**ლექცია 3 კომენტარები, გამოცხადება,ფუნქციები**

**კომენტარები JavaScript-ში , JavaScrip**t -ის **კოდი კომენტარში.**

გარკვეული ბრაუზერის გარკვეულ ვერსიებს არ შეუძლია **JavaScrip**t -ის დამუშავება. შესაძლებელია ასევე, რომ მომხმარებელმა თავის ბრაუზერში გამორთოს **JavaScript** -ის გამოყენება.

ამ შემთხვევაში კოდი, რომელიც მოთავსებულია <script> და </script> ტეგებს შორის, შესაძლებელია არასწორედ დამუშავდეს და ვებ-გვერდზე გამოვიდეს როგორც ტექსტი.

ამ პრობლემის მოსაგვარებლად, **JavaScrip**t -ის კოდი შეიძლება განვათავსოთ **HTML** -ის კომენტარების ტეგებს შორის. ეს მიდგომა ხელს არ შეუშლის იმ ბრაუზერს, რომელშიც მხარდაჭერილია **JavaScrip**t -ი, დაამუშავოს კოდი სწორედ; მაშინ როცა ბრაუზერს, რომელშიც არაა მხარდაჭერილია **JavaScrip**t -ი, უგულველყოს ის, რადგანაც **JavaScrip**t -ის კოდი აღქმული იქნება, როგორც კომენტარი.

მაგალითად

ლისტინგი 0 **JavaScrip**t -ის კოდი კომენტარებში

<script language=”JavaScript”>

<!-- კომენტარის დასაწყისი, რომელივ დაფარავს **JavaScrip**t -ს, იმ ბრაუზერებისგან, რომლებშიც არაა იგი მხარდაჭერილი

// **JavaScrip**t კოდი

//-->

</script>

გავიხსენოთ, რომ **HTML**-ში კომენტარები ჩაიწერება შემდეგი სახის ტეგში:

**<!--** ................... **-->** თავად კომენტარი ჩაიწერება მრავალწერტილის ადგილას და ბრაუზერის მიერ არ იქნება დამუშავებული, ანუ გამოტანილი მონიტორზე.

რაც შეეხება კომენტარებს **JavaScrip**t -ში, მას აქვს შემდეგი სახე:

ა) როცა კომენტარი ერთ სტრიქონში ეტევა, კომენტარის დასაწყისი და გამოცხადებაა ორი სლეში **//**

ბ) როცა კომენტარი ერთ სტრიქონში არ ეტევა, კომენტარის დასაწყისი და კომენტარის დასასრული : სლეში ვარსკლავით და ვარსკლავი სლეშით **/\*** ..... **\*/**

ჩავწეროთ კონკრეტული **JavaScrip**t-ის კოდი კომენტარით **HTML**-ში:

ლისტინგი 01

<html>

<head>

<title> JavaScript -ი კომენტარში </title>

<script language=”JavaScript”>

<!--

document.write(“HELLO, WORLD! გამარჯობა მსოფლიოვ!”)

//-->

</script>

</head>

<body> </body>

</html>

მნიშვნელოვანია ის თანამიმდევრობა რომელიც უნდა ჰქონდეს **<script>** -ს და კომენტარს. დასამახსოვრებელია, რომ პირველად ჩაიწერება **<script>**, რომელიც ატყობინებს ბრაუზერს, რომ ამის შემდეგ ჩართულია **JavaScript**-ის კოდი, რომელიც მოთავსებულია **HTML**-ს კომენტარების ტეგში. **HTML**-ის კომენტარების ტეგის დამამთავრებელი ნაწილის **-->** წინ იწერება ორმაგი სლეში **//,** რომელიც ბრაუზერს მიუთითებს, რომ ამის შემდეგ იწყება **JavaScript**-ის კომენტარი. ეს აუცილებელი იმიტომ, რომ ბრაუზერი არ დაიბნეს და შეცდომით არ მოახდიმოს **HTML**-ის ინტერპრეტირება, მისი დამხურავი ნაწილი, როგორც **JavaScript**-ის კოდი.

**გამოცხადება** **JavaScript**-ში

**JavaScript**-ში პროგრამის ყოველი სტრიქონი განიხილება როგორც *გამოცხადება.* ახალი გამოცხადების შესაქმნელად საჭიროა მხოლოდ გადავიდეთ ახალ სტრიქონზე ან დავსვათ წერტილ-მძიმე წინა სტრიქონის ბოლოში. ფორმალურად წერტილ-მძიმე არაა საჭირო ყოველი გამოცხადებისას, მაგრამ უმჯობესია **JavaScript**-ში ის ყოველი გამოცხადების შემდეგ.

მაგალითად:

X = 100

Y = 100; // ორი გამოცხადება, გამოყოფილნი კარეტის გადატანით

X=100; Y=200: // ორი გამოცხადება, გამოყოფილნი წერტილ-მძიმით

შენიშვნა - როცა კოდის ბლოკის კომენტირება ხდება, და ბლოკი ერთზე მეტ სტრიქონს შეიცავს, ბრაუზერი ახდენს ინტერპრეტირებას ამ ბლოკისა, როგორც ერთ სტრიქონს, თუ კი არ ჩავსვამთ კარეტის გადაყვანას და ამით მივიღებთ ახალ სტრიქონს.

*მივცეთ ვებ-გვერდს მიმზიდველი სახე*. დავიწყოთ მდგომარეობის ამსახველი სტრიქონიდან ს **(Status Bar)**. მიზნად დავისახოთ მასში მორბენალი სტრიქონის გამოყვანა. Notepad-ში ავკრიფოთ შემდეგი კოდი:

ლისტინდი 02

<HTML>

<HEAD>

<TITLE>შევქმნათ მორბრნალი სტრიქონი!</title>

<STYLE>

H2,P {font-family: LitNusx}

</style>

<SCRIPT LANGUAGE="JavaScript">

var msg= "Hello, Baby!";

var spacer= " ... ... " ;

var pos=0;

function ScrollMessage() { // **საკომენტარო I სტრიქონი**

window.status= msg.substring(pos, msg.length) + spacer + msg.substring(0,pos); // **საკომენტ. II სტრ**

pos++; //**საკომენტარო III სტრიქონი**

if (pos > msg.length) pos=0; /// **საკომენტარო IV სტრიქონი**

window.setTimeout("ScrollMessage()", 200); // **საკომენტარო V სტრიქონი**

} // **საკომენტარო VI სტრიქონი**

ScrollMessage();

</script>

</head>

<BODY>

<CENTER><H2>მორბენალი სტრიქონის მაგალითი</h2></center>

<P> შეხედეთ სტატუსის სტრიქონს! </p>

</body>

</html>

ამ კოდში გასარკვევად დაგვჭირდება **JavaScript -**ის რიგი შესაძლებლობების შესწავლა, მაგრამ ეხლა გავარჩიოთ ლისტინგ 02-ის ექვსი სტრიქონი, რომელიც **ფუნქციის** კოდს განმარტავს.

**საკომენტარო I სტრიქონი** - შეიცავსიმ ფუნქციის სახელს რომელიც განსაზღვრავს მის მიერ შესასრულებელ ოპერაციას.

**საკომენტარო II სტრიქონი -** შეიცავს ტექსტს, რომელიც იქნება გამოსახული მდგომარეობის სტრიქონში როგორც მოძრავი შეტყობინება. შეტყობინება შესდგება სტრიქონული ცვლადის - **msg** -ს მნიშვნელობის ნაწილისგან (განსაზღვრული ინდექსიდან **pos** ცვლადის ბოლომდე), სორისისგან და სტრიქონული ცვლადის - **msg** -ს მნიშვნელობის ნაწილისგან (განსაზღვრული ცვლადის დასაწყისიდან **pos** ინდექსამდე).

**საკომენტარო III სტრიქონი** - შეიცავს **pos** ცვლადის მნიშვნელობის ერთი ერთეულით გაზრდის ოპერატორს.

**საკომენტარო IV სტრიქონი** - შეიცავს **pos** ცვლადის მნიშვნელობის შედარების ოპერატორს **msg** ცვლადის სიგრძესთან. თუ ის მეტია, მაშინ **pos** ცვლადი მიიღებს ახალ მნიშვნელობას - 0-ს.

**საკომენტარო V სტრიქონი** - შეიცავს მეთოდს **window.setTimeout,** რომელიც განსაზღვრავს შეყოვნების იმ დროს, რომლის შემდეგადაც შესრულდება ოპერატორი. ჩვენს შემთხვევაში **ScrollMessage** ფუნქცია შესრულდება 200 მილიწამიანი შეყოვნებით.

**საკომენტარო VI სტრიქონი -** შეიცავს დამხურავ ფიგურულ ფრჩხილს, რომელიც აღნიშნავს ფუნქციის კოდის დასასრულს.

**ლიტერალები**

მარტივი მონაცემები რაზედაც შეიძლება აგებული იქნას პროგრამა არის ლიტერალები. მთელი ტიპის მუდმივები შეიძლება მოცემული იქნას ათობითი თექვსმეტობითი, ან რვაობითი ფორმით.

მთელი ტიპის ათობითი მუდმივები შეიძლება წარმოდგენილი იყოს ნიშნით ან ნიშნის გარეშე მაგალითად 15; 125; -156; +3546.

თექვსმეტობითი რიცხვები შეიცავენ ციფრებს 0-9-მდე და ასოებს a,b,c, d, e, f.

თექვსმეტობითი რიცხვები იწერება რიცხვის წინ სიმბოლოთი 0x, მაგალითად 0x25, 0xa1, 0xff.

რვაობითი რიცხვები შეიცავენ მხოლოდ ციფრებს 0-დან 7-ის ჩათვლით დაიწერება ასე 03, 0543. 011.

ათწილადი რიცხვების ჩასაწერად, როგორც ყველა დაპროგრამების ენაში მძიმის ნაცვლად გამოიყენება წერტილი. მაგალითად, 123. 45, -56.789.

ათწილადი რიცხვების ჩასაწერად გამოიყენება ექსპონენციალური ფორმა მაგალითად, რიცხვი 0,000000234 რომელიც შეიძლება ასე წარმოვადგინოთ 2.34x10-7 JavaScript დაპროგრამების ენაზე ჩაიწერება 2.34e-7.

ათობითი და მთელი მუდმივების გარდა არსებობს ლოგიკური მუდმივები, რომლებიც იღებენ მხოლოდ ორ მნიშვნელობას ჭეშმარიტი – true ზოგიერთ წარმოდგენაში - 1; მცდარი- false – 0.

სტრიქონული ლიტერალი წარმოადგენს სიმბოლოების თანმიმდევრობას რომელიც მოთავსებულია ‘ ‘ ან “ “ სიმბოლოებში. მაგალითად “ შედეგი” ან ’შედეგი’. ცარიელი სტრიქონული ცვლადი აღინიშნება ‘ ‘ ან “ ” სიმბოლოებით.

**ცვლადი**

ცვლადები გამოიყენება მონაცემების შესანახად. ცვლადების აღწერა ხდება იდენტიფი­კატორის საშუალებით. იდენტიფიკატორი აუცილებლად უნდა იწყებოდეს ლათინური ალფავიტის ასოთი ან გახაზვის სიმბოლოთი , შემდეგ შეიძლება ჩაწერილი იყოს ციფრი ან გახაზვის სიმბოლო. მაგალითად test1, \_my-test, test-1. ცვლადის ტიპი განისაზღვრება მათში შენახული მონაცემების ტიპით, მონაცემების ტიპის შეცვლით იცვლება ცვლადის ტიპი. ცვლადის განსაზღვრა ხდება სიტყვით var ის აუცილებლად უნდა იყოს გამოყენებული ცვლადების აღწერისას ან ინიციალიზაციისას. მაგალითად,

**var t1.**

აქ ცვლადის ტიპი არ არის განსაზღვრული და ის ცნობილი გახდება მას შემეგ რაც მას მიენიჭება მნიშვნელობა ეს კი ხდება მინიჭების ოპერატორის საშუალებით. მაგალითად:

**var t1=345**

ამ შემთხვევაში **t1** ცვლადს მინიჭებული აქვს მნიშვნელობა 345.

ცვლადის მნიშვნელობა შეიძლება შეიცვალოს ასევე მინიჭების ოპერატორით. მინიჭების ოპერატორი შეიძლება გამოყენებული იყოს ნებისმიერ ადგილზე და შეუძლია შეცვალოს არა მარტო მონაცემი არამედ ცვლადის ტიპიც. მინიჭების ოპერატორს აქვს სახე:

a=b

სადაც a - ცვლადია, b - გამოსახულება.

დასაშვებია ცვლადების შემდები აღწერა:

vat n=3245

var x=2.89

var p=true

var s=”გამოთვლა დამთავრებულია”

n და x ცვლადებს აქვთ რიცხვითი ტიპი, p-ს აქვს ლოგიკური ტიპი, ხოლო s-ს **string.**

**JavaScript** -ში განსაზღვრულია ასევე **function** ტიპი ყველა სტანდარტული ფუნქციისათვის და ასევე მომხმარებლის მიერ განსაზღვრული ფუნქციებისათვის.

**JavaScript** -ში ობიექტებს აქვთ მონაცემთა ტიპი **object**.

**ცვლადების მოქმედების არე**

**JavaScript**-ში არსებობს ორი ტიპის ცვლადები გლობალური და ლოკალური. გლობალური ცვლადი განსაზღვრულია მთელ პროგრამაში, ლოკალური ცვლადი განსაზღვრულია მხოლოდ კონკრეტული ფუნქციის შიგნით.

**გამოსახულება**

გამოსახულება შედგება ლიტერალებისაგან, ცვლადებისაგან, მოქმედებათა ნიშნებისაგან და ფრჩხილებისაგან. გამოსახულების გამოთ¬ვლის შემდეგ მიიღება ერთი მნიშვნელობა, რომელიც შეიძლება იყოს რიცხვი, სტრიქონი, ან ლოგიკური.

გამოსახულებაში არსებულ ყველა ცვლადი უნდა იყოს განსაზღვრული. თუ გამოსახულე­ბაში არსებულ რომელიმე ცვლადს არ აქვს მნიშვნელობა ან არ არის განსაზღვრული მაშინ ხდება შეცდომის დაფიქსირება.

**JavaScript** ში არსებობს **null** ლიტერალი რომელიც აღნიშნავს განუსაზღვრელ მნიშვნელობას.

ნებისმიერი გამოსახულება შედგება ოპერანდების და ოპერაციის ნიშნებისაგან. მაგალითად, გამოსახულება **a\*b** -ში, **a** და **b** ოპერანდებია, **\*** ოპერაციის ნიშანია.

ოპერაციები არსებობენ უნარული (ერთადგილიანი) და ბინარული (ორადგილიანი). გამოსახულება ჩაიწერება +A, თუ + - უნარული ოპერაციის აღნიშვნაა, ან A+B. თუ + - ბინარული ოპერაციის აღნიშვნაა.

გამოთვლილი მნიშვნელობის მიხედვით, გამოსახულება შეიძლება არსებობდეს არითმეტიკული, ლოგიკური და სტრიქონული.

არითმეტიკული გამოსახულების ჩასაწერად გამოიყენება შემდეგი ოპერატორები:

**+** შეკრება

**-** გამოკლება

 **\*** გამრავლება

 **/** გაყოფა

**%** ნაშთის მიღება გაყოფის შედეგად

**++** ოპერანდის მნიშვნელობის გაზრდა 1-ით

**--** ოპერანდის მნიშვნელობის შემცირება1-ით

\*\* ახარისხება, მაგალითად a3 ჩაიწერება შემდეგნაირად: a\*\*3 ; ასევე შესაძლებელია ახარისხება ჩაიწეროს როგორც Math.pow(a, 3); ფესვის ამოღება შეიძლება ასე ჩაიწეროს - Math.pow(a, (1/2));

ოპერაციების შესრულება ხდება მარცხნიდან მარჯვნივ არითმეტიკული ოპერაციების პრიორიტეტის მიხედვით.

**JavaScript**-ში გამოიყენება ოპერატორების ჩაწერის შემოკლებული ფორმა:

X+=Y X=X+Y

X-=Y X=X-Y

X\*=Y X=X\*Y

X/=Y X=X/Y

X%=Y X=X%Y

**JavaScript** -ში გამოიყენება შედარების შემდეგი ოპერაციები. ისინი შესრულების შედეგად იღებენ მხოლოდ ორ მნიშვნელობას **true** თუ შედარება ჭეშმარიტია ან **false** წინააღმდეგ შემთხვევაში. ეს ოპერაციებია:

**<**  ნაკლებია

**<=** ნაკლებია ან ტოლია

**==** ტოლია

**!=** არ არის ტოლი

**>** მეტია

**>=** მეტია ან ტოლია

**!**  ოპერაცია (ლოგიკური არა) გამოიყენება ლოგიკური ტიპის ოპერანდებზე. თუ **a** ოპერანდის მნიშვნელობა **true**, მაშინ **!a** – მნიშვნელობა არის მცდარი, თუ **a** ოპერანდის მნიშვნელობა არის **false**, მაშინ **!a** - მნიშვნელობა არის ჭეშმარიტი. გარდა აღნიშნულისა ლოგიკური ოპერაციების შესასრულებლად გამოიყენება შემდეგი ლოგიკური ოპერაციები:

**&&** (ლოგიკური და), **||**  (ლოგიკური ან)

მათი შესრულების შედეგი მოყვანილია ცხრილში

A B A&&B A||B

­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

true true true true

true false false true

false true false true

false false false false

სტრიქონულ მნიშვნელობებზე დასაშვებია კონკატენციის (შეერთების) ოპერაცია. ის აღინიშნება **+** ნიშნით.

მაგალითად,

st1= “მიმდინარე”

st2=”მდგომარეობა”

st3=st1+st2

ან

st1+=st2

მაშინ st3 მიიღებს მნიშვნელობას “მიმდინარე მდგომარეობა”

st1-იც მიიღებს მნიშვნელობას “მიმდინარე მდგომარეობა”

**JavaScript**-ში გამოიყენება შემდეგი ოპერატორები (ისინი მოცემულია პრიორიტეტის შემცირების მიხედვით):

*ინკრემენტი* **++**

*დეკრემენტი* **--**

*უარყოფა* **!**

*უნარული მინუსი* **-**

*გამრავლება* **\***

*გაყოფა, გაყოფა ნაშთის შენახვით* **/,%**

*გამოკლება* **-**

*შედარება* **<,>,<=,>=**

*ტოლობა* **==**

*არ არის ტოლი* **!=**

*ლოგიკური და* **&&**

ლოგიკური ან  **||**

მინიჭება **+, +=, -=, \*=, /=, %=, !=**

**მონაცემთა ტიპების გარდაქმნა**

მონაცემთა ტიპები გავლენას ახდენს ოპერატორების შესრულებაზე, განსაკუთრებით მაშინ თუ ოპერანდებს აქვთ სხვადასხვა ტიპი. ავიღოთ მაგალითად ორი რიცხვის შეკრების მაგალითი

3+3 შედეგი იქნება 6,

მაგრამ როცა ოპერანდებს სხვადასხვა ტიპი აქვთ

3+”3” შედეგი იქნება”33”

რადგან მეორე ოპერანდის ტიპი არის სტრიქონული ე.ი შედეგი იქნება სტრიქონული ტიპის. პირველი ოპერანდი ავტომატურად გარდაიქმნება სტრიქონულ ტიპად და შედეგად მიიღება ორი სტრიქონის გაერთიანება. თუ ადგილი ექნება ასეთ შემთხვევას:

3+3+”3” მაშინ შედეგი იქნება “63”.

**JavaScript** ენას აქვს შესაძლებლობა სტრიქონული სიმბოლოები გარდაქმნას რიცხვით მონაცემებად. ამისათვის იყენებს ფუნქციებს: **parselnt ()** და **parseFloat()**.

მაგალითად:

**parselnt**(“34”) შედეგი იქნება 34

**parselnt**(“23.56”) შედეგი იქნება 23

ე. ი მეორე შემთხვევაში შედეგის დამრგვალება მოხდება მთელამდე.

**parseFloat** (“42”) შედეგი იქნება 42

**parseFloat** (“42.33”) შედეგი იქნება 42.33

რომ გამოვიყენოთ ეს ფუნქციები იქ სადაც საჭიროა გარდაქმნა უნდა ჩავსვათ შესაბამისი ფუნქცია.

 მაგალითად:

3+3+ **parselnt(**”3”) შედეგი იქნება 9.

რიცხვითი მნიშვნელობის გარდაქმნა სტრიქონულად განვიხილოთ მარტივ მაგალითებზე:

(“ ”+2500) შედეგი იქნება2500

**(“ ”+2500).length** შედეგი იქნება 4.

**length** ნიშნავს რომ დათვლილი იქნება სიმბოლოების რაოდენობა. მაგალითად, “2500” სიმბოლოების რაოდენობა მასში ტოლია 4-ის. **length** - სიგრძე , მისი ტიპი არის რიცხვი და სტრიქონი.

ლისტინგი 03. კვადრატული სამწევრი

<html>

<head>

<title> კვადრატული განტოლების ამოხსნა </title>

</head>

<body>

<h3> კვადრატული განტოლების ფესვების პოვნა </h3><br>

<h1> შეგვყავს კვადრატული სამწევრის კოეფიციენტები</h1>

<br>

<script LANGUAGE="javascript">

a=prompt("შეიყვანეთ პირველი კოეფიციენტი a-ს მნიშვნელობა");

document.write("a=",a);

document.write("<br>");

b=prompt("შეიყვანეთ მეორე კოეფიციენტი b-ს მნიშვნელობა<br>");

document.write("b=",b);

document.write("<br>");

c=prompt("შეიყვანეთ მესამე კოეფიციენტი c-ს მნიშვნელობა");

document.write("c=",c);

document.write("<br>");

a=parseFloat(a);

b=parseFloat(b);

c=parseFloat(c);

document.write("კოეფიციენტების ჯამი=",a+b+c);

document.write("<br>");

D=b\*\*2-4\*a\*c;

document.write("დისკრიმინანტი D=",D);

document.write("<br>");

d=Math.pow(D,(1/2));

document.write("ფესვი დისკრიმინანტიდან D^(1/2)=",d);

document.write("<br>");

</script>

</body>

</html>