ऑपरेटर 1 (True) मान तभी रिटर्न करता है, जब दोनों ऑपरेण्ड्स में से केवल एक ऑपरेण्ड 1 (True) हो।
5. **तार्किक ऑपरेटर्स (Logical Operators)** तार्किक ऑपरेटर्स, बूलियन एक्सप्रेशन्स की conditions को इवैल्यूएट करने की अनुमति प्रदान करते हैं। जावास्क्रिप्ट में उपलब्ध विभिन्न तार्किक ऑपरेटर्स निम्नांकित हैं—

ऑपरेटर	परिभाषा
!	Logical NOT
&&	Logical AND
	Logical OR

बूलियन एक्सप्रेशन का Opposite प्राप्त करने के लिए (!) ऑपरेटर अत्यन्त उपयोगी है।

6. **विशेष ऑपरेटर्स (Special Operators)** जावास्क्रिप्ट में उपलब्ध विभिन्न विशेष ऑपरेटर्स निम्नांकित हैं—

- (i) **Conditional (Ternary) Operator** यह जावास्क्रिप्ट द्वारा ऑफर किया गया एकमात्र ternary ऑपरेटर है। इसके लिए तीन ऑपरेण्ड्स की आवश्यकता होती है। दी गई कण्डीशन (जो पहला ऑपरेण्ड है) के मान के अनुरूप यह ऑपरेटर दोनों ऑपरेण्ड्स में से किसी एक को, या तो रन (Run) या फिर रिटर्न करता है। इस ऑपरेटर का सिन्टेक्स निम्न है—
condition ? val1 : val2
यदि दी गई कण्डीशन का मान true है, तो जावास्क्रिप्ट val1 को रन (Run) या रिटर्न करती है अन्यथा जावास्क्रिप्ट val2 को रन (Run) या रिटर्न करती है। यह ऑपरेटर if...else के सरल संस्करण के बराबर है।
- (ii) **delete Operator** जावास्क्रिप्ट में इस ऑपरेटर का प्रयोग Implicit वेरिएबल अर्थात् ऐसे वेरिएबल ऑब्जेक्ट के लिए होता है, जिसको घोषित करने के लिए var कीवर्ड का प्रयोग नहीं किया गया है अथवा एक ऑब्जेक्ट की किसी प्रोपर्टी अथवा ऐरे के किसी एक एलिमेण्ट को डिलीट करने अर्थात् मिटाने के लिए किया जाता है। यदि यह ऑपरेटर डिलीशन प्रक्रिया को पूर्ण करने योग्य होता है, तो true मान रिटर्न करता है अन्यथा यह false मान रिटर्न करता है। यदि यह ऑपरेटर डिलीशन प्रक्रिया को करने योग्य परिस्थिति में नहीं होता है, तो ऑब्जेक्ट के रेफरेन्सेज अथवा ऑब्जेक्ट की प्रोपर्टी अथवा ऐरे के एलिमेण्ट के मान में Undefined मान होता है।
- (iii) **new Operator** जावास्क्रिप्ट में इस ऑपरेटर का प्रयोग नया ऑब्जेक्ट क्रिएट करने के लिए किया जाता है। इस ऑपरेटर के सिन्टेक्स निम्नानुसार है—
new constructor ([argument1 [...], argumentN])
इस ऑपरेटर का प्रयोग करके हम मैम्बर के बिना भी ऑब्जेक्ट बना सकते हैं। यह इस ऑब्जेक्ट के कन्स्ट्रक्टर को कॉल करता है। कन्स्ट्रक्टर दिए गए ऑर्ग्युमेण्ट्स के अनुरूप ऑब्जेक्ट इनिशियलाइज (Initialize) करता है और फिर ऑपरेटर नए ऑब्जेक्ट के एक प्वाइण्टर को रिटर्न करता है।
- (iv) **typeof Operator** जावास्क्रिप्ट में यह ऑपरेटर एक एक्सप्रेशन के डेटा टाइप को आइडेन्टिफाई करके कैरेक्टर स्ट्रिंग को रिटर्न करता है। इस ऑपरेटर का सिन्टेक्स निम्नानुसार है—
typeof [() objectClass []]

(v) **void Operator** जावास्क्रिप्ट में void ऑपरेटर किसी ऐसे एक्सप्रेशन को इवैल्यूएट (Evaluate) अथवा एक्जिक्यूट (Execute) करने की अनुमति प्रदान करता है, जो कोई मान रिटर्न नहीं करता। इस ऑपरेटर का सिन्टेक्स निम्नानुसार है—

```
void [ ( ) expression [ ] ]
```

उपरोक्त दो ऑपरेटर्स typeof और void ऑपरेटर्स में ब्रैकेट्स () वैकल्पिक हैं।

(vi) **string Operator** जावास्क्रिप्ट में, (+) ऑपरेटर का उपयोग स्ट्रिंग को जोड़ने के लिए भी किया जाता है। जब (+) ऑपरेटर का उपयोग स्ट्रिंग्स पर किया जाता है, तो इसे concatenation operator कहते हैं। इसका सिन्टेक्स निम्नानुसार है—

```
string3 = string1 + string2
```

ऑपरेटर प्रीसीडेन्स (Operator Precedence)

जब किसी एक्सप्रेशन को इवैल्यूएट किया जाता है, तब ऑपरेटर्स के एक्जिक्यूट होने का क्रम ऑपरेटर प्रीसीडेन्स द्वारा निर्धारित किया जाता है। जिस ऑपरेटर्स की प्रीसीडेन्स अधिक होती है, वह कम प्रीसीडेन्स वाले ऑपरेटर्स से पहले एक्जिक्यूट होता है। जैसे—जोड़ से पहले गुणा होती है। निम्नलिखित सारणी सबसे अधिक से कम प्रीसीडेन्स की तरफ का क्रम प्रदर्शित कर रही है। जिन ऑपरेटर्स की प्रीसीडेन्स समान है, वे बाएँ से दाएँ की तरफ इवैल्यूएट होंगे।

ऑपरेटर	विवरण
++ -- -!	Unary ऑपरेटर्स, delete ऑपरेटर, new ऑपरेटर, typeof ऑपरेटर,
delete new type of void	void ऑपरेटर
* / %	गुणा, भाग, शेष
+ - +	जोड़, घटा, concatenation ऑपरेटर
<< >> >>>	Bit Shifting
< <= > >=	Less than, less than or equal to, greater than, greater than or equal to
= != == !=	Equality, inequality, identify, non-identify
&	Binary AND
^	Binary Exclusive OR
	Binary OR
&&	Logical AND
	Logical OR
?:	Conditional ऑपरेटर
= OP=	एसाइनमेंट, ऑपरेटर्स के साथ एसाइनमेंट (जैसे + = &=)

1.8 जावास्क्रिप्ट में ब्रान्चिंग और लूपिंग स्टेटमेंट्स (Branching and Looping Statements in JavaScript)

प्रोग्राम के प्रवाह को नियंत्रित करने के लिए ब्रान्चिंग और लूपिंग स्टेटमेंट्स आवश्यक होते हैं। वस्तुतः अधिकांश स्क्रिप्ट्स तभी उपयोगी होती हैं, जब वे भिन्न परिस्थितियों में भिन्न ऑपरेटर्स कर सकें। ब्रान्चिंग स्टेटमेंट्स को कंट्रोल स्ट्रक्चर (Control Structure) भी कहा जाता है।

ब्रान्चिंग (Branching)

प्रोग्राम की ब्रान्चिंग के लिए कंट्रोल स्ट्रक्चर्स का प्रयोग किया जाता है। जावास्क्रिप्ट में प्रयोग किए जाने वाले कंट्रोल स्ट्रक्चर्स निम्नलिखित हैं—

1. **if स्ट्रक्चर** If स्ट्रक्चर की सहायता से हम किसी परिस्थिति (Condition) के सही होने पर एक अथवा एक से अधिक स्टेटमेंट्स को Run करा सकते हैं। परिस्थिति की जाँच कराने पर वह बूलियन True अथवा False रिटर्न करता है। इसका सिन्टेक्स निम्नानुसार है—

```
if (<condition>) <statement>;
    या
if (<condition>)
    <sequence of statements>;
```

उपरोक्त स्टेटमेंट परिस्थिति के सत्य होने पर ही रन होता है। यदि परिस्थिति सत्य नहीं होती है, तो यह False मान रिटर्न करता है।

2. **if else** स्ट्रक्चर if के साथ else का प्रयोग करके दो परिस्थितियों (Conditions) की जाँच कर सकते हैं। if...else स्ट्रक्चर में यदि पहली परिस्थिति के सत्य होने पर if स्ट्रक्चर के स्टेटमेंट्स रन होते हैं और यदि पहली परिस्थिति सत्य नहीं होती है, तो else में दिए गए स्टेटमेंट्स रन होते हैं अर्थात् प्रोग्राम का कण्ट्रोल इन स्टेटमेंट्स पर आ जाता है। इस स्ट्रक्चर का सिन्टेक्स निम्नानुसार है—

```

if(<condition>)
{
    <sequence of statements 1>;
}
else
{
    <sequence of statements 2>;
}

```

guru rana

if...else स्ट्रक्चर के स्थान पर Conditional (Ternary) Operator का प्रयोग भी किया जा सकता है। दो से भी अधिक परिस्थितियों को जाँचने के लिए if...else if...else स्ट्रक्चर का प्रयोग किया जा सकता है। इस स्ट्रक्चर का सिन्टेक्स निम्नानुसार है—

```

if(<condition 1>)
{
    <sequence of statements 1>;
}
else if (<condition 2>)
{
    <sequence of statements 2>;
}
else
{
    <sequence of statements 3>;
}

```

3. **switch** स्ट्रक्चर यदि हमारे पास जाँचने के लिए अनेक परिस्थितियाँ हैं, तो if...else if...else स्ट्रक्चर अत्यन्त कठिन हो जाता है। ऐसी परिस्थिति को हैण्डल करने के लिए switch स्ट्रक्चर का प्रयोग किया जाता है। इस स्ट्रक्चर की सहायता से हम अनेक परिस्थितियों में से किसी भी परिस्थिति के स्टेटमेंट्स को रन (Run) कर सकते हैं। इस स्ट्रक्चर का सिन्टेक्स निम्नानुसार है—

```

switch (<expression_to_be_checked>)
{
    case<expression 1>:
        <sequence of statements 1>;
        break;
    case<expression 2>:
        <sequence of statements 2>;
        break;
    case<expression 3>:
        <sequence of statements 3>;
        break;
    default :
        <other sequence of statements>;
}

```

इसमें हमें break स्टेटमेंट का प्रयोग करना आवश्यक है, क्योंकि हमें प्रत्येक केस (Case) को पृथक् करना होता है। अन्यथा एक परिस्थिति के सही होने पर उससे नीचे के सभी केसिज़ run हो जाएँगे। इसमें default लेबल की सहायता से किसी भी केस के सत्य न होने पर run किए जाने पर default ब्लॉक वाले स्टेटमेंट्स को लिखा जाता है।

लूपिंग स्टेटमेंट्स (Looping Statements)

सामान्यतः लूप्स (Loops) तीन प्रकार के होते हैं— for लूप, while लूप और do...while लूप।

1. for लूप

for लूप का प्रयोग उस स्थिति में किया जाता है, जब हमें पहले से ही यह पता होता है, कि स्क्रिप्ट को कितनी बार Run किया जाना है। for लूप का सिन्टेक्स निम्नानुसार है—

```
for (variable=startvalue; variable <= endvalue;
variable=variable+incrementfactor)
{
    //Here goes the script lines you want to loop.
}
```

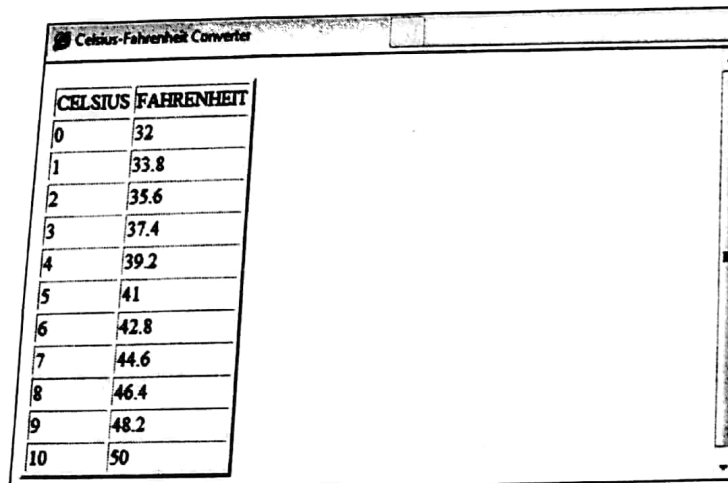
इस सिन्टेक्स में variable उस वेरिएबल का नाम है, जिसके मान के आधार पर लूपिंग की जानी है। startvalue इस वेरिएबल का प्रारम्भिक और endvalue इस वेरिएबल का अन्तिम मान है। incrementfactor लूप में प्रत्येक स्तर पर वेरिएबल का वृद्धि मान है। हम यह मान ऋणात्मक भी निर्धारित कर सकते हैं।

उदाहरण, हमें for लूप का प्रयोग करके सेल्सियस और फारेनहाइट की तुलना करते हुए एक सारणी बनानी है। सारणी की प्रत्येक पंक्ति 1 डिग्री सेल्सियस का मान बढ़ाए और इसी पंक्ति के दूसरे कॉलम में इसका फारेनहाइट मान दर्शाए।

```
<html>
  <head>
    <title>Celsius-Fahrenheit Converter</title>
  </head>
  <body>
    <table border=3>
      <tr><td>CELSIUS</td><td>FAHRENHEIT</td></tr>
      <script language="javascript">
        for(celsius=0; celsius<=10; celsius=celsius+1)
        {
          document.write("<tr><td>" +celsius+"</td><td>"
          +((celsius*9/5)+32)+"</td></tr>");
        }
      </script>
    </table>
  </body>
</html>
```

उपरोक्त HTML डॉक्यूमेंट, जिसे FOR-LOOP.html के नाम से Save किया गया है, इसमें स्क्रिप्ट का प्रयोग करके for लूप के प्रयोग को दर्शाया गया है।

आउटपुट



CELSIUS	FAHRENHEIT
0	32
1	33.8
2	35.6
3	37.4
4	39.2
5	41
6	42.8
7	44.6
8	46.4
9	48.2
10	50

Execution of HTML codes

2. while लूप

while लूप का प्रयोग उस स्थिति में किया जाता है जब हम स्क्रिप्ट को तब तक लगातार run करना चाहते हैं, जब तक दी गई आवश्यक condition सत्य न हो जाए। while लूप का सिन्टेक्स निम्नानुसार है—

```
while (variable <= endvalue)
{
    //Here goes the script lines you want
}
```

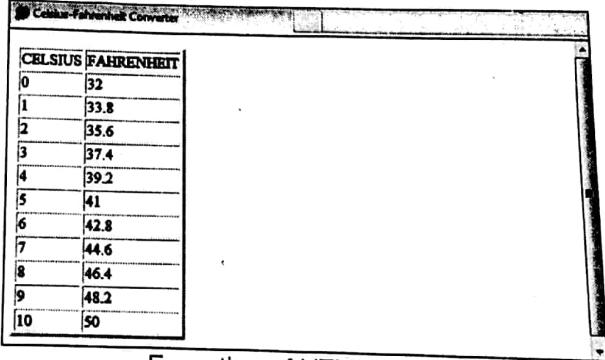
इस सिन्टेक्स में variable उस वेरिएबल का नाम है, जिसके मान के आधार पर लूपिंग की जानी है। endvalue इस वेरिएबल का अन्तिम मान है। less than or equal to ऑपरेटर (<=) के स्थान पर हम लूप के उद्देश्य के अनुरूप कोई अन्य ऑपरेटर भी प्रयोग कर सकते हैं। उदाहरण, हमें while लूप का प्रयोग करके सेल्सियस और फारेनहाइट की तुलना करते हुए एक सारणी बनानी है। सारणी की प्रत्येक पंक्ति 1 डिग्री सेल्सियस का मान बढ़ाए और इसी पंक्ति के दूसरे कॉलम में इसका फारेनहाइट मान दर्शाए।

```
<html>
<head>
<title>Celsius-Fahrenheit Converter</title>
</head>
<body>
<table border=3>
<tr><td>CELSIUS</td><td>FAHRENHEIT</td></tr>
<script language="javascript">
    celsius=0;
    while ( celsius<=10)
    {
        document.write("<tr><td>" +celsius+"</td><td>"
            +((celsius*9/5)+32)+"</td></tr>");
        celsius=celsius+1;
    }
</script>
</table>
</body>
</html>
```

guru rana

उपरोक्त में HTML डॉक्युमेन्ट, जिसे WHILE-LOOP.html के नाम से Save किया गया है, में स्क्रिप्ट का प्रयोग करके while लूप के प्रयोग को दर्शाया गया है।

आउटपुट



Execution of HTML codes

3. do...while लूप

do...while लूप में पहले स्ट्रक्चर एक्जिक्यूट होता है, फिर कण्डीशन की जाँच होती है। इस लूप का प्रयोग उस स्थान पर किया जाता है, जहाँ पर प्रत्येक परिस्थिति में स्ट्रक्चर का कम-से-कम एक बार एक्जिक्यूट होना आवश्यक होता है। do...while लूप का सिन्टेक्स निम्नानुसार है—

```
do
{
    <sequence of statements>;
}
while<condition>
```

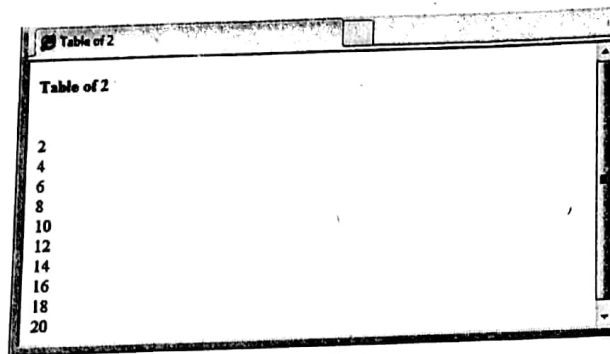

इस सिन्टेक्स में do के अन्तर्गत वे स्टेटमेन्ट्स लिखे जाते हैं, जिनका एक्ज़िक्यूशन कम-से-कम एक बार होना आवश्यक है और while के सामने उस कण्डीशन को घोषित किया जाता है, जिसके सत्य होने तक do के अन्तर्गत लिखे गए स्टेटमेन्ट्स एक्ज़िक्यूट होंगे।
उदाहरण, हम do...while लूप का प्रयोग करके 2 का पहाड़ा बनाते हैं।

```
<html>
  <head>
    <title>Table of 2</title>
  </head>
  <body>
    <h4>Table of 2</h4>
    <script language="javascript">
      var i=0;
      do
      {
        i=i+2;
        document.write("<br>" + i);
      }
      while(i != 20)
    </script>
  </body>
</html>
```

guru rana

उपरोक्त HTML डॉक्यूमेंट, जिसे DO_WHILE-LOOP.html के नाम से Save किया गया है, में स्क्रिप्ट का प्रयोग करके do...while लूप के प्रयोग को दर्शाया गया है।

आउटपुट



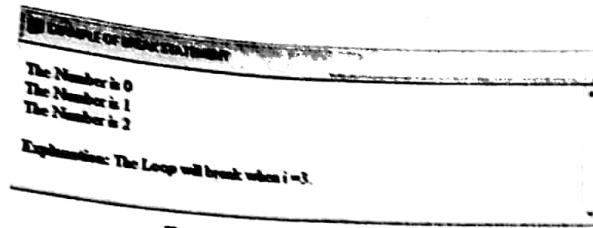
Execution of HTML codes

Break और Continue स्टेटमेन्ट्स (Break and Continue Statements)

Break स्टेटमेन्ट का प्रयोग किसी लूप को मध्य में ही छोड़कर प्रोग्राम का नियन्त्रण लूप के बाद वाले स्टेटमेन्ट पर जाने के लिए किया जाता है।

उदाहरण,

```
<html>
  <head>
    <title>EXAMPLE OF BREAK STATEMENT</title>
  </head>
  <body>
    <script type="text/javascript">
      var i=0;
      for(i=0; i<10; i++)
      {
        if(i == 3){ break; }
        document.write("The Number is " + i + "<br/>" );
      }
    </script>
    <p><b>Explanation:</b> The Loop will break when i =3.</p>
  </body>
</html>
```



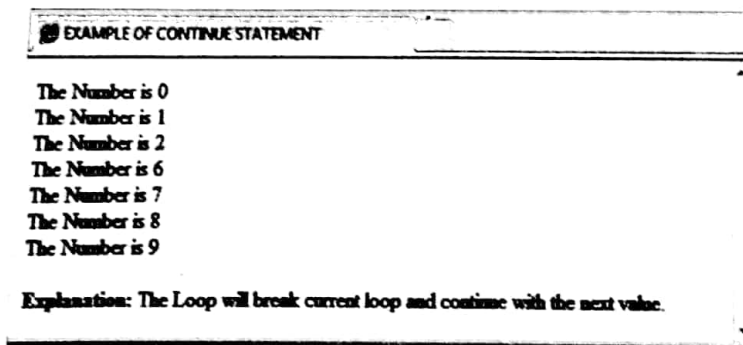
Execution of HTML codes

उपरोक्त उदाहरण में, एक वेरिएबल *i* प्रारम्भिक मान 0 के साथ घोषित किया गया है और एक for लूप को इस वेरिएबल के मान 10 के बराबर होने तक चलाने के लिए प्रयोग किया गया है। इस लूप में if का प्रयोग करके एक कण्डीशन लगाई गई कि यदि *i* का मान 3 के बराबर हो जाए, तो यह लूप break हो जाए। लूप के चलने पर जैसे ही *i* का मान 3 होता है, प्रोग्राम का नियन्त्रण इस लूप से बाहर आ जाता है और इसके बाद दी गई लाइन प्रिण्ट होती है। continue स्टेटमेन्ट का उपयोग लूप के किसी वर्तमान स्टेटमेन्ट को स्किप करके लूप के शेष अन्य स्टेटमेन्ट को run करने के लिए किया जाता है।

उदाहरण इस उदाहरण में, एक वेरिएबल *i* प्रारम्भिक मान 0 के साथ घोषित किया गया है और एक for लूप को इस वेरिएबल के मान 10 के बराबर होने तक चलाने के लिए प्रयोग किया गया है। इस लूप में if का प्रयोग करके कण्डीशन लगाई गई कि *i* का मान 3 के बराबर, *i* का मान 4 के बराबर और *i* का मान 5 के बराबर हो, तो लूप continue हो जाए। लूप के चलने पर जैसे ही *i* का मान 3, 4 और 5 के होने पर यह लूप नहीं चलता है और जैसे ही *i* का मान 5 से अधिक होता है, लूप पुनः चलना प्रारम्भ हो जाता है अर्थात् इस लूप की तीन लाइन्स The number is 3, The number is 4 और The number is 5 लाइन्स प्रिण्ट नहीं होंगी।

```
<html>
  <head>
    <title>EXAMPLE OF CONTINUE STATEMENT</title>
  </head>
  <body>
    <script type="text/javascript">
      var i=0;
      for(i=0; i<10; i++)
      {
        if(i == 3)           { continue: }
        if(i == 4)           { continue: }
        if(i == 5)           { continue: }
        document.write("The Number is "+ i + "<br/>" );
      }
    </script>
    <p><b>Explanation:</b> The Loop will break current loop
    and continue with the next value.</p>
  </body>
</html>
```

आउटपुट



Execution of HTML codes

1.9 जावा स्क्रिप्ट में फंक्शन्स (Functions in Javascript)

फंक्शन्स जावास्क्रिप्ट में फण्डामेंटल ब्लॉक्स बनाने में प्रयोग किया जाता है। फंक्शन एक जावास्क्रिप्ट प्रोसीजर है। फंक्शन अथवा प्रोसीजर, स्टेटमेंट्स का एक समूह है, जो किसी एक विशेष कार्य के लिए लिखे जाते हैं। एक फंक्शन और प्रोसीजर में केवल एक अन्तर होता है— फंक्शन सदैव कुछ-न-कुछ रिटर्न करता है, जबकि प्रोसीजर कुछ भी रिटर्न नहीं करता। फंक्शन की परिभाषा में निम्नलिखित मूलभूत भाग होते हैं—

1. कीवर्ड function
2. फंक्शन का नाम
3. फंक्शन के लिए parentheses में कॉमा से पृथक् किए गए आरग्युमेन्ट्स की सूची
4. Curly Brackets { } में फंक्शन के विभिन्न स्टेटमेंट्स

फंक्शन को परिभाषित करना (Defining a Function)

जावास्क्रिप्ट में फंक्शन को परिभाषित करते समय इसके सिन्टेक्स का ध्यान रखना आवश्यक है। जावास्क्रिप्ट एक case sensitive लैंग्वेज है और इसके स्टेटमेंट्स को कर्ली ब्रेकेट्स { } में लिखा जाना चाहिए। फंक्शन के पैरामीटर्स को कॉमाज़ द्वारा पृथक् किया जाना चाहिए और प्रत्येक स्टेटमेंट्स के अन्त में सेमीकॉलन (;) का प्रयोग किया जाना चाहिए।

फंक्शन को परिभाषित करते समय फंक्शन का नाम और इस फंक्शन को कॉल करने पर होने वाले कार्य का निर्धारण किया जाता है। फंक्शन को `<head>...</head>` टैग के अन्दर `<script>...</script>` के अन्तर्गत परिभाषित किया जाता है। फंक्शन को परिभाषित करने में फंक्शन में कॉल किए जाने वाले वेरिएबल्स को भी घोषित किया जाना चाहिए।

फंक्शन को घोषित करने का सिन्टेक्स निम्नानुसार है—

```
function <ProcedureName>([agr1] [.....,argN]
{
  <sequence of statements>;
  //Optionally, you can interrupt
  //the procedure execution:
  [return:;]
}
```

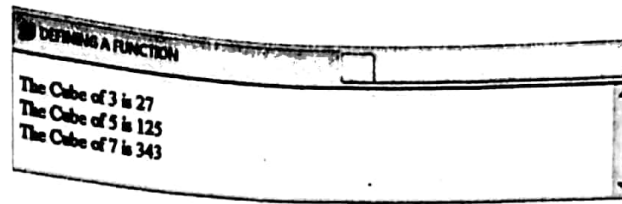
उदाहरण

```
<html>
<head>
  <title>DEFINING A FUNCTION</title>
  <script language="javascript">
    <!--
      function showCube(X)
      {
        document.write("The Cube of "+ X + " is " +X*X*X
          + "<br/>" );
      }
    <!-->
  </script>
</head>
<body>
  <script language="javascript">
    showCube(3);
    showCube(5);
    showCube(7);
  </script>
</body>
</html>
```

उपरोक्त HTML डॉक्यूमेण्ट को FUNCTION-DEFINE.html के नाम से Save किया गया है, जिसमें स्क्रिप्ट का प्रयोग करके एक फंक्शन show cube को परिभाषित किया गया है। इसमें पहले `<script>...</script>` टैग में फंक्शन को परिभाषित किया गया है और दूसरे `<script>...</script>` टैग में इस फंक्शन को तीन बार कॉल किया गया है।

guru rana

आउटपुट



Execution of HTML codes

फंक्शन को कॉल करना (Calling a Function)

फंक्शन को कॉल करना वास्तव में फंक्शन में निर्धारित कार्यों को करना है। इन फंक्शन्स को किसी HTML डॉक्युमेंट की `<body>` ...`</body>` टैग में ही कॉल किया जाता है और सामान्यतः इसमें एक पैरामीटर भी पास किया जाता है। यह पैरामीटर, फंक्शन से बाहर परिभाषित किया गया एक वैरिएबल होता है, जिस पर फंक्शन को कॉल किया जाता है।

1.10 जावास्क्रिप्ट में ऑब्जेक्ट प्रोग्रामिंग (Object Programming in JavaScript)

जावास्क्रिप्ट एक ऑब्जेक्ट ओरिएण्टेड प्रोग्रामिंग लैंग्वेज नहीं है, परन्तु जावास्क्रिप्ट में ऑब्जेक्ट बेस्ड प्रोग्रामिंग की जा सकती है। जावास्क्रिप्ट में उपलब्ध मुख्य ऑब्जेक्ट्स निम्नलिखित हैं—

1. Array ऑब्जेक्ट

इस ऑब्जेक्ट की सहायता से हम ऐसे ऐरे जो सभी प्रकार के data रखते हैं, उन्हें क्रिएट और हैण्डल कर सकते हैं, इसके विषय में हम इसी अध्याय में पहले चर्चा कर चुके हैं।

2. Boolean ऑब्जेक्ट

इस ऑब्जेक्ट की सहायता से हम एक एक्सप्रेशन से एक नया बूलियन मान क्रिएट कर सकते हैं, इसका सिन्टेक्स निम्नानुसार है—

```
var varObj = new Boolean ([expression]);
```

इसमें varObj एक वैरिएबल है, जो हमारे द्वारा क्रिएट किए गए बूलियन ऑब्जेक्ट का रेफरेन्स देने की अनुमति प्रदान करता है। New एक ऑपरेटर है, जो हमें बूलियन ऑब्जेक्ट के कन्स्ट्रक्टर फंक्शन को कॉल करने की अनुमति प्रदान करता है। Expression नए ऑब्जेक्ट का प्रारम्भिक बूलियन मान है।

3. Date ऑब्जेक्ट

ये ऑब्जेक्ट हमें अपनी स्क्रिप्ट में दिनांक (Date) और समय (Time) पर कार्य करने की अनुमति प्रदान करती है। डेट ऑब्जेक्ट्स को पढ़ने, लिखने और हैण्डल करने के लिए, जावास्क्रिप्ट में डेट ऑब्जेक्ट्स के लिए अनेक मैथड्स उपलब्ध हैं। डेट ऑब्जेक्ट्स की कोई प्रोपर्टी नहीं होती है। डेट ऑब्जेक्ट को क्रिएट करने के सिन्टेक्स निम्न हैं—

```
var varObj = new Date ( ) ;
var varObj = new Date (dateValue) ;
varObj = new Date (year, month, day [, hours
[, minutes [, second [,milliseconds] ]]]) ;
```

इनमें पहला सिन्टेक्स एक डेट ऑब्जेक्ट क्रिएट करने की अनुमति देता है जिसका प्रारम्भिक मान वही होता है, जो इस ऑब्जेक्ट के क्रिएशन (Creation) की दिनांक और समय होता है। दूसरा सिन्टेक्स एक ऐसे डेट ऑब्जेक्ट क्रिएट करने की अनुमति देता है, जिसको प्रारम्भिक मान dateValue द्वारा प्रदान किया गया है। तीसरा सिन्टेक्स एक ऐसा डेट ऑब्जेक्ट क्रिएट करने की अनुमति देता है जिसको प्रारम्भिक मान, इसमें पास किए गए पैरामीटर्स के मान का प्रयोग करके प्रदान किया गया है।

इस ऑब्जेक्ट के द्वारा उपलब्ध कराए गए मुख्य मैथड निम्नलिखित हैं—

- `getYear ()` यह मैथड दो डिजिट का इन्टीजर रिटर्न करता है, जो ऑब्जेक्ट द्वारा contain किए जाने वाले लोकल टाइम के सन् और सन् 1900 का अन्तर दर्शाता है। सन् 2000 और उससे आगे के लिए यह मैथड अप्रचलित हो गया है। इसके लिए `getFullYear ()` मैथड का प्रयोग किया जाना चाहिए।
- `getFullYear ()` यह मैथड चार डिजिट का इन्टीजर रिटर्न करता है, जो ऑब्जेक्ट द्वारा contain किए जाने वाले लोकल टाइम के सन् को दर्शाता है।

- (iii) **getMonth ()** यह मैथड ऑब्जेक्ट में contain लोकल टाइम के अनुरूप शून्य (0) और 11 के मध्य का इन्टीजर रिटर्न करता है। यह जनवरी के लिए शून्य (0), फरवरी के लिए 1 और इसी प्रकार दिसम्बर के लिए 11 मान रिटर्न करता है।
- (iv) **getDate ()** यह मैथड ऑब्जेक्ट में contain लोकल टाइम के अनुरूप शून्य (0) और 30 के मध्य का इन्टीजर रिटर्न करता है। यह पहली तारीख के लिए शून्य (0), दूसरी के लिए 1 और इसी प्रकार इक्कीस के लिए 30 मान रिटर्न करता है।
- (v) **getDay ()** यह मैथड ऑब्जेक्ट में contain लोकल टाइम के अनुरूप शून्य (0) और 6 के मध्य का इन्टीजर रिटर्न करता है। यह रविवार के लिए शून्य (0), सोमवार के लिए 1, इसी प्रकार शनिवार के लिए 6 मान रिटर्न करता है।
- (vi) **getHours ()** यह मैथड ऑब्जेक्ट में contain लोकल टाइम के अनुरूप शून्य (0) और 23 के मध्य का इन्टीजर रिटर्न करता है। यह मध्य रात्रि के बाद एक बजे के लिए शून्य (0), दो बजे के लिए 1 और इसी प्रकार मध्यरात्रि बारह बजे के लिए 23 मान रिटर्न करता है।
- (vii) **getMinutes ()** यह मैथड ऑब्जेक्ट में contain लोकल टाइम के अनुरूप शून्य (0) और 59 के मध्य का इन्टीजर रिटर्न करता है। यह पहले मिनट के लिए शून्य (0), दूसरे मिनट के लिए 1 और इसी प्रकार साठवें मिनट के लिए 59 मान रिटर्न करता है।
- (viii) **getSeconds ()** यह मैथड ऑब्जेक्ट में contain लोकल टाइम के अनुरूप शून्य (0) और 59 के मध्य का इन्टीजर रिटर्न करता है। यह पहले सेकेण्ड के लिए शून्य (0), दूसरे सेकेण्ड के लिए 1 और इसी प्रकार साठवें सेकेण्ड के लिए 59 मान रिटर्न करता है।
- (ix) **getMilliseconds ()** यह मैथड ऑब्जेक्ट में contain लोकल टाइम के अनुरूप शून्य (0) और 999 के मध्य का इन्टीजर रिटर्न करता है। यह पहले मिलीसेकेण्ड के लिए शून्य (0), दूसरे मिलीसेकेण्ड के लिए 1 और इसी प्रकार हजारवें मिलीसेकेण्ड के लिए 999 मान रिटर्न करता है।
- (x) **setYear (yearnumber)** यह मैथड ऑब्जेक्ट में वर्ष का निर्धारण करता है। इसमें पास किए गए पैरामीटर yearnumber, सम्बन्धित वर्ष और सन् 1900 का अन्तर दर्शाता है। सन् 2000 और उससे आगे के लिए यह मैथड अप्रचलित हो गया है। **setFullYear (year[month[,day]])** मैथड का प्रयोग इसके लिए किया जाना चाहिए।
- (xi) **setFullYear (year [month[,day]])** यह मैथड ऑब्जेक्ट में contain किए जाने वाले निर्धारित लोकल टाइम के सन् को दर्शाता है। इसमें सन् के साथ-साथ महीने और दिन का भी निर्धारण Integer के रूप में किया जा सकता है।
- (xii) **setMonth (month[,day])** यह मैथड ऑब्जेक्ट में contain किए जाने वाले लोकल टाइम के माह का निर्धारण करता है। इसमें महीने के साथ-साथ दिन का भी निर्धारण integer के रूप में किया जा सकता है।
- (xiii) **setDate (day)** यह मैथड ऑब्जेक्ट में contain लोकल टाइम के दिन का निर्धारण करता है।
- (xiv) **setHours (hours[,minutes[,seconds[,milliseconds]])** यह मैथड ऑब्जेक्ट में दिन के hours का निर्धारण, शून्य (0) और 23 के मध्य के इन्टीजर मान द्वारा करता है। इस मैथड में घण्टों के साथ-साथ मिनट्स, सेकेण्ड्स और मिलीसेकेण्ड्स का भी निर्धारण किया जा सकता है।
- (xv) **setMinutes (minutes[,seconds[,milliseconds]])** यह मैथड ऑब्जेक्ट में मिनट्स का निर्धारण शून्य (0) और 59 के मध्य के इन्टीजर मान द्वारा करता है। इस मैथड में मिनट्स के साथ-साथ सेकेण्ड्स और मिलीसेकेण्ड्स का भी निर्धारण किया जा सकता है।
- (xvi) **setSeconds (seconds[,milliseconds])** यह मैथड ऑब्जेक्ट में सेकेण्ड्स का निर्धारण शून्य (0) और 59 के मध्य के इन्टीजर मान द्वारा करता है। इस मैथड में सेकेण्ड्स के साथ-साथ मिलीसेकेण्ड्स का भी निर्धारण किया जा सकता है।
- (xvii) **setMilliseconds (milliseconds)** यह मैथड ऑब्जेक्ट में मिलीसेकेण्ड्स का निर्धारण शून्य (0) और 999 के मध्य के इन्टीजर मान द्वारा करता है।

4. Function ऑब्जेक्ट

फंक्शन ऑब्जेक्ट क्लास हमें नए फंक्शन क्रिएट करने की अनुमति प्रदान करता है। फंक्शन ऑब्जेक्ट को क्रिएट करने के सिन्टेक्स निम्नानुसार हैं—

```
var functionName = new Function
([arg1. [...argn. ]] statementsequence);
```

इसमें functionName एक वेरिएबल है, जो हमारे द्वारा क्रिएट किए जा रहे फंक्शन ऑब्जेक्ट का रेफरेन्स देने की अनुमति प्रदान करता है।

5. Global ऑब्जेक्ट

यह ऑब्जेक्ट क्लास अन्य ऑब्जेक्ट क्लास से कुछ भिन्न है। इस क्लास के ऑब्जेक्ट को हम new ऑपरेटर की सहायता से क्रिएट नहीं कर सकते हैं। जावास्क्रिप्ट के स्टार्ट होने के साथ ही जावास्क्रिप्ट इंजन global का एक instance क्रिएट करता है।

6. Math ऑब्जेक्ट

इस क्लास का ऑब्जेक्ट भी जावास्क्रिप्ट startup पर ही बना देता है। इसको भी इस प्रकार इसीलिए बनाया जाता है ताकि इसके मैथड प्रोपर्टीज को पूरे प्रोग्राम में से कहीं भी एक्सेस किया जा सके। हम इसके मैथड व प्रोपर्टीज को निम्नलिखित सिन्टेक्स की सहायता से एक्सेस कर सकते हैं—

Math.val ("var myDate = new Date ();")
इस ऑब्जेक्ट के मुख्य मैथड निम्नलिखित हैं—

- (i) **abs (numericExpression)** यह मैथड numericExpression आरग्युमेण्ट का Absolute मान रिटर्न करता है।
- (ii) **acos (numericExpression)** यह मैथड numericExpression आरग्युमेण्ट का Arccosine मान रिटर्न करता है।
- (iii) **asin (numericExpression)** यह मैथड numericExpression आरग्युमेण्ट का Arcsine मान रिटर्न करता है।
- (iv) **atan (y, x)** यह मैथड x-axis और origin तथा (y, x) बिन्दु को मिलाने वाली रेखा के मध्य बनने वाले कोण का मान Radians में रिटर्न करता है।
- (v) **cell (numericExpression)** यह मैथड न्यूनतम इन्टीजर मान रिटर्न करता है, जो या तो इस मैथड के आरग्युमेण्ट numericExpression के बराबर अथवा इससे बड़ा होता है।
- (vi) **cos (numericExpression)** यह मैथड numericExpression आरग्युमेण्ट का Cosine मान रिटर्न करता है।
- (vii) **exp (numericExpression)** यह मैथड numericExpression की घात e, जो Euler's constant है, का मान रिटर्न करता है। Euler's Constant का मान लगभग 2.178 होता है।
- (viii) **log (numericExpression)** यह मैथड numericExpression का natural logarithm मान रिटर्न करता है।
- (ix) **max (numericExpression1, numericExpression2)** यह मैथड आरग्युमेण्ट के रूप में दिए गए दो न्युमैरिक एक्सप्रेशन में से बड़े वाले न्युमैरिक एक्सप्रेशन को रिटर्न करता है।
- (x) **min (numericExpression1, numericExpression2)** यह मैथड आरग्युमेण्ट के रूप में दिए गए दो न्युमैरिक एक्सप्रेशन में से छोटे वाले न्युमैरिक एक्सप्रेशन को रिटर्न करता है।
- (xi) **sin (numericExpression)** यह मैथड numericExpression आरग्युमेण्ट का Sine मान रिटर्न करता है।
- (xii) **sqrt (numericExpression)** यह मैथड numericExpression आरग्युमेण्ट का वर्गमूल (Square Root) मान रिटर्न करता है। यदि numericExpression आरग्युमेण्ट एक ऋणात्मक मान है, तो यह मैथड शून्य मान रिटर्न करता है।
- (xiii) **tan (numericExpression)** यह मैथड numericExpression आरग्युमेण्ट का tangent मान रिटर्न करता है।

7. Number ऑब्जेक्ट

जब हम किसी न्युमैरिक मान वाले वेरिएबल को घोषित करते हैं, तो जावास्क्रिप्ट स्वतः ही नम्बर ऑब्जेक्ट बना देती है। फिर भी यदि हमें Explicitly नम्बर ऑब्जेक्ट बनाने की आवश्यकता होती है, तो उसके लिए सिन्टेक्स निम्नानुसार है—

```
var varObj = new number (<numericValue>);
```

चूँकि नम्बर ऑब्जेक्ट हमें न्युमैरिक नाम से सम्बन्धित विभिन्न प्रोपर्टीज उपलब्ध कराते हैं। अतः ये अत्यन्त उपयोगी होते हैं। इसकी प्रोपर्टीज को एक्सेस करने के लिए इनको क्रिएट करने की आवश्यकता नहीं होती। हम ऑब्जेक्ट क्लास का नाम अर्थात् number के बाद डॉट (.) लगाकर वांछित प्रोपर्टी का नाम टाइप करके भी प्रोपर्टी को एक्सेस कर सकते हैं।

8. Object ऑब्जेक्ट

जावास्क्रिप्ट के सभी ऑब्जेक्ट्स, Object ऑब्जेक्ट को inherit करते हैं, इसका तात्पर्य यह है कि जावास्क्रिप्ट के सभी ऑब्जेक्ट्स के पास Object ऑब्जेक्ट के मैथड्स और प्रोपर्टीज होती हैं। तदनुसार, Object ऑब्जेक्ट क्लास उन प्रोपर्टीज और मैथड्स को परिभाषित करता है, जो जावास्क्रिप्ट लैंग्वेज द्वारा उपलब्ध कराई गई सभी ऑब्जेक्ट क्लासेज के लिए common होते हैं। हम जावास्क्रिप्ट ऑब्जेक्ट को निम्न सिन्टेक्स के अनुसार क्रिएट कर सकते हैं—

```
var varObj = new Object ([expression]);
```

इसमें expression न्युमैरिक मान अथवा बूलियन मान अथवा फंक्शन अथवा कैरेक्टर स्ट्रिंग भी हो सकता है। यदि हम expression आरग्युमेण्ट को नहीं देते हैं, तो एक Empty ऑब्जेक्ट क्रिएट कर सकते हैं।

9. String ऑब्जेक्ट

जावास्क्रिप्ट में जब हम कोई वेरिएबल कोटेशन मार्क्स में स्ट्रिंग प्रकार के मान के साथ घोषित करते हैं, तो जावास्क्रिप्ट स्ट्रिंग ऑब्जेक्ट को implicitly क्रिएट करती है। साथ ही, स्ट्रिंग ऑब्जेक्ट को explicitly क्रिएट करने की आवश्यकता हो सकती है। स्ट्रिंग ऑब्जेक्ट को निम्न सिन्टेक्स के अनुसार क्रिएट कर सकते हैं—

```
var varObj = new String ("string value");
```

स्ट्रिंग ऑब्जेक्ट्स कैरेक्टर स्ट्रिंग को हैण्डल करने के लिए मैथड्स और प्रोपर्टीज उपलब्ध कराते हैं।

इस ऑब्जेक्ट के द्वारा उपलब्ध कराए गए मैथड निम्नलिखित हैं—

- (i) **charAt ()** यह मैथड निर्धारित इन्डेक्स के एक कैरेक्टर को रिटर्न करता है।
- (ii) **charCodeAt ()** यह मैथड एक संख्या रिटर्न करता है जो दिए गए इन्डेक्स के कैरेक्टर की यूनिकोड वैल्यू को दर्शाता है।
- (iii) **concat ()** यह मैथड दो स्ट्रिंग को जोड़कर एक नई स्ट्रिंग रिटर्न करता है।
- (iv) **indexOf ()** यह मैथड दिए गए मान की तुलना कॉलिंग स्ट्रिंग के ऑब्जेक्ट से करते समय जहाँ यह मान सबसे पहले प्राप्त होगा, उसकी इन्डेक्स वैल्यू को रिटर्न करता है। यदि कोई वैल्यू प्राप्त नहीं होती है, तो -1 रिटर्न करता है।
- (v) **lastIndexOf ()** ये मैथड विशिष्ट वैल्यू में से केवल अन्तिम स्ट्रिंग ऑब्जेक्ट को एक इन्डेक्स में रिटर्न करता है। यदि कोई वैल्यू प्राप्त नहीं होती है, तो -1 रिटर्न करता है।
- (vi) **localeCompare ()** ये मैथड सॉर्ट (Sort) क्रम में दी हुई स्ट्रिंग में से, उसकी रेफरेंस (Reference) स्ट्रिंग के बराबर एक संख्या को रिटर्न करता है जो संख्या उसके बराबर या उससे छोटी, बड़ी होती है।
- (vii) **match ()** ये मैथड, एक स्ट्रिंग के लिए उसके रेग्युलर अभिव्यक्ति को मैच (Match) करने के लिए प्रयोग किए जाते हैं।
- (viii) **replace ()** ये मैथड एक नियमित अभिव्यक्ति या स्ट्रिंग के मध्य के सम्बन्ध को प्राप्त करने के लिए प्रयोग किया जाता है या मैच (match) हुई सब-स्ट्रिंग को नई सब-स्ट्रिंग में बदलने के लिए भी प्रयोग किया जाता है।
- (ix) **search ()** ये मैथड एक नियमित अभिव्यक्ति (Expression) या दी हुई स्ट्रिंग के मध्य के सम्बन्ध को सर्च (Search) करने के लिए प्रयोग किया जाता है।
- (x) **slice ()** ये मैथड, एक नई स्ट्रिंग के लिए किसी विभाग या उसके ऐलोकेशन (Allocation) के लिए स्थान (Space) प्रदान करता है।
- (xi) **split ()** ये मैथड, स्ट्रिंग्स के ऐरे में से स्ट्रिंग ऑब्जेक्ट को अलग (Separate) करता है जो उसकी सब-स्ट्रिंग (Substring) कहलाती है।
- (xii) **substr ()** ये मैथड, एक दी हुई विशिष्ट लोकेशन (Location) के शुरुआत से दिए हुए कैरेक्टर की संख्या के द्वारा एक स्ट्रिंग के रूप में कैरेक्टर को रिटर्न करता है।
- (xiii) **toString ()** ये मैथड, एक स्ट्रिंग को रिटर्न करता है जो उसके विशिष्ट ऑब्जेक्ट को प्रदर्शित करती है।
- (xiv) **toLowerCase ()** ये मैथड, लोअर केस (Lower Case) में बदलने के बाद स्ट्रिंग की वैल्यू को रिटर्न करता है।
- (xv) **toUpperCase ()** ये मैथड, अपर केस (Upper Case) में बदलने के बाद स्ट्रिंग की वैल्यू को रिटर्न करता है।

1.11 जावास्क्रिप्ट में पॉपअप बॉक्स (Popup Boxes in JavaScript)

जावास्क्रिप्ट में तीन प्रकार के पॉपअप बॉक्स होते हैं— Alert डायलॉग बॉक्स, Prompt डायलॉग बॉक्स तथा Confirm डायलॉग बॉक्स।

Alert डायलॉग बॉक्स

Alert मैथड हमारे वेब पेज को विज़िट करने वाले व्यक्ति को हमारे द्वारा दिए जाने वाले मैसेज को दर्शाने के लिए एक OK बटन के साथ Alert मैसेज बॉक्स प्रदर्शित करता है। इस मैथड का प्रयोग हम निम्नांकित सिन्टेक्स के अनुरूप कर सकते हैं—

```
alert ("message")
```

उदाहरण

```
<html>
```

```
<head>
```

```
<title>EXAMPLE OF ALERT BOX</title>
```

```
<script language="javascript">
```

```
function valid(form)
```

```
{
```

```
var.input=0;
```

guru rana


```

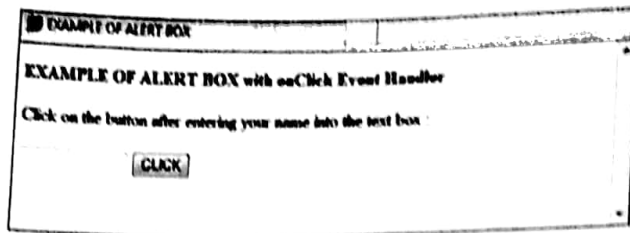
    input=document.myform.data.value;
    alert("Hello " + input + "! Welcome ....!");
  }
</script>
</head>
<body>
  <h4>EXAMPLE OF ALERT BOX with onClick Event Handler</h4>
  Click on the button after entering your name into the text box :<br>
  <form name="myform">
    <input type="text" name="data" value=" " size=12>
    <input type="button" value="CLICK" onClick="valid(this.form)">
  </form>
</body>
</html>

```

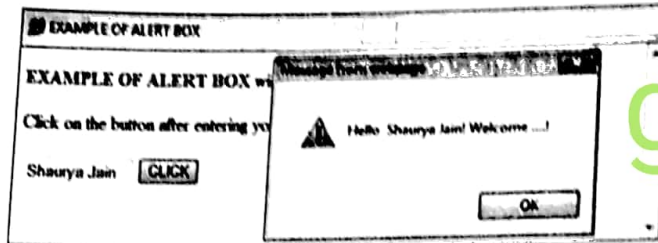
उपरोक्त HTML डॉक्युमेंट, जिसे ALERT-BOX.html के नाम से Save किया गया है, में Alert बॉक्स इवेंट के प्रयोग को दर्शाया गया है। इस डॉक्युमेंट को वेब ब्राउज़र में खोलने पर एक फॉर्म प्रदर्शित होगा। इस फॉर्म पर एक टेक्स्ट बॉक्स और एक बटन जिस पर Click लिखा हुआ है, प्रदर्शित होता है।

इस फॉर्म पर दिए गए टेक्स्ट बॉक्स में कुछ भी टाइप करके Click बटन पर क्लिक करने पर alert() मैथड को कॉल किया जा सकता है। जो निम्न चित्र में प्रदर्शित किया है।

आउटपुट



इस आउटपुट में टेक्स्ट बॉक्स में हम कोई नाम, उदाहरण के लिए Shaurya Jain, टाइप करके Click बटन पर क्लिक करते हैं, तो मॉनीटर स्क्रीन पर निम्नांकित चित्र की भाँति Alert बॉक्स का प्रदर्शन होता है—



Prompt डायलॉग बॉक्स

Prompt मैथड एक प्रॉम्प्ट डायलॉग बॉक्स प्रदर्शित करता है, जिसमें हमारा दिया हुआ मैसेज, OK और Cancel कमाण्ड बटन एवं एक इनपुट फील्ड के साथ प्रदर्शित होता है। हम इस इनपुट फील्ड के लिए एक डिफॉल्ट मान (Default Value) भी निर्धारित कर सकते हैं। यदि प्रयोगकर्ता OK कमाण्ड बटन पर क्लिक करता है, तो यह मैथड प्रयोगकर्ता के द्वारा इनपुट फील्ड में प्रविष्ट किए गए मान को रिटर्न करता है और यदि प्रयोगकर्ता Cancel कमाण्ड बटन को क्लिक करता है, तो यह मैथड कुछ भी रिटर्न नहीं करता है। इस मैथड का प्रयोग हम निम्नांकित सिन्टेक्स के अनुरूप कर सकते हैं—

```
prompt (["message" [."defaultValue"]])
```

उदाहरण

```

<html>
  <head>
    <title>EXAMPLE OF PROMPT BOX</title>
  </head>
  <body>
    <script language="javascript">
      <!--

```

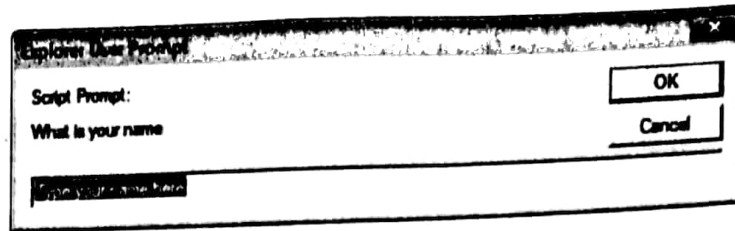
```

var name=prompt("What is your name", "Type your name here");
document.write("<center><h3>Hi "+name +
"<br> Welcome to Mypage!</h3></center>");
//-->
</script>
</body>
</html>

```

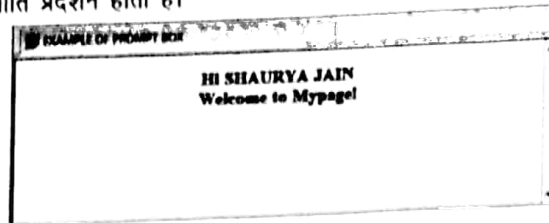
उपरोक्त HTML डॉक्युमेंट, जिसे PROMPT-BOX.html के नाम से Save किया गया है, में Prompt मैथड के प्रयोग को दर्शाया गया है।

आउटपुट



Explorer User Prompt Dialog Box

इस प्रदर्शन में टेक्स्ट बॉक्स में अपना नाम, उदाहरण के लिए SHAURYA JAIN, टाइप करके OK बटन पर क्लिक करते हैं, तो मॉनीटर स्क्रीन पर निम्नांकित चित्र की भाँति प्रदर्शन होता है।



Confirm डायलॉग बॉक्स

Confirm मैथड एक कन्फर्म डायलॉग बॉक्स प्रदर्शित करता है, जिसमें हमारा दिया हुआ मैसेज OK और Cancel कमाण्ड बटन के साथ प्रदर्शित होता है।

यह मैथड प्रयोगकर्ता के द्वारा चुने गए बटन के बूलियन मान को रिटर्न करता है। इस मैथड का प्रयोग हम निम्नांकित सिन्टेक्स के अनुरूप कर सकते हैं—

```
confirm ("message")
```

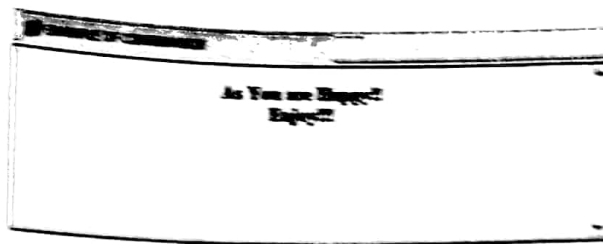
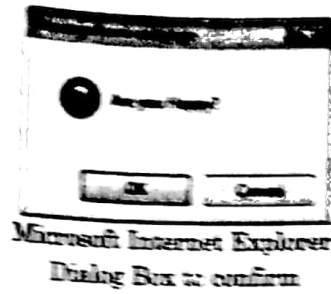
उदाहरण

```

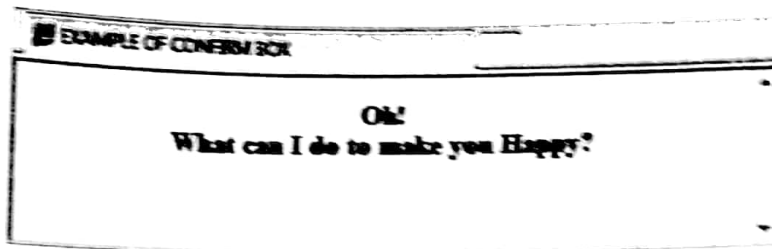
<html>
  <head>
    <title>EXAMPLE OF CONFIRM BOX</title>
  </head>
  <body>
    <script language="javascript">
      <!--
      var response=confirm("Are you Happy?");
      if( response == 1)
        document.write("<center><h3>As You are Happy!!<br>Enjoy!!!</h3></center>" );
      else
        document.write("<center><h3>Oh!<br> What can I do to make you Happy?
        </h3></center>");
      //-->
    </script>
  </body>
</html>

```

उपरोक्त HTML डॉक्युमेंट, जिसे CONFIRM-BOX.html के नाम से Save किया गया है, आगे दिए गए चित्र में Confirm मैथड के प्रयोग को दर्शाया गया है।



उस कार्यलयी बॉक्स में हमारे मैसैज के साथ OK और Cancel बटन होने से प्रत्येक को उसे एक ही प्रयोगकर्ता का प्रयोग में OK बटन प्र क्लिक करता है, वो मैसैज खोजता प्र प्रयोगकर्ता कि प्र क्लिक और यदि बटन Cancel प्र क्लिक करता है, वो मैसैज खोजता प्र कि प्र क्लिक को प्रती प्रदर्शित होता है-



12 Document Object Model (DOM)

DOM एक Application Programming Interface (API) है, जिसे XML के लिए define किया गया था। DOM किसी भी डॉक्यूमेंट को मैमोरी में नोड्स की एक hierarchy के रूप में तैयार करता है। HTML या XML document का हर element tag, attribute व text आदि DOM के नोड्स को प्रदर्शित करते हैं।

उदाहरण, निम्न HTML कोड देखिए-

```
<html><head>
<title>Sample page< title>
< /head><body>
<p>Hello World!< /p>
< /body>< /html>
```

जब ये HTML code, Web browser की memory में load होता है, तब निम्नानुसार form में विभिन्न HTML elements की एक hierarchy बन जाती है-

```
<html>
<head>
<title>Sample page< title>
< /head>
<body>
<p>Hello world!< /p>
< /body>
< /html>
```


किसी डॉक्युमेंट के विभिन्न एलिमेंट्स के मैमोरी में इस प्रकार से मॉडल होने की व्यवस्था को ही DOM या Document Object Model कहा जाता है, जिसमें डॉक्युमेंट के विभिन्न एलिमेंट्स DOM के एक नोड के साथ प्रदर्शित किए जाते हैं और हर नोड एक आब्जेक्ट की भाँति व्यवहार करता है, जिसकी स्वयं की कुछ properties व behaviour होते हैं।

डॉक्युमेंट के विभिन्न कन्टेन्ट्स की एक tree बनाकर DOM, किसी वेब डेवलपर को अपने डॉक्युमेंट पर पूर्ण रूप से कण्ट्रोल करने की सुविधा प्रदान करता है क्योंकि जावास्क्रिप्ट की किसी स्क्रिप्टिंग भाषा का प्रयोग करके वेब डेवलपर अपने डॉक्युमेंट के किसी नोड को मिटा सकता है, DOM में नया नोड जोड़ सकता है, किसी अवांछित नोड को replace कर सकता है अथवा DOM API का प्रयोग करते हुए किसी नोड को modify कर सकता है।

चूँकि web browser में document प्रस्तुत होने से पहले उस डॉक्युमेंट का DOM Tree क्रिएट होता है, जो उस डॉक्युमेंट का In-Memory Model होता है और वेब ब्राउज़र के विण्डो में वही दिखाई देता है, जो DOM Tree में होता है, इसलिए DOM में किए जाने वाले परिवर्तनों का प्रभाव तुरन्त वेब ब्राउज़र में प्रस्तुत होता है।

इसलिए DOM tree किसी भी client side scripting language के लिए एक मुख्य स्रोत होता है, जिस पर वह स्क्रिप्टिंग भाषा द्वारा विभिन्न प्रकार के कार्यों को प्रदर्शित करके डॉक्युमेंट को अधिक प्रभावित (Interactive) बनाने में सक्षम हो पाता है।

चूँकि DOM को विभिन्न कम्पनियों ने अपने-अपने वेब ब्राउज़र में अपनी सुविधानुसार अलग-अलग तरीकों से डेवलप किया था, इसलिए वेब को cross platform यानी platform independent बनाए रखने के लिए सभी वेब ब्राउज़र में किसी डॉक्युमेंट को एक जैसा दर्शाने के लिए फिर से एक प्रमाणिक (Standard) तरीके की आवश्यकता को महसूस किया गया।

फलस्वरूप एक नई संस्था अस्तित्व में आई, जिसका नाम World Wide Web Consortium (W3C) था। ये संस्था विभिन्न प्रकार के web related standards develop करने का कार्य करती है। इस संस्था (Organization) में विभिन्न बड़ी कम्पनियों जैसेकि Microsoft, Google, Yahoo, AOL आदि के मैम्बर्स भाग लेते हैं और वेब किस दिशा में आगे बढ़ेगा इस बात का निर्णय लेकर standards क्रिएट करते हैं।

DOM के आज तक में कुल तीन लेवल्स (Levels) W3C द्वारा निर्धारित किए गए हैं। DOM Level 1 सबसे पहले October, 1998 में recommend किया गया था। इस DOM के अन्य दो लेवल्स DOM Core व DOM HTML हैं।

DOM Core किसी XML आधारित डॉक्युमेंट को संरचना की सुविधा प्रदान करता है ताकि डेवलपर्स किसी XML डॉक्युमेंट के विभिन्न हिस्सों को सरलता से प्राप्त कर सकें तथा DOM HTML वास्तव में DOM Core का ही एक extension है, जिसमें HTML के साथ कुछ विशेष ऑब्जेक्ट्स व मैथड्स को ऐड करके HTML का विस्तार किया गया है।

DOM जावास्क्रिप्ट नहीं है, ECMAScript की भाँति ही इसे भी कई अन्य प्रोग्रामिंग भाषाओं में कार्यान्वित किया गया है। हालाँकि वेब ब्राउज़र्स में DOM को ECMAScript का प्रयोग करके कार्यान्वित किया गया है और अब ये DOM जावास्क्रिप्ट भाषा का एक सबसे बड़ा व सबसे महत्वपूर्ण हिस्सा है।

DOM भी ECMAScript की भाँति ही केवल एक विनिर्देश (Specification) है। जिस प्रकार से ECMAScript के आधार पर विभिन्न प्रकार की स्क्रिप्टिंग भाषाओं को विकसित किया गया है, उसी प्रकार से DOM के आधार पर विभिन्न प्रकार की प्रोग्रामिंग भाषा में किसी डॉक्युमेंट को एक्सेस व manipulate करने के तरीकों को विकसित (Develop) किया जाता है ताकि एक प्रोग्रामिंग भाषा में डेवलप किया गया डॉक्युमेंट किसी दूसरी प्रोग्रामिंग भाषा में भी सरलता से उपयोग में लिया जा सके।

हालाँकि DOM Level 1 का मूल उद्देश्य किसी डॉक्युमेंट को संरचना प्रदान करना था, ताकि डेवलपर्स जावास्क्रिप्ट जैसी client side scripting language द्वारा डॉक्युमेंट के विभिन्न हिस्सों को सरलता से access व manipulate कर सकें, जबकि DOM Level 2 को डेवलप करने का मूल उद्देश्य DOM के साथ माउस व user interface events, ranges, traversals तथा cascading style sheets को सपोर्ट करवाना था, ताकि डॉक्युमेंट का न केवल अच्छे तरीके से स्ट्रक्चर किया जा सके बल्कि उसे सरलता से डिज़ाइन भी किया जा सके। साथ ही उसे आकर्षक भी बनाया जा सके। इसलिए DOM Level 1 के कोर (Core) को XML Namespaces को सपोर्ट करने के लिए बढ़ाया गया। Dom Level 2 में निम्न नए मॉड्युल्स (Modules) को बढ़ाया गया था। जो निम्नलिखित हैं

1. DOM व्यूज़
2. DOM इवेन्ट्स
3. DOM स्टाइल्स
4. DOM ट्रेवर्सल और रेंज

डॉक्युमेंट की स्टाइलिंग करने से पहले व स्टाइलिंग करने के बाद एक ही डॉक्युमेंट के अनेक व्यूज़ हो जाते हैं। इन व्यूज़ को हैण्डल करने के लिए DOM व्यूज़ का सिद्धान्त विस्तारित किया गया।

डॉक्युमेंट को व्यूज़ के लिए अधिक आकर्षक बनाने के लिए विभिन्न प्रकार के इवेण्ट्स (Events) व इवेण्ट हैण्डलर्स (Event Handlers) को DOM Events के रूप में परिभाषित किया गया।

डॉक्युमेंट की स्टाइलिंग को कंट्रोल करने व डॉक्युमेंट के स्ट्रक्चर से अलग रखने के लिए DOM Styles का विस्तार किया गया ताकि डॉक्युमेंट की स्टाइलिंग को कंट्रोल, मैनेज व हैंडल करना सरल हो सके। DOM ट्रेवर्सल तथा रेंज का विस्तार करके DOM को access, manipulate व traverse करने के लिए नए उल्लेख को परिभाषित किया गया।

वर्तमान समय में DOM Level 3 का वर्णन किया जा रहा है, जिसमें ऐसे मैथड्स को सपोर्ट किया जा रहा है जिससे वेब ब्राउज़र या host environment के डॉक्युमेंट को लोकल डिवाइस पर सेव किया जा सके व लोकल डिवाइस से host environment में लोड किया जा सके। एक प्रकार से देखा जाए, तो अब वेब टेक्नालॉजी पूरी प्रकार से डेस्कटाप टेक्नोलॉजी के समकक्ष आने वाली है, क्योंकि DOM Level 2 तक, किसी भी डॉक्युमेंट को लोकल डिवाइस में सेव नहीं किया जा सकता था। इसीलिए कोई भी यूजर केवल वैसे ही डॉक्युमेंट देख सकता था या एक्सेस कर सकता था, जैसा डेवलपर ने उसे अधिकृत (Authorized) किया था।

लेकिन DOM Level 3 के पूर्ण निर्माण के बाद में बात पूर्ण रूप से बदल जाएगी, क्योंकि उस स्थिति में यूजर अपनी इच्छानुसार किसी डॉक्युमेंट को मॉडिफाई कर सकेगा और अपनी निजी डिवाइस पर सेव कर सकेगा, जिससे एक ही डॉक्युमेंट को अलग-अलग यूजर अपनी इच्छानुसार अलग-अलग तरीके से access व manipulate कर सकेंगे।

DOM Level 3 का निर्माण धीरे-धीरे होने लगा है और HTML5 DOM Level 3 व CSS3 का ही एक निर्माण है, जो धीरे-धीरे विभिन्न वेब ब्राउज़र में सपोर्ट किया जाने लगा है। इन मूल DOMs के स्थान पर कुछ अन्य DOMs भी हैं, जिन्हें अलग प्रकार की आवश्यकताओं को पूरा करने के लिए परिभाषित किया गया है।

उदाहरण, SVG 1.0 व Math ML 1.0 का अपना DOM है।

SVG Host Environment में ग्राफिक्स डेवलप करने से सम्बन्धित स्टैण्डर्स को हैंडल करता है। इसी प्रकार से SMIL के लिए डॉक्युमेंट में multimedia integration से सम्बन्धित DOM को स्पेसिफाई (Specify) किया गया है।

इनके स्थान पर अन्य भाषाओं ने अपनी आवश्यकतानुसार अपना स्वयं का DOM डेवलप किया है।

उदाहरण, mozilla ने XML का प्रयोग करके XUL (XML User Interface Language) विकसित किया है, जिसका प्रयोग mozilla व Firefox वेब ब्राउज़र्स के front end को डेवलप करने के लिए किया गया है, लेकिन इस भाषा को व ऐसी ही अनेक और भाषाओं को W3C ने स्टैण्डर्ड के रूप में स्वीकार नहीं किया है, जिन्हें अलग-अलग companies ने XML के आधार पर अपनी specific आवश्यकताओं को पूरा करने के लिए develop किया है।

guru rana

वर्णनात्मक प्रश्नोत्तर

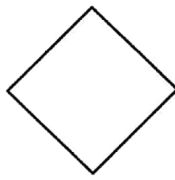
→ प्रश्नि सु-स्तर से भी बहुविकल्पीय प्रश्नों को आधार मानकर अन्य प्रकृति के प्रश्नों की तैयारी भी उनको बहुविकल्पीय में बदलकर कर सकते हैं।

1. स्यूडोकोड तथा फ्लोचार्ट में कोई एक अन्तर बताइए।
उत्तर प्रोग्राम के लॉजिक को चित्रानुसार रेखांकित करना फ्लोचार्ट (Flowchart) एवं प्रोग्राम के लॉजिक को पदों में क्रमानुसार अपनी भाषा में लिखना स्यूडोकोड (Pseudocode) कहलाता है।
2. जावास्क्रिप्ट में parseInt () फंक्शन का क्या कार्य है?
उत्तर यह फंक्शन एक कैरेक्टर स्ट्रिंग को आरग्युमेण्ट के रूप में लेता है और इण्टीजर के रूप में रिटर्न करता है, इस फंक्शन का प्रयोग निम्न सिन्टेक्स के अनुसार किया जाता है
parseInt (String [,base])
3. फ्लोचार्ट कितने प्रकार के होते हैं?
उत्तर फ्लोचार्ट मुख्यतः दो प्रकार के होते हैं
(i) प्रोग्राम फ्लोचार्ट
(ii) सिस्टम फ्लोचार्ट
4. जावास्क्रिप्ट में प्रयोग होने वाले Alert डायलॉग बॉक्स का सिन्टेक्स लिखिए।
उत्तर Alert डायलॉग बॉक्स का प्रयोग हम निम्नांकित सिन्टेक्स के अनुरूप करते हैं
alert ("message")
5. वेरिएबल को जावास्क्रिप्ट में किस प्रकार से घोषित किया जाता है?
उत्तर वेरिएबल को घोषित करने के कुछ उदाहरण निम्नलिखित हैं
var my first name = "Manish"
var avar = 11;
var nvar = null;
var n2var;
6. सिंगल लाइन कमेण्ट तथा मल्टीलाइन कमेण्ट में अन्तर बताइए।
उत्तर सिंगल लाइन कमेण्ट में एक लाइन का कमेण्ट (Comment) लिखने के लिए लाइन के पहले दो slashes (//) लगाए जाते हैं, जबकि मल्टीलाइन कमेण्ट में एक से अधिक लाइन का कमेण्ट लिखने के लिए पहली लाइन के प्रारम्भ में /* और अन्तिम लाइन के अन्त में */ चिन्ह का प्रयोग किया जाता है।
7. रिज़र्व्ड (Reserved) कीवर्ड्स से आप क्या समझते हैं? किन्हीं दो रिज़र्व्ड कीवर्ड्स के नाम लिखिए।
उत्तर आरक्षित कीवर्ड्स (Reserved Keywords) का प्रयोग जावास्क्रिप्ट द्वारा आन्तरिक ऑपरेशन्स (Internal Operations) के लिए किया जाता है। आरक्षित कीवर्ड्स के उदाहरण हैं— break, continue।
8. स्थिरांक (Constant) से क्या तात्पर्य है?
उत्तर जो मान हमेशा निश्चित होते हैं, उन मानों को होल्ड करने वाले आइडेण्टिफायर्स (Identifiers) को स्थिरांक कहा जाता है।
9. किन्हीं दो तार्किक ऑपरेटर्स के नाम बताइए।
उत्तर दो तार्किक ऑपरेटर्स निम्नांकित हैं
(i) ! — Logical NOT
(ii) && — Logical AND
10. यूनरी ऑपरेटर से आप क्या समझते हैं? ये कितने प्रकार के होते हैं?
उत्तर यूनरी ऑपरेटर को एक ही ऑपरेण्ड की आवश्यकता होती है। इन ऑपरेटर्स को निम्नलिखित दो प्रकार के नोटेशन के साथ प्रयोग किया जा सकता है
(i) prefix notation (++i)
(ii) post fix notation (i++)

guru rana

अभ्यास प्रश्न

1. जावास्क्रिप्ट एक साइड स्क्रिप्टिंग भाषा है।
 (a) सर्वर
 (b) ब्राउज़र
 (c) आई.एस.पी
 (d) इनमें से कोई नहीं
2. Dom Events को किस लेवल में extent किया गया था?
 (a) लेवल 1
 (b) लेवल 2
 (c) लेवल 3
 (d) लेवल 4
3. किस प्रकार के नोटेशन में ऑपरेटर ऑपरेन्ड के पहले प्रयोग किया जाता है?
 (a) पोस्टफिक्स
 (b) इनफिक्स
 (c) प्रिफिक्स
 (d) सुपरफिक्स
4. जावास्क्रिप्ट को कहाँ लिख सकते हैं?
 (a) Body के अन्दर
 (b) Head के अन्दर
 (c) Jsp फाइल में
 (d) ये सभी
5. निम्न में कौन डायलॉग बॉक्स नहीं है?
 (a) सॉफ्ट बॉक्स
 (b) प्रॉस्ट बॉक्स
 (c) कन्फर्म बॉक्स
 (d) एलर्ट बॉक्स
6. Date ऑब्जेक्ट में getDay () शून्य से कहाँ तक की वैल्यू रिटर्न करता है?
 (a) 5
 (b) 6
 (c) 4
 (d) 7
7. जावास्क्रिप्ट को ऑब्जेक्ट ओरिएण्टेड लैंग्वेज से प्रेरित होकर विकसित किया गया।
 (a) पास्कल
 (b) सी
 (c) जावा
 (d) HTML
8. जावास्क्रिप्ट में किसी कण्डिशन को test करने के लिए किस स्टेटमेन्ट का प्रयोग किया जा सकता है?
 (a) While
 (b) IF
 (c) Select
 (d) Break
9. Dom के अभी तक कितने लेवल हैं?
 (a) 2
 (b) 3
 (c) 4
 (d) 5
10. फ्लोचार्ट का प्रारम्भ तथा अन्त दर्शाने के लिए निम्न में से क्या उपयोगी है?
 (a) डिसीज़न
 (b) फ्लोलाइन
 (c) टर्मिनल
 (d) कनेक्टर
11. parseInt ("Arihant") जावास्क्रिप्ट में क्या रिटर्न करेगा?
 (a) Arihant
 (b) 54387
 (c) 0
 (d) NaN
12. बिटवाइज़ लेफ्ट शिफ्ट के लिए किस चिह्न का प्रयोग किया जाता है?
 (a) > =
 (b) < =
 (c) >> =
 (d) << =
13. जावास्क्रिप्ट में कौन-सा ऑपरेटर एक एक्सप्रेशन के डेटा टाइप को पहचान कर कैरेक्टर स्ट्रिंग को रिटर्न करता है?
 (a) String Operator
 (b) Delete Operator
 (c) Typeof Operator
 (d) New Operator
14. निम्न में से कौन-सा मैथड दो स्ट्रिंग को जोड़कर एक नई स्ट्रिंग रिटर्न करता है?
 (a) charAt ()
 (b) concat ()
 (c) codeAt ()
 (d) mergeAt ()
15. SVG होस्ट एनवॉयरमेंट में डेवलप किया जाता है।
 (a) ग्राफिक्स
 (b) पिक्चर
 (c) ब्राउज़र
 (d) मॉडल
16. जावास्क्रिप्ट हैण्डलर है
 (a) on Click
 (b) on Blur
 (c) on Abort
 (d) ये सभी
17. नीचे दिया गया चित्र निम्न में किसको दर्शाता है?



 (a) इनपुट
 (b) आउटपुट
 (c) डिसीज़न
 (d) इनमें से कोई नहीं
18. eval () किस प्रकार का फंक्शन है?
 (a) कम्पोजिट
 (b) यूज़र डिफाइण्ड फंक्शन
 (c) बिल्ट-इन-फंक्शन
 (d) उपरोक्त में से कोई नहीं
19. जावास्क्रिप्ट को इन्सर्ट किया जाता है
 (a) <head> सेक्शन में
 (b) <body> सेक्शन में
 (c) किसी भी एक्सटर्नल फाइल
 (d) ये सभी
20. जावास्क्रिप्ट सपोर्ट करने वाला पहला ब्राउज़र (Browser) है
 (a) Netscape
 (b) Opera
 (c) Google Chrome
 (d) Mozilla firefox

21. एक जावास्क्रिप्ट प्रोग्राम किसी डाक्यूमेण्ट को ट्रेवर्स एवं मैनीपुलेट किसके द्वारा करता है?

- (a) एलीमेण्ट ऑब्जेक्ट
(b) डॉक्यूमेण्ट ऑब्जेक्ट
(c) 'a' और 'b' दोनों
(d) उपरोक्त में से कोई नहीं

22. HTML के किस तत्त्व (Element) के अन्दर Java Script लिखी जाती है?

- (a) <is>
(b) <scripting>
(c) <script>
(d) <javascript>

23. निम्न में से किस पर जावास्क्रिप्ट execute होती है?

- (a) वेब ब्राउज़र पर
(b) नेटस्केप के सर्वर पर
(c) वेब सर्वर पर
(d) इनमें से कोई नहीं

24. कौन-सा एट्रीब्यूट जावास्क्रिप्ट वर्ज़न को होल्ड करता है?

- (a) version
(b) script
(c) language
(d) javascript

25. किसी फंक्शन के पैरामीटर, उस फंक्शन के लिए होते हैं

- (a) Local
(b) Global
(c) Dependent
(d) इनमें से कोई नहीं

उत्तरमाला

1. (b) 2. (b) 3. (c) 4. (d) 5. (a) 6. (b) 7. (c) 8. (b) 9. (b) 10. (c)
11. (d) 12. (d) 13. (c) 14. (b) 15. (a) 16. (d) 17. (c) 18. (c) 19. (d) 20. (a)
21. (c) 22. (c) 23. (a) 24. (c) 25. (a)