**ინტერნეტ დაპროგრამება. ლექცია № 11. ობიექტები**

**ობიექტები**

ჩვენ უკვე შეგვექმნა წარმოდგენა ობიექტებზე, მათ თვისებებსა და მეთოდებზე. ვიცით, რომ JavaScript-ში საქმე გვაქვს 3 სახის ობიექტებთან. სანამ დაწვრილებით გავეცნობოდეთ თითოეულ მათგანს, ვნახოთ, თუ როგორ ხდება ობიექტების შექმნა.

 JavaScript ამ მიზნით იყენებს სპეციალურ ფუნქციებს, ე.წ. კონსტრუქტორებს. მაგალითად, ჩვენ შეგვიძლია შევქმნათ სტრიქონული ცვლადის სახის მქონე ობიექტი (მიღებულია გამოთქმა -“ობიექტის ეგზემპლარი”) **String**  ჩაშენებულ ფუნქციაზე დაყრდნობით:

**myname = new String(“ესეც ასე!”); new** გასაღებური სიტყვა JavaScript -ს აცნობებს, რომ საჭიროა **String** ობიექტის ახალი ეგზემპლარის შექმნა. ეს იქნება სტრიქონული ცვლადი **myname**  სახელით და “ესეც ასე!” მნიშვნელობით.

ამგვარივე წესით ახალი ეგზემპლარები შეიძლება შევქმნათ არა მარტო **String**, **Date, Array** და სხვა ჩაშენებული, არამედ ე.წ. მომხმარებელთა ობიექტებისთვისაც.

გამონაკლისს წარმოადგენს **Math** ჩაშენებული ობიექტი (იხ. ქვემოთ).

ობიექტები ხასიათდება ერთი ან მეტი თვისებით (ატრიბუტით).

თვისება გახლავთ ობიექტში შენახული, რაიმე მნიშვნელობის მქონე ცვლადი.

ჩვენ უკვე ვიცით, როგორ მივმართოთ თვისებას, მაგალითად, მასივის სიგრძეს:

**names.length (names** მასივის სახელია**).**

შევნიშნოთ, რომ თვისება, თავის მხრივ, შეიძლება თვითონაც წარმოადგენდეს ობიექტს. მაგალითად, მასივის თითოეული ობიექტი ამ მასივისთვის წარმოადგენს სპეციალური ტიპის თვისებას, რომელიც აღინიშნება ინდექსით.

ამრიგად, მასივის ელემენტის სიგრძე, ვთქვათ, **names** [7]. **length**, ასე ვთქვათ, თვისების თვისების როლში გვევლინება.

მეთოდი გახლავთ ობიექტის თვისებათა ფუნქცია ანუ ოპერატორების ერთობლიობა, რომელთა შესრულების შედეგი ამავე ობიექტშივე შეინახება როგორც ერთ-ერთი თვისება.

მეთოდის გამოძახება ფუნქციის გამოძახების ანალოგიურად ხდება.

მაგალითად:

value.toUpperCase(); (value სტრიქონული ტიპის ცვლადის მნიშვნელობა მეთოდის შესრულების შემდეგ მხოლოდ პატარა ასოებით იქნება გამოსახული.)

მეთოდის მიერ დაბრუნებული მნიშვნელობა (შეგახსენებთ, მეთოდი ფუნქცია

 გახლავთ! ) შეიძლება რომელიმე ცვლადს მივანიჭოთ:

**finish = Math.round(num)**;

(**Math** ჩაშენებული ობიექტის round მეთოდი ამრგვალებს **num** ცვლადის მნიშვნელობას, რომელიც ენიჭება **finish** ცვლადს.)

აღსანიშნავია, რომ **with** საკვანძო სიტყვის მეშვეობით შესაძლებელია კოდის

გამარტივება:

with (lastname) {

window.alert ( “ “გვარის სიგრძეა: “ + length);

toUpperCase();

}

 (**length** თვისება და **toUpperCase()** მეთოდი კავშირდება **lastname** ობიექტთან.)

**Math** ობიექტი

ზემოთ აღვნიშნეთ, რომ **Math** ჩაშენებული ობიექტისთვის საჭირო არ არის ახალი ეგზემპლარების შექმნა. ამ ობიექტის თვისებები მათემატიკური კონსტანტებია, ხოლო მეთოდები - მათემატიკური ფუნქციები.

გავეცნოთ **Math** ობიექტთან დაკავშირებულ, რამდენიმე ხშირად გამოყენებულ მეთოდს:

**Math.ceil()** - რიცხვი მრგვალდება მეტობით;

**Math.floor()** - რიცხვი მრგვალდება ნაკლებობით;

**Math.round()** - რიცხვი მრგვალდება უახლოეს მთელ რიცხვამდე.

რომელიმე ათობით ნიშნამდე (მაგალითად, მეასედამდე) რიცხვის დასამრგვალებ­ლად შეიძლება ასე მოვიქცეთ:

**function round(num) {**

**return Math.round (num\*100)/100;**

**}**

იგულისხმება, რომ num ცვლადს მინიჭებული აქვს რაიმე მნიშვნელობა. round ფუნქციის გამოძახების შემდეგ აღნიშნული მნიშვნელობა ორ ათობით თანრიგამდე დამრგვალდება.

მსგავს გადაწყვეტას შეიძლება მივმართოთ 1-a დიაპაზონში შემთხვევითი რიცხვის გენერირებისათვის:

function rand(num) {

return Math.floor ( Math .random()\*num) + 1;

}

Math.random() მეთოდით ხდება 0-1 დიაპაზონში შემთხვევითი რიცხვის მიღება, Math.floor() მეთოდით შესაბამისი ნამრავლის ნაკლებობით დამრგვალება. დაბოლოს, rand ფუნქციას უბრუნდება მიღებული სიდიდის ერთით გადიდებული მნიშვნელობა.

დავწეროთ შემთხვევითი რიცხვების გენერატორის პროდუქტის “შემთხვე-

ვითობაზე” შემოწმების კოდი:

**ლისტინგი** 1. ფსევდოშემთხვევითი რიცხვების გენერატორის საიმედოობის დადგენა

<HTML>

<HEAD>

<TITLE>>შემთხვევითი რიცხვების გენერატორი</title>

<STYLE>

H2, P {font-family: LitNusx}

</style>

</head>

<BODY>

<H2> გენერატორის შემოწმება</h2>

<P> რამდენად შემთხვევითია შემთხვევითი რიცხვების გენერატორით

მიღებული რიცხვები? გამოვთვალოთ 100 000 ასეთი რიცხვის საშუალო

მნიშვნელობა. <HR>

<SCRIPT LANGUAGE="JavaScript">

total = 0;

for (i = 0; i < 100000; i++) {

num = Math.random();

total += num;

if (( i/10000-Math.round(i/10000)) ==0)

window.status = "გენერირებულია " + i + " რიცხვი";

}

average = total/100000;

average = Math.round (average\*1000) / 1000;

document.write (“<H2> შემთხვევითი რიცხვების საშუალო

არითმეტიკული ტოლია: “ + average + “</h2>”);

window.status = “ გენერირებულია “ + i + " რიცხვი";

</script> </body>

</html>

**Date** ობიექტი

ჩვენ უკვე გავეცანით ამ ჩაშენებულ ობიექტს. მისი შექმნა სხვა ობიექტების ანა­ლოგიურად ხდება. დასაშვებია ამ დროს ობიექტისთვის მნიშვნელობის

განსაზღვრაც:

**birthday = new Date();**

**birthday = new Date(“Jan 20 2002 11:00:0”);**

**birthday = new Date(5, 26, 2002);**

**birthday = new Date(5, 26, 2002, 11, 0, 0);**

აღსანიშნავია, რომ **Date** ობიექტს არც ერთი თვისება არ ახასიათებს, რის გამოც შექმნილი ობიექტისთვის მნიშვნელობის მისანიჭებლად ან მინიჭებული მნიშვნელობის გასაგებად იყენებენ ქვემომოყვანილ მეთოდებს:

 მინიჭება -

ა)

მინიჭება -

**setDate()** – განსაზღვრავს დღის აღმნიშვნელ რიცხვს;

**setMonth()** – განსაზღვრავს თვეს;

**setFullYear()** – განსაზღვრავს წელიწადს;

**setTime()**  – განსაზღვრავს პერიოდს მილიწამებში 1970 წლის 1 იანვრიდან;

**setHours(), setMinutes** და **setSeconds** დროის შესაბამისი სიდიდეების

განსაზღვრისთვის გამოიყენებიან.

ბ) რაც შეეხება თარიღის ტიპის ობიექტებიდან ინფორმაციის მიღებას, ამ მიზნით გამოიყენება მსგავსი მეთოდები, რომლებშიც **set** თავსართი შეცვლილია **get** -ით.

მაგალითად, უკვე გამოცხადებული **Date** ტიპის მქონე **holiday** ობიექტისთვის წლის მნიშვნელობის განსაზღვრა და იმავე მნიშვნელობის შესახებ ინფორმაციის მიღება მოხდება შემდეგი ოპერატორების მეშვეობით:

**holiday. setFullYear(2002)**;

**holiday. getFullYear()**;

JavaScripte-ში თარიღის ჩაწერის სამი მეთოდი არსებობს:
ISO თარიღი - "2020-11-09" - ეს საერთაშორისო სტანდარტია;

მოკლე თარიღი -"11/09/2020";

გრძელი თარიღი - "Nov 09 2020" ან "09 Nov 2020".

ISO სტანდარტით მუშაობა მისასალმებელია,

მიუხედავად იმისა, თუ რა ფორმით შეგვყავს თარუღის მონაცემები, ჩუმათობის პრინციპით თარიღის მონაცემები გამოდის სრული ტექსტის სახით სტრიქონულ ფორმატში, მაგალითად:

Mon Nov 09 2020 04:00:00 GMT+0400 (საქართველო, სტანდარტული დრო)

ISO სტანდარტულ დროის ფორმატში შეყვანისას შესაძლებელია დროის დამატებაც, ამ შემთხვევაში ასო T ემატება, მაგალითად:

 var d=new Date("2020-11-09T12:00:00Z")

Z აღნიშნავს გრინვიჩის UTC დროს, ღუკი საჭიროა UTC დროის მიმართ ცვლილება, მაშინ Z აღარ იწერება და მის მაგიერ ჩაიწერება +-ით ან - -ით დრო, იმის და მიხედვით გინდა დაასწროთ გრინვიჩის დროს თუ არა, მაგალითად ("2020-11-09T12:00:00-06:30")

**Date**  ობიექტის მნიშვნელობის ჩაწერის ფორმატის შესაცვლელად გამოიყენება შემდეგი ორი მეთოდი:

**Date.parse()** - ეს მეთოდი **Date** ტიპის ობიექტის მნიშვნელობის ტექსტური სახით ჩაწერას ცვლის მილიწამების რიცხვით, რომლებიც ათვლილია 1/1/1970 თარიღიდან.

**Date.UTC() -** ეს მეთოდი **Date** ტიპის ობიექტის მნიშვნელობის ტექსტური სახით ჩაწერას ცვლის მილიწამების რიცხვით, რომლებიც ათვლილია 1/1/1970 თარიღიდან გრინვიჩის დრომდე.

გამვიხილოთ საიტზე დროის ტაიმერის გამოტანის სკრიპტი, რომლის მაგალითზე ჩვენ სხვადასხვა მომენტებს განვმარტავთ.

**ლისტინგი** 2. საიტზე ტაიმერის გამოტანა

 1: <html>

 2: <head>

 3: <style> //სტილი: გვერდის ფონი - ყვითელი, ტექსტი - ლურჯი

 4: body {

 5: background-color: yellow;

 6: color: blue;

 7: }

 8: </style>

 9: <script language = 'javascript'> // სცენარი ჯავასკრიპტში

10: function startTime() { // startTime() ფუნქციის განსაზღვრა

11: var date = new Date(); /\* Date() ობიექტის ახალი date ეგზემპლარის აგება,

12: რომელშიც მიმდინარე თარიღის მნიშვნელობები ჩაიწერება \*/

13: var hours = date.getHours(); // ცვლად hours-ში მიმდინარე საათის ჩაწერა

 14: var minutes = date.getMinutes(); //ცვლად minutes-ში მიმდინარე წუთის ჩაწერა

 15: var = date.getSeconds(); // ცვლად seconds-ში მიმდინარე წამების ჩაწერა

 16: if (hours < 10) hours = "0" + hours;/\* სტრიქონებში 16-18 პირობითი ოპერატორის საშუალებით ერთნიშნა მნიშვნელობების ორნიშნად წარმოდგენა, ცვლადები სტრინგებია\*/

 17: if (minutes < 10) minutes = "0" + minutes;

 18: if (seconds < 10) seconds = "0" + seconds;

 19: document.getElementById("time").innerHTML = hours + ":" + minutes + ":" + seconds; /\*DOM-ის საშუალებით html ელემენტში, კონკრეტულად 24 სტრიქონში მოცემულ გამოყოფილ არეში, რომლის იდენტიფიკატორია time (document.getElementById("time")), ჩაწერა (innerHTML) საათის, წუთის, წამის (hours + ":" + minutes + ":" + seconds)\*/

 20: setTimeout(startTime, 1000); /\* setTimeout() - ეს არის ფუნქციის გამოძახების დაგეგმვა, კერძოდ ამ შემთხვევაში ფუნქცია startTime გამოიძახება 1000 მილიწამის ანუ 1 წამის შემდეგ. გამოიძახება ერთხელ. მაგრამ რადგანაც დამგეგმავი იძახებს იმ ფუნქციას რომელშიც თვითონ არის (რეკურსია) მაშინ ეს დამგეგმავი სულ იმუშავებს - ხდება ჩაციკვლა \*/

 21: }

22: </script>

23: </head>

23: <body onLoad = 'startTime()' > /\* Load მოვლენის (გვერდის ჩატვირთვის) დამმამუშავებელი onLoad იძახებს ფუნქციას 'startTime()'-ს, ანუ დრო გამოჩნდება მაშინვე, როგორც კი ჩაიტვირთება გვერდი\*/

24: <b>დრო საიტზე: <span id="time"></span></b>

25: </body>

26: </html>

ჩვენ შეგვიძლია თავი ავარიდოთ ამგვარ რეკურსიას, თუკი გამოვიყენებთ მეორე დამგეგმავს - setInterval(startTime, 1000), რომელიც startTime ფუნქციას გამოიძახებს ყოველ 1 წამში.

**ლისტინგი** 3. ტაიმერის გამოტანის სხვა მეთოდი

<html>

<head>

<style>

body {

 background-color: yellow;

 color: blue;

}

</style>

<script language = 'javascript'>

 function startTime() {

 var date = new Date();

 var hours = date.getHours();

 var minutes = date.getMinutes();

 var seconds = date.getSeconds();

 if (hours < 10) hours = "0" + hours;

 if (minutes < 10) minutes = "0" + minutes;

 if (seconds < 10) seconds = "0" + seconds;

 document.getElementById("time").innerHTML = hours + ":" + minutes + ":" + seconds;

 }

</script>

</head>

<body>

<script language = 'javascript'>

setInterval(startTime, 1000);

</script>

 <b>დრო საიტზე: <span id="time"></span></b>

</body>

</html>

ლისტინგი 3-ის გაშვებისას ვნახავთ, რომ დროის ათვლა იწყება გარკვეული ყოვნით, რაც გამოწვეულია იმით, რომ ჩვენ ამ შემთხვევაში გავაუქმეთ გვერდის ჩატვირთვის მოვლენაზე დამუშავება.

შემდეგი ლისტინგი გვაძლევს საშუალება გავიგოთ კვირის რომელი დღეა ესა თუ ის თარიღი:

ლისტინგი 4. კვირის დღის განსაზღვრა

<html>

 <head>

<script>

p=prompt("შეიყვანეთ წელი, თვე, დღე ფორმატით: წელი-თვე-რიცხვი");

var date1 = new Date(p);

var year = date1.getFullYear();

var month = date1.getMonth() + 1;

var day = date1.getDate();

var dgea=0;

 var dge = date1.getDay();

 if (day < 10) day = "0" + day;

 switch(dge)

 {

 case 0: dgea = "კვირა"; break;

 case 1: dgea = "ორშაბათი"; break;

 case 2: dgea = "სამშაბათი"; break;

 case 3: dgea = "ოთხშაბათი"; break;

 case 4: dgea = "ხუთშაბათი"; break;

 case 5: dgea = "პარასკევი"; break;

 case 6: dgea = "შაბათი"; break;

 }

 if (month < 10) month = "0" + month;

</script>

<script language = 'javascript'>

 function startTime()

 {

 var date=new Date();

 var hours = date.getHours();

 var minutes = date.getMinutes();

 var seconds = date.getSeconds();

 if (hours < 10) hours = "0" + hours;

 if (minutes < 10) minutes = "0" + minutes;

 if (seconds < 10) seconds = "0" + seconds;

 document.getElementById("year1").innerHTML = year + "/" + month + "/" + day;

 document.getElementById("day1").innerHTML = "ეს იყო კვირის შემდეგი დღე - " + dgea;

 document.getElementById("time").innerHTML = "ახლა არის - " + hours + ":" + minutes + ":"+seconds;

 setTimeout(startTime, 1000);

 }

</script>

</head>

<body onLoad = 'startTime()' bgcolor=yellow>

 <b>შეყვანილია თარიღი: <span id="year1"></span></b><br><br>

<div id="day1"> </div><br><br>

<div id="time"></div>

</body>

</html>

 **ბრაუზერის ობიექტების მოდელთან მუშაობა**

ბრაუზერის ობიექტების მართვა **JavaScript**-ის დიდი ღირსება გახლავთ. ეს ობიექტები შემდეგი იერარქიული სტრუქტურის სახით შეიძლება წარმოვადგინოთ:

**window**

**location**

**history**

**document**

elements [ ]

forms [ ]

images [ ]

anchors [ ]

link [ ]

შევნიშნოთ, ჩვენ განვიხილავთ დოკუმენტის ე.წ. ნულოვანი დონის ობიექტურ მოდელს. ინტერნეტისთვის სტანდარტების შემმუშავებელ **W3C** კონსორციუმს (**W3C** – **World Wide Web Consortium**) დამტკიცებული აქვს **DOM1** და **DOM2** დონეთა შესატყვისი სტანდარტებიც.

ზემოთ, ნახაზზე ასახულია ბროუზერის მხოლოდ უმნიშვნელოვანესი ობიექტები. იერარქიული სტრუქტურის სათავეში იმყოფება **window**-ობიექტი. ჩვენ უკვე

ვიცნობთ მის ზოგიერთ თვისებასა და მეთოდს. ესენია:

**window.status** თვისება

და **window.alert(), window.confirm(), window.prompt()** მეთოდები.

დასაშვებია, ბროუზერს ერთდროულად გავახსნევინოთ რამდენიმე ფანჯარა.

აღსანიშნავია, რომ ფრეიმებიც და შრეებიც **window** - ობიექტის სახესხვაობებს

წარმოადგენს.

გადავიდეთ **document** -ობიექტზე. მისი სხვა სახელწოდებებია **Web**-დოკუმენტი და **Web** -ფურცელი.

დავუშვათ, ეკრანზე ერთდროულად გამოგვყავს რამდენიმე ფანჯარა და გვსურს, რომელიმე მათგანისთვის გამოვიყენოთ ჩვენთვის უკვე ცნობილი **document.write()** მეთოდი. ცხადია, ასეთ შემთხვევაში აუცილებელი გახდება ფანჯრის სახელის დაკონკრეტება - **window.document.write()**. აღვნიშნოთ, რომ მსგავსი როლი აკისრია **document.writeln()** მეთოდსაც, მხოლოდ მისი შესრულების შემდეგ ეკრანზე ინფორმაცია ახალი სტრიქონიდან გამოდის.

გავეცნოთ დოკუმენტის სხვა მნიშვნელოვან თვისებებსაც:

**URL** - გვიჩვენებს ბრაუზერში ჩატვირთული მიმდინარე **Web**-ფურცლის მისამართს. ცხადია, ჩვენ მიერ ამ თვისების შეცვლა არ შეიძლება, მაგრამ თუკი საჭიროა ბრაუზერში სხვა ფურცლის გამოყვანა, ვიყენებთ **window.location obieqts** (იხ. ქვემოთ).

**title** - შეიცავს მიმდინარე **Web** -ფურცლის სათაურს - **<TITLE> </title>** დესკრიპტორული წყვილის შიგნით.

**referrer** - გვიჩვენებს წინა ფურცლის მისამართს, რომელზეც, როგორც წესი, ფიგურირებს მიმდინარე ფურცელზე დაყრდნობა (ჰიპერკავშირი).

**lastModified** - შეიცავს **Web** -ფურცლის ბოლო კორექტირების თარიღს. ეს მონაცემი ინახება სერვერზე და ბრაუზერს გადაეგზავნება **Web**-ფურცელთან ერთად (აქვე შევნიშნოთ, რომ ზოგი სერვერი ამ სერვისს არ უზრუნველყოფს). ერთ-ერთ პროგრამაში ჩვენ გავეცანით, თუ როგორ შეიძლება **Web**-ფურცლის ბოლო კორექტირების თარიღის ეკრანზე გამოყვანა.

**document** - ობიექტში, window-ისაგან განსხვავებით, open და close მეთოდები დოკუმენტის თუ ფანჯრის გაღება-დახურვას კი არ ემსახურება, არამედ **open** -ით ხდება დოკუმენტის გასუფთავება და მისი მომზადება **write (writeln)** მეთოდებით ტექსტური ნაკადის ეკრანზე გამოყვანისთვის. რეალურად კი ეს პროცესი მაშინ იწყება, როცა **JavaScript** -ინტერპრეტატორი კოდში close მეთოდს იპოვის.

**document.open** ბრძანებაში შესაძლებელია ეკრანზე გამოსაყვანი მონაცემების ტიპის განსაზღვრაც. ქსელში გადასაგზავნი დოკუმენტი უმეტესწილად **HTML** -ტიპისაა

(შეესატყვისება **text/html** ტიპს).

**დაყრდნობები, ანკერები**

**document** - ობიექტს შეიძლება ჰქონდეს შვილობილი ობიექტებიც. განვიხილოთ ორი მათგანი: **anchor** და **link**.

ანკერი **HTML** -დოკუმენტის იმ ადგილს მონიშნავს, რომელზეც შესაძლებელია დაყრდნობის (**link)**  მეშვეობით გადავიდეთ. მოვიყვანოთ ანკერ-ობიექტის

მაგალითი:

**<A NAME =” top”>**

ამ ანკერზე გადასვლა კი განხორციელდება შემდეგი დაყრდნობა - ობიექტის მეშვეობით:

**<A HREF=”#top”>**

 დაყრდნობა-ობიექტით გადასვლა შესაძლებელია არა მარტო მიმდინარე **Web**-ფურცლის ფარგლებში, არამედ სხვა ფურცლის ნებისმიერ უბანზეც. რადგანაც დაყრდნობები დოკუმენტში საკმაოდ ბევრი შეიძლება იყოს, მათი მართვისთვის მიზანშეწონილია ლინკს მასივის გამოყენება. ასევე, ანკერების სამართავად განკუთვნილია **anchors** მასივი.

შესაბამისი თვისებით შეიძლება მივიღოთ ინფორმაცია ამ მასივების სიგრძის

შესახებ:

**document.links.length da document.anchors.length.**

ლინკს მასივის თითოეული ელემენტი ხასიათდება შემდეგი თვისებებით:

დაყრდნობის ნომერი, სახელწოდება, **Web** -ფურცლის მისამართი. თუ გვაინტერე­სებს, რომელი ფურცლის მისამართი ფიგურირებს ლინკს მასივის, ვთქვათ, პირველ ელემენტში, რომელიმე ცვლადს ამგვარად მივანიჭებთ შესაბამის მნიშვნელობას:

**link1 = links[0].href**

**location ობიექტი**

**window** -ობიექტის შვილობილია **location -** ობიექტიც. ახალი **Web** -ფურცლის

ჩატვირთვა მხოლოდ ამ ობიექტის **href** თვისების გამოყენებით ხორციელდება.

მაგალითად:

**window.location.href = “http://home.netscape.com”;**

თუ საჭიროა, შეგვიძლია **URL -** მისამართის ნაწილის შესახებაც მივიღოთ

ინფორმაცია. მაგალითად, მისამართის პროტოკოლური ნაწილი (უმეტეს შემთხ-

ვევაში ეს გახლავთ **http: )**

ინახება **location.protocol** თვისებაში**.**

მართალია, **location.href** თვისება იმავე **URL -** მისამართს შეიცავს, რომელიც

არის **document.URL .** თვისების მნიშვნელობა, მაგრამ, რადგანაც უკანასკნელის

შეცვლა არ შეიძლება, ახალი **Web -** ფურცლის გამოსაძახებლად მიმართავენ

მხოლოდ **location -** ობიექტს**.**

**location.reload** მეთოდით დოკუმენტი ბროუზერში ხელახლა ჩაიტვირთება.

**history ობიექტი**

**history** -ობიექტიც **window** -ობიექტის შვილობილების რიცხვში შედის. იგი

იმ მასივის სახით ინახება, რომლის თითოეული ელემენტი შეიცავს უკვე

ნანახი **Web** -ფურცლების **URL** -მისამართს. მიმდინარე ფურცლისთვის

გათვალისწინებულია მასივის პირველი ელემენტი **- history[0].**

**history** -ობიექტი 4 თვისებით ხასიათდება:

**history.length –** ერთი სეანსის მანძილზე ჩათვალიერებული ფურცლების რიცხვი;

**history.current** – მიმდინარე ფურცლის **URL** -მისამართი;

**history.next –** ფურცლის **URL** -მისამართი, რომელზეც მოვხვდებით ბრაუზერის ღილაკების პანელში მყოფ **Forward** ღილაკზე დაწკაპუნების შემდეგ;

**history.previous** – შესაბამის ფურცელზე გადავალთ, თუ **Back** ღილაკზე დავაწკაპუნებთ.

რაც შეეხება მისამართების მასივში ნებისმიერი სასურველი ფურცლის

მისამართის არჩევასა და გადასვლის განხორციელებას, ეს მიიღწევა

**history.go(n) და history.go(-n)** მეთოდების გამოყენებით. მაგალითად,შესაძლებელია ეკრანზე გამოვიყვანოთ ნახატები - ერთ მათგანზე ასახული იქნება მარჯვნივ, ხოლო მეორეზე - მარცხნივ მიმართული ისრები. დავწეროთ კოდი, რომლითაც ამ ისრებზე დაწკაპუნებისას **history.go(+1) და history.go(-1)**

მეთოდებით მიიღწევა **Forward** და **Back** ღილაკების გამოყენების ეფექტი:

**<HTML>**

**<HEAD>**

**<TITLE>Back და Forward ღილაკების ანალოგების შექმნა </title>**

**<STYLE>**

**H2, P {font-family: LitNusx}**

**</style>**

**</head>**

**<BODY>**

**<H2> Back და Forward ღილაკების ანალოგები</h2>**

**<HR>**

**<P>** **მოვინახულოთ უკვე უკვე ნანახი ფურცლები!**

**<HR>**

**<A HREF="javascript:history.go(-1);">**

 **<IMG border=0 src="left.gif">**

**</a>**

**<A HREF="javascript:history.go(1);">**

 **<IMG border=0 src="right.gif">**

**</a>**

**<HR>**

**</script>**

**</body>**

**</html>**

მივაქციოთ ყურადღება - დაყრდნობებში გამოყენებულია **javascript:URL** და

**history.go** მეთოდების კომბინაცია. ამრიგად, შეიძლება **JavaScript**-ის

ოპერატორებს **<SCRIPT> </script>** წყვილის გარეშეც მივმართოთ. ნახატების

ასახვა კი ხდება **HTML** ენის საშუალებებით.