**ლექცია №1-Part1 of 3**

*WEB-ტექნოლოგიების მიმოხილვა, დინამიური და სტატიკური WEB-გვერდები და მათი შექმნის ინსტრუმენტები, ბრაუზერები, შესავალი JavaScript-ში, HTML დოკუმენტის სტრუქტურა, HTML დოკუმენტთან .js ფაილის მიბმა, ცვლადები, მონაცემთა ტიპები, მმართველი კომბინაციები, გამოსახულებები და ოპერატორები, არითმეტიკული მინიჭების ოპერატორები.*

WWW ან W3 [World Wide Web] - არის სერვისი, რომლის ძირითადი ფუნქციაა ჰიპერტექსტური [HyperText] ტექნოლოგიით აწყობილი ინფორმაციის შენახვა და გადაცემა. ამ მომსახურებაზე პასუხისმგებელია ვებ-სერვერები, სადაც განთავსებულია უამრავი ვებ-საიტი. ტერმინი ჰიპერტექსტი შემოიღო ტედ ნელსონმა [Theodor Holm Nelson]. ჰიპერტექსტურ ინფორმაციას წარმოადგენს ნებისმიერი ვებ-გვერდი, რომელთანაც ჰიპერბმულების [Hyperlink] მეშვეობით შეიძლება დაკავშირებული იყოს სხვა რესურსებიც.

ვებ-გვერდებიდან ინფორმაციის მისაღებად გამოიყენება ბრაუზერი [WEB Browser]. დღეისათვის უამრავი ბრაუზერი არსებობს, რომელთაგან ყველაზე პოპულარულებია:

* **Internet Explorer** - იგივე IE. იგი არის კომპანია მაიკროსოფტის [MicroSoft Inc.] ბრაუზერი, რომელიც, როგორც წესი, ჩაშენებულია MS Windows ოპერაციულ სისტემაში.
* **Mozilla Firefox** - ცეცხლოვანი მელია, უფასო, მრავალპლატფორმიანი ბრაუზერია. იგი შექმნა ორგანიზაციამ Mozilla Foundation. მისი პოპულარობა დღითიდღე იზრდება. თავდაპირველად იგი იყო მცირე ზომის სანავიგაციო კომპონენტი Gecko ძრავზე. პირველი Gecko ძრავის დანიშნულება იყო ძველი Netscape Communicator ძრავის შეცვლა.
* **Safari** - Apple-ს მიერ გამოშვებული ვებ-ბრაუზერია. მისი პირველი ვერსია გამოვიდა ახალი ათასწლეულის დასაწყისში ოპერაციული სისტემა MAC OS X-სთვის.
* **Google Chrome** - ორგანიზაცია Google-ს მიერ შემოთავაზებული ბრაუზერია. მასში გათვალისწინებულია ღია კოდის პრინციპი.
* **Opera** - ეს ბრაუზერი შექმნა კომპანიამ Opera Software. იგი მუშაობს დამოუკიდებელ ძრავზე [Presto] და მუშაობს სხვადასხვა ოპერაციულ სისტემებში - Windows, MAC OS X, Linux, Solaris და ა.შ.

ყველა ვებ-ბრაუზერი ვებ-საიტებიდან ინფორმაციას იღებს HTTP [HyperText Transfer Protocol] განაწესით.

სტატიკური ვებ-გვერდების შესაქმნელად გამოიყენება HTML [HyperText Markup Language] ტექნოლოგია. იგი შეიქმნა XX საუკუნის 80-იან წლებში ორგანიზაციაში CERN ტიმ ბერნერს ლის მიერ. [Tim Berners-Lee]. CERN არის ბირთვული კვლევის ლაბორატორია, რომელიც განთავსებულია ნაწილობრივ შვეიცარიისა და ნაწილობრივ საფრანგეთის ტერიტორიაზე.

ტექნოლოგიების განვითარებასთან ერთად საჭირო გახდა ვებ-გვერდების დაპროექტებისა და აგების ინსტრუმენტების დახვეწაც, დინამიურობის შეტანა ვებ-გვერდებზე და ამისათვის შეიქმნა პროგრამირების ენა JavaScript. ის გამოიყენება ისეთი HTML დოკუმენტების შესაქმნელად, რომლებსაც ექნება DHTML [Dynamic HTML] ტექნოლოგიის მხარდაჭერა.

JavaScript შეიქმნა ორგანიზაციაში Netscape Communications Corporations. იგი შექმნა Brendan Eich-მა. ამ ორგანიზაციის ერთ-ერთ პროდუქტს წარმოადგენს XX საუკუნის 90-იან წლებში არსებული ერთ-ერთი პოპულარული ბრაუზერი Netscape Navigator, რომელმაც ვერ გაუძლო ბრაუზერების ომს და დღეისათვის თითქმის არ გამოიყენება.

JavaScript წარმოადგენს პროგრამირების სპეციფიურ ენას. მისი კოდის შესრულება შესაძლებელია მაშინ, როცა HTML დოკუმენტში მოვათავსებთ შემდეგნაირად: <SCRIPT> სცენარი </SCRIPT>. JavaScript-ზე შექმნილი სკრიპტები გასაგებია ყველა პოპულარული ბრაუზერისათვის.

<SCRIPT> თეგი შეიცავს ორ არააუცილებელ პარამეტრს - LANGUAGE და TYPE. LANGUAGE პარამეტრის მნიშვნელობაა JavaScript, ხოლო TYPE პარამეტრის მნიშვნელობა კი - text/javascript. ფრაგმენტს ექნება შემდეგი სახე:

<SCRIPT LANGUAGE=”JavaScript” TYPE=”text/Javascript”>

JavaScript-ის კოდი ანუ სცენარი

</SCRIPT>

გამომდინარე იქიდან, რომ ძველმა ბრაუზერებმა შეიძლება JavaScript-ის კოდის შემცველი HTML დოკუმენტის ინტერპრეტაცია შეცდომით დაასრულოს, ზემოთ მოცემული ფრაგმენტი სასურველია, მოთავსებული იყოს HTML კომენტარების თეგებს შორის და ფრაგმენტი მიიღებს შემდეგ სახეს:

<SCRIPT LANGUAGE=”JavaScript” TYPE=”text/Javascript”>

<!--

JavaScript-ის კოდი ანუ სცენარი

//-->

</SCRIPT>

HTML კომენტარის დასრულების --> ნიშანთან წინ დგას ერთსტრიქონიანი კომენტარის // ნიშანი იმიტომ, რომ -- არის JavaScript-ის ოპერატორი და იგი არ უნდა განიხილებოდეს ოპერატორად.

პროგრამირების ტრადიციულ ენებში, კომპილირებული პროგრამული კოდი არ შეიცავს კომენტარებს. JavaScript-ში კი HTML დოკუმენტის ინტერპრეტაციისას კომენტარი ბრაუზერის ფანჯარაში არ გამოისახება, მაგრამ HTTP სერვერის მიერ გადაეცემა HTML დოკუმენტთან ერთად და მისი ხილვა შეუძლია HTML დოკუმენტის ნებისმიერ მომხმარებელს.

HTML სპეციფიკაციის მიხედვით, JavaScript-ის პროგრამული კოდის განთავსება რეკომენდირებულია HTML დოკუმენტის HEAD განყოფილებაში შემდეგნაირად:

<HTML>

<HEAD>

<TITLE> Document\_01 </TITLE>

<SCRIPT>

document.write (“First Program”)

</SCRIPT>

</HEAD>

<BODY>

<P> Example\_01 </P>

</BODY> </HTML>

სცენარის განთავსება შესაძლებელია HTML დოკუმენტის BODY განყოფილებაშიც შემდეგნაირად:

<HTML>

<HEAD>

<TITLE> Document\_02 </TITLE>

</HEAD>

<BODY>

<P> Example\_02 </P>

<SCRIPT>

document.write (“Second Program”)

</SCRIPT>

</BODY>

</HTML>

JavaScript-ის პროგრამული კოდის მოთავსება შესაძლებელია HTML თეგების შემოადგენლობაშიც შესაბამისი პარამეტრის მნიშვნელობად. ქვემოთ მოცემული ფრაგმენტის შესრულებისას, სკრიპტი შესრულდება Load მოვლენის მოხდენისას, მაშასადამე HTML დოკუმენტის ბრაუზერში ჩატვირთვის დასრულების შემდეგ:

<HTML>

<HEAD>

<TITLE> Document\_03 </TITLE>

</HEAD>

<BODY onLoad=”document.bgColor=’yellow’”>

<P> Example\_03 </P>

</BODY>

</HTML>

JavaScript-ის პროგრამული კოდის განთავსება შესაძლებელია სპეციალურ ტექსტურ ფაილში, რომლის გაფართოებაც იქნება js. როდესაც სკრიპტი მოთავსებულია სპეციალურ ტექსტურ ფაილში, იგი არ შეიცავს არც სკრიპტის დასაწყისისა და დასასრულის აღმნიშვნელ თეგებს და არც კომენტარის თეგებს. ტექსტურ ფაილში მოთავსებული სკრიპტის HTML დოკუმენტთან მისაერთებლად გამოიყენება <SCRIPT> თეგის SRC პარამეტრი, რომლის მნიშვნელობაა სკრიპტის შემცველი ტექსტური ფაილის სახელი. მაგალითად, თუ ტექსტური ფაილის სახელია myfile.js, მაშინ მიბმა მოხდება შემდეგნაირად:

<SCRIPT SRC=”myfile.js”>

</SCRIPT>

მოცემულ შემთხვევაში იგულისხმება, რომ სკრიპტის შემცველი ფაილი მოთავსებულია იმავე დირექტორიაში, სადაც HTML დოკუმენტია. თუ ეს ფაილი არის სხვა საქაღალდეში, მაშინ აუცილებელია, მივუთითოთ სრული გზა.

ნებისმიერი პროგრამა გამოიყენება ინფორმაციის დასამუშავებლად. ინფორმაციის გადასაცემად და დასაბრუნებლად საჭიროა ცვლადების გამოყენება. ცვლადი არის უნიკალური სახელის მქონე მეხსიერების უბანი, სადაც დროებით იწერება ინფორმაცია. ცვლადებთან მუშაობის დაწყებამდე საჭიროა ცვლადების გამოცხადება. JavaScript-ში ცვლადის აღწერა შესაძლებელია შემდეგი კონსტრუქციით: var abc ,სადაც var არის გასაღები სიტყვა [keyword], ხოლო abc კი ცვლადის სახელია.

**ცვლადის სახელის შედგენისას აუცილებელია შემდეგი წესების დაცვა:**

* ცვლადის სახელის პირველი სიმბოლო აუცილებლად უნდა იყოს ლათინური ანბანის დიდი ან პატარა ასო, ან ქვედა ტირე ან დოლარის ნიშანი
* ცვლადის სახელის პირველი სიმბოლო არ უნდა იყოს ციფრი
* ცვლადის სახელის დანარჩენი ნაწილი შეიძლება იყოს ლათინური ანბანის დიდი ან პატარა ასოები, ციფრები, დოლარის ნიშნები და ქვედა ტირეები [ნებისმიერი კომბინაციით]
* ცვლადის სახელში დაუშვებელია სასვენი ნიშნებისა და space-ების გამოყენება
* ცვლადის სახელი მგრძნობიარეა სიმბოლოთა რეგისტრის მიმართ
* ცვლადის სახელში სიმბოლოთა რაოდენობა არ არის რეგლამენტირებული
* ცვლადის სახელად დაუშვებელია JavaScript-ის რეზერვირებული და გასაღები სიტყვების გამოყენება

JavaScript-ში ცვლადის გამოყენება შესაძლებელია მისი წინასწარი აღწერის გარეშეც, თუმცა ამაზე მოგვიანებით ვისაუბრებთ.

JavaScript-ში გვაქვს მონაცემთა სამი ტიპი - რიცხვითი, ტექსტური და ბულის.

პროგრამირების სხვა ენებისაგან განსხვავებით JavaScript-ში გვაქვს რიცხვითი ცვლადის მხოლოდ ერთი ტიპი, რომელის მეშვეობითაც წარმოიდგინება ნებისმიერი რიცხვითი ცვლადი. ქვემოთ აღწერილი x, y და z ცვლადები რიცხვითი ცვლადებია:

var x=6

var y=-1.23

var z=2.34E+6

ტექსტური ცვლადები არის ერთმაგ ან ორმაგ ბრჭყალებში მოთავსებული სიმბოლოთა ერთობლიობა. ქვემოთ აღწერილი a, b და c ცვლადები ტექსტური ცვლადებია:

var a, b, c

a=”JavaScript”

b=’0678’

c=””

ტექსტური ტიპის ინფორმაციის ჩაწერისას დაუშვებელია ერთნაირი ბრჭყალების ერთმანეთში ჩალაგება მმართველი კომბინაციების გარეშე, ხოლო სხვადასხვანაირი ბრჭყალების ჩალაგება ერთმანეთში - შესაძლებელია.

მმართველი კომბინაცია არის სიმბოლოთა ერთობლიობა, რომელთაგან პირველია დახრილი ხაზი. ყველაზე ხშირი მმართველი კომბინაციებია:

\’ - ერთმაგი ბრჭყალი

\” - ორმაგი ბრჭყალი

\n - ახალი სტრიქონი

\t - ტაბულაცია

\\ - დახრილი ხაზი

განვიხილოთ ერთნაირი ბრჭყალების ერთმანეთში ჩასმის პროგრამა მმართველი კომბინაციების გამოყენებით:

<HTML>

<HEAD>

<TITLE> Document\_04 </TITLE>

</HEAD>

<BODY>

<SCRIPT >

var string1

string1=”string \” JavaScript \”“

document.write (string1)

</SCRIPT>

</BODY>

</HTML>

ზემოთ მოცემულ პროგრამაში ცვლადის მიმდინარე მნიშვნელობა ბრაუზერში გამოისახება document ობიექტის write( ) მეთოდით.

ბულის ცვლადებმა შეიძლება მიიღოს მხოლოდ ორი შესაძლო მნიშვნელობა - true და false.

JavaScript-ში ცვლადის ტიპი განისაზღვრება მასში მოთავსებული მონაცემის ტიპით და შეიძლება შეიცვალოს პროგრამის შესრულების პროცესში.

ნებისმიერი გამოსახულება შედგება ოპერანდებისა და ოპერატორებისაგან. არსებობს შეკრების, გამოკლების, გამრავლების, გაყოფის, ნაშთის აღების, პრე ინკრემენტის, პოსტ ინკრემენტის, პრე დეკრემენტის, პოსტ დეკრემენტისა და ნიშნის შებრუნების ოპერატორები.

JavaScript-ის შემადგენლობაში შედის ასევე არითმეტიკული მინიჭების ოპერატორები: +=, -=, \*=, /=, %=.

**კითხვები და სავარჯიშოები:**

№1. დაწერეთ სკრიპტი, რომელიც ბრაუზერის ფანჯარაში გამოსახავს “Hello World!”-ს:

ა) .js ფაილის გარეშე

ბ) .js ფაილის გამოყენებით

გ) დააფორმატეთ ტექსტი და მთლიანად დოკუმენტი ვიზუალური ფორმატირების საშუალებებით HTML-დან.

№2. დაწერეთ სკრიპტი, რომელიც HTML დოკუმენტის სხვადასხვა ტექსტური ობიექტებისათვის გამოიყენებს cursor თვისების სხვადასხვა მნიშვნელობებს.

№3. აღწერეთ სხვადასხვა ტიპის რამდენიმე ცვლადი:

ა) რიცხვით ცვლადებზე შეასრულეთ ინკრემენტის, დეკრემენტისა და არითმეტიკული მინიჭების ოპერაციები.

ბ) ტექსტური ცვლადების მნიშვნელობები გამოსახეთ ბრაუზერში მმართველი კომბინაციების გამოყენებით.

**ლექცია №1-Part2 of 3**

*ტექსტური გამოსახულებები, შედარების ოპერატორები, სიმბოლოთა კოდირების ცხრილები, მონაცემთა გარდაქმნა, ლოგიკური ოპერატორები, ოპერატორების პრიორიტეტი, პირობითი ოპერატორები.*

JavaScript-ში ტექსტურ ოპერანდებთან მუშაობისას გამოიყენება მხოლოდ ერთი - კონკატენაციის (შეკრების) ოპერატორი. კონკატენაცია ნიშნავს ტექსტური ცვლადების მნიშვნელობების მიწებებას. ისეთი გამოსახულების შესრულების შედეგად, რომლის ერთი ოპერანდი მაინც ტექსტური ტიპისაა, მიიღება ყოველთვის ტექსტური სტრიქონი, თუმცა არსებობს გამონაკლისებიც:

* თუ გამოსახულების პირველი ოპერანდი ტექსტური ტიპისაა, მაშინ ყველა დანარჩენი ოპერანდიც გარდაიქმნება ტექსტურ ფორმატში.
* თუ გამოსახულების პირველი რამდენიმე ოპერანდი რიცხვითი ტიპისაა, ხოლო დანარჩენები ტექსტურია, მაშინ ჯერ შეიკრიბება რიცხვითი ოპერანდები, ხოლო შემდეგ შეიკრიბება ტექსტურ ოპერანდთან და შედეგი მიიღება ტექსტური ტიპის.

განვიხილოთ კონკრეტული მაგალითი:

<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">

<!--

var a="6",b=7,c=8,d,e,f,result

d=a+b+c

e=b+c+a

f=c+a+b

result=d+e+f

document.write(result)

//-->

</SCRIPT>

მოცემული ფრაგმენტის შესრულების შემდეგ მიიღება შედეგი - 678156867 ტექსტური სტრიქონი. ეცადეთ, ახსნათ რატომ

JavaScript-ის შედარების ოპერატორების შესრულების შედეგი არის ბულის ცვლადის მნიშვნელობა - ან ჭეშმარიტი, ან მცდარი. შეიძლება შეგვხვდეს შემთხვევები, როდესაც ერთმანეთს ვადარებთ სხვადასხვა ტიპის ცვლადებს. შედარების ოპერატორიდან გამომდინარე, შესაძლოა, მოხდეს ტიპების გარდაქმნა და შემდეგ შესრულდეს შედარების ოპერაცია.

* თუ ერთი ოპერანდი რიცხვითი ტიპისაა, მეორე კი - ტექსტური, მაშინ JavaScript შეეცდება ტექსტური ტიპის ოპერანდის რიცხვითში გარდაქმნას. თუ ტექსტური ტიპის ოპერანდი რიცხვითში ვერ გარდაიქმნება, მაშინ შედარების ოპერაციის შედეგი იქნება false.
* თუ ერთი ოპერანდი რიცხვითი ტიპისაა, მეორე ოპერანდი კი ბულის ცვლადია, მაშინ JavaScript შეეცდება ბულის ტიპის ოპერანდის რიცხვით ფორმატში გარდაქმნას: true მნიშვნელობა გარდაიქმნება 1-ად, ხოლო false - 0-ად და ამის შემდეგ შესრულდება შედარების ოპერაცია.
* თუ ერთი ოპერანდი ბულის ტიპისაა, მეორე ოპერანდი კი ტექსტურია, JavaScript შეეცდება ორივე ოპერნადის რიცხვით ფორმატში გარდაქმნას და შემდეგ შესრულდება შედარების ოპერაცია. თუ ტექსტური ტიპის ოპერანდი ვერ გარდაიქმნა რიცხვითში, მაშინ შედარების ოპერაციის შედეგი იქნება false.

შედარების ოპერატორებია: == > < >= <= != === !==

ტიპების გარდაქმნა ხდება ყველა ჩამოთვლილი ოპერაციის შესრულების დროს, გარდა ბოლო ორისა. === არის შედარება იდენტურობაზე, ხოლო !== არის შედარება არაიდენტურობაზე.

ტექსტური ოპერანდების შედარების დროს ოპერაცია სრულდება სიმბოლოთა კოდირების ცხრილების მიხედვით. ASCII კოდირების ცხრილით 1 ბაიტიანი კოდირებისას შესაძლებელია 256 სიმბოლოს წარმოდგენა. ყოველ სიმბოლოს შეესაბამება თავისი უნიკალური კოდი. ციფრი 0-ის კოდია 48, ციფრი 1-ის - 49 და ა.შ. სიმბოლო A-ს კოდია 65, სიმბოლო B-ს - 66 და ა.შ. სიმბოლო a-ს კოდია 97, სიმბოლო b-ს - 98 და ა.შ.

JavaScript-ში არსებობს ლოგიკური ოპერატორები - && [AND], || [OR] და ! [NOT]. მათთან მუშაობის პრინციპი იგივეა, რაც ნებისმიერ პროგრამირების ენასა და ლოგიკურ გამოსახულებაში.

JavaScript-ში გასათვალისწინებელია ოპერაციათა პრიორიტეტი, რომელიც მსგავსია მისი წინაპარი პროგრამირების ენებისა. ოპერატორების პრიორიტეტულობის ცხრილი მკაცრად განსაზღვრავს ოპერატორების შესრულების თანმიმდევრობას. ფრჩხილების გამოყენებით კი შესაძლებელია ოპერატორების შესრულების თანმიმდევრობის შეცვლა.

პირობითი ოპერატორებია if( ), switch( ) და ? :

* if( ) პირობითი ოპერატორის ზოგადი სახე შემდეგნაირია: if (expression) {statement} . expression არის შედარების ან ლოგიკური გამოსახულება-პირობა, რომლის შესრულების შედეგი შეიძლება იყოს მხოლოდ ბულის მნიშვნელობა - true ან false. statement არის JavaScript-ის ნებისმიერი დასაშვები ოპერატორებისაგან შედგენილ გამოსახულებას. შენიშვნა: false მნიშვნელობა შესაძლოა წარმოდგენილი იყოს ექვივალენტური მნიშვნელობებითაც, მაგალითად 0, “”, null და undefined. ყველა დანარჩენი მნიშვნელობა true-ს ექვივალენტურია. მოცემული ოპერატორის შედარებით რთულ ფორმას აქვს შემდეგი სახე: if (expression) {true statements} else {false statements} .
* შემდეგი პირობითი ოპერატორია switch( ). მისი ზოგადი ფორმა შემდეგნაირად გამოიყურება:

switch (expression) {

case Case1: Case1 statements break

case Case2: Case2 statements break

case Case3: Case3 statements break

. . .

default: Default statements }

* ოპერატორ ? : -ის სინტაქსი ასე გამოიყურება: expression ? true result : false result .

**კითხვები და სავარჯიშოები:**

რა იქნება მოცემული ფრაგმენტების შესრულების შედეგები?

<SCRIPT>

var b=6, h=56

if((b>h)||((b/3)<(h%11))||h) b=3

else b=0

document.write(b)

</SCRIPT>

<SCRIPT>

var b=6,h=5

b+=h+b

document.write(b)

document.write(h)

</SCRIPT>

<SCRIPT>

var b=6,h=5

b=b+4\*h-(10+3)\*b

document.write(b)

</SCRIPT>

<SCRIPT>

var x=10,y=2,z=7,t

if((x>y)&&!z||!(x>y)) t=1

else t=0

if(t) document.write("YES")

else document.write("NO")

</SCRIPT>

<SCRIPT>

var x=3.7,y=-0.25,z

if(x<0) if(y<=0) z=2

else z=5

else if(y>=0) z=3

else z=10

document.write(x)

</SCRIPT>

<SCRIPT>

var n=-2,m=-1,j=2,k

k=n--+(m-(m<4?12:5))+--j

document.write(k)

</SCRIPT>

<SCRIPT>

var x=2,y=3

x<y?x:y=x>y?(x-3):(y+2)

document.write(x-y)

</SCRIPT>

<SCRIPT>

var i=2

if(++i==i++) document.write("YES")

else document.write("NO")

</SCRIPT>

<SCRIPT>

var a=b=1,c=0

document.write((a>b?a:b)>c?(a>b?a:b):c)

</SCRIPT>

**ლექცია №1-Part 3 of 3**

*ციკლის ოპერატორები - for( ), while( ), do...while(). ფუნქციები. ლოკალური და გლობალური ცვლადები*

პროგრამირებაში ხშირად გვხვდება ისეთი კოდები, როდესაც საჭიროა ერთი და იგივე მოქმედებების მრავალჯერ შესრულება. ასეთი ფრაგმენტების დასამუშავებლად გამოიყენება ციკლის ოპერატორები. მათი ზოგადი სქემა შემდეგნაირად გამოიყურება:

for(var counter=start; counter\_expression; counter\_statement) { statements }

while (expression) { statements }

do {

statements

}

while (expression)

გაარკვიეთ, რა შედეგი მიიღება ბრაუზერის ფანჯარაში მოცემული ფრაგმენტების შესრულების შემდეგ:

var k=6,m=11,d=7,n

for(n=2;n<=k;n++) {m+=n+d; d-=2;}

document.write(m+"<BR>"+d)

var n=2,k=7

while(n<=7) {

n+=1; if(n%5==0) k-=2 }

document.write(k+"<BR>"+n)

var k=7,m=3,n=4

while(k>n) {

--k; if(k%3==0) m+=4 }

document.write(k+"<BR>"+m)

var n=1

for(var i=1;i<=n;i++) {

document.write("Hello")

n=0

n=i+1

i=0 }

var i=0

do {

++i

if(i) ++i

else --i }

while(i<10)

document.write(i)

ფუნქცია წარმოადგენს ოპერატორთა ერთობლიობას, რომელთა შესრულებაც შესაძლებელია ფუნქციის გამოძახებისას. ფუნქცია გამოიძახება სახელით. ფუნქციის ზოგადი სახე გამოიყურება შემდეგნაირად:

function function\_name (arguments) { statements }

ფუნქციის სახელის შედგენის წესები ემთხვევა ცვლადის სახელის შედგენის წესებს.

ფუნქციის კოდი აუცილებლად უნდა იყოს მოთავსებული <SCRIPT> და </SCRIPT> თეგებს შორის HTML დოკუმენტის ნებისმიერ ადგილას. განვიხილოთ კონკრეტული მაგალითი:

<HTML>

<HEAD>

<TITLE> Example </TITLE>

<SCRIPT>

function my\_message(new\_message) {

document.write(new\_message)

}

</SCRIPT>

</HEAD>

<BODY>

<SCRIPT>

my\_message("Hello1")

my\_message("Hello2")

</SCRIPT>

</BODY>

</HTML>

როგორც ვხედავთ, ფუნქცია აღწერილია HTML დოკუმენტის HEAD განყოფილებაში, ხოლო გამოძახებულია ორჯერ BODY განყოფილებიდან. შეცდომების თავიდან ასაცილებლად აუცილებელია, რომ ობიექტის შესაბამისი მოვლენის დასამუშავებლად განკუთვნილი ფუნქციის კოდი ობიექტის აღწერაზე უფრო წინ იყოს მოთავსებული HTML დოკუმენტის შემადგენლობაში. ფუნქციის პროგრამული კოდის შესრულება შესაძლებელია მხოლოდ ფუნქციის გამოძახების შემდეგ. გამოძახების გარეშე ფუნქციის პროგრამული კოდი ბრაუზერის მიერ არ შესრულდება. ფუნქციის გამოძახებისას ძირითადი სკრიპტის მოქმედება შეწყდება და განახლდება მხოლოდ ფუნქციის მოქმედების დასრულებისას. ფუნქციის მოქმედების შედეგის დაბრუნება შესაძლებელია return ( ) ოპერატორის გამოყენებითაც, რომლის საშუალებითაც ფუნქციის მოქმედების შედეგი ფუნქციის გამომძახებელ ოპერატორს უბრუნდება. როგორც წესი, იგი არის ფუნქციის ბოლო ოპერატორი, ანუ return ( ) ოპერატორის შესრულების შემდეგ ფუნქცია დასრულებულად ჩაითვლება და მართვა გადეცემა სკრიპტის იმ ოპერატორს, საიდანაც მოხდა ფუნქციის გამოძახება. ამ ოპერატორის შემდეგ მოთავსებულ გამოსახულებებს JavaScript აღარ განიხილავს, მაშასადამე ფუნქციის მოქმედება მხოლოდ ერთი შედეგით შეიძლება დასრულდეს.

JavaScript-ში განსაზღვრულია ცვლადის მოქმედების ორი ზონა - ლოკალური და გლობალური.

ფუნქციაში აღწერილი ყველა ცვლადი არის ლოკალური, ანუ ლოკალური ცვლადის მნიშვნელობის როგორც წაკითხვა, ისე შეცვლა, შესაძლებელია მხოლოდ იმ ფუნქციის შემადგენლობაში მოთავსებულ გამოსახულებებში, სადაც თავადაა აღწერილი. ფუნქციის გარეთ ან რომელიმე სხვა ფუნქციაში მოთავსებულ გამოსახულებებში ლოკალური ცვლადის მნიშვნელობის წაკითხვა შეუძლებელია. მოცემულია კონკრეტული მაგალითი:

<HTML>

<HEAD>

<TITLE> Example </TITLE>

<SCRIPT>

function my\_function(x, y) {

var result

result = (x + y) / 2

return (result)

}

</SCRIPT>

</HEAD>

<BODY>

<SCRIPT>

var a, b, c

a = 3

b = 5

c = my\_function(a, b)

document.write (result)

</SCRIPT>

</BODY>

</HTML>

რადგან result ცვლადი აღწერილია ფუნქციაში, მისი დანახვა ფუნქციის გარედან შეუძლებელია, ამიტომ მოცემული ფრაგმენტის შესრულების შედეგი იქნება ცარიელი ფანჯარა.

ფუნქციის გარეთ აღწერილი ყველა ცვლადი ხასიათდება მოქმედების გლობალური ზონით, ანუ გლობალური ცვლადის მნიშვნელობის როგორც წაკითხვა, ისე შეცვლა შესაძლებელია HTML დოკუმენტთან დაკავშირებულ ყველა სკრიპტსა და ფუნქციაში. განვიხილოთ კონკრეტული მაგალითი:

<HTML>

<HEAD>

<TITLE> Example </TITLE>

<SCRIPT>

function my\_funqction(x, y) {

c = (x + y) / 2

}

</SCRIPT>

</HEAD>

<BODY>

<SCRIPT>

var a, b, c

a = 3

b = 5

my\_funqction(a, b)

document.write (c)

</SCRIPT>

</BODY>

</HTML>

მოცემულ ფრაგმენტში ფუნქციის გარეთ აღწერილია გლობალური ცვლადი calculate\_result და მისი დანახვა ფუნქციას შეუძლია აღწერის გარეშეც. მაშასადამე მოცემული ფრაგმენტის შესრულების შედეგი იქნება set\_number1 და set\_number2 ცვლადების საშუალო არითმეტიკული, ანუ 4.

**მაგალითები:**

პროგრამირების ენა JavaScript სასწავლად და გამოსაყენებლად არც ისე რთულია, მითუმეტეს თუ ადამიანმა იცის რომელიმე სხვა პროგრამირების ენა. ჩვენი კურსის განმავლობაში სკრიპტების რეალიზებას მოვახდენთ Notepad++ რედაქტორში [მაგალითად ვერსია 6.4.5 ან სხვა].

* დაიმახსოვრეთ, რომ ფორმატირების საშუალებების გამოყენება შესაძლებელია document.write მეთოდშიც:

<body>

<script>

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

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

</script>

</body>

* გავისწროთ წინ ლექციის თემატიკას და შევქმნათ ფუნქციური ღილაკი HTML დოკუმენტში:

<body>

<button type="button" onclick="document.write('hi')">Click Me!</button>

</body>

HTML დოკუმენტში მივიღებთ მისალმების ტექსტს.

* გადავაკეთოთ წინა მაგალითი:

<body>

<button type="button" onclick="alert('Welcome!')">Click Me!</button>

</body>

alert( ) დიალოგურ ფანჯარაში მივიღებთ მისალმების ტექსტს.

* გამოვიყენოთ getElementById მეთოდი და დავწეროთ მარტივი სკრიპტი:

<body>

<p id="demo"> 0678 </p>

<script>

function myFunction()

{

x=document.getElementById("demo");

x.innerHTML="0699";

}

</script>

<button type="button" onclick="myFunction()">Click Me!</button>

</body>

* კიდევ ერთი სკრიპტი:

<body>

<p id="demo"> 0667 </p>

<script>

function myFunction()

{

x=document.getElementById("demo")

x.style.color="#FFFF00";

}

</script>

<button type="button" onclick="myFunction()">Click Me!</button>

</body>

ფერის მითითება შესაძლებელია სახელითაც [yellow] და ათობითი კოდის მეშვეობითაც [rgb(255,255,0)].