ინტერნეტ დაპროგრამება, ლექცია 4, ჯავასკრიპტი, **ფუნქციები**

**ფუნქცია**

პროგრამის შედგენის დროს გამოიყენება ლოგიკურად დამოუკიდებელი ნაწილები (ქვეპროგრამები). პროგრამის ყოველი ნაწილი შეიძლება დავყოთ ცალკეულ ქვე­პროგრამად. ერთხელ შექმნილი პროგრამა შეიძლება გამოვიყენოთ მრავალჯერ, ბევრ ენაში ქვეპროგრამის გაგება რეალიზდება პროცედურების, ფუნქციის, მოდულების და ა.შ დახმარებით.

**JavaScript**-ში მთავარი ელემენტი არის ფუნქცია.

*ფუნქცია ესაა ის სამშენებლო ბლოკი, რომელიც გამოიყენება მრავალჯერ*.

**ფუნქციის აღწერას აქვს სახე**: **function** F (V) {S}

სადაც, F ფუნქციის სახელია, რომლის საშუალებითაც ხდება მასზე მიმართვა, V- *ფუნქციის პარამერტების (არგუმენტების) სიაა*, ისინი ერთმანეთისგან გამოიყოფიან მძიმით; S- *ფუნქციის ტანია,* მისი საშუალებით მოიცემა ის შესასრულებელი მოქმე­დება, რომლითაც უნდა მივიღოთ შედეგი.

ფუნქციის აღწერისას უნდა გავითვალისწინოთ, რომ ის არ უნდა შეიცავდეს სხვა ფუნქციის აღწერას.

თუ ფუნქციის აღწერას აქვს სახე:

**function** F (v1,v2, ... vn) {s}

მაშინ ფუნქციის გამოძახებას უნდა ქონდეს სახე:

F (e1,e2, ... ,en)

სადაც, e1,e2, ... en– გამოსახულებებია რომლებიც ფაქტიურ პარამეტრებს (არგუმენ­ტებს) განსაზღვრავენ.

v1,v2, ..., vn პარამეტრებს, რომლებიც ფუნქციის აღწერაში არის მითითებული ქვია ***ფორმალური პარამეტრები***. თუ ფუნქციას არ აქვს პარამეტრები მაშინ მის აღწერას აქვს სახე:

function F () {s}

ფრჩხილების არსებობა გამოძახების ოპერატორში აუცილებელია ანუ ფუნქციის გამოძახებას ასეთ შემთხვევაში ექნება სახე:

F ().

*შენიშვნა*: ყველა განსაზღვრა და ფუნქცია უნდა შეტანილი იყოს <HEAD> განყოფილე­ბაში.

განვიხილოთ მართკუთხა სამკუთხედის ფართობის გამოთვლის პროგრამა ფუნქციის გამოყეენებით, როდესაც მოცემულია მისი კათეტები.

<HTML>

<HEAD>

<TITLE> პროგრამა ფუნქციის გამოყენებით </TITLE>

<script language=”JavaScript”>

<!-- //

function ss (a,b)

{return a\*b/2 }

// -->

</script>

</HEAD>

<BODY>

<P> ფურცლის გამოსახვა ფუნქციით</P>

<script>

<!- -

var a1=6; b1=9

var s= ss(a1,b1)

document.write (“მიიღება მნიშვნელობა ”, s,”.”);

//-->

</script>

<P> ფურცლის ფორმირების დასასრული </P>

</BODY>

</HTML>

ფუნქციის ტანი შედგება მხოლოდ ერთი **return** ოპერატორით, რომელიც განსაზღვრავს

ფუნქციის გამოძახება ხდება ფუნქციის ტანში მინიჭების ოპერატორით:

s=ss(a1,b1)

ფორმალურ a და b პარამეტრს მიენიჭება a1-ის და b1-ის ფაქტიური მნიშვნელობა და შესრულდება ფუნქციის ტანი. მიღებული მნიშვნელობა თავსდება დოკუმენტში write ოპერატორით.

*განხილულ შემთხვევებში მომხმარებელს არ ქონდა შესაძლებლობა შეეტანა მნიშვნელობები და მიეღო შესაბამისი შედეგი!*

მოვიყვანოთ ფუნქციის კიდევ ერთი მაგალითი:

**function Greet**() {

**alert** (“აბა, ჰე!”)

}

ამ მაგალითში **function** საკვანძო სიტყვის გამოყენებით განვსაზღვრეთ Greet() ფუნ­ქცია, რომლის გამოძახებისას (იხ. ქვემოთ) შესრულდება ფიგურულ ფრჩხილებში მოყვანილი ოპერატორების მიმდევრობა \_ მოცემულ შემთხვევაში ეს გახლავთ ერთადერთი **alert** () ოპერატორი, რომელიც წარმოადგენს ჩაშენებულ ფუნქციას.

მისი დანიშნულებაა ეკრანზე რაიმე შეტყობინების გამოყვანა.

ფუნქციისათვის უფრო მეტი მოქნილობის მისანიჭებლად უმეტეს შემთხვევაში იყენებენ პარამეტრებს (არგუმენტებს).

პარამეტრები წარმოადგენს იმ ცვლადებს, რომელთა მნიშვნელობები ფუნქციას გადაეცემა მისი გამოძახების მომენტში.

მაგალითად, ზემომოყვანილი კოდის ფრაგმენტი შეიძლებოდა ასე გაგვერთუ-

ლებინა:

**function Greet** (who) {

**alert** ( “აბა, ჰე, ” + who);

}

ცხადია, ფუნქციის გამოძახების მომენტში who ცვლადს რაიმე მნიშვნელობა

უნდა ჰქონდეს მინიჭებული.

ფუნქციას ტრადიციულად <HEAD> უბანში განსაზღვრავენ, ხოლო გამოძახებას ახდენენ კოდის სხეულში \_ მიუთითებენ ფუნქციის სახელს და, საჭიროების შემთხვევაში, განსაზღვრავენ პარამეტრების მნიშვნელობას.

მაგალითად:

**ლისტინგი 1**

<HTML>

<HEAD>

<TITLE>ფუნქციების გამოყენება</title>

<STYLE>

H2, P {font-family: LitNusx}

</style>

<SCRIPT LANGUAGE="javascript">

**function Greet** (who) {

alert("აბა, ჰე, " + who );

}

</script>

</head>

<BODY>

<center><H2>ვისწავლოთ ფუნქციებთან მუშაობა!</h2>

<SCRIPT LANGUAGE="javascript">

Greet("გიგლა!");

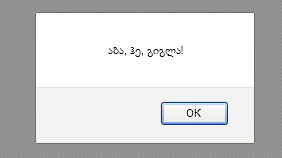
Greet("გაგა!");

</script>

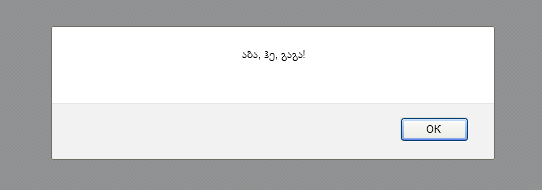
</body>

</html>

Greet (“გიგლა!”) ოპერატორით ფუნქციის პირველი გამოძახებისას ეკრანზე გამოდის შესაბამისი შეტყობინების ფანჯარა:



OK-ზე დაწკაპუნების შემდეგ მას შეცვლის ასეთივე ფანჯარა, ოღონდ ამჟამად შეტყობინებაში გამოვა “აბა, ჰე, გაგა!”.



ფუნქციას არა მარტო შეტყობინების გამოყვანა შეუძლია - მისი მეშვეობით ხში­­რად გამოთვლიან რაიმე მნიშვნელობასაც. ამ მიზნით, ფუნქციას შესაძლოა რამდენიმე პარამეტრი გადაეცეს, მაგრამ სცენარისთვის უკან დასაბრუნებელი მნიშვნელობა კი მხოლოდ ერთადერთი შეიძლება იყოს და ეს მნიშვნელობა თვით ამ ფუნქციის სახელთან დაკავშირდება.

ქვემოთ მოყვანილი კოდის ფრაგმენტში Average ფუნქციას **return** ოპერატორით უბრუნდება result ცვლადის მნიშვნელობა, რომელიც წარმოადგენს ამ ფუნქციისათ­ვის გადაცემული 4 პარამეტრის მნიშვნელობების საშუალო არითმეტიკულ სიდიდეს:

<SCRIPT LANGUAGE=”javascript”>

function Average (a,b,c,d) {

result=(a+b+c+d)/4;

return result;

}

</script>

Average ფუნქციის მნიშვნელობას კი შემდგომ სხვა ოპერატორებში გამოვიყენებთ. ცხადია, ფუნქციის გამოძახების მომენტში a, b, c, d ცვლადების მნიშვნელობა განსაზ­ღვრული უნდა იყოს. ეს შეიძლება პირდაპირი გზითაც მოხდეს, მაგალითად:

score = Average(3,4,5,6);

დასაშვებია ფუნქციის გამოძახება გამოსახულების შემადგენელი ნაწილის სახითაც:

alert (Average(1,2,3,4));

**ლისტინგი 2,** საშუალო არითმეტიკულის გამოთვლა წინასწარი ოთხი რიცხვისთვის

<HTML>

<HEAD>

<TITLE>საშუალო არითმეტიკული</title>

<STYLE>

H2, P, body {font-family: LitNusx;

}

</style>

<SCRIPT LANGUAGE="javascript">

function Average (a,b,c,d) {

result=(a+b+c+d)/4;

return result;

}

</script>

</head>

<BODY>

<center><H2>ვისწავლოთ ფუნქციებთან მუშაობა!</h2>

<SCRIPT LANGUAGE="javascript">

score = Average(3,4,5,6);

document.write ("მიიღება მნიშვნელობა საშუალო არითმეტიკულისა=", score,".");

alert (Average(1,2,3,4));

</script>

</body>

</html>

ჩვენ უკვე გავეცანით ჩაშენებულ ფუნქციას **alert** ()-ს, რომელსაც გამოყავს ფანჯარა და კოდის შესრულება არ გაგრძელდება, სანამ მომხმარებელი არ დააჭერს ღილაკს - **ok**. ასეთ ფანჯარას **მოდალური** ფანჯარა ჰქვია. ეს ნიშნავს იმას, რომ მომხმარებელი ვერ განახორციელებს ურთიერთობას კოდის სხვა ნაწილებთან, სანამ არ დაასრულებს ურთიერთობას ამ ფანჯარასთან. ფუნქციას **alert** ()-ის შემთხვევაში აუცილებელია ღილაკი - **ok**-ის დაჭერა.

მოდალური ფანჯარა გამოაქვს ასევე შემდეგ ჩაშენებულ ფუნქციას - **confirm**(). ამ ფანჯარაში გამოდის ტექსტი, რომელიც ფრჩხილებშია ჩაწერილი და ორი ღილაკი **ok** და **cancel**. ამ ფუნქციას შემდეგი სინტაქსისი აქვს:

result = **confirm**(question);

მოდალურ ფანჯარაში გამოვა ტექსტი - question და ორი ღილაკი - **ok** და **cancel**. თუ მომხმარებელი დააჭერს ღილაკს - **ok**, მაშინ ცვლად result -ს მიენიჭება ლოგიკური მნიშვნელობა true, იგივე 1; ხოლო, თუ მომხმარებელი დააჭერს ღილაკს - **cancel**, მაშინ ცვლად result -ს მიენიჭება ლოგიკური მნიშვნელობა false, იგივე 0. ფაქტიურად **confirm**() ჩაშენებული ფუნქციით ხდება ვებ-გვერდთან იტერაქტიულობა, ამასთან შესაძლებელია მონაცემების შეყვანა, კერძოდ ლოგიკური მონაცემების.

არსებობს ტექსტური ინფორმაციის შეყვანისთვის განკუთვნილი ჩაშენებული ფუნქცია - **prompt**(). ამ ფუნქციის სინტაქსისია

result = **prompt**(title, [default]);

**prompt**() ფუნქციის შესრულებისას გამოდის მოდალური ფანჯარა: title-ის ადგილას ჩაწერილი ტექსტით; სტრინგის ტიპის მონაცემის შესაყვანი ველით, რომელშიც არასავალდებულო [default]-ის არსებობისას, ნიმუშის სახით იქნება ჩაწერილი default-ის ადგილას ჩაწერილი სიმბოლოები; ორი ღილაკი - **ok** და **cancel**. თუ მომხმარებელი დააჭერს ღილაკს - **ok**, მაშინ ცვლად result -ს მიენიჭება შეყვანილი ტექსტური მნიშვნელობა, ხოლო, თუ მომხმარებელი დააჭერს ღილაკს - **cancel**, მაშინ ცვლად result -ს მიენიჭება მნიშვნელობა null. არსებითია გვახსოვდეს, რომ შეყვანილი მონაცემები არის სტრიქონული ტიპის, იმ შემთხვევაშიც კი, როცა აკრიფეთ რაიმე რიცხვი - მთელი, თუ ნამდვილი.

ამიტომ, თუ გვინდა რომ ცვლადმა მიიღოს რიცხვითი მნიშვნელობა, საჭიროა შეყვანილი სტრიქონული მნიშვნელობა გადავიყვანოთ რიცხვით ცვლადში, მაგალითად შემდეგი ორი ფუნქცის - **parselnt ()** და **parseFloat()** საშუალებით.

ჩვენ მოვახდენთ ლისტინგ 2-ის მოდიფიცირებას, რათა გამოვთვალოთ ნებისმიერი ოთხი რიცხვის საშუალო არითმეტიკულის მნისვნელობა, რისთვისაც საჭიროა, მოვახდინოთ ამ ოთხი რიცხვის შეყვანის პროცესის ორგანიზება, რისთვისაც გამოყენებულია ჩაშენებული ფუნქციები **prompt**(), **parseFloat**(). ასევე მომხმარებლის ფუნქცია - Average (a1, b1, c1, d1). შედეგად მივიღებთ შემდეგ ლისტინგს.

ლისტინგი 3, საშუალო არითმეტიკულის გამოთვლა ნებისმიერი ოთხი რიცხვისთვის

<HTML> <HEAD> <TITLE>საშუალო არითმეტიკული</title>

<STYLE>

H2, P, body {font-family: LitNusx;

text-align: center

}

body {background-color: coral; }

</style> <SCRIPT LANGUAGE="javascript">

function Average (a, b, c, d) { result=(a + b + c + d) / 4; return result;

} </script> </head> <BODY> <H2>ვისწავლოთ ფუნქციებთან მუშაობა!</h2>

<SCRIPT LANGUAGE="javascript"> a1 = prompt("შეიყვანეთ პირველი მნიშვნელობა", 7); b1 = prompt("შეიყვანეთ მეორე მნიშვნელობა", 5); c1 = prompt("შეიყვანეთ მესამე მნიშვნელობა", 8); d1 = prompt("შეიყვანეთ მეოთხე მნიშვნელობა", 6); a1 = parseFloat(a1);b1 = parseFloat(b1);c1 = parseFloat(c1);d1 = parseFloat(d1);result1 = Average (a1, b1, c1, d1);

alert("საშუალო არითმეტიკულის გამოთვლა = " + result1);

document.write("საშუალო არითმეტიკულის გამოთვლა = ", result1); </script>

</body>

</html>

ჩვენ შეგვიძლია ფუნქციები გამოვიყენოთ არა მარტო არითმეტიკული ოპერაციების შესასრულებლად, კერძოდ სტრიქონული ტიპის მონაცემების გაერთიანებისთვის. კერძოდ, ვთქვათ შეგვყავს ცალ-ცალკე პირის საიდენტიფიკაციო მონაცემები და შემდეგ საჭიროა ამ მონაცემების ერთიანი ჩანაწერით გამოტანა/გაერთიანება. ამისთვის ჩვენ მოვახდენთ ლისტინგ 4-ის მოდიფიცირებას და მივიღებთ

ლისტინგი 5, ჩანაწერების გაერთიანება

<HTML>

    <HEAD>

    <TITLE>ჩანაწერების გაერთიანება</title>

    <STYLE>

    H2, P  {font-family: LitNusx}

    h3 {color:rgb(243, 3, 3)}

    </style>

    <SCRIPT LANGUAGE="javascript">

     function Gaertianeba (a, b, c, d)

     {

        return (a + " " + b + " " + " " + c + " " + d);

     }

    </script>

    </head>

    <BODY>

    <H2>ვისწავლოთ ფუნქციებთან მუშაობა!</h2>

    <SCRIPT LANGUAGE="javascript">

        a1 = prompt("სახელი");

        b1 = prompt("მამის სახელი");

        c1 = prompt("გვარი");

        d1 = prompt("დაბადების წელი");

        result1 = Gaertianeba (a1, b1, c1, d1);

    alert("ერთიანი ჩანაწერის სახით: " + result1);

    document.write("ერთიანი ჩანაწერი: " + "<h3>" + result1 + "</h5>");

    </script>

    </body>

    </html>

გავეცნოთ ენის სხვა, ასევე უმნიშვნელოვანეს კომპონენტს - **ობიექტებს**.

თუ ცვლადი მხოლოდ ერთ მნიშვნელობას შეიცავს (რიცხვითს, ტექსტურს, თარიღის ტიპის და სხვ.), ობიექტი შეიძლება წარმოვიდგინოთ, როგორც ერთ სახელთან დაკავ­შირებული (ცხადია, ამავე დროს საკუთარი სახელის მქონეც) ცვლადების კრებული. ამ ცვლადთაგან თითოეულს უწოდებენ თვისებას. ობიექტს, მაგალითად, შეიძლება წარმოადგენდეს მომხმარებელი, ხოლო მისი თვისებები გახლდეთ:

სახელი, გვარი, მისამართი, ტელეფონის ნომერი და სხვ.

თითოეული ამ თვისებათაგანი რომელიმე კონკრეტული ობიექტისათვის, მაგალი-

თად, Bob-ისთვის, ასე შეიძლება გამოვსახოთ:

Bob.addres ან Bob.phone

თვისებების გარდა, ობიექტები ხშირად შეიცავენ მეთოდებსაც. მეთოდი შეიძლება განვმარტოთ როგორც ფუნქცია, რომელიც გარკვეული წესით დაამუშავებს ობიექტის თვისებას (ან თვისებებს).

მაგალითად, Bob.display() მეთოდი საშუალებას იძლევა, ეკრანზე აისახოს Bob-ობიექტის ყველა თვისება.

**javascript**-ში საქმე გვაქვს 3 ტიპის ობიექტებთან:

* ***ჩაშენებული ობიექტები***. ზემოთ ჩვენ უკვე გავეცანით ორ ასეთ ობიექტს: Date() და Math().
* ***ბრაუზერის ობიექტები***. ჩვენ მიერ განხილული alert() ფუნქცია, ფაქტობრივად, windows-ობიექტის ერთ-ერთ მეთოდს წარმოადგენს.
* ***მომხმარებლის მიერ შექმნილი ობიექტები.***

****

სამივე ტიპის ობიექტებს უფრო დაწვრილებით შემდგომში გავეცნობით.