საქართველოს ტექნიკური უნივერსიტეტი

გელა ღვინეფაძე, ნინო ჩორხაული

Web-ტექნოლოგიები

(W3.JS, Vue.JS, Kotlin)

დამტკიცებულია:

სტუ-ს „IT კონსალტინგის სამეცნიერო ცენტრის“ სარედაქციო კოლეგიის მიერ - ოქმი N1 2.02.24

თბილისი - 2024

უაკ 004.5

განხილულია Web–დოკუმენტებში სცენარების მაფორმირებელი Javascript ენის ბაზაზე შემუშავებული W3.JS და Vue.JS თანამედროვე ტექნოლოგიები, აგრეთვე, მობილურ აპარატებში ფუნქციონირებადი Android ოპერაციული სისტემისთვის აპლიკაციების შექმნისათვის განკუთვნილი Kotlin დაპროგრამების ენის შესაძლებლობები. დამხმარე სახელმძღვანელო განკუთვნილია ინფორმატიკის სპეციალობის შემსწავლელი სტუდენტებისა და ამ საკითხით დაინტერესებული პირებისთვის.

რეცენზენტები:

- პროფ. თ. სუხიაშვილი – საქართველოს ტექნიკური უნივერსიტეტი,
- პროფ. თ. თოდუა – ბიზნესისა და ტექნოლოგიების უნივერსიტეტი

რედკოლეგია:

ა. ფრანგიშვილი (თავმჯდომარე), ზ. აზმაიფარაშვილი, მ. ახობაძე, გ. გოგიჩაიშვილი, ზ. ბოსიკაშვილი, ე. თურქია, რ. კაკუბავა, ვ. კვარაცხელია, თ. ლომინაძე, ნ. ლომინაძე, ჰ. მელაძე, ლ. პეტრიაშვილი, გ. სურგულაძე. ბ. შანშიაშვილი, ო. შონია, ზ. წვერაიძე

ISBN 978-9941-8-6745-3

სარჩევი

თავი 1. W3.JS

1.1. შესავალი

W3.JS არის Javascript-ის ბაზაზე შექმნილი ბიბლიოთეკა, უფასოდ გავრცელებადი პროგრამული პროდუქტი, რომელიც აადვილებს და აჩქარებს ვებდანართების შექმნას როგორც პერსონალური კომპიუტერების, ასევე - პლანშეტებისა და მობილური მოწყობილობებისათვის.

W3.JS-ის მეშვეობით შესაძლებელია:

- HTML-ელემენტების დამალვა და ხელახლა ასახვა;
- ელემენტებისათვის სტილების დამატება და ამოგდება;
- ოპერაციების ჩატარება კლასებზე დაყრდნობით;
- ელემენტებისათვის კლასისა და CSS-ის დამატება-ამოგდება;
- HTML-ელემენტების შემცველობის ფილტრაცია;
- მონაცემების დახარისხება;
- სლაიდ-შოუს გაშვება;
- კოდში HTML-ფაილების და ელემენტების იმპორტირება;
- W3.JS-ის დახმარებით მონიტორზე html-მონაცემებისათვის ადგილის დარეზერვირება;
- myCallback() ფუნქციის ჩართვით არასასურველი ქმედებების პრევენცია;
- ცხრილებზე მანიპულაციების ჩატარება.

HTML-ის ნებისმიერი W3.JS პროდუქტით სარგებლობისათვის ვიყენებთ შემდეგ ინსტრუქციას:

```
<script src="https://www.w3schools.com/lib/w3.js">
</script>
```

მეორე გზაა w3.js ფაილის ჩამოტვირთვა და მასზე დაყრდნობა:

```
<script src="w3.js"></script>
```

შენიშვნა: *აქ იგულისხმება, რომ ინტერნეტიდან ჩამოტვირთულ w3.js ფაილი იმავე საქაღალდეში გვაქვს განთავსებული, რომელშიც ვინახავთ მის გამომძახებელ ფაილს.*

W3.JS-ის მეშვეობით ვირჩევთ ელემენტებს (ანუ ვახდენთ სელექციას) და მათზე ვასრულებთ ამა თუ იმ მოქმედებას:

w3.*action*(*selector*)

მოვიყვანოთ HTML-ელემენტის სელექტორით მოძიების და მასზე ქმედების განხორციელების (დამალვის) მაგალითი:

w3.hide('p') // დაიმალოს ყველა <p> ელემენტი.

სელექცია ხორციელდება იმ სელექტორების და იმავე სინტაქსის მეშვეობით, რომელთაც გავეცანით CSS შესაძლებლობების შესწავლისას.

სელექციისათვის გამოიყენება დასახელებანი:

- HTML-ელემენტების ტეგის;
- იდენტიფიკატორის;
- კლასის;
- ტიპის;
- ატრიბუტის და სხვ.

მომდევნო პარაგრაფებში განვიხილავთ ზემოთ მოყვანილ სიაში ჩამოთვლილი თითოეული პუნქტისთვის შესაბამის მაგალითებს.

1.2. HTML-ის ელემენტების დამალვა და ხელახლა ასახვა

1) **ყველა p ელემენტის დამალვა**

```
<!DOCTYPE html>
<html>
```

```
<title>W3.JS</title>
<script
src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<center>

<h3> html-ელემენტის (ელემენტების) ამორჩევისა და მათზე მანიპულაციების ჩასატარებლად w3.js იყენებს css-სინტაქსს.
</h3>
<h3>ზოგადი სახით მიდგომა ასეთი სახისაა: </h3>
<h3><i>w3.აქციის_სახელწოდება
(სელექტორი)</i></h3>
<br/>

<h3>ჯერ გავეცნოთ html-ელემენტის (ელემენტების) ამორჩევისა და დამალვის ხერხებს ამ მაგალითზე.
</h3>
<button onclick="w3.hide('p')"><h3>დამალე ყველა p
ელემენტი!</h3></button>
</center>
<h2>ლონდონი</h2>
<p>London is the capital city of England.</p>
<p>ლონდონი ინგლისის დედაქალაქია.</p>
<p>It is the most populous city in the United Kingdom.</p>
<p>ის არის გაერთიანებული სამეფოს ყველაზე
მჭიდროდ დასახლებული ქალაქი.</p>
<p>Population over 13 million inhabitants.</p>
<p>მისი მოსახლეობა 13 მილიონს აჭარბებს.</p>

</body>
</html>
```

მოგვყავს სხვა სახელის მქონე ელემენტის დამალვის მაგალითიც (ამჯერად h2 ელემენტის):

2) ყველა h2 ელემენტის დამალვა

```
<!DOCTYPE html>
<html>
  <title>W3.JS</title>
  <script src="https://www.w3schools.com/lib/w3.js"></script>
<body>

<center>
 <h3>დააწკაპუნეთ ქვემოთ ყველა "h2" ელემენტის დასამალად!</h3>
 <button onclick="w3.hide('h2')"><h3>C l i c k !</h3></button>
</center>

<div id="Tbilisi" class="city">
  <h2>თბილისი</h2>
  <h4>თბილისი საქართველოს დედაქალაქია.</h4>
</div>
<div id="Paris" class="city">
  <h2>პარიზი</h2>
  <h4>პარიზი საფრანგეთის დედაქალაქია.</h4>
</div>
<div id="Tokyo" class="city">
  <h2>ტოკიო</h2>
  <h4>ტოკიო იაპონიის დედაქალაქია.</h4>
</div>
</body>
</html>
```

3) იმ ელემენტის დამალვა, რომლის id="London"

ამ მიზნის მისაღწევად hide() მეთოდში პარამეტრად მითითებულ იდენტიფიკატორს წინ უნდა უძღვოდეს (#) ხეშ-ტეგი (იხ. კოდი).

```
<!DOCTYPE html>
<html>
  <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>

<body>
 <h3 id="London" align='center'> ამ მაგალითში ვმალავთ სხვადასხვა სახის ყველა იმ ელემენტს, რომლებისთვისაც id ატრიბუტის მნიშვნელობა არის London.</h3>

 <button onclick = "w3.hide ('#London')"><h3 align='center'>Hide London</h3></button>
   <!-- მიაქციეთ ყურადღება, აუცილებელია ამ სინტაქსის დაცვა - იდენტიფიკატორის წინ # სიმბოლოს წამძღვარება, -->
   <!-- ასევე მას, რომ ცენტრირება button-ის შიგნით მისი მნიშვნელობისთვის არ მუშაობს! -->

 <p id="London"> ქვემოთ გაეცნობით ინფორმაციას ლონდონის შესახებ!</p>

 <div id="London">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
 </div>
```

```
<div id="Paris">
 <h2>პარიზი</h2>
 <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>

<div id="Tokyo">
 <h2>ტოკიო</h2>
 <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>

</body>
</html>
```

4) ყველა იმ ელემენტის დამალვა, რომლის id="London"

```
<!DOCTYPE html>
<html>
  <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>

<body>
 <h3 id="London" align='center'> ამ მაგალითში ვმალავთ
სხვადასხვა სახის ყველა იმ ელემენტს, რომლებისთვისაც id
ატრიბუტის მნიშვნელობა არის London.
</h3>

   <button id="London"  onclick = "w3.hide ('#London')"><h3
align='center'>Hide London</h3></button>  <!-- განსხვავება
ისაა, რომ აქ  id="London" გამო წაიშლება ეს ელემენტიც! -->
```

```
<p id="London"> ქვემოთ გაეცნობით ინფორმაციას
ლონდონის შესახებ!</p>

<div id="London">
 <h2>ლონდონი</h2>
 <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>

<div id="Paris">
 <h2>პარიზი</h2>
 <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>

<div id="Tokyo">
 <h2>ტოკიო</h2>
 <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body>
</html>
```

5) კლასის სახელის მიხედვით ელემენტების ამორჩევა-დამალვა

ამ მიზნის მისაღწევად hide() მეთოდში პარამეტრად მითითებულ კლასს წინ უნდა უძღვოდეს წერტილის სიმბოლო:

```
<!DOCTYPE html>
<html>
  <title>W3.JS</title>
  <script src="https://www.w3schools.com/lib/w3.js"></script>
```

```
<body>
  <center>
  <button class="city" onclick="w3.hide ('.city')">
<!--კლასით იდენტიფიცირებისას აუცილებელია ასეთი
სინტაქსის დაცვა - სახელისთვის წინ წერტილის
წამძღვარება! -->

     <h3>ვმალავთ (სხვადასხვა სახის) ყველა იმ ელემენტს,
რომლებისთვისაც class="city"</h3>
  </button>
 </center>

 <div id="Georgia" class="country">
   <h2>საქართველო</h2>
   <p>თბილისი საქართველოს დედაქალაქია.</p>
 </div>

 <div id="London" class="city">
   <h2>ლონდონი</h2>
   <p>ლონდონი ინგლისის დედაქალაქია.</p>
 </div>

 <div id="Paris" class="city">
   <h2>პარიზი</h2>
   <p>პარიზი საფრანგეთის დედაქალაქია.</p>
 </div>

 <div id="Tokyo" class="city">
```

```
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
 </div>
</body>
</html>
```

1.3. სელექტორების სხვა მაგალითები

W3.JS-ს შეუძლია ითანამშრომლოს CSS-თან, რათა მიღებული იქნას მეტი ვიზუალური ეფექტი. ქვემოთ მოყვანილ ორ ფაილში გავეცნობით W3.JS-ის მიერ მოწოდებული ზოგიერთ დამატებითს შესაძლებლობას CSS-სერვისით სარგებლობის მაგალითზე:

მაგალითი_1:

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
 <link rel="stylesheet"
href="https://www.w3schools.com/w3css/4/w3.css"> <!-- დ ა ე მ ა ტ ა ეს ლინკი ! -->
 <script src="https://www.w3schools.com/lib/w3.js"></script>
 <body class="w3-container">
 <h3 align='center'> ამ მაგალითში ვეცნობით W3.JS-ის მიერ მოწოდებული CSS-სერვისის ზოგიერთ დ ა მ ა ტ ე ბ ი თ ს შესაძლებლობას.</h3>
 <p>
  <button class="w3-button w3-green" onclick =
     "w3.hide('#London')"> დამალე ლონდონთან
     დაკავშირებული ინფორმაცია!
 </button>
 </p>
```

```
<div id="London" class="w3-panel w3-red city">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>

<div id="Paris" class="w3-panel w3-red city">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>

<div id="Tokyo" class="w3-panel w3-red city">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>

</body>
</html>
```

მაგალითი 2:

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
<link rel="stylesheet"
href="https://www.w3schools.com/w3css/4/w3.css">
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body class="w3-container">
```

```
<h3 align='center'> ამ მაგალითში ვეცნობით W3.JS-ის მიერ მოწოდებული CSS-სერვისის ზოგიერთ დ ა მ ა ტ ე ბ ი თ ს შესაძლებლობას.</h3>
<p align="center">
<button class="w3-button w3-blue-grey" onclick="w3.hide('h2')"><h3>დამალე h2-ელემენტები!</h3></button>
</p>
<div id="London" class="w3-panel w3-blue-grey city">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris" class="w3-panel w3-blue-grey city">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo" class="w3-panel w3-blue-grey city">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body>
</html>
```

ამჯერად, წინა მაგალითებზე დაყრდნობით, ვაჩვენოთ, თუ როგორ ხდება ელემენტების **არა მარტო დამალვა,** არამედ მონიტორზე მათი **ხელახლა ასახვაც:**

მაგალითი 1:

გავეცნოთ კოდს, თუ როგორ შეიძლება მოხდეს ყველა p ელემენტის დამალვა-გამოტანა:

```
<!DOCTYPE html>
```

```
<html>
  <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
 <center>
 <h2 align='center'>W3.JS</h2>
  <button onclick="w3.hide('h4')"><h3>დამალე  ყველა h4 ელემენტი!</h3></button>
  <button onclick="w3.show('h4')"><h3>აჩვენე ყველა h4 ელემენტი!</h3></button>
  </center>
 <h3>ლონდონი</h3>
 <h4>ლონდონი ინგლისის დედაქალაქია.</h4>
 <h4>მისი მოსახლეობა 13 მილიონს აჭარბებს.</h4>
</body>
</html>
```

მაგალითი 2:

ამ მაგალითში კი ყველა <p>-ელემენტის დამალვა-ასახვა მოხდება w3.toggleShow გადართვის მეთოდის მეშვეობით:

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
 <script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<center>
 <h2 align='center'>W3.JS</h2>
 <h3 align='center'>ამ ვარიანტში <b> .toggleShow </b>
                მეთოდის მეშვეობით ერთი ღილაკი
```

```
                    ითავსებს ელემენტების დამალვა-გამოყვანის
                    ფუნქციებს<br/> (აქ p ელემენტებისთვის).
</h3>
 <button onclick="w3.toggleShow('p')"><h2>გადართვა
                    დამალვა/ჩვენება</h2></button>
</center>
 <div id="London">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
 </div>
<div id="Paris">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
 </div>
 <div id="Tokyo">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
 </div>
<div id="London">
  <h2>ლონდონი</h2>
  <p>კიდევ ერთი-ორი სიტყვა ლონდონის შესახებ</p>
 </div>
</body>
</html>
```

დავალება: *მოახდინეთ ქვემოთ მოყვანილი მაგალითების მოდიფიცირება, ზემოთ წაჩვენებ მიდგომაზე დაყრდნობით.*

დ1) ყველა h2 ელემენტის დამალვა-გამოტანა

```
<!DOCTYPE html>
<html>
    <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<p>ვაწკაპუნებთ "Hide h2"-ღილაკზე ყველა h2-ელემენტის დასამალად.</p>
<p>ვაწკაპუნებთ "Show h2"-ღილაკზე ყველა h2-ელემენტის გამოსატანად</p>
<p>
<button onclick="w3.hide('h2')">Hide h2</button>
<button onclick="w3.show('h2')"> show h2</button>
</p>
<div id="London">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo">
  <h1>ტოკიო</h1>
  <p>ტოკიო იაპონიის დედაქალაქია</p>
</div>
</body></html>
```

დ2) იმ ელემენტის დამალვა-გამოტანა, რომლის id="London"

ამ მიზნის მისაღწევად შესაბამის მეთოდებში პარამეტრად მითითებულ იდენტიფიკატორს წინ უნდა უძღვოდეს (#) ხეშ-ტეგი.

```
<!DOCTYPE html>
<html>
  <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<p>ვაწკაპუნებთ შესაბამის ღილაკზე id="London" მქონე
ელემენტის დასამალავად (გამოსატანად).</p>
<button onclick = "w3.hide ('#London')">Hide London</button>
<button onclick = "w3.show ('#London')">Show
London</button>
<div id="London">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body>
</html>
```

დ3) კლასის სახელის მიხედვით ელემენტების ამორჩევა და დამალვა-გამოტანა

ამ მიზნის მისაღწევად hide() მეთოდში პარამეტრად მითითებულ კლასს წინ უნდა უძღვოდეს წერტილის სიმბოლო (იხ. კოდი).

```
<!DOCTYPE html>
<html>
  <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<p>ქვემოთ შესაბამის ღილაკზე დაწკაპუნებით ვმალავთ / გამოვაჩენთ  class="city"-ის ყველა ელემენტს.</p>
<button onclick="w3.hide ('.city')">დ ა ვ მ ა ლ ო თ !</button>
<button onclick="w3.show ('.city')">ა ვ ს ა ხ ო თ ! </button>
<div id="London" class="city">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris" class="city">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo" class="city">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
<div id="Georgia" class="country">
  <h2>ქვეყანა</h2>
  <p>თბილისი საქართველოს დედაქალაქია.</p>
```

```
</div>
</body>
</html>
```

სელექტორების მაგალითები

სელექტორი	ამორჩევა (მონიშვნა)
("*")	დოკუმენტში ყველა ელემენტის
(this)	მიმდინარე HTML ელემენტის
("p.intro")	class="intro"-ში ყველა <p> ელემენტის
("div p")	ყველა ელემენტში ყველა <p> ელემენტის
("div p:first-child")	ყველა ელემენტში პირველი <p> ელემენტის
("[href]")	href ატრიბუტის მქონე ყველა ელემენტის
("a[target=_blank]")	ყველა <a> ელემენტის ატრიბუტების "_blank" მნ-ბით
("p:nth-child(even)")	ლუწი ნომრის ყველა <p> ელემენტის

(სელექტორების სრული სია იხ. https://w3schoolsrus.github.io/cssref/css_selectors.html#gsc.tab=0)

1.4. HTML ელემენტებისათვის სტილების დამატება და ამოგდება

W3.JS-ის მეშვეობით შესაძლებელი არის ელემენტებისათვის სტილების დამატება და ამოგდებაც.

ჯერ გავეცნოთ სტილების დამატების შესაძლებლობას:

1) იდენტიფიკატორის მიხედვით ამორჩეული ელემენტისათვის ფონად წითელი ფერის დაყენება სტილის დამატებით

```
<!DOCTYPE html>
<html>
    <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<p>ღილაკზე დაწკაპუნებით id="London"-ის
    ელემენტისათვის წითელი ფონის დამატება.</p>
<p>
<button onclick="w3.addStyle('#London','background-
color','red')">
  <h2>დაამატე სტილი!</h2>
</button>
</p>
<div id="London" class="city">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris" class="city">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo" class="city">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body></html>
```

2) ყველა h2 ელემენტისათვის სტილის დამატებით ფონად წითელი ფერის შერჩევა

```
<!DOCTYPE html>
<html>
    <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<p>ყველა h2 ელემენტს ფონად განვუსაზღვრავთ წითელ ფერს.</p>
<button onclick="w3.addStyle('h2','color','red')">დაამატე სტილი!</button>
<div id="London" class="city">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris" class="city">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo" class="city">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body>
</html>
```

შემდეგ, დიდი მოცულობის ფაილებში ელემენტებისათვის სტილების დამატება-ამოგდების ოპერაციები, როგორც წესი, მნიშვნელოვნად მარტივდება კლასებზე დაყრდნობით.

ქვემოთ განვიხილავთ შესაბამის მაგალითებს:

3) იდენტიფიკატორის მიხედვით ამორჩეული ელემენტისათვის კლასის დამატებით ფონის ფერის განსაზღვრა

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
<style>
.marked {
   background-color: orange;
   font-size: 30px;
   padding: 16px;
   color: white;
}
</style>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<p>ღილაკზე დაწკაპუნებით id="London"-ის მქონე
      ელემენტს ვუმატებთ "marked"-კლასს.
</p>
<p>
<button onclick="w3.addClass('#London','marked')">კლასის
დამატება!
</button>
```

```
</p>
<div id="London" class="city">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris" class="city">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo" class="city">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქაია.</p>
</div>
</body>
</html>
```

4) მოცემული კლასის ყველა ელემენტისათვის კლასის დამატებით სხვადასხვა პარამეტრის განსაზღვრა

```
<!DOCTYPE html>
<html>
    <title>W3.JS</title>
<style>
.marked {
   background-color: orange;
   font-size: 20px;
   padding: 16px;
   color: white;
}
</style>
```

```
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<p>ღილაკზე დაწკაპუნებით class='city'-ის ელემენტებს
    ვუმატებთ "marked"-კლასს.
</p>
<p>
<button onclick="w3.addClass('.city','marked')">Add Class
</button>
</p>
<div id="London" class="city">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<br>
<div id="Paris" class="city">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<br>
<div id="Tokyo" class="city_1">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body>
</html>
```

5) მოცემულ იდენტიფიკატორიანი ელემენტისათვის რამდენიმე კლასის დამატება (კლასები სიაში ერთმანეთისგან უნდა განცალკევდნენ შუალედით)

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
<style>
.class1 {
   background-color: orange;
   font-size: 30px;
   color: white;
}
.class2 {
   padding: 20px;
   border: 5px solid red;
}
</style>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<p>class='class1 class2' კლასების დამატება id='London'-ის
    მქონე ელემენტისათვის.
</p>
<button onclick="w3.addClass('#London','class1 class2')">
   კლასების დამატება!
</button>
<div id="London" class="city">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
```

```
</div>
<div id="Paris" class="city">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo" class="city">
  <h2>ტოკიოTokyo</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body>
</html>
```

6) მოცემულ იდენტიფიკატორიანი ელემენტისათვის კლასის ამოგდება

```
<!DOCTYPE html>
<html>
      <title>W3.JS</title>
<style>
.marked {
   background-color: orange;
   padding: 16px;
   font-size: 30px;
   color: white;
}
</style>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
```

```
<p>  "marked"  კლასს  ვანადგურებთ  id="London"
      ელემენტისათვის.
</p>
<p>
   <button onclick="w3.removeClass('#London','marked')">
     ამოაგდე კლასი!
   </button>
</p>
<div id="London" class="marked">
 <h2>ლონდონი</h2>
 <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris">
 <h2>პარიზი</h2>
 <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo">
 <h2>ტოკიო</h2>
 <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body>
</html>
```

7) ყველა h2 ელემენტისათვის კლასის ამოგდება

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
<style>
.marked {
```

```
    color: red;
    font-size: 30px;
}
</style>
    <script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<p>ვახდენთ "marked" კლასის მოხსნას ყველა h2-ელემენტიდან.</p>
    <button onclick="w3.removeClass ('h2','marked')">მოხსენი კლასი!
</button>
<div id="London">
  <h2 class="marked">ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris">
  <h2 class="marked">პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo">
  <h2 class="marked">ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body>
</html>
```

8) მოცემული კლასის ყველა ელემენტისათვის კლასის ამოგდება

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
<style>
.marked {
   background-color: orange;
   font-size: 30px;
   padding: 16px;
   color: white;
}
</style>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<p>ვახდენთ "marked"-კლასის მოხსნას class="city"-ის ყველა
 ელემენტიდან.
</p>
<p>
<button onclick="w3.removeClass('.city','marked')">
      მოხსენი კლასი!
</button>
</p>
<div id="London" class="city marked">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<br>
```

```
<div id="Paris" class="city marked">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<br>
<div id="Tokyo" class="city marked">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body>
</html>
```

9) მოცემულ იდენტიფიკატორიანი ელემენტისათვის რამდენიმე კლასის ამოგდება

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
<style>
.class1 {
    background-color: orange;
    color: white;
    padding:16px;
}
.class2 {
    font-size: 30px;
}
</style>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
```

```
<p>
"class1" და "class2" კლასებს ვუნადგურებთ id="London"-ის მქონე ელემენტს.
</p>
<p>
<button onclick="w3.removeClass('#London','class1 class2')">
    ამოვაგდოთ კლასები!
</button>
</p>
<div id="London" class="city class1 class2">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris" class="city">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo" class="city">
  <h2>ტოკიო</h2>
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body>
</html>
```

10) მოცემულ იდენტიფიკატორიანი ელემენტის კლასზე გადართვა-გადმორთვა

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
```

```
<style>
.marked {
   background-color: orange;
   color: white;
   font-size: 30px;
   padding: 16px;
}
</style>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<p>id="London" ელემენტისათვის ვახდენთ marked-კლასის ჩართვა-ამოგდებას.
</p>
<p>
<button onclick="w3.toggleClass('#London','marked')">
    კლასის ჩართვა-ამოგდება!
</button>
</p>
<div id="London" class="city">
  <h2>ლონდონი</h2>
  <p>ლონდონი ინგლისის დედაქალაქია.</p>
</div>
<div id="Paris" class="city">
  <h2>პარიზი</h2>
  <p>პარიზი საფრანგეთის დედაქალაქია.</p>
</div>
<div id="Tokyo" class="city">
  <h2>ტოკიო</h2>
```

```
  <p>ტოკიო იაპონიის დედაქალაქია.</p>
</div>
</body></html>
```

დავალება: კლასზე გადართვა-გადმორთვისათვის მსგავსი წესი გამოიყენეთ ადრე განხილული შემთხვევებისთვისაც.

1.5. W3.JS-ის მეშვეობით ელემენტების ფილტრაცია

1) სიის ელემენტების ფილტრაცია

```
<!DOCTYPE html>
<html>
  <title>W 3 . J S </title>
 <meta charset="utf-8">
<script src="w3.js"></script>
<body>
 <center>
  <h2 align="center">W3.JS-ის მეშვეობით ელემენტების
ფილტრაციის ხერხები</h2>
  <h3>
     დავიწყოთ სიაში მოსაძებნი ობიექტის სახელის რაიმე
ფრაგმენტის აკრეფვა:
  </h3>
  <input oninput="w3.filterHTML('#id01', 'li', this.value)"
placeholder="მოიძიეთ ფაკ. ნომრის ან სახელის
მიხედვით:">
 </center>
 <ul id="id01">
 <li>01. სამშენებლო ფაკულტეტი</li>
  <li>02. ენერგეტიკის ფაკულტეტი</li>
```

```
  <li>03. სამთო-გეოლოგიური ფაკულტეტი</li>
  <li>04. ქიმიური ტექნოლოგიის და მეტალურგიის
ფაკულტეტი</li>
  <li>05. სატრანსპორტო სისტემებისა და მექანიკის
ინჟინერიის ფაკულტეტი</li>
  <li>06. არქიტექტურის, ურბანისტიკის და დიზაინის
ფაკულტეტი</li>
  <li>07. სამართლისა და საერთაშორისო ურთიერთობების
ფაკულტეტი</li>
  <li>08. ბიზნესტექნოლოგიების ფაკულტეტი</li>
  <li>09. საინჟინრო ეკონომიკის, მედიატექნოლოგიებისა და
სოციალურ მეცნიერებათა ფაკულტეტი</li>
  <li>10. ინფორმატიკისა და მართვის სიტემების
ფაკულტეტი</li>
  <li>11. დიზაინის საერთაშორისო სკოლა</li>
  <li>12. აგრარული მეცნიერებების და მიოსისტემების
ინჟინერინგის  ფაკულტეტი</li>
  <li>13. მთის მდგრადი განვითარების ფაკულტეტი</li>
  <li>14. მედიცინის ფაკულტეტი</li>
 </ul>
 </body>
</html>
```

1_1) სიის ელემენტების ფილტრაცია

```
<!DOCTYPE html>
<html>
   <title>W 3 . J S </title>
  <meta charset="utf-8">
   <script src="w3.js"></script>
```

```
<body>
 <center>
  <h1 align="center">W3.JS-ის მეშვეობით ელემენტების
ფილტრაციის ხერხები</h1>
  <h2>დავიწყოთ სიაში მოსაძებნი ობიექტის შესახებ რაიმე
ფრაგმენტის - ფაკ. ნომრის ან სახელის - აკრეფვა:
   </h2>
        <input oninput="w3.filterHTML('#id01', 'li', this.value)">
 </center>
 <h3>
 <ul id="id01">
  <li>01. სამშენებლო ფაკულტეტი</li>
  <li>02. ენერგეტიკის ფაკულტეტი</li>
  <li>03. სამთო-გეოლოგიური ფაკულტეტი</li>
  <li>04. ქიმიური ტექნოლოგიის და მეტალურგიის
ფაკულტეტი</li>
  <li>05. სატრანსპორტო სისტემებისა და მექანიკის
ინჟინერიის ფაკულტეტი</li>
  <li>06. არქიტექტურის, ურბანისტიკის და დიზაინის
ფაკულტეტი</li>
  <li>07. სამართლისა და საერთაშორისო ურთიერთობების
ფაკულტეტი</li>
  <li>08. ბიზნესტექნოლოგიების ფაკულტეტი</li>
  <li>09. საინჟინრო ეკონომიკის, მედიატექნოლოგიებისა და
სოციალურ მეცნიერებათა ფაკულტეტი</li>
  <li>10. ინფორმატიკისა და მართვის სიტემების
ფაკულტეტი</li>
  <li>11. დიზაინის საერთაშორისო სკოლა</li>
```

```
        <li>12. აგრარული მეცნიერებების და მიოსისტემების
      ინჟინერინგის  ფაკულტეტი</li>
        <li>13. მთის მდგრადი განვითარების ფაკულტეტი</li>
        <li>14. მედიცინის ფაკულტეტი</li>
       </ul>
      </h3>

</body>
</html>
```

1_2) ცხრილში არსებული ინფორმაციის ფილტრაცია

```
      <!DOCTYPE html>
      <html>
       <title> W 3 . J S </title>
       <meta charset="utf-8">
       <script src="w3.js"></script>
      <body>
      <center>
       <h2>Filter Table</h2>
       <h3>დაიწყეთ ცხრილში მოყვანილი პარამეტრების
მნიშვნელობებიდან თქვენ მიერ გახსენებული რაიმე ფრაგმენტის
აკრეფვა:
      </h3>
       <p>
        <input oninput="w3.filterHTML('#id01', '.item', this.value)"
placeholder="დაიწყეთ აკრეფვა!">
      </p>
       <table id="id01" border="1">
        <tr>
         <th>შემკვეთი</th>
```

```
  <th>ქალაქი</th>
  <th>ტელ.</th>
 </tr>
 <tr class="item">
  <td>აბაშიძე გიორგი</td>
  <td>ბათუმი</td>
  <td>294-94-94</td>
 </tr>
 <tr class="item">
  <td>ასათიანი გოჩა</td>
  <td>თბილისი</td>
  <td>293-21-12</td>
 </tr>
 <tr class="item">
  <td>ბაღათურია გიორგი</td>
  <td>თბილისი</td>
  <td>293-67-25</td>
 </tr>
 <tr class="item">
  <td>ბერიძე თეონა</td>
  <td>ქობულეთი</td>
  <td>595-22-78</td>
 </tr>
<tr class="item">
  <td>გიორგაძე ნინო</td>
  <td>ზესტაფონი</td>
  <td>293-11-57</td>
 </tr>
 <tr class="item">
```

```
  <td>დვალი ნიკოლოზი</td>
  <td>ამბროლაური</td>
  <td>593-18-45</td>
 </tr>
 <tr class="item">
  <td>ელიავა მედეა</td>
  <td>სენაკი</td>
  <td>255-55-94</td>
 </tr>
  <tr class="item">
  <td>თალაშვილი ოთარი</td>
  <td>ნატახტარი</td>
  <td>275-74-75</td>
 </tr>
 <tr class="item">
  <td>ინასარიძე მალხაზი</td>
  <td>გორი</td>
  <td>299-58-59</td>
 </tr>
 <tr class="item">
  <td>მესტვირიშვილი მზია</td>
  <td>საგარეჯო</td>
  <td>277-45-69</td>
 </tr>
<tr class="item">
  <td>ნემსაძე ალექსანდრე</td>
  <td>ხაშური</td>
  <td>234-33-77</td>
 </tr>
```

```
  <tr class="item">
    <td>ობგაიძე გიორგი</td>
    <td>თიანეთი</td>
    <td>234-55-29</td>
  </tr>
  <tr class="item">
    <td>რამიშვილი ნესტანი</td>
    <td>ოზურგეთი</td>
    <td>299-89-65</td>
  </tr>
  <tr class="item">
    <td>შავიშვილი თორნიკე</td>
    <td>ჟინვალი</td>
    <td>299-09-67</td>
  </tr>
  <tr class="item">
    <td>ცერცვაძე რამაზი</td>
    <td>ზესტაფონი</td>
    <td>277-01-99</td>
  </tr>
 </table>
</center>
</body>
</html>
```

1.6. W3.JS-ის მეთოდებით მონაცემების დახარისხება

1) სიის ელემენტების დახარისხება

```
<!DOCTYPE html>
<html>
```

```
    <title>W3.JS</title>
  <meta charset="utf-8">
  <script src="https://www.w3schools.com/lib/w3.js"></script>
 <body>
  <center>
   <h2>W3.JS-ის მეთოდებით მონაცემების დახარისხება
   </h2>
   <h3>სიის ელემენტების დახარისხება</h3>
  </center>
  <button onclick =" w3.sortHTML('#id01','li')">
    <h3>დააწკაპუნეთ აქ (მეორედაც)!</h3>
   </button>
 <h3>
  <ul id="id01">
   <li>ოსლო</li>
   <li>სტოკჰოლმი</li>
   <li>ჰელსინკი</li>
   <li>რომი</li>
   <li>ლისაბონი</li>
   <li>მადრიდი</li>
  </ul>
 </h3>
 </body>
 </html>
```

2) ცხრილებში არსებული მონაცემების დახარისხება

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
```

```
<meta charset="utf-8">
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<center>
<h2>W3.JS-ის მეთოდებით მონაცემების დახარისხება</h2>
<h3>ცხრილის ელემენტების დახარისხება</h3>
</center>
<button onclick="w3.sortHTML('#id01','li')"><h3>დააწკაპუნეთ
ცხრილში ს ვ ე ტ ე ბ ზ ე (მეორედაც)!</h3>
</button>
<h3>
<table id="myTable">
<tr>
<th onclick = "w3.sortHTML('#myTable', '.item', 'td:nth-
child(1)')" style="cursor:pointer">დ ე დ ა ქ ა ლ ა ქ ი</th>
<th onclick="w3.sortHTML('#myTable', '.item', 'td:nth-
child(2)')" style="cursor:pointer">ქ ვ ე ყ ა ნ ა </th>
</tr>
<tr class="item">
<td>ოსლო</td>
<td>ნორვეგია</td>
</tr>
<tr class="item">
<td>სტოკჰოლმი</td>
<td>შვედეთი</td>
</tr>
<tr class="item">
<td>ჰელსინკი</td>
<td>ფინეთი</td>
```

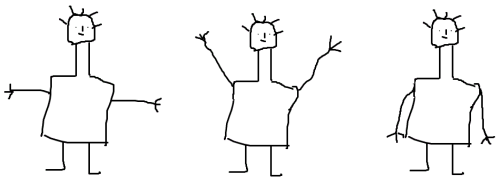
```
  </tr>
  <tr class="item">
    <td>რომი</td>
    <td>იტალია</td>
  </tr>
  <tr class="item">
    <td>ლისაბონი</td>
    <td>პორტუგალია</td>
  </tr>
  <tr class="item">
    <td>მადრიდი</td>
    <td>ესპანეთი</td>
  </tr>
  <tr class="item">
    <td>რეიკიავიკი</td>
    <td>ისლანდია</td>
  </tr>
  <tr class="item">
    <td>ბრაზილია</td>
    <td>ბრაზილია</td>
  </tr>
 </table>
</h3>
</body>
</html>
```

1.7. W3.JS-ის მეთოდებით სლაიდ-შოუს ორგანიზება

```
<!DOCTYPE html>
<html>
```

```
 <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
 <center>
  <h2>Testing W3.JS in HTML</h2>
  <h3>იხილეთ სლაიდ-შოუ class="gymnastics"-ისათვის
 </h3>
  <img class="gymnastics" src="1.gif" width="35%">
  <img class="gymnastics" src="2.gif" width="35%">
  <img class="gymnastics" src="3.gif" width="35%">
  <script>
    w3.slideshow(".gymnastics", 1200);
  </script>
 </center>
</body></html>
```

მიმაგრებული ფაილები:



თუკი არ გვსურს, რომ სლაიდ-შოუ ავტომატურად იქნას გაშვებული, მაშინ w3.slideshow(".gymnastics", 0); ოპერატორში მეორე პარამეტრს ვანიჭებთ ნულის ტოლ მნიშვნელობას, ხოლო სლაიდიდან სლაიდზე გადასვლების ორგანიზების მიზნით პროგრამაში ვამატებთ 2 ღილაკს. მათი მეშვეობით მივმართავთ ენაში არსებულ შესაბამის ფუნქციებს. სლაიდ-შოუში ჩართული

ფუნქცია w3.slideshow() აბრუნებს ობიექტს, რომელიც ვიცით, რომ შეიცავს თვისებებს და მეთოდებს - ამ კონკრეტულ შემთხვევაში ეს მეთოდებია: next () და previous ().

ზემოთ თქმულის გათვალისწინებით, პროგრამა საბოლოოდ ასეთ სახეს მიიღებს:

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
 <center>
  <h2>Testing W3.JS in HTML</h2>
  <h3>იხილეთ შოუს სხვა ვარიანტი class="gymnastics"-ისათვის</h3>
  <img class="gymnastics" src="1.gif" width="35%">
  <img class="gymnastics" src="2.gif" width="35%">
  <img class="gymnastics" src="3.gif" width="35%">
<p>
<button onclick="myShow.previous()">Previous</button>
<button onclick="myShow.next()">Next</button>
</p>
  <script>
      myShow = w3.slideshow(".gymnastics", 0);
  </script>
 </center>
</body>
</html>
```

სლაიდ-შოუში შესაძლებელია ნებისმიერი html-ელემენტის ჩართვაც:

```
<!DOCTYPE html>
<html>
  <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
  <center>
   <h2>Testing W3.JS in HTML</h2>
   <h3>იხილეთ შოუში სხვა html-ელემენტების ჩართვის
   შესაძლებლობაც
    </h3>
   <img class="gymnastics" src="1.gif" width="35%">
   <img class="gymnastics" src="2.gif" width="35%">
   <img class="gymnastics" src="3.gif" width="35%">
   <h1 class="city">განმკლავი!</h1>
   <h1 class="city">ზემკლავი!</h1>
   <h1 class="city">თავისუფლად!</h1>
    <button onclick="myShow.previous()">Previous</button>
   <button onclick="myShow.next()">Next</button>
  </p>
   <script>
      myShow = w3.slideshow(".gymnastics", 1000);
      w3.slideshow(".city", 1000);
   </script>
 </center>
</body>
</html>
```

აქვე აღვნიშნოთ, რომ კოდში ამ დაყრდნობის

```
<link rel="stylesheet"  href =
"https://www.w3schools.com/w3css/4/w3.css">
```

ჩართვით შესაძლებელი ხდება მასში html-ელემენტების ფორმატი ვცვალოთ, კერძოდ, მივანიჭოთ მათ პარამეტრებს დუმილით დანიშნულთა ნაცვლად CSS-ით შერჩეული მნიშვნელობები:

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
  <link rel="stylesheet" href =
   "https://www.w3schools.com/w3css/4/w3.css">
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body class="w3-container">
<h2>Testing W3.JS with CSS</h2>
<p>Elements with a class="city" will behave like a slideshow:</p>
<h1 class="city"> ლ ო ნ დ ო ნ ი</h1>
<h1 class="city"> პ ა რ ი ზ ი</h1>
<h1 class="city">ტოკიო</h1>
<script>
w3.slideshow(".city");
</script></body>
</html>
```

1.8. W3.JS-ის დახმარებით კოდში html-ფაილების ჩართვა

W3.JS-ისათვის კუთვნილი w3-include-html ატრიბუტის გამოყენებით შესაძლებელი არის მოცემულ კოდში html-ფაილების ჩართვა უფრო მოხერხებული გავხადოთ, მით უფრო მაშინ, როდესაც ასეთი ჩართვები საკმაოდ ხშირად გვჭირდება.

ამ მიზნით, თავდაპირველად ვქმნით .html გაფართოების მქონე დამხმარე ფაილს, მაგალითად, content.html-ს, ვთქვათ, ასეთი შიგთავსით:

```
<a href="https://www.w3schools.com/html/">HTML</a><br>
```

```
<a href="https://www.w3schools.com/css/">CSS</a><br>
<a href="https://www.w3schools.com/bootstrap/">Bootstrap</a><br>
<a href="https://www.w3schools.com/js/">JavaScript</a><br>
<a href="https://www.w3schools.com/sql/">SQL</a><br>
<a href="https://www.w3schools.com/php/">PHP</a><br>
<a href="https://www.w3schools.com/w3css/">W3.CSS</a><br>
```

გამომძახებელი ფაილისათვის კი ვწერთ შემდეგ კოდს, მაგალითად, ისეთი შემთხვევისათვის, როდესაც გვჭირდება w3schools.com საიტიდან html-ენის შესახებ არსებული სასწავლო მასალის გადმოქაჩვა:

```
<!DOCTYPE html>
<html>
  <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<div w3-include-html="content.html"></div>
<script>
w3.includeHTML();
</script>
</body>
</html>
```

ზემოთ მოყვანილი ფაილების წყვილი ერთმანეთთან მაშინ ითანამშრომლებს, თუ ამ ფაილებს სერვერზე, მაგალითად, XAMPP სერვერული პაკეტის htdocs-საქაღალდეში განვათავსებთ და ბროუზერის სამისამართო სტრიქონში გამომძახებელი ფაილის http://localhost/... მისამართს ავკრეფთ!

აქვე შევნიშნავთ, რომ გამომძახებელი ფაილის კოდში html-ფაილების ჩართვა შესაძლებელია უფრო მარტივადაც მოვახდინოთ „წმინდა“ Javascript-ში უკვე ნასწავლი გზებით, თუ გამოძახებული ფაილიც იმავე საქაღალდეში იმყოფება, რომელშიც განთავსებულია მისი გამომძახებელი ფაილი:

```
<!DOCTYPE html>
<html>
  <title>W3.JS</title>
<body>
<h2 align="center">ფაილის გამოძახება</h2>
<h2 align="center">
 <a href="რწყილი_და_ჭიანჭველა.html">
  </a> ვიძახებთ დამხმარე ფაილს!
</h2>
</body>
</html>
```

1.9. W3.JS-ის დახმარებით მონიტორზე html-მონაცემებისათვის ადგილის დარეზერვირება

{{}} ფიგურული ფრჩხილების დახმარებით შესაძლებელია ნებისმიერი html-ელემენტისათვის მოხდეს ადგილის დარეზერვირება, რათა მონიტორზე ავსახოთ ამ ელემენტისათვის გამოთვლების შემდგომ ეტაპებზე მინიჭებული მნიშვნელობა:

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
```

```
<h2>Testing W3.JS in HTML</h2>
<div id="id01">
{{firstName}} {{lastName}}
</div>
<script>
var myObject= {"firstName" : "გიორგი", "lastName" : "ბერიძე"};
w3.displayObject("id01", myObject);
</script>
</body>
</html>
```

შემდეგ იმავე **{{}}** ფიგურული ფრჩხილებისა და მონაცემების სელექციის განკუთვნილი შემდეგი ფრაგმენტის დახმარებით:

```
<select id="id01">
  <option w3-repeat="შემკვეთი">{{კლიენტი}}</option>
</select>
```

გადმოვწეროთ ყველა მონაცემი მათი შემცველი უფრო რთული ობიექტიდან და ავსახოთ ისინი ჩამოშლად სიაში:

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
<meta charset="utf-8">
<script src="https://www.w3schools.com/lib/w3.js"></script>
<script>
var myObject= {"შემკვეთი":[
{"კლიენტი" : "აბაშიძე რევაზი","ქალაქი" : "ბათუმი","ტელ" : "111-11- 11-11"},
{"კლიენტი" : "ბერიძე გიორგი","ქალაქი" : "ქუთაისი","ტელ" : "222-22-22-22"},
```

{"კლიენტი" : "ბიბილაშვილი თეონა","ქალაქი" : "თბილისი","ტელ" : "333-33-33-33"},
{"კლიენტი" : "გიორგაძე დავითი", "ქალაქი" : "გორი","ტელ" : "444-44-44-44"},
{"კლიენტი" : "დვალი ზურაბი","ქალაქი" : "სამტრედია","ტელ" : "555-55-55-55"},
{"კლიენტი" : "ველიაშვილი ზაზა","ქალაქი" : "თელავი","ტელ" : "111-11-11-22"},
{"კლიენტი" : "თუთბერიძე ლიანა", "ქალაქი" : "ქუთაისი","ტელ" : "222-22-22-333"},
{"კლიენტი" : "ლომიძე ოთარი","ქალაქი" : "გორი","ტელ" : "333-33-33-444"},
{"კლიენტი" : "მონიავა ნინო","ქალაქი" : "სენაკი","ტელ" : "444-44-44-555"},
{"კლიენტი" : "მრევლიშვილი ნინო","ქალაქი" : "ხაშური","ტელ" : "555-55-55-99"},
{"კლიენტი" : "ნარეშელაშვილი ლევანი","ქალაქი" : "ქარელი","ტელ" : "111-11-11-333"},
{"კლიენტი" : "ოდიშარია კორნელი","ქალაქი" : "თბილისი","ტელ" : "222-22-22-444"},
{"კლიენტი" : "სტურუა ელენე","ქალაქი" : "ქუთაისი","ტელ" : "333-33-33-777"},
{"კლიენტი" : "უგულავა გიორგი","ქალაქი" : "ზუგდიდი","ტელ" : "444-44-44-888"},
{"კლიენტი" : "ღაღანიძე გიგა","ქალაქი" : "ზესტაფონი","ტელ" : "555-55-55-999"},
{"კლიენტი" : "შავიშვილი თორნიკე","ქალაქი" : "თბილისი","ტელ" : "111-11-11-444"},

{"კლიენტი" : "ჩაჩანიძე გეგა","ქალაქი" : "თბილისი","ტელ" : "222-22-22-777"}

]};

</script>

<body>

<h2 align='center'>Testing W3.JS in HTML</h2>

<h3 align='center'> ამჯერად მონაცემების სელექციისათვის განკუთვნილ select-ელემენტში განთავსებული ფრაგმენტის დახმარებით

უფრო რთულ myObject ობიექტში ავირჩევთ და გადმოვწერთ "კლიენტი"- ელემენტისთვის ფიქსირებულ ყველა მონაცემს და

 მათ მონიტორზე ავსახავთ ჩამოშლადი სიის სახით (შემდგომში რომელიმე პუნქტის არჩევის შესაძლებლობით).

</h3>

<select id="id01">

<option w3-repeat = "შემკვეთი">{{კლიენტი}}</option>

<!-- ზემოთ w3-repeat ატრიბუტი ასრულებს ციკლის ფუნქციას myObject-ის "შემკვეთი"-ობიექტის შიგთავსისთვის, რომლიდანაც არჩეული "კლიენტი"-ელემენტის მნიშვნელობებით შეივსება სიაში ყველა შესაბამისი პუნქტი.

-->

</select>

<script>

// ქვემოთ მოყვანილ ოპერატორში .displayObject მეთოდით ხდება id01-თი იდენტიფიცირებული select ელემენტისადმი მიმართვა და მისთვის myObject ობიექტის გადაცემა შემდგომი დამუშავებისათვის.

w3.displayObject("id01", myObject);

</script>

```
</body>
</html>
```

ამ ფაილის შესრულებაზე გაშვების შემდეგ კლიენტების შესახებ ინფორმაცია მონიტორზე აისახება ჩამოშლადი სიის სახით.

ამჯერად მოვიყვანოთ წინა ფაილის ვარიანტი, რომელშიც w3.displayObject("id01", myObject); ოპერატორისადმი მიმართვა მოხდება ამ ოპერატორის შემცველი ფუნქციის გამოძახებით:

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
  <script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
 <h2>Testing W3.JS in HTML</h2>
<select id="id01">
   <option w3-repeat = "შემკვეთი">{{კლიენტი}}</option>
</select>
<script>
var myObject= {"შემკვეთი":[
{"კლიენტი" : "აბაშიძე რევაზი","ქალაქი" : "ბათუმი","ტელ" : "111-11- 11-11"},
{"კლიენტი" : "ბერიძე გიორგი","ქალაქი" : "ქუთაისი","ტელ" : "222-22-22-22"},
{"კლიენტი" : "ბიბილაშვილი თეონა","ქალაქი" : "თბილისი","ტელ" : "333-33-33-33"},
{"კლიენტი" : "გიორგაძე დავითი", "ქალაქი" : "გორი","ტელ" : "444-44-44-44"},
```

```
{"კლიენტი" : "დვალი ზურაბი","ქალაქი" : "სამტრედია","ტელ" : "555-55-55-55"},
{"კლიენტი" : "ველიაშვილი ზაზა","ქალაქი" : "თელავი","ტელ" : "111-11-11-22"},
{"კლიენტი" : "თუთბერიძე ლიანა", "ქალაქი" : "ქუთაისი","ტელ" : "222-22-22-333"},
{"კლიენტი" : "ლომიძე ოთარი","ქალაქი" : "გორი","ტელ" : "333-33-33-444"},
{"კლიენტი" : "მონიავა ნინო","ქალაქი" : "სენაკი","ტელ" : "444-44-44-555"},
{"კლიენტი" : "მრევლიშვილი ნინო","ქალაქი" : "ხაშური","ტელ" : "555-55-55-99"},
{"კლიენტი" : "ნარეშელაშვილი ლევანი","ქალაქი" : "ქარელი","ტელ" : "111-11-11-333"},
{"კლიენტი" : "ოდიშარია კორნელი","ქალაქი" : "თბილისი","ტელ" : "222-22-22-444"},
{"კლიენტი" : "სტურუა ელენე","ქალაქი" : "ქუთაისი","ტელ" : "333-33-33-777"},
{"კლიენტი" : "უგულავა გიორგი","ქალაქი" : "ზუგდიდი","ტელ" : "444-44-44-888"},
{"კლიენტი" : "ღაღანიძე გიგა","ქალაქი" : "ზესტაფონი","ტელ" : "555-55-55-999"},
{"კლიენტი" : "შავიშვილი თორნიკე","ქალაქი" : "თბილისი","ტელ" : "111-11-11-444"},
{"კლიენტი" : "ჩაჩანიძე გეგა","ქალაქი" : "თბილისი","ტელ" : "222-22-22-777"}
]};
    myFunction(myObject);  // ვიძახებთ ფუნქციას.
```

```
function myFunction(myObject) {
    w3.displayObject("id01", myObject);
 }
</script>
</body>
</html>
```

იგივე მანიპულაცია ჩავატაროთ სიის მხოლოდ შესავსებად (ანუ არჩევანის გაკეთების შესაძლებლობის გარეშე) კოდის შემდეგ ფრაგმენტზე დაყრდნობით:

```
<ul id="id01">
  <li w3-repeat="შემკვეთი">{{კლიენტი}}</li>
</ul>
```

მოგვყავს ამ კოდის შემცველობა:

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
<meta charset="utf-8">
<script src="https://www.w3schools.com/lib/w3.js"></script>

<script>
var myObject= {"შემკვეთი":[
{"გვარი" : "აბაშიძე რევაზი","ქალაქი" : "ბათუმი","ტელ" : "111-11- 11-11"},
{"გვარი" : "ბერიძე გიორგი","ქალაქი" : "ქუთაისი","ტელ" : "222-22-22-22"},
```

{"გვარი" : "ბიბილაშვილი თეონა","ქალაქი" : "თბილისი","ტელ" : "333-33-33-33"},

{"გვარი" : "გიორგაძე დავითი", "ქალაქი" : "გორი","ტელ" : "444-44-44-44"},

{"გვარი" : "დვალი ზურაბი","ქალაქი" : "სამტრედია","ტელ" : "555-55-55-55"},

{"გვარი" : "ველიაშვილი ზაზა","ქალაქი" : "თელავი","ტელ" : "111-11-11-22"},

{"გვარი" : "თუთბერიძე ლიანა", "ქალაქი" : "ქუთაისი","ტელ" : "222-22-22-333"},

{"გვარი" : "ლომიძე ოთარი","ქალაქი" : "გორი","ტელ" : "333-33-33-444"},

{"გვარი" : "მონიავა ნინო","ქალაქი" : "სენაკი","ტელ" : "444-44-44-555"},

{"გვარი" : "მრევლიშვილი ნინო","ქალაქი" : "ხაშური","ტელ" : "555-55-55-99"},

{"გვარი" : "ნარეშელაშვილი ლევანი","ქალაქი" : "ქარელი","ტელ" : "111-11-11-333"},

{"გვარი" : "ოდიშარია კორნელი","ქალაქი" : "თბილისი","ტელ" : "222-22-22-444"},

{"გვარი" : "სტურუა ელენე","ქალაქი" : "ქუთაისი","ტელ" : "333-33-33-777"},

{"გვარი" : "უგულავა გიორგი","ქალაქი" : "ზუგდიდი","ტელ" : "444-44-44-888"},

{"გვარი" : "ღაღანიძე გიგა","ქალაქი" : "ზესტაფონი","ტელ" : "555-55-55-999"},

{"გვარი" : "შავიშვილი თორნიკე","ქალაქი" : "თბილისი","ტელ" : "111-11-11-444"},

```
{"გვარი" : "ჩაჩანიძე გეგა","ქალაქი" : "თბილისი","ტელ" : "222-22-22-777"}

]};

</script>
<body>
 <center>
  <h2>Testing W3.JS in HTML</h2>
  <h2>იგივე მანიპულაცია ამჯერად მხოლოდ სიის შესავსებად (ანუ არჩევანის გაკეთების შესაძლებლობის გარეშე). </h2>
 </center>
 <!-- begin - იგივე მანიპულაცია ჩავატაროთ ამჯერად <ul> სიის შესავსებად კოდის შემდეგ ფრაგმენტზე დაყრდნობით. -->
  <h3>
 <ul id="id01">
  <li w3-repeat="შემკვეთი">{{გვარი}}</li>
 </ul>
 </h3>
 <!-- e n d - იგივე მანიპულაცია ჩავატაროთ ამჯერად <ul> სიის შესავსებად კოდის შემდეგ ფრაგმენტზე დაყრდნობით. -->
  <script>
     w3.displayObject("id01", myObject);
  </script>

</body>
</html>
```

მონიტორზე აისახება ასეთი შედეგი:

Testing W3.JS in HTML

1.10. myCallback() ფუნქცია

ვებფურცელზე განთავსებულ HTML-ელემენტებზე დამოკიდებულმა ფუნქციებმა რომ მანამდე არ დაიწყონ შესრულება, ვიდრე ყველა ეს ელემენტი თავის ადგილზე არ ჩაიტვირთება, აღნიშნული არასასურველი ქმედების აღსაკვეთად w3.includeHTML() ფუნქციას არგუმენტად განუსაზღვრავენ უკუგამოძახების myCallback() ფუნქციას.

მოგვყავს შესაბამისი გადაწყვეტილების მაგალითი:

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
<meta charset="utf-8">
<script src="https://www.w3schools.com/lib/w3.js"></script>
<script>
var myObject= {"შემკვეთი":[
{"კლიენტი" : "აბაშიძე რევაზი","ქალაქი" : "ბათუმი","ტელ" : "111-11-11-11"},
{"კლიენტი" : "ბერიძე გიორგი","ქალაქი" : "ქუთაისი","ტელ" : "222-22-22-22"},
{"კლიენტი" : "ბიბილაშვილი თეონა","ქალაქი" : "თბილისი","ტელ" : "333-33-33-33"},
{"კლიენტი" : "გიორგაძე დავითი", "ქალაქი" : "გორი","ტელ" : "444-44-44-44"},
{"კლიენტი" : "დვალი ზურაბი","ქალაქი" : "სამტრედია","ტელ" : "555-55-55-55"},
{"კლიენტი" : "ველიაშვილი ზაზა","ქალაქი" : "თელავი","ტელ" : "111-11-11-22"},
```

{"კლიენტი" : "თუთბერიძე ლიანა", "ქალაქი" : "ქუთაისი","ტელ" : "222-22-22-333"},
{"კლიენტი" : "ლომიძე ოთარი","ქალაქი" : "გორი","ტელ" : "333-33-33-444"},
{"კლიენტი" : "მონიავა ნინო","ქალაქი" : "სენაკი","ტელ" : "444-44-44-555"},
{"კლიენტი" : "მრევლიშვილი ნინო","ქალაქი" : "ხაშური","ტელ" : "555-55-55-99"},
{"კლიენტი" : "წარეშელაშვილი ლევანი","ქალაქი" : "ქარელი","ტელ" : "111-11-11-333"},
{"კლიენტი" : "ოდიშარია კორნელი","ქალაქი" : "თბილისი","ტელ" : "222-22-22-444"},
{"კლიენტი" : "სტურუა ელენე","ქალაქი" : "ქუთაისი","ტელ" : "333-33-33-777"},
{"კლიენტი" : "უგულავა გიორგი","ქალაქი" : "ზუგდიდი","ტელ" : "444-44-44-888"},
{"კლიენტი" : "ღაღანიძე გიგა","ქალაქი" : "ზესტაფონი","ტელ" : "555-55-55-999"},
{"კლიენტი" : "შავიშვილი თორნიკე","ქალაქი" : "თბილისი","ტელ" : "111-11-11-444"},
{"კლიენტი" : "ჩაჩანიძე გეგა","ქალაქი" : "თბილისი","ტელ" : "222-22-22-777"}
]};
</script>
<body>
<h2 align='center'>Testing W3.JS in HTML</h2>
<h3 align='center'> ამჯერად მონაცემების სელექციისათვის განკუთვნილ select-ელემენტში განთავსებული ფრაგმენტის დახმარებით

უფრო რთულ myObject ობიექტში ავირჩევთ და გადმოვწერთ "კლიენტი"- ელემენტისთვის ფიქსირებულ ყველა მონაცემს და

 მათ მონიტორზე ავსახავთ ჩამოშლადი სიის სახით (შემდგომში რომელიმე პუნქტის არჩევის შესაძლებლობით).

```
</h3>
 <select id="id01">
   <option w3-repeat = "შემკვეთი">{{კლიენტი}}</option>
<!-- ზემოთ w3-repeat ატრიბუტი ასრულებს ციკლის ფუნქციას myObject-ის "შემკვეთი" ობიექტის შიგთავსისთვის, რომლიდანაც არჩეული "კლიენტი"- ელემენტის მნიშვნელობებით შეივსება ეს {{გვარი}} უბანი. -->
</select>
<script>
        w3.includeHTML(myCallback);
        function myCallback() {
         w3.displayObject("id01", myObject);
      }
 </script>
</body>
</html>
```

1.11. მანიპულაციები ცხრილებზე

table ცხრილის tr სტრიქონებში, რომლებიც მოცემულ შემთხვევაში თითო-თითო td უჯრას მოიცავენ, თითოეულ სტრიქონში w3-repeat="შემკვეთი" ატრიბუტის მნიშვნელობის მიხედვით ვახდენთ „შემკვეთი“-ობიექტის ელემენტების ჩათვალიერებას და მათზე ვატარებთ შემდეგ მანიპულაციებს:

- td უჯრებს ვავსებთ კლიენტების შესახებ „კლიენტი“-ველში დაფიქსირებული ინფორმაციით;

- tr სტრიქონებისათვის (ფაქტობრივად, მოცემულ შემთხვევაში სტრიქონში შემავალი ერთი უჯრისათვის) class="{{ფერი}} ატრიბუტითა და მისი მეშვეობით „ფერი“-ველის მნიშვნელობაზე გასვლით ვაყენებთ შესაბამისი ფონის ფერს:

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
<style>
  .red {background-color: red}
  .green {background-color: green}
  .blue {background-color: blue}
  .yellow {background-color: yellow}
 </style>
 <script src="https://www.w3schools.com/lib/w3.js"></script>
<script>
  var myObject = {"შემკვეთი":[
  {"კლიენტი":"აბაშიძე გიორგი","ფერი":"red"},
  {"კლიენტი":"გოცირიძე დავითი","ფერი":"green"},
  {"კლიენტი":"მახვილაძე ნინო","ფერი":"blue"},
  {"კლიენტი":"ჯელია თამარი","ფერი":"yellow"}
  ]};
 </script>
<body>
<h2 align='center'>Testing W3.JS in HTML</h2>
  <h2 align='center'><i>ცხრილის შევსებისას w3.js რამდენადმე განსხვავებულ
     სინტაქსს იყენებს! (იხ. კოდი) </i>
  </h2>
```

```
<h2>
<table id="id01">
    <tr w3-repeat="შემკვეთი" class="{{ფერი}}"> <!-- მივაქციოთ ყურადღება -
        ცხრილის შევსებისას ამ და მომდევნო სტრიქონზე w3.js რამდენადმე განსხვავებულ სინტაქსს იყენებს! -->
    <td>{{კლიენტი}}</td> <!-- ამასთან, ელემენტისათვის ზემოთ შერჩეული ფერი, თავის მხრივ, უფრო ზემოთ <style> ელემენტში კონკრეტდება, -->
    </tr> <!-- თუ ეს ფერი სად უნდა იქნეს გამოყენებული (მოცემულ შემთხვევაში ფონის ფერად.) -->
</table>
        </h2>
        <script>
          w3.displayObject("id01", myObject);
        </script>
       </body>
       </html>
```

დავალება: შეამოწმეთ, რა მოხდება, თუ ზემოთ მოყვანილ კოდში წითელი ფერის ფრაგმენტს ასეთზე შევცვლით:

```
<tr w3-repeat="შემკვეთი">
  <td>{{კლიენტი}}  {{Color}}</td>
```

განვიხილოთ კიდევ ორი, წინასთან შედარებით რამდენადმდე უფრო მარტივი მაგალითი:

მაგალითი 1:

```
<!DOCTYPE html>
<html>
```

```
<title>W3.JS</title>

<style>
 .red {background-color: red}
 .green {background-color: green}
 .blue {background-color: blue}
 .yellow {background-color: yellow}
</style>

<script src="https://www.w3schools.com/lib/w3.js"></script>

<script>

 var myObject = {"შემკვეთი":[
  {"გვარისახელი":"აბაშიძე გიორგი","ფერი":"red"},
  {"გვარისახელი":"გოცირიძე დავითი","ფერი":"green"},
  {"გვარისახელი":"მახვილაძე ნინო","ფერი":"blue"},
  {"გვარისახელი":"ჯელია თამარი","ფერი":"yellow"}
 ]};

</script>
<body>

 <h2 align='center'>Testing W3.JS in HTML</h2>
 <h3 align='center'><i>აქ, წინა ფაილისგან განსხვავებით, უფრო მარტივი მაგალითი განიხილება.(იხ. კოდი)</i></h3>
  <h3>
 <table id="id01">
  <tr w3-repeat="შემკვეთი">
   <td>{{გვარისახელი}}</td>
  </tr>
 </table>
 </h3>
```

```
 <script>
   w3.displayObject("id01", myObject);
 </script>

</body>
</html>
```

მაგალითი 2:

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
 <style>
     .red {background-color: red}
     .green {background-color: green}
     .blue {background-color: blue}
     .yellow {background-color: yellow}
 </style>
 <script src="https://www.w3schools.com/lib/w3.js"></script>
 <script>
  var myObject = {"შემკვეთი":[
    {"გვარისახელი":"აბაშიძე გიორგი","ფერი":"red"},
    {"გვარისახელი":"გოცირიძე დავითი","ფერი":"green"},
    {"გვარისახელი":"მახვილაძე ნინო","ფერი":"blue"},
    {"გვარისახელი":"ჯელია თამარი","ფერი":"yellow"}
  ]};
</script>
<body>
 <center>
  <h2>Testing W3.JS in HTML</h2>
```

```
<h3><i>სრულდება წინა ამოცანა ამჯერად ღილაკზე დაწკაპუნების შემდეგ.</i>
</h3>
<h3>
<table id="id01">
 <tr w3-repeat="შემკვეთი" class="{{Color}}">
  <td>{{გვარისახელი}}</td>
 </tr>
</table>
</h3>
<script>
//  w3.displayObject("id01", myObject);  აღარაა საჭირო!
</script>
<!-- ქვემოთ ღილაკზე დაწკაპუნებით .displayObject მეთოდით id01-იდენტიფიკატორიან ელემენტს გადავცემთ myObject ობიექტის შიგთავსს. -->
<button onclick="w3.displayObject('id01', myObject);"><h3>Click!<h3></button>
</center>
</body>
</html>
```

ამჯერად მოვნიშნოთ ვებფურცელზე table ცხრილში განთავსებულ უჯრებში checkbox-ის ელემენტების ნაწილი:

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<script>
var myObject= {"შემკვეთი":[
```

```
 {"გვარისახელი":"აბაშიძე გიორგი","Ok":false},        //  Ok !!!
 {"გვარისახელი":"გოცირიძე დავითი", "checked":"checked"},
 {"გვარისახელი":"მახვილაძე ნინო", "checked":""},
 {"გვარისახელი":"ჯელია თამარი", "checked":""}
]};
</script>

<body>
 <h2 align='center'>Testing W3.JS in HTML</h2>
 <h3 align='center'><i>აქ ობიექტში ერთ-ერთ ელემენტად
        გათვალისწინებულია checked.</i>
</h3>
 <h3>
 <table id="id01">
  <tr w3-repeat="შემკვეთი">
    <td>{{გვარისახელი}}</td>
    <td><input type="checkbox" {{checked}}></td>
  </tr>
 </table>
  </h3>
 <script>
   w3.displayObject("id01", myObject);
 </script>

</body>
</html>
```

დავალება: ქვემოთ მოყვანილი table ცხრილის შევსების კიდევ ერთი მაგალითის ბაზაზე შეადგინეთ კოდი, რომელშიც ობიექტად არჩეული იქნება სხვა საგნობრივი გარემო (ბიბლიოთეკა, ჯგუფი, მეგობრების წრე და სხვ.)

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
 <meta charset="utf-8">
<script src="https://www.w3schools.com/lib/w3.js"></script>
<script>
  var myObject= {"შემკვეთი":[
        {"გვარი_სახელი" : "აბაშიძე რევაზი","ქალაქი" : "ბათუმი","ტელ" : "111-11- 11-11"},
        {"გვარი_სახელი" : "ბერიძე გიორგი","ქალაქი" : "ქუთაისი","ტელ" : "222-22-22-22"},
        {"გვარი_სახელი" : "ბიბილაშვილი თეონა","ქალაქი" : "თბილისი","ტელ" : "333-33-33-33"},
        {"გვარი_სახელი" : "გიორგაძე დავითი", "ქალაქი" : "გორი","ტელ" : "444-44-44-44"},
        {"გვარი_სახელი" : "დვალი ზურაბი","ქალაქი" : "სამტრედია","ტელ" : "555-55-55-55"},
        {"გვარი_სახელი" : "ველიაშვილი ზაზა","ქალაქი" : "თელავი","ტელ" : "111-11-11-22"},
        {"გვარი_სახელი" : "თუთბერიძე ლიანა", "ქალაქი" : "ქუთაისი","ტელ" : "222-22-22-333"},
        {"გვარი_სახელი" : "ლომიძე ოთარი","ქალაქი" : "გორი","ტელ" : "333-33-33-444"},
        {"გვარი_სახელი" : "მონიავა ნინო","ქალაქი" : "სენაკი","ტელ" : "444-44-44-555"},
        {"გვარი_სახელი" : "მრევლიშვილი ნინო","ქალაქი" : "ხაშური","ტელ" : "555-55-55-99"},
```

```
{"გვარი_სახელი" : "ნარეშელაშვილი ლევანი","ქალაქი" : "ქარელი","ტელ" : "111-11-11-333"},
{"გვარი_სახელი" : "ოდიშარია კორნელი","ქალაქი" : "თბილისი","ტელ" : "222-22-22-444"},
{"გვარი_სახელი" : "სტურუა ელენე","ქალაქი" : "ქუთაისი","ტელ" : "333-33-33-777"},
{"გვარი_სახელი" : "უგულავა გიორგი","ქალაქი" : "ზუგდიდი","ტელ" : "444-44-44-888"},
{"გვარი_სახელი" : "ღაღანიძე გიგა","ქალაქი" : "ზესტაფონი","ტელ" : "555-55-55-999"},
{"გვარი_სახელი" : "შავიშვილი თორნიკე","ქალაქი" : "თბილისი","ტელ" : "111-11-11-444"},
{"გვარი_სახელი" : "ჩაჩანიძე გეგა","ქალაქი" : "თბილისი","ტელ" : "222-22-22-777"}
]};
</script>
<body>
 <h2 align='center'>Testing W3.JS in HTML</h2>
 <h3 align='center'><i>წარმოვადგენთ ობიექტიდან წამოღებული მონაცემების ცხრილში ასახვის კიდევ ერთ ვარიანტს.</i></h3>
 <h3>
 <table id="id01">
  <tr>
    <th>გვარი და სახელი</th>
    <th>ქალაქი</th>
    <th>ტელეფონი</th>
  </tr>
  <tr w3-repeat="შემკვეთი">
```

```
    <td>{{გვარი_სახელი}}</td>
    <td>{{ქალაქი}}</td>
    <td>{{ტელ}}</td>
  </tr>
 </table>
 </h3>
 <script>
   w3.displayObject("id01", myObject);
 </script>
</body>
</html>
```

განვიხილოთ <select> ელემენტის დახმარებით სიის შევსების მაგალითი და მისი შედეგი:

```
<!DOCTYPE html>
<html>
 <title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>
<h2>Testing W3.JS in HTML</h2>
<select id="id01">
  <option w3-repeat="x in cars">{{x}}</option>
</select>
<script>
w3.displayObject("id01", {"cars": ["Volvo", "Ford", "BMW",
"Mercedes" ]});
</script>
</body>
</html>
```

მონიტორზე აისახება ჩამოშლადი სია არჩევანის გასაკეთებლად:

Testing W3.JS in HTML

1.12. ცხრილებზე მანიპულაციის ჩატარების უფრო რთული მაგალითი

```
<!DOCTYPE html>
<html>
<title>W3.JS</title>
<script src="https://www.w3schools.com/lib/w3.js"></script>
<body>

<h2>Testing W3.JS in HTML</h2>

<table id="id01" border="1">
 <tr>
  <th>კლიენტი </th>
  <th>ქალაქი</th>
  <th >თანხა</th>
 </tr>
 <tr w3-repeat="შემკვეთი">
  <td>{{კლიენტი}}</td>
  <td>{{ქალაქი}}</td>
  <td style="text-align:right">{{თანხა}}</td>
 </tr>
 <tr>
```

```
        <td></td>
        <td></td>
        <td><b>{{total}}</b></td>
      </tr>
    </table>

    <script>
    var myObject= {"შემკვეთი":[
    {"კლიენტი" : "აბაშიძე რევაზი","ქალაქი" : "ბათუმი","თანხა" :
"111"},
    {"კლიენტი" : "ბერიძე გიორგი","ქალაქი" : "ქუთაისი","თანხა" :
"222"},
    {"კლიენტი" : "ბიბილაშვილი თეონა","ქალაქი" :
"თბილისი","თანხა" : "333"},
    {"კლიენტი" : "გიორგაძე დავითი", "ქალაქი" : "გორი","თანხა" :
"77"},
    {"კლიენტი" : "დვალი ზურაბი","ქალაქი" :
"სამტრედია","თანხა" : "222"},
    {"კლიენტი" : "ველიაშვილი ზაზა","ქალაქი" :
"თელავი","თანხა" : "111"},
    {"კლიენტი" : "შავიშვილი თორნიკე","ქალაქი" :
"თბილისი","თანხა" : "111"},
    {"კლიენტი" : "ჩაჩანიძე გეგა","ქალაქი" : "თბილისი","თანხა" :
"222"}
    ]};
        myFunction(myObject);
    function myFunction(myObject) {
      var i, total = 0;
      var myArray = myObject.შემკვეთი;
      for (i = 0; i < myArray.length; i++) {
```

```
  total += Number(myArray[i].თანხა);
 }
 myObject.total = total.toFixed(2);
 w3.displayObject("id01", myObject);
}
</script>
</body>
</html>
```

ფაილის შესრულებაზე გაშვების შემდეგ მონიტორზე აისახება ასეთი სურათი:

Testing W3.JS in HTML

კლიენტი	ქალაქი	თანხა
აბაშიძე რევაზი	ბათუმი	111
ბერიძე გიორგი	ქუთაისი	222
ბიბილაშვილი თეონა	თბილისი	333
გიორგაძე დავითი	გორი	77
დვალი ზურაბი	სამტრედია	222
ველიაშვილი ზაზა	თელავი	111
შავიშვილი თორნიკე	თბილისი	111
ჩაჩანიძე გეგა	თბილისი	222
		1409.00

თავი 2. Vue.js

2.1. შესავალი

Vue.js არის Javascript-ენის საფუძველზე აგებული ფრეიმვორკი. შესაბამისად, ამ პროგრამული პროდუქტის შესწავლის დაწყებამდე უნდა აღნიშნოს, რომ აუცილებელია გაცნობილი ვიყოთ HTML, CSS და JavaScript პროგრამული საშუალებების დანიშნულებასა და შესაძლებლობებს.

Vue.js-ის პირველი სამუშაო ვერსია გამოქვეყნდა 2015 წელს, იგი განიხილებოდა ReactJS და Angular ფრეიმვორკების კონკურენტად. თუმცა, სპეციალისტების შეხედულებით, ამ სისტემებთან შედარებით Vue.Js არის უფრო კომპაქტური, სწრაფქმედი და შედარებით ადვილად ასათვისებელი პროგრამული პროდუქტი, ამასთან ერთად, სტრუქტურულად უფრო მოქნილი და ადვილად მასშტაბირებადი სხვადასხვა მოცულობის და სიმძლავრის დანართების (აპლიკაციების) შექმნისას.

Vue.js-ის შესაძლებლობებით სარგებლობა ხდება მარტივად, პროექტის სცენარში შემდეგი JS-ფაილის ჩართვით:

```
<script
  src="https://unpkg.com/vue@3/dist/vue.global.js">
</script>
```

გარდა ზემოთ აღნიშნულისა, Vue.js-ის ღირსებებია:

- HTML-ელემენტების შემცველობასა და Javascript-ენის ოპერატორებს შორის კავშირების უზრუნველყოფა არა პირდაპირი მითითების გზით, არამედ შაბლონებზე დაყრდნობით;
- ამასთან, ამ 2 სახის მონაცემების ერთმანეთთან ორმხრივი ბმების მეშვეობით დაკავშირებით;

- სცენარების შესრულების პროცესის ცენტრალიზებული მართვა.

Vue.Js-ის თითოეული გამოყენება (დანართი) იწყება Vue ფესვური ეგზემპლარიდან და ექვემდებარება Model-view-ViewModel (MVVM) შაბლონებზე დაყრდნობის შედეგად შემუშავებულ კონცეფციას.

ეს კი ნიშნავს, რომ Vue-ეგზემპლარის (ეგზემპლარების), როგორც შუამავალთა, დახმარებით ხდება მონიტორზე ასახულ მონაცემებსა და მათ წარმოდგენებს შორის კავშირის დამყარება და **რენდერინგი.**

რენდერინგი (ინგლ. rendering) კომპიუტერული გრაფიკის სფეროს კუთვნილი ტერმინია და ზოგადად გულისხმობს ობიექტისათვის შექმნილ კომპიუტერულ მოდელში პარამეტრების ცვლილებისას ამ ობიექტის მონიტორზე ასახვას კომპიუტერული პროგრამისვე მეშვეობით. ანუ, რენდერინგისას ხდება აღნიშნულ მოდელში შეტანილი ცვლილებების ვიზუალიზება.

რაც შეეხება უშუალოდ Vue.js-ფრეიმვორკს, მასში მიმდინარე რენდერინგის (ვიზუალიზების) თავისებურებაა ის, რომ ამ შემთხვევაშიც ხდება Javascript-ენაზე შექმნილი მოდელის დაკავშირება HTML-ელემენტებთან და ამ უკანასკნელების შემცველობის განსაზღვრა, ოღონდ არა ტრადიციული, არამედ ამ ენისათვის სპეციალურად შემუშავებული, გაფართოებული სინტაქსის საფუძველზე.

სწორედ ასეთი მიდგომა უზრუნველყოფს HTML-ელემენტებისათვის Javascript-ენაზე შექმნილ მოდელში ცვლილებების შეტანისას მონიტორზე მათ ადვილად და ავტომატურად ასახვას.

წინაპირობები Vue.js-ფრეიმვორკის შესწავლისათვის გახლავთ HTML, CSS Javascript პროგრამული პროდუქტების

დანიშნულებისა და შესაძლებლობების ცოდნა და გამოყენების უნარი.

შენიშვნა: Vue.js-ში შესაძლებელია გამოყენებული იქნეს კოდის დაწერის 2 ხერხი:

ერთი მათგანი ეყრდნობა API-პარამეტრებით მანიპულირებას, მეორე კი იყენებს API-კომპოზიციებს. კონცეფციები ორივესათვის მსგავსია. დეველოპერები თვლიან, რომ ფრეიმვორკის შესაძლებლობების შესწავლის დაწყება უმჯობესია პირველი გზით, როგორც უფრო ადვილად აღქმადით, და ამ სახელმძღვანელოშიც სწორედ იგი იქნება გამოყენებული.

ამრიგად, Vue.js აფართოებს HTML-ენის შესაძლებლობებს, უმატებს რა ამ ენას მეტ ფუნქციონალობას. კერძოდ, Vue.js-ის მეშვეობით ხდება Javascript ენაზე აღწერილ მოდელში არსებული მიმდინარე მდგომარეობის ავტომატური ასახვა მონიტორზე HTML-ელემენტების განახლების შედეგად და შესაძლებელია - პირიქითაც.

თუ ვიყენებთ „სუფთა“ Javascript-ს, მაშინ ჩვენ გვიხდება ამ ენის ოპერატორებსა და HTML-ელემენტებს შორის კავშირების დეტალურად აღწერა, ხოლო Vue.js-თი სარგებლობის შემთხვევაში კი მხოლოდ მიეთითება, რომ მათ შორის კავშირი არსებობს. დანარჩენი მანიპულაციების ჩატარებაზე ფრეიმვორკი თვითონვე იზრუნებს.

Vue.js ფრეიმვორკის შესაძლებლობების შემსწავლელი მაგალითების დემონსტრირებამდე, ხაზი უნდა გაესვას შემდეგ გარემოებას:

საყოველთაოდ ცნობილია, რომ ინფორმატიკის დარგის განვითარება მეტად სწრაფი ტემპებით ხდება, მასში კი ამ მხრივ ვებტექნოლოგიების მიმართულება გამოირჩევა და პროგრამული პროდუქტები და მიდგომები, რომლებიც ჯერ კიდევ, ასე ვთქვათ,

გუშინ პროგრამისტების არსენალში მყარად დამკვიდრებული ჩანდნენ, დღეს (და მით უფრო ხვალ) ახლებით იცვლება ანდა უნდა შეიცვალოს!

და გამონაკლისი არც პროგრამული პროდუქტი Vue.js ფრეიმვორკია.

მართლაც, ერთი წელიც არ არის გასული, რაც ამ პროდუქტის შესწავლა სპეციალიზებულ საიტებზე იწყებოდა ქვემოთ მოყვანილი სახის სცენარებზე დაყრდნობით.

2.2. Vue.js ფრეიმვორკის შესაძლებლობების მაგალითები

მაგალითი_1

```
<div id="app">
  <h1>{{ message }}</h1>
</div>

<script>
  var myObject = new Vue({
    el: '#app',
    data: {message: 'Hello Vue!'}
  })
</script>
```

როგორც უკვე იყო ნათქვამი, ფაილის შესრულებაზე გაშვების წინ სასურველია უფრო სრულყოფილი სახე მივცეთ ზემოთ მოყვანილ მაგალითს, რაც უნდა მოხდეს მასში დამატებითი ინფორმაციის ჩართვით.

გარდა ამისა, მოყვანილი სცენარის არსის უკეთ გაგების მიზნით, მასში ოპერატორებთან უნდა მიერთდეს კომენტარების სახით წარმოდგენილი შესაბამისი განმარტებებიც.

ასეთი ქმედებების შედეგად ლაბორატორიული მეცადინეობების ჩატარების პროცესში გაგვიადვილდება როგორც სცენარში თავიდანვე არსებული ოპერატორების, ასევე - ჩვენ მიერ წამოყენებული სხვადასხვა მოსაზრებების შესამოწმებლად სცენარში შეტანილი კორექტივების დანიშნულების გაგება.

მაგალითი_1_1

```
<!DOCTYPE html>
<html>
  <script rc="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
<body>
<center>
 <div id="app">
  <h1>{{ message }}</h1>
 </div>
</center>

<script>
 var app = new Vue({
   el: '#app',
  data: {message: 'Hello Vue!'}  // შეტყობინება მონიტორის შესაბამის უბანზე (html-ელემენტში) თავიდანვე აისახება და ავტომატურად - ამ ობიექტში  შემდგომში ცვლილებების შეტანისას.
 })
</script>

<!--
 HTML-ელემენტთან კ ა ვ შ ი რ ი ხორციელდება არა უ შ უ ა ლ ო დ, არამედ - დ ა ნ ა რ თ ი ს მეშვეობით.
```

ამასთან, გადასაგზავნი მონაცემების ყოველი ცვლილებისას იქმნება Vue-ობიექტის ახალი ეგზემპლარი.

ნებისმიერ დანართს დასაწყისში უნდა გააჩნდეს თუნდაც ერთი Vue-ეგზემპლარი.

ეს ეგზემპლარი მიებმება DOM-ში რომელიმე HTML-კვანძს და შ ე მ დ ე გ მართავს მას. მიბმა ხდება el-თვისებით.

HTML-ფაილისათვის შესაძლებელია შეიქმნას ნებისმიერი რაოდენობის ეგზემპლარები.

-->

</body>

</html>

ქვემოთ წარმოდგენილი არის წინა სცენარის მოდიფიცირებული „*მაგალითი_1_2“* და „*მაგალითი_1_3“* ვარიანტები:

მაგალითი_1_2

```
<!DOCTYPE html>
<html>
 <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js">
</script>
<body>
<center>
 <h1 id="app">  <!-- წინა ფაილთან შედარებით ამ უბანზე მოხდა სახეცვლილება. შედეგი იგივეა. -->
  {{ message }}
 </h1>
</center>
<script>
  var app = new Vue({
   el: '#app',
```

```
  data: {message: 'Hello Vue!'}  // შეტყობინება მონიტორის
შესაბამის უბანზე (html-ელემენტში) ეგრევე აისახება და ასევე
ავტომატურად ამ ობიექტში შემდგომ ცვლილებების შეტანისას.
 })
</script>
</body>
</html>
```

განვიხილოთ ისეთი სცენარების მაგალითები, რომლებშიც მოდელში არსებული მონაცემების (ან მასში შეტანილი ცვლილებების) მონიტორზე ასახვა ხდება <button> ღილაკზე ხელის დაჭერით ნებართვის გაცემის შედეგად:

მაგალითი_1_3

```
<!DOCTYPE html>
<html>
  <script
src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
<body>
<center>
 <h1>Vue.js</h1>
<h3>ამ ვარიანტში, წინათაგან განსხვავებით, დამატებულია
    button-ღილაკი (არჩევანის მოსაფიქრებლად).
</h3>
<div id="app">
 <h2> {{ message }} </h2>
</div>
<h2>
 <button onclick="myFunction()"><h2>Click!</h2>
  </button>
```

```
</h2>
</center>
<script>
  var myObject = new Vue({
    el: '#app',
    data: {message: 'Hello Vue!'}
  })
  function myFunction() {
    myObject.message = "John Doe";
  }
</script>
</body>
</html>
```

Vue.js ფრეიმვორკის უახლეს ვერსიაში, პროგრამისტებისათვის სამუშაოს უფრო გამარტივების მიზნით, მოხდა ზემოთ დემონსტრირებული კონცეფციის მოდიფიცირება.

კერძოდ, პირველივე სცენარში დასახული მიზანი ამჯერად ასეთი გზით მიიღწევა:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>My first Vue page</title>
</head>
<body>

  <div id="app">
    {{ message }}
  </div>
```

```
<!-- ქვემოთ script-უბანში ჩამოიტვირთება js-ფაილი Vue-
ფრეიმვორკის კოდის ინტერპრეტირებისათვის
-->
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
  <script>
    const app=Vue.createApp({ // Vue კოდით ვქმნით app სახელის
Vue-ეგზემპლარს.
      data() {
        return {
          message: "Hello World!"
        }
      }
    })
   app.mount('#app')  // app-ეგზემპლარი მიუერთდება id="app" მქონე
ელემენტს.
    </script>
</body>
</html>
```

მონიტორზე ასახული ინფორმაციის ვიზუალური მხარის სრულყოფისათვის კი ასეთი სცენარით:

```
<!DOCTYPE html>
<html>
<head>
  <title>My first Vue page</title>
  <style>
    #app {
      display: inline-block;
      padding: 10px;
      font-size: x-large;
      background-color: lightgreen;
```

```
   }
  </style>
</head>
<body>
<h1>Vue Example</h1>
<h3>შეტყობინება (მესიჯი) Vue-ობიექტის app-ეგზემპლარის
        message-თვისებიდან იგზავნება id="app" იდენტიფიკატორით
        კვალიფიცირებული html-ელემენტის {{message}} უბანში.
</h3>
<div id="app">
  {{ message }}
</div>
<!-- ქვემოთ script-უბანში ჩამოიტვირთება js-ფაილი Vue-
ფრეიმვორკის კოდის ინტერპრეტირებისათვის -- >
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
   data() {
    return {
     message: "Hello World!"JS
    }
   }
  })
    app.mount('#app')
</script>
</body>
</html>
```

მივაქციოთ ყურადღება - ზემოთ მოყვანილ სცენარებში შესაბამის უბანში ახალ მისამართზე დაყრდნობით შეცვლილია ადრინდელი ფაილი, რომლის სცენარში ჩამოტვირთითაც საშუალება გვეძლევა გამოვიყენოთ Vue.Js ფრეიმვორკის შესაძლებლობები:

```
<script
  src="https://unpkg.com/vue@3/dist/vue.global.js">
</script>
```

გარდა Vue.Js-ეგზემპლარიდან წამოღებული ტექსტის ამ თუ იმ HTML-ელემენტის {{message }} უბანში განთავსებისა, დასაშვებია {{ }} ფიგურულ ფრჩხილებში განვათავსოთ Javascript ენაზე დაწერილი გამოსახულებებიც.

ქვემოთ მოყვანილ მაგალითში პროგრამის შესრულებაზე გაშვებისას დამატებით გამოდის ინფორმაცია შემთხვევით არჩეული რიცხვის თაობაზეც წინასწარ განსაზღვრული დიაპაზონიდან:

```
<!DOCTYPE html>
<html>
<head>
  <title>My first Vue page</title>
  <style>
    #app {
      display: inline-block;
      padding: 10px;
      font-size: x-large;
      background-color: lightgreen;
    }
  </style>
</head>
<body>

  <h3>ამ მაგალითში, წინასგან განსხვავებით, მონიტორზე აისახება ინფორმაცია <br/>
    1 – 6 დიაპაზონში შემთხვევითობის წესით  არჩეული რიცხვის შესახებაც.
  </h3>
    <div id="app">
```

```
        {{ message }} <br>
        {{'Random number: ' + Math.ceil(Math.random()*6) }}
      </div>
      <script
src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
      <script>
        const app = Vue.createApp({
         data() {
          return {
           message: "Hello World!"
          }
         }
        })
          app.mount('#app')
      </script>
      </body>
      </html>
```

მომდევნო პარაგრაფებში გავეცნობით მონიტორზე გამოსაყვანი ინფორმაციის უფრო სრულყოფილად მართვისათვის განკუთვნილ საშუალებებს - **დირექტივებს.**

2.3. დირექტივები

ზემოთ განხილული მაგალითებისაგან განსხვავებით, უფრო რთული მიზნების რეალიზებისათვის Vue.js ფრეიმვორკი HTML-ელემენტებისათვის იყენებს სპეციფიკურ ატრიბუტებს, რომელთაც ეწოდებათ **დირექტივები**.

დირექტივების მეშვეობით ხდება HTML-ელემენტე-ბისათვის დანიშნული ატრიბუტების შესაძლებლობების გაფართოება და მათი მეშვეობით - ამ ელემენტებზე სხვადასხვა ოპერაციის ჩატარება გამოსახულებების დახმარებით.

გარეგნულად ასეთი ატრიბუტების განმასხვავებელი ნიშანია ის, რომ მათ წინ უძღვით v-პრეფიქსი.

დირექტივები 2 სახისაა:

1. ჩაშენებული;
2. მომხმარებლების მიერ ფორმირებული.

დირექტივებზე რეაგირების მომხდენ HTML-ელემენტებშიც Vue-ობიექტის ეგზემპლარიდან გადმოგზავნილი მონაცემების ჩასმა ხდება {{ }} ფიგურულ ფრჩხილებში.

Vue-დირექტივებზე დაყრდნობით მომხმარებლისათვის იქმნება დინამიკური და რეაქტიული სახის ინტერფეისები და, ამასთან ერთად, გაცილებით ადვილდება ასეთი ადაპტური ვებგვერდების შექმნა, ვიდრე ეს ხერხდება JavaScript-ენაში არსებული ტრადიციული მეთოდებით.

შენიშვნა: მდგომარეობა, რომლის ცვლილებასაც შეუძლია მონიტორზე გამოსახულების განახლება, იწოდება **რეაქტიულად.** ასეთი მდგომარეობის გამოცხადება შესაძლებელია reactive() ფუნქციით შექმნილი ობიექტებისათვის.

ქვემოთ ცხრილში მოყვანილია ის დირექტივები, რომლებიც გამოყენებულია წინამდებარე სახელმძღვანელოში:

დირექტივა	დანიშნულება
v-bind	HTML-ელემენტის ატრიბუტს აკავშირებს Vue-ეგზემპლარში გამოცხადებული ცვლადის შიგთავსთან.

დირექტივა	დანიშნულება
v-if	ქმნის HTML-ტეგებს რაიმე პირობის შესრულებისას. მასთან ერთად შესაძლებელია გამოყენებული იქნეს v-else-if და v-else დირექტივებიც.
v-show	განსაზღვრული პირობის შესრულებისას განაპირობებს მონიტორზე HTML-ელემენტის ჩვენებას.
v-for	Vue-ეგზემპლარში გამოცხადებული მასივიდან for-ციკლის მეშვეობით აფორმირებს ტეგების სიას.
v-on	HTML-ტეგში (ან ტეგზე) განხორციელებული ხდომილობისას მყარდება კავშირი JavaScript გამოსახულებასა ანდა Vue-ეგზემპლარის მეთოდთან.
v-model	დირექტივას იყენებენ <form>, <input> და <button> ტეგებში. მისი მეშვეობით შეტანის ელემენტსა და Vue-ეგზემპლარში მონაცემის თვისებას შორის

დირექტივა	დანიშნულება
	შორის ხორციელდება ორმხრივი მიბმა.

ვაჩვენოთ ზემოთ მოყვანილი დირექტივების გამოყენების მაგალითები.

დავიწყოთ v-bind დირექტივის დანიშნულების შესწავლით (Vue.Js ფრეიმვორკის წინა ვერსიიდან).

ამ დირექტივის მეშვეობით title ელემენტი ებმება Vue-ობიექტის ეგზემპლარში message-თვისებას და მასზე შესაბამისი ქმედებისას რეაგირებს ამ თვისებაში მომხდარ ცვლილებებზე:

```
<script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js">
</script>
<h2 align='center' id="app-2">
  <span v-bind:title="message"> ამ უბანზე კურსორი
     დააყოვნეთ 2-3 წამით და დაინახავთ, რომ title
     ატრიბუტის მნ-ბა განახლდება!
  </span>
</h2>
<script>
 var app2 = new Vue({
   el: '#app-2',
   data: {
    message: 'ფურცელი ჩაიტვირთა: ' + new
Date().toLocaleString()
   }
 })
```

/*

აქ სიახლეა v-bind ატრიბუტის შემოტანა. ჩვეულებრივი ატრიბუტისგან მას განასხვავებს შემდეგი ნიშანი - v-პრეფიქსი, რომელიც მასთან კავშირში მყოფ ელემენტს ანიჭებს რეაქტიულობის უნარს.

მოცემულ შემთხვევაში title ელემენტი მასზე განხორციელებული შესაბამისი ქმედებისას რეაგირებს Vue-ობიექტის ეგზემპლარში არსებულ message-თვისებაში მომხდარ ცვლილებებზე.

*/

</script>

ამრიგად, Vue-ობიექტის მიბმა HTML-ელემენტთან განაპირობებს მისი შიგთავსის მოდიფიცირებას ამ Vue-ობიექტში ცვლილებების შეტანისას.

მოვიყვანოთ v-bind დირექტივით სარგებლობის ისეთი მაგალითიც, რომელშიც ბროუზერს ევალება გაარკვიოს, თუ Vue-ობიექტის ეგზემპლარში არსებულ რომელ კლასს უნდა მიუერთოს ელემენტი:

```
<!DOCTYPE html>
<html>
<head>
 <title>My first Vue page</title>
 <style>
  div {
   margin: 10px;
   padding: 10px;
   border: solid black 1px;
   display: inline-block;
```

```
  }
  .pinkBG{
    background-color: lightpink;
  }
 </style>
</head>
<body>
<h1>'v-bind' Example</h1>
<p>ამ მაგალითში ბროუზერი კლასს ეძებს Vue-ეგზემპლარში.</p>
<div id="app">
  <p> ბროუზერი კლასს პოულობს Vue-ეგზემპლარში და ამ div-
      ელემენტში ყველგან იქნება შესაძლებელი ეგზემპლარში
      არსებული ინფორმაციებით სარგებლობა.
  </p>
  <div v-bind:class="vueClass">ეს ელემენტი კავშირშია "pinkBG"-
      კლასთან.
    </div>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
 const app = Vue.createApp({
  data() {
   return {
    vueClass: "pinkBG"
   }
  }
 })
 app.mount('#app')
</script>
```

```
</body>
</html>
```

ზემოთ მოყვანილ მაგალითებში HTMI-ელემენტზე მანიპულაცია ხორციელდება არა {{ }} ფრჩხილების შიგთავსის ცვლილებით, არამედ გამოყენებულია განსხვავებული მიდგომა, კერძოდ, ეს HTMI-ელემენტი ცხადდება Vue-ეგზემპლარში არსებულ კლასთან მიბმულად (რის შედეგადაც, მაგალითად, ბოლო შემთხვევაში იცვლება HTML-ელემენტის დიზაინი).

შენიშვნა: მსგავსი დანიშნულების ამოცანების გადაწყვეტა, ცხადია, უფრო მარტივი გზებითაც არის შესაძლებელი, კერძოდ, Vue-კოდის გამოყენების გარეშე, მაგრამ ასეთი მიდგომები მნიშვნელოვან ეფექტს იძლევა მაშინ, როდესაც საჭირო ხდება გაცილებით რთული და უფრო მოცულობითი საიტების ადაპტური გვერდების სახით წარმოდგენა.

2.4. პირობის შესრულებაზე დამოკიდებული რენდერინგი (v-if, v-else-if და v-else დირექტივები)

ცნობილია, რომ JavaScript-ენა თავისი ოპერატორების მეშვეობით იძლევა HTM-ელემენტებისა და მათი ატრიბუტების შექმნის შესაძლებლობას, მაგრამ ეს პროცესები გაცილებით უფრო მარტივად ხორციელდება v-if, v-else-if და v-else დირექტივების დახმარებით.

ჯერ ვაჩვენოთ v-if და v-else დირექტივების ერთმანეთისაგან განცალკევებით გამოყენების მაგალითი:

მაგალითი_1_1

```
<html>
<head>
```

```
 <title>Typewriters</title>
 <style>
  #app {
    border: dashed blue 2px;
    width: 99px;
    padding-left: 15px;
    font-weight: bold;
    background-color: lightgreen;
  }
 </style>
</head>
<body>
<h2 align="center">წარმოვადგენთ 'v-if' და 'v-else'დირექტივების
            გამოყენების მაგალითს.</h2>
<h3 align="center">Vue-ეგზემპლარში 'typewritersInStock'-ისათვის
            «true» მნიშვნელობა შეცვალეთ «false»-თი და სცენარი
            ხელახლა გაუშვით შესრულებაზე!
</h3>
<div id="app">
 <p v-if="typewritersInStock">
  გაყიდვაშია
 </p>
 <p v-else>
  შემოგვიარეთ ხვალ!
 </p>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
 const app = Vue.createApp({
```

```
  data() {
   return {
     typewritersInStock: true
   }
  }
 })
 app.mount('#app')
</script>
</body>
</html>
```

მომდევნო სცენარში კი წარმოდგენილია ამ სამივე დირექტივის გამოყენების მაგალითი:

```
<!DOCTYPE html>
<html>
<head>
  <title>My first Vue page</title>
  <style>
    #app {
      border: dashed black 1px;
      width: 130px;
      padding-left: 20px;
      font-weight: bold;
      background-color: lightgreen;
    }
  </style>
</head>
<body>
<h1>Example with 'v-if', 'v-else-if' and 'v-else'</h1>
```

```
<h3 align="center"> Vue-ეგზემპლარში 'typewriterCount-ისათვის
        მნიშვნელობა შეცვალეთ ჯერ 0-ისა და შემდეგ 5-ის ტოლი
        სიდიდეებით და სცენარი ხელახლა გაუშვით შესრულებაზე!
</h3>
<h2>We sell typewriters</h2>
<img src="img_typewriter.png" alt="typewriter" width="202"
            height="135">
<div id="app">
 <p v-if="typewriterCount>3">
        გაყიდვაშია
 </p>
 <p v-else-if="typewriterCount>0">
  გვაქვს მცირე რაოდენობით!
 </p>
 <p v-else> შემოგვიარეთ ერთი კვირის შემდეგ! </p>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
 const app = Vue.createApp({
  data() {
   return {
    typewriterCount: 3
   }
  }
 })
 app.mount('#app')
</script>
</body>
</html>
```

ამავე სცენარში ნაჩვენებია, თუ როგორი გზით არის შესაძლებელი მონიტორზე ნახატის (გრაფიკის) გამოყვანაც:

სურ. 2.1

ამგვარი სახის ამოცანების გადასაწყვეტად Vue.js_ში გათვალისწინებულია სხვა მიდგომებიც.

განვიხილოთ ისინი:

v-if დირექტივისათვის Vue-ფრეიმვორკში დასაშვები არის პარამეტრის დამბრუნებული ფუნქციისათვის «true» ან «false» მნიშვნელობების გამოყენებაც.

მაგალითად, თუ რაიმე ტექსტი თავის თავში შეიცავს ტერმინ „პიცას“ (მოვთავსოთ ეს ტექსტი, დავუშვათ, <h3>…. </h3> ტეგებში), JavaScript-ენისავე კუთვნილი includes() მეთოდით შესაძლებელი ხდება გავიგოთ, ამ ელემენტში მოთავსებული ტექსტი შეიცავს თუ არა ჩვენთვის საინტერესო ტერმინს.

```
<!DOCTYPE html>
<html>
<head>
  <title>'v-if' დირექტივის დახმარებით შეკვეთის
განხორციელება</title>
  <style>
    #app {
      border: dashed black 1px;
      width: 130px;
      padding: 0 20px 20px 20px;
      font-weight: bold;
      background-color: lightgreen;
    }
    img {
```

```
    width: 100%;
   }
  </style>
</head>
<body>
<h2> ტექსტის შემოწმების მეშვეობით არჩევანის გაკეთება</h2>
<h2> ამ მაგალითში 'v-if' დირექტივა შედარების ოპერატორის
ნაცვლად იყენებს 'includes()' მეთოდს.
</h2>
    <h3>Vue-ს ეგზემპლარის 'text'-თვისების შემცველობიდან
    ამოაგდეთ ტერმინი 'პიცა' და სცენარი ხელახლა გაუშვით
    შესრულებაზე.
</h3>
<div id="app">
  <div v-if="text.includes('პიცა')">
    <p> text-თვისება (ანუ მენიუ) შეიცავს ტერმინ 'პიცას'. </p>
  </div>
  <p v-else>ტერმინი 'პიცა' ტექსტში (მენიუში) ვერ იქნა ნაპოვნი.</p>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
   data() {
    return {
      text: 'მწვადი, ჩიხირთმა, სალათა, პიცა, ნაყინი, ყავა.'
    }
   }
  })
    app.mount('#app')
</script>

</body>
</html>
```

სცენარში შევიტანოთ გრაფიკაც:

```
<!DOCTYPE html><html>
<head>
  <title>'v-if' დირექტივის დახმარებით შეკვეთის
განხორციელება</title>
  <style>
    #app {
      border: dashed black 1px;
      width: 130px;
      padding: 0 20px 20px 20px;
      font-weight: bold;
      background-color: lightgreen;
    }
    img {
      width: 100%;
    }
  </style>
</head>
<body>
 <center>
  <h2> ტექსტის შემოწმების მეშვეობით არჩევანის გაკეთება</h2>
  <h2> ამ მაგალითში 'v-if' დირექტივა შედარების ოპერატორის
ნაცვლად იყენებს 'includes()' მეთოდს.</h2>
  <h3>Vue-ს ეგზემპლარის 'text'-თვისების შემცველობიდან
ამოაგდეთ ტერმინი 'პიცა' და სცენარი ხელახლა გაუშვით
შესრულებაზე.
  </h3>
 </center>
<div id="app">
  <div v-if="text.includes('პიცა')">
```

```
   <p> text-თვისება შეიცავს ტერმინ 'პიცას'. </p>
   <img src="პიცა.png">
  </div>
  <p v-else>ტერმინი 'პიცა' 'text'-თვისებაში (ანუ მენიუში) ვერ იქნა ნაპოვნი.</p>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
   data() {
    return {
      text: 'მწვადი, ჩიხირთმა, პიცა, სალათა, ნაყინი, ყავა.'
    }
   }
  })
   app.mount('#app')
</script>
</body>
</html>
```

მენიუში საძიებო ელემენტის მოძებნის შემთხვევაში მონიტორზე აისახება მისი გრაფიკული გამოსახულებაც.

ამრიგად, როგორც ზემოთ მოყვანილი მაგალითებიდან ჩანს, Vue-ფრეიმვორკის დახმარებით შესაძლებელია დაიწეროს ისეთი სცენარები, რომელთა მეშვეობით გაცილებით უფრო მარტივდება მონიტორზე HTML-ელემენტების ასახვა და მათი მართვა, ვიდრე მაშინ, როდესაც სცენარში გამოიყენება მხოლოდ ტრადიციული JavaScript ენისათვის კუთვნილი ოპერატორები.

2.5. v-show დირექტივა

v-show დირექტივის მეშვეობით შესაძლებელი ხდება შესაბამისი სახის პირობის შემოწმების შედეგად მონიტორზე ელემენტების ასახვა ან დამალვა. ამასთან, ეს პირობა ჩაიწერება უშუალოდ HTML-ელემენტის საწყისი ტეგის ატრიბუტში.

მოვიყვანოთ მაგალითი:

```
<div v-show="showDiv">
    ეს div ელემენტი შესაძლებელია როგორც დავმალოთ,
    ასევე ხელახლა ავსახოთ მონიტორზე.
</div>
```

ზემოთ მოყვანილ ფრაგმენტში «showDiv» წარმოადგენს Vue-ფრეიმვორკის ეგზემპლარის თვისებას - ლოგიკური ტიპის მონაცემს, რომელსაც შეუძლია მიიღოს «true» ან «false» მნიშვნელობა.

პირველ შემთხვევაში მონიტორზე აისახება ელემენტი, მეორეში - არა.

მოვიყვანოთ მაგალითი:

```
<div v-show="showDiv">This div tag can be hidden</div>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
  <script>
    const app = Vue.createApp({
      data() {
        return {
          showDiv: true
        }
```

```
  }
 })
 app.mount('#app')
</script>
```

აქვე აღვნიშნავთ, რომ v-if დირექტივის მეშვეობით ელემენტი იქმნება შესაბამისი პირობის შესრულების შემთხვევაში, ხოლო v-show დირექტივა კი ახდენს უკვე შექმნილი ელემენტის მონიტორზე ასახვა-დამალვას.

მიჩნეულია, რომ v-show დირექტივის გამოყენების შემთხვევაში პროცესი უფრო სწრაფად მიმდინარეობს და მომხმარებელთან ურთიერთობაც მარტივდება, v-if დირექტივის უპირატესობა კი ის გახლავთ, რომ მასთან ერთად დასაშვებია v-else-if და v-else დირექტივების გამოყენებაც.

ქვემოთ მოყვანილ მაგალითში ორი სხვადასხვა -ელემენტისათვის ცალ-ცალკე არის გამოყენებული v-show და v-if დირექტივები Vue-ეგზემპლარის ერთი და იმავე თვისებისათვის. ამასთან, v-show დირექტივის შესრულებისას არ ნადგურდება შესაბამისი -ელემენტი, მხოლოდ ხდება ის, რომ მასთან დაკავშირებულ CSS-ეფექტებს მნიშვნელობად განისაზღვრებათ «none».

რაც შეეხება v-if დირექტივის შესრულების შედეგს, ანუ მასთან დაკავშირებულ ელემენტზე ზემოქმედებას, ეს ელემენტი ნადგურდება.

რჩევა: ორი ელემენტი მხოლოდ მაშინ ასახეთ მონიტორზე, თუ showDiv თვისებისათვის მნიშვნელობად დანიშნულია «true». «false»-ს არჩევის შემთხვევაში მონიტორზე შევამჩნევთ, რომ v-if დირექტივის შესრულებამ გამოიწვია მასთან მიბმული -ის **განადგურება,** v-show-მა კი გავლენა მოახდინა

ასეთივე -ელემენტის შიგთავსის მხოლოდ ვიზუალურ მხარეზე (CSS-ისთვის მნიშვნელობად «none»-ის დანიშვნის გამო).

ჯერ ყურადღება მივაქციოთ სცენარში შესატან მთავარ ცვლილებებს, შემდეგ კი ვიხილოთ მთელი ფაილი, შესრულებაზე გავუშვათ იგი და გავითვალისწინოთ მონიტორზე გამოსული მითითებები:

```
<div id="app">
  <div v-show="showDiv">Div tag with v-show</div>
  <div v-if="showDiv">Div tag with v-if</div>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        showDiv: true
      }
    }
  })
  app.mount('#app')
</script>
```

მაგალითი_1

```
<!DOCTYPE html>
<html>
```

```
<head>
  <title>v-show vs. v-if</title>
  <style>
    #app {
      border: black dashed 1px;
      width: 80%;
      padding: 0 20px 20px 20px;
    }
    #app div {
      padding: 20px;
      margin: 10px;
      display: inline-block;
      font-weight: bold;
      border: solid black 1px;
    }
    #app p {
      padding: 10px;
      background-color: rgb(200, 246, 200);
    }
  </style>
</head>
<body>
<h1>Example: v-show vs. v-if</h1>
<div id="app">
<p> showDiv-ს მნიშვნელობად განუსაზღვრეთ false და სცენარი
    ხელახლა გაუშვით შესრულებაზე. თაგვის მარჯვენა ღილაკით
    დააწკაპუნეთ მწვანედ შეფერილ ამ p-ელემენტზე, ამოირჩიეთ
    პუნქტი 'Inspect' (შემოწმდეს) ანდა (შემოწმდეს ელემენტი) და
    დავინახავთ, რომ v-show-დირექტივის მქონე div ელემენტი
```

კვლავ არსებობს, მაგრამ მისთვის შეიცვალა მონიტორზე ასახვის CSS-თვისებები მათთვის 'none' მნიშვნელობის მინიჭების გამო, ხოლო v-show-დირექტივის მქონე v-if ელემენტი განადგურებულია.

```
</p>
  <div v-show="showDiv">Div-ელემენტი, მართული v-show-დირექტივით</div>
  <div v-if="showDiv">Div-ელემენტი, მართული v-if-დირექტივით</div></div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
 const app = Vue.createApp({
  data() {
   return {
    showDiv: true
   }
  }
 })
app.mount('#app')
</script>
</body>
</html>
```

2.6. v-for დირექტივა

"სუფთა" Javascript-ის ოპერატორებით სარგებლობისას, კერძოდ, for-ციკლში მასივის დამუშავებისას, შესაძლებელია ამ მასივის ელემენტების ამა თუ იმ სახით ასახვა მონიტორზე, მაგრამ, ამის შემდეგ მასივში შეტანილ ცვლილებებზე მონიტორი აღარ რეაგირებს.

სწორედ ამ ნაკლის გამოსწორება არის შესაძლებელი v-for დირექტივის მეშვეობით.

განვიხილოთ ეს მიდგომა მონიტორზე ასახული ამ სიის მაგალითზე:

```
<ol>
  <li v-for="x in manyFoods">{{ x }}</li>
</ol>
```

მაგალითი_1

```
<!DOCTYPE html>
<html>
<head>
  <title>My first Vue page</title>
  <style>
    #app > div {
      border: solid black 1px;
      width: 80%;
      padding: 10px;
      display: flex;
      flex-wrap: wrap;
    }
    img {
      width: 70px;
      margin: 10px;
    }
  </style>
</head>
<body>
<center>
```

```
<h2> ამ სცენარში მოყვანილია li-ტეგების შესაქმნელად «v-for» დირექტივის გამოყენების მაგალითი.
</h2>
<p>'v-for' დირექტივით ფორმირებული თითოეული 'li'-ტეგი შეიცავს ამა თუ იმ პროდუქტის დასახელებას, გადმოკოპირებულს Vue-ეგზემპლარში არსებული მასივიდან.
</p>
</center>
<div id="app">
  <ol>
    <li v-for="x in manyFoods">{{ x }}</li>
  </ol>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
   data() {
    return {
      manyFoods: [
        'Burrito',
        'Salad',
        'Cake',
        'Soup',
        'Fish',
        'Pizza',
        'Rice'
      ]
    }
   }
```

```
 })
   app.mount('#app')
</script>
</body>
</html>
```

დავალება: გაეცანით ქვემოთ მოყვანილი ორი ფაილის შემცველობას, გააანალიზეთ მათი დანიშნულება, მიღებული შედეგები და მათ ბაზაზე დაწერეთ მსგავსი ამოცანების გადაწყვეტისათვის საჭირო კოდები სხვადასხვა საგნობრივი სფეროსათვის, როგორიც არის, მაგალითად, თქვენი ჯგუფი, ბიბლიოთეკა, ფეხბურთის ან სხვა სპორტის სახეობებში გუნდი, ქართული ღვინოები, სასწავლო საგნების ჩამონათვალი, ...)

მაგალითი_1

```
<!DOCTYPE html>

<html>
<head>
 <title>My first Vue page</title>
 <style>
   #app > div {
    border: solid black 1px;
    width: 80%;
    padding: 10px;
    display: flex;
    flex-wrap: wrap;
    justify-content: space-around;
   }
   img {
```

```
    width: 70px;
    margin: 10px;
   }
  </style>
</head>
<body>

<h1>Example 'v-for' to create images</h1>
<p>The 'v-for' directive is used to create images based on the 'manyFoods'
                array in the Vue instance.</p>

<div id="app">
  <div>
   <img v-for="x in manyFoods" v-bind:src="x">
  </div>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

<script>
  const app = Vue.createApp({
   data() {
    return {
     manyFoods: [
      'img_burrito.svg',
      'img_salad.svg',
      'img_cake.svg',
      'img_soup.svg',
      'img_fish.svg',
      'img_pizza.svg',
      'img_rice.svg'
```

```
    ]
   }
  }
 })
  app.mount('#app')
</script>
</body>
</html>
```

მაგალითი_2

```
<!DOCTYPE html>
<html>
<head>
 <title>Vue page</title>
 <style>
  #app > div {
   border: solid black 1px;
   width: 80%;
   padding: 10px;
   display: flex;
   flex-wrap: wrap;
   justify-content: space-around;
  }
  figure {
   width: 80px;
   padding: 10px;
   margin: 10px;
   background-color: lightskyblue;
   border-radius: 5px;
```

```
    }
    figcaption {
      text-align: center;
    }
    img {
      width: 100%;
    }
  </style>
</head>
<body>

<h1>Example 'v-for' to create images and text</h1>
<p>The 'v-for' directive is used to create images and text based on the
                    'manyFoods' array in the Vue instance.</p>
<div id="app">
  <div>
    <figure v-for="x in manyFoods">
      <img v-bind:src="x.url">
      <figcaption>{{ x.name }}</figcaption>
    </figure>
  </div>
 </div>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

<script>
  const app = Vue.createApp({
   data() {
    return {
      manyFoods: [
       {name: 'Burrito', url: 'img_burrito.svg'},
```

```
      {name: 'Salad', url: 'img_salad.svg'},
      {name: 'Cake', url: 'img_cake.svg'},
      {name: 'Soup', url: 'img_soup.svg'},
      {name: 'Fish', url: 'img_fish.svg'},
      {name: 'Pizza', url: 'img_pizza.svg'},
      {name: 'Rice', url: 'img_rice.svg'}
    ]
   }
  }
 })
   app.mount('#app')
</script>

</body>
</html>
```

შემდეგ, v-for დირექტივის დახმარებით შესაძლებელია მასივში არსებული ობიექტების როგორც ინდექსის, ასევე, მენიუს თითოეული პუნქტისათვის ამ ობიექტის სახელის და მაილუსტრირებელი გრაფიკული ფაილების URL-მისამართების მონიტორზე გამოყვანა.

განვიხილოთ მაგალითი:

```
<!DOCTYPE html>
<html>
<head>
 <title>Vue page</title>
 <style>
  #app > div {
   display: inline-block;
   border: dashed black 1px;
```

```
    padding: 10px;
    background-color: lightgreen;
   }
   #app p {
    font-weight: bold;
    margin: 5px 0;
   }
  </style>
</head>
<body>
<p> v-for დირექტივა გამოიყენება «manyFoods» მასივში არსებული
      ობიექტების როგორც ინდექსის, ასევე - მენიუს თითოეული
      პუნქტისათვის სახელისა და URL-მისამართის
      დასაფიქსირებლად.
</p>
<div id="app">
  <div>
   <p v-for="(x, index) in manyFoods">
    {{ index }}: "{{ x.name }}", url: "{{ x.url }}" <br>
   </p>
  </div>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
   data() {
    return {
     manyFoods: [
      {name: 'Burrito', url: 'img_burrito.svg'},
```

```
      {name: 'Salad', url: 'img_salad.svg'},
      {name: 'Cake', url: 'img_cake.svg'},
      {name: 'Soup', url: 'img_soup.svg'},
      {name: 'Fish', url: 'img_fish.svg'},
      {name: 'Pizza', url: 'img_pizza.svg'},
      {name: 'Rice', url: 'img_rice.svg'}
    ]
  }
 }
})
   app.mount('#app')
</script>
</body>
</html>
```

2.7. Vue-ფრეიმვორკში ხდომილობები (Events) და მათი დამუშავება

Javascript-ენის სცენარების შესრულებისას, როგორც წესი, ადგილი აქვს როგორც მომხმარებლის, ასევე სისტემის მიერ ინსპირირებულ მოვლენებს - **ხდომილობებს.**

მათზე რეაგირებენ **ხდომილობათა დამმუშავებლები** - კონკრეტული ხდომილობის დამუშავებისათვის გათვალისწინებული კოდები.

Vue-ფრეიმვორკში ამა თუ იმ ხდომილობაზე შესაბამისი რეაგირება ხორციელდება **v-on დირექტივის** მეშვეობით.

v-on დირექტივასთან კავშირდება ხდომილების დასახელება, რათა ბროუზერს ეცნობოს, რომლის მოლოდინში იყოს ის (რომელს „უსმენდეს“) და, იმისდა მიხედვით, იქნება ეს,

მაგალითად, თაგვის მარცხენა თუ მარჯვენა ღილაკზე დაწკაპუნება, რომელი წესების მიხედვით იმოქმედოს.

ხდომილობების დასამუშავებლად **Vue**-ფრეიმვორკში იყენებენ **v-on მეთოდებსაც.**

მეთოდში განთავსდება შედარებით უფრო რთული კოდი, რომლის გაწყობაც ხდება პროცესის უკეთ მართვისათვის. მასზე მიმართვა ხდება HTML-ელემენტის ატრიბუტიდან, მაგალითად, ამგვარად:

```
<p v-on:click="changeColor">დააწკაპუნეთ აქ!</p>
```

ამავე მიზნით, ასევე გამოიყენება **v-on მოდიფიკატორებიც.**

მოდიფიკატორების შესაძლებელობებზე დაყრდნობით შესაძლებელი ხდება, კიდევ უფრო დეტალურად აღიწეროს, თუ როგორ უნდა განხორციელდეს კოდის მოდიფიცირება და შესაბამისად, ხდომილობებზე რეაგირება დასახული მიზნის უფრო მარტივად მისაღწევად.

ამრიგად, **Vue**-ფრეიმვორკში ხდომილობების ინსპირი-რებისა და მათზე რეაგირებისათვის გათვალისწინებულია 3 შესაძლებლობა:

1. დირექტივები,
2. მეთოდები,
3. მოდიფიკატორები

2.8. v-on დირექტივა

მომხმარებლის მიერ v-on დირექტივის გამოყენება - ამა თუ იმ ხდომილობისათვის გათვალისწინებული კოდის გაშვება - ხდება <button>-ღილაკზე თაგვით დაწკაპუნებისას.

ვაჩვენოთ ამ მიდგომის გამოყენების მაგალითი:

```
<!DOCTYPE html>
```

```
<html>
<head>
  <title>Count Moose</title>
  <style>
    #app {
      border: dashed black 1px;
      width: 200px;
      padding: 20px;
    }
    img {
      width: 100%;
    }
  </style>
</head>
<body>
 <center>
 <h3>ოპერატიულად დავითვალოთ "button"-ელემენტზე
       თაგვით დაწკაპუნებების რიცხვი და ავსახოთ ის
       მონიტორზე!
 </h3>
 <div id="app">
<!-- ქვემოთ ელემენტში მოთავსებულ ფიგურულ
ფრჩხილებში აისახება <button>-ღილაკზე დაწკაპუნებათა რიცხვი,
რომლის საწყის ტეგში არის განთავსებული არის v-on დირექტივა
და იქვე მითითებულია  მასთან დაკავშირებული click ხდომილობა
-->
   <h4>{{ "თქვენ button-ღილაკზე დააწკაპუნეთ " + count +"-
ჯერ!" }}</h4>
```

```
        <button v-on:click="count++"><b>დააწკაპუნეთ!</b></button>
      </div>
      <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
      <script>
        const app = Vue.createApp({
          data() {
            return {
              count: 0
            }
          }
        })
        app.mount('#app')
      </script>
      </body>
      </html>
```

შევნიშნავთ, რომ ზემოთ მოყვანილ მაგალითში დასახული ამოცანის შესასრულებლად თუ მხოლოდ „სუფთა“ JavaScript-ის ოპერატორებით ვისარგებლებდით, ასეთ შემთხვევაში ღილაკზე დაწკაპუნებათა რიცხვის მონიტორზე ოპერატიულად ჩვენებისათვის საჭირო იქნებოდა კოდში კიდევ ერთი სტრიქონის დამატება.

ამ მაგალითში მივმართავდით <button>-ღილაკზე თაგვის ღილაკზე დაწკაპუნებით გამოწვეულ ხდომილობას, საერთოდ კი, Javascript-ენასა და შესაბამისად, Vue.js ფრეიმვორკში მრავალი სახის ხდომილობა არსებობს.

ქვემოთ ცხრილის სახით წარმოდგენილია რამდენიმე ყველაზე უფრო მეტად გამოყენებადი ხდომილობები, ხოლო მათი თითქმის სრული სია განთავსებულია **დანართი_1**-ში:

Event	Occurs When	Belongs To
click	ელემენტზე დაწკაპუნება	UiEvent, Event
keypress	კლავიშზე ხელის დაჭერა	MouseEvent
mouseover	კურსორის (თაგვის მაჩვენებლის) დაყენება	MouseEvent
copy	ელემენტის შემცველობის კოპირება	ClipboardEvent
cut	ელემენტის შემცველობის ამოჭრა	ClipboardEvent
paste	ელემენტში კონტენტის ჩასმა	MouseEvent
input	ელემენტს მომხმარებლისგან მიეწოდება ინფორმაცია	DragEvent
submit	ფორმა გაიგზავნა	ClipboardEvent

2.9. მეთოდები

Vue ფრეიმვორკში მეთოდი წარმოდგენილი არის ფრეიმვორკის ეგზემპლარის თვისებაში ჩაწერილი ფუნქციის სახით და მისი გამოყენება შესაძლებელია არა მხოლოდ ხდომილობებზე რეაგირებისათვის.

ერთ-ერთ წინა მაგალითში ჩვენ უკვე გამოყენებული გვქონდა Vue-ეგზემპლარის «data» თვისება.

ამჯერად ამ თვისებით ვისარგებლოთ შემოვლითი გზით, როდესაც მისდამი მიმართვა ხდება დირექტივიდან გამომახებული მეთოდიდან.

აქვე ხაზს ვუსვამთ იმ გარემოებას, რომ თვითონ **მეთოდიც (მეთოდები) Vue-ეგზემპლარის თვისებად** ითვლება, უფრო ზუსტად მის ისეთ ნაირსახეობად, რომელიც წარმოადგენს ფუნქციას (ან ფუნქციებს).

ამასთან, ხშირ შემთხვევაში ფუნქცია Vue-ფრეიმვორკის იმავე ეგზემპლარში დაფიქსირებულ თვისებებს იყენებს, ოღონდ მათდამი მიმართვისას თვისების სახელწოდებას წინ წარუმძღვარებს this პრეფიქსს.

მაგალითი:

```
<!DOCTYPE html>
<html>
<head>
 <title>Click To Run Method</title>
 <style>
  #app {
   border: black dashed 1px;
   width: 200px;
   padding: 0 20px 20px 20px;
  }
```

```
    #app > div {
      width: 140px;
      height: 60px;
      background-color: lightgreen;
      padding: 20px;
      font-weight: bold;
      font-family: 'Courier New', Courier, monospace;
    }
  </style>
</head>
<body>
<h1> მეთოდის გამოძახება</h1>
<div id="app">
  <p>მეთოდის გამოსაძახებლად დააწკაპუნეთ ქვემოთ არეზე:</p>
  <div v-on:click="changeText">
   {{ text }}
  </div>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        text: ''
      }
              },
    methods: {
                changeText() {
        this.text = 'Hello World!'
      }
    }
  })
      app.mount('#app')
```

```
</script>
</body>
</html>
```

მონიტორზე გამოსული ინსტრუქციის შესრულების შედეგი იქნება Vue-ფრეიმვორკის ეგზემპლარში არსებული მეთოდის გამოძახება, რომელიც თავის მხრივ მიმართავს ამავე ეგზემპლარში არსებულ ერთ-ერთ თვისებას და ეკრანზე გამოიტანს ამ თვისებისათვის მნიშვნელობად მინიჭებულ ტექსტს.

განვიხილოთ უფრო რთული და შესაბამისად, მომხმარებლისათვის უფრო მეტი შესაძლებლობების მიწოდების მაგალითი.

ქვემოთ წარმოდგენილ სცენარში ელემენტში მითითებულ v-on დირექტივის შესრულებაზე გაშვებით და შესაბამისად, mousemove-ხდომილობის გამოწვევით, გამოიძახება mousePos მეთოდი.

შედეგად, ადგილზე გადმოიცემა **ხდომილობის ობიექტის** შესახებ ინფორმაცია, კერძოდ, თუ რა თვისებებით ხასიათდება ეს მიზნობრივი ობიექტი, რომელი სახის ხდომილობასთან გვაქვს საქმე, როგორია მოცემულ მომენტში სცენარის შესასრულებლად საჭირო პარამეტრების მნიშვნელობები, კერძოდ, ოპერატიული მონაცემები თაგვის მაჩვენებლის პოზიციების შესახებ.

ადრეც აღვნიშნავდით და კიდევ ერთხელ **მივაქციოთ ყურადღება**, რომ მეთოდიდან ფრეიმვორკის მოცემული ეგზემპლარის შიგნით არსებულ თვისებისადმი მიმართვისას ამ თვისების სახელს წინ უნდა წარემძღვაროს This-პრეფიქსი.

მაგალითი

```
<!DOCTYPE html>
<html>
<head>
  <title>თაგვის მაჩვენებლის პოზიციის დაფიქსირება</title>
```

```
  <style>
    #app {
      border: black dashed 1px;
      width: 200px;
      padding: 0 20px 20px 20px;
    }
    #app > div {
      width: 160px;
      height: 80px;
      background-color: lightcoral;
      padding: 20px;
    }
    #app span {
      font-weight: bold;
      font-family: 'Courier New', Courier, monospace;
    }
  </style>
</head>
<body>
<h1></h1>
<div id="app">
  <p>თაგვის მაჩვენებელი მოათავსეთ ქვემოთ არეზე და ამოძრავეთ
მასზე:</p>
  <div v-on:mousemove="mousePos">
    <span>xPos: {{ xPos }}</span><br><span>yPos: {{ yPos }}</span>
  </div>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        xPos: 0,
```

```
      yPos: 0
    }
  },
  methods: {
    mousePos(event) {
      this.xPos = event.offsetX
      this.yPos = event.offsetY
    }
  }
 })
        app.mount('#app')
</script>
</body>
</html>
```

რამდენადმე გავართულოთ ამოცანა: მიზნად დავისახოთ, რომ არეზე თაგვის მაჩვენებლის მოძრაობისას დინამიკურად იცვლებოდეს ამ არის ფონის ფერი.

ვახდენთ კოდის მოდიფიცირებას - სცენარში ვამატებთ **v-bind** დირექტივას და hsl-ის მნიშვნელობის xPos-ისთან დაკავშირებით ვქმნით style-ში ფონის ფერის ცვლილების შესაძლებლობას.

კოდში ვითვალისწინებთ ასეთ ფრაგმენტს:

```
<div
 v-on:mousemove="mousePos"
 v-bind:style="{backgroundColor:'hsl('+xPos+',80%,80%)'}">
</div>
```

მაგალითი

```
<!DOCTYPE html>
<html>
<head>
 <title>თაგვის მაჩვენებლის პოზიციის დაფიქსირება</title>
 <style>
```

```
    #app {
      border: black dashed 1px;
      width: 400px;
      padding: 0 20px 20px 20px;
    }
    #app > div {
      width: 360px;
      height: 100px;
      background-color: lightcoral;
    }
    #app span {
      font-weight: bold;
      font-family: 'Courier New', Courier, monospace;
      margin: 20px;
    }
  </style>
</head>
<body>
<h3 align="center">თაგვის მაჩვენებლის პოზიციის ცვლით
     ზემოქმედებას ვახდენთ <br/> მონიტორზე გამოტანილ
     გამოსახულებაზე!
</h3>
<div id="app">
 <p>თაგვის მაჩვენებელი მოათავსეთ ქვემოთ არეზე და ამოძრავეთ
მასზე:</p>
  <div v-on:mousemove="mousePos" v-bind:style  =
        "{backgroundColor:'hsl('+xPos+',80%,80%)'}">
    <br><span>xPos: {{ xPos }}</span><br><span>yPos: {{ yPos }}</span>
  </div>

<p>CSS:<br>backgroundColor:'hsl(<strong>{{xPos}}</strong>,80%,80%)'<
/p>
</div>
```

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        xPos: 0,
        yPos: 0
      }
    },
    methods: {
      mousePos(event) {
        this.xPos = event.offsetX
        this.yPos = event.offsetY
      }
    }
  })
          app.mount('#app')
</script>
</body>
</html>
```

ამჯერად განვიხილოთ ისეთი ამოცანა, როდესაც გვსურს ხდომილობის ობიექტი შეიცავდეს ტექსტს, აკრეფილს მომხმარებლის მიერ <textarea>-ელემენტში, რათა შესაძლებელი გახდეს მისით, როგორც ბლოკნოტით, სარგებლობა.

ქვემოთ მოყვანილ მაგალითში v-on დირექტივა შეიცავს <textarea> ელემენტს, რომელიც „უსმენს“ input-ხდომილობას, რომლის ამოქმედებაც ხდება ამ ელემენტში ტექსტის ყოველი ცვლილებისას.

ამ დროს გამოძახებული writeText მეთოდი დუმილით გადმოაგზავნის ხდომილობის ობიექტს, რათა წაკითხულ იქნას ტექსტი <textarea> ელემენტიდან.

Vue-ეგზემპლარში text თვისების განახლება ხდება writeText მეთოდით.

დაბოლოს, Vue -ფრეიმვორკი ზრუნავს span ელემენტში არსებულ ორმაგ ფიგურულ ფრჩხილებში, არსებული ტექსტის ავტომატურად განახლებაზე.

ვხედავთ, რომ ამ ორმაგ ფრჩხილებში მითითებულია სწორედ ეს text თვისება.

მაგალითი

```
<!DOCTYPE html>
<html>
<head>
  <title>Notebook-ის იმიტაცია</title>
  <style>
    #app {
      border: black dashed 1px;
      width: 300px;
      padding: 20px;
    }
    #app > div {
      width: 100%;
      position: relative;
      margin-top: 10px;
      aspect-ratio: 1;
      background-image: url("Note.png");
      background-size: 340%;
      background-position: -345px 0;
      overflow: hidden;
    }
    #app span {
      width: 80%;
```

```
    font-weight: bold;
    font-family: 'Courier New', Courier, monospace;
    line-height: 1.2em;
    transform-origin: 0 0;
    rotate: 33deg;
    position: absolute;
    top: 50px;
    left: 70px;
   }
   #app textarea {
    width: 100%;
    box-sizing: border-box;
   }
  </style>
</head>
<body>
<h1>Notebook</h1>
<div id="app">
  <textarea v-on:input="writeText" rows="8" cols="30" placeholder="Start
writing.."></textarea>
  <div>
   <span>{{ text }}</span>
  </div>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
   data() {
    return {
     text: ''
    }
   },
   methods: {
```

```
    writeText(event) {
      this.text = event.target.value
    }
  }
})
    app.mount('#app')
</script>
</body>
</html>
```

სურ. 2.2

2.10. მეთოდისათვის არგუმენტის გადაცემა

ვაჩვენოთ ერთი და იმავე მეთოდის გამოძახებისას მისთვის <button>-ღილაკების მეშვეობით არგუმენტის სხვადასხვა მნიშვნელობის გადაცემის მაგალითი:

```
<div id="app">
  <img src="img_moose.jpg">
  <p>{{ "Moose count: " + count }}</p>
  <button v-on:click="addMoose(+1)">+1</button>
  <button v-on:click="addMoose(+5)">+5</button>
  <button v-on:click="addMoose(-1)">-1</button>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
```

```
  data() {
    return {
      count: 0
    }
  },
  methods: {
    addMoose(number) {
      this.count+=number
    }
  }
 })
 app.mount('#app')
</script>
```

დავალება: სცადეთ ზემოთ მოყვანილ კოდის მოდიფიცირება იმგვარად, რომ მეთოდისათვის გადასაცემი პარამეტრის მნიშვნელობა შეირჩეოდეს დიალოგის რეჟიმში და ამის შემდეგ თქვენი ვარიანტი შეადარეთ ქვემოთ მოყვანილს.

```
<!DOCTYPE html>
<center>
<div id="app">
  <img src="ნიკალა.png">
  <h4>{{ "ვამატებთ შემოსწრებულ სტუმართა რიცხვს. ამ მომენტისათვის მათი რაოდენობა არის " + count + "." }}</h4>
  <button v-on:click="addMoose(+0)"><h3> ვამატებთ ამ ღილაკით!</h3></button>
</div>
</center>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
```

```
  data() {
    return {
     count: 0.0
    }
  },
  methods: {
    addMoose(number) {
      number = prompt("შეიტანეთ რიცხვი ", "0")
      this.count += Number(number);
    }
  }
 })
app.mount('#app')
</script>
```

სურ. 2.3 ნიკალა - ხუთი თავადის ქეიფი

2.11. მეთოდი არგუმენტისა და ხდომილობის ობიექტის პარამეტრებით

დასახული მიზანი - *მეთოდის გამოძახებისას მის მიერ Vue-ფრეიმვორკის ეგზემპლარში არგუმენტის მნიშვნელობისა და ხდომილობის ობიექტის მოძიება და გამოძახების ადგილზე*

გადაცემა - მიიღწევა მისი გამოძახების ადგილზე კოდის შემდეგი ფრაგმენტის შეტანით:

```
<button v-on:click="addAnimal($event, 5)">+5</button>
```

ამ შემთხვევაში მეთოდის სახელად შერჩეულია addAnimal, თუმცა შესაძლებელია, ის, ფაქტობრივად, ნებისმიერი იყოს, განსხვავებით $event პარამეტრის სახელწოდებისაგან.

თვით Vue-ფრეიმვორკის ეგზემპლარში განთავსებული მეთოდისათვის კოდი კი, მაგალითად, ასეთი შეიძლება იყოს:

```
methods: {
  addAnimal(e, number) {
    if(e.target.parentElement.id==="tigers"){
      this.tigers = this.tigers + number
    }
  }
}
```

ქვემოთ მოყვანილია გამოძახებული მეთოდის მეშვეობით ადგილზე ხდომილობის ობიექტისა და კიდევ ერთი არგუმენტის (მოცემულ შემთხვევაში ტექსტური შეტყობინების) გადაცემის მაგალითი:

```
<!DOCTYPE html>
<html>
<head>
  <title> ხდომილობის ობიექტის და კიდევ ერთი არგუმენტის გადმოგზავნა </title>
  <style>
    #app {
      border: dashed black 1px;
      width: 180px;
      padding: 10px;
    }
    #app img {
```

```
    width: 100%;
   }
   #green {
    background-color: lightgreen;
    display: inline-block;
   }
  </style>
</head>
<body>
<h4 align = "center">მეთოდით არგუმენტის მნიშვნელობისა და
ხდომილობის ობიექტის მოძიება და გამოძახების ადგილზე
გადაცემა $event პარამეტრის გამოყენებით. </h4>
<h4 align = "center"> დააწკაპუნეთ გამოსახულებაზე! </h4>
<div id="app"><img src="img_tiger_square.jpeg" id="tiger" v-on:click =
"myMethod($event,'Hello')" width="733" height="733">
<p>აქ აისახება მეთოდის მიერ გამოგზავნილი შეტყობინება და
img-ელემენტის id:</p>
<p id="green">"{{ msgAndId }}"</p>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
   data() {
    return {
     msgAndId: ''
    }
   },
   methods: {
    myMethod(e,msg) {
     this.msgAndId =  msg + ', '
     this.msgAndId += e.target.id
    }
   }
```

```
 })
 app.mount('#app')
</script>
</body>
</html>
```

სურ. ვეფხია ვეფხვაძე

2.12. Vue-ხდომილობების მოდიფიკატორები

Vue-ფრეიმვორკში ხდომილობების უფრო ეფექტიანად დამუშავებისათვის დასაშვებია მათი მოდიფიცირება. და ამ შემთხვევაშიც გამოიყენება v-on დირექტივა, მაგალითად, შემდგომი დამატებითი შესაძლებლობების უზრუნველსაყოფად:

1. v-on:submit.prevent - დუმილის წესით, მოხდეს HTML-ფორმებში ცვლილებების შეტანის აკრძალვა მათი გაგზავნისას;
2. v-on:click.once - ნიშნავს, რომ ხდომილობა **მხოლოდ ერთხელ** გაიშვება <button>-ღილაკზე დაწკაპუნებით მას შემდეგ, როგორც კი დამთავრდება ვებფურცლის ჩატვირთვა.
3. v-on:keyup.enter - მიუთითებს კლავიშს, რომელიც უნდა იქნას გამოყენებული მასზე დაჭერისას გამოწვეული ხდომილობით შესაბამისი მეთოდის გამოსაძახებლად.

ამრიგად, მოდიფიკატორების მოდიფიცირების მეშვეობით ხდება ხდომილობებზე რეაგირებისათვის უფრო მეტი საშუალებების გამოყენება.

ხდომილობის მომენტში მისი მოდიფიკატორთან დაკავშირება ამ შემთხვევაშიც სტანდარტული წესით ხდება:

```
<button v-on:click="createAlert">Create alert</button>
```

რეაგირების სახის უფრო დასაკონკრეტებლად, მაგალითად, ზემოთ სიაში მეორე პუნქტში აღწერილი დავალების შესასრულებლად, შესაძლებელი არის ასეთი სინტაქსის გამოყენება:

```
<button v-on:click.once="createAlert">Create alert</button>
```

მაგალითი_1

```
<!DOCTYPE html>
<html>
<head>
  <title> თაგვის მაჩვენებლის პოზიცია</title>
  <style>
    #app {
      border: black dashed 1px;
      width: 200px;
      padding: 0 20px 20px 20px;
    }
  </style>
</head>
<body>
<h1> შეტყობინების უფლება გაიცემა მხოლოდ ერთხელ! </h1>
<div id="app">
 <p>ქვემოთ ღილაკზე დაწკაპუნებით მონიტორზე აისახება
       შეტყობინება, მაგრამ ეს დასაშვებია მხოლოდ
       <strong>ერთხელ.</strong>
</p>
  <button v-on:click.once="createAlert">გაცნობებთ, რომ...</button>
```

```
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    methods: {
      createAlert() {
        alert("მეტად ნუ შემაწუხებთ, სამუშაო დღე დამთავრდა!")
      }
    }
  })
       app.mount('#app')
</script>
</body>
</html>
```

ზემოთ მოყვანილი ამოცანის გადაწყვეტა დასაშვებია ხდომილობის დამუშავებით მეთოდის შიგნითაც, მაგრამ განხილული მიდგომით საქმე უფრო მარტივდება.

შემდეგ, ხდომილობის მოდიფიკატორებისადმი მიმართვა შესაძლებელია სხვა სიტუაციებშიც, და მეტიც, მაგალითად, თაგვისა და კლავიატურის ერთობლივად გამოყენებისას.

ჯერ განვიხილოთ კლავიატურისათვის გათვალისწინებული ხდომილობები, ამ ხდომილობებთან კავშირებული მოდიფიკატორები და წარმოვადგინოთ მათთან მუშაობის მაგალითები.

კლავიატურაზე განთავსებული კლავიშებისათვის იყენებენ სამის სახის ხდომილობას:

keydown: keypress და keyup.

თითოეული მათგანისათვის უნდა მიეთითოს ის კლავიში, რომელზე ზემოქმედება გამოიწვევს ხდომილობას.

აი, როგორი წესით შეიძლება მოხდეს ზოგიერთ მათგანზე მითითება:

.space, .enter, .w,.up

კლავიშზე ზემოქმედებით გამოწვეული ხდომილობისათვის ჩანაწერის გაკეთება შესაბამისი კლავიშის განსაზღვრისათვის შესაძლებელია როგორც ვებფურცელზე, ასევე, - კონსოლშიც ამ ოპერატორის მეშვეობით:

console.log(event.key)

მოვიყვანოთ მაგალითი - კლავიატურისათვის დანიშნული keydown ხდომილობით შესრულებაზე გაიშვება getKey მეთოდი, შედეგად ხდომილობის ობიექტიდან key მნიშვნელობა აისახება კონსოლსა და ვებფურცელზე.

მაგალითი_2

```
<!DOCTYPE html>
<html>
<head>
 <title> კლავიშების დაკავშირება ხდომილობებთან </title>
 <style>
  #app {
   border: black dashed 1px;
   width: 200px;
   padding: 0 20px 20px 20px;
  }
  #green {
   background-color: lightgreen;
  }
 </style>
</head>
<body>
<div id="app">
 <h4>ტექსტურ ველში ჩაწერეთ კლავიშის დასახელება და დააკავშირეთ ის რომელიმე ძირითადი ფუნქციის
```

```
        შემსრულებელ ღილაკთან.<br/>პროცედურა რამდენჯერმე
        გაიმეორეთ და გადადით consol-ჩანართზე.
      </h4>
       <input type="text" v-on:keydown="getKey">
  <p id="green">{{ keyValue }}</p>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        keyValue: ''
      }
    },
    methods: {
      getKey(evt) {
        this.keyValue = evt.key
        console.log(evt.key)
      }
    }
  })
          app.mount('#app')
</script>
</body>
</html>
```

შენიშვნა: ამ მაგალითებში შექმნილი ფაილების სამუშაო გარემოში გადატანისას გამოყენებული უნდა იქნეს (*.prod.js) კრებული.

დასაშვებია ხდომილობისათვის შეირჩეს კომბინირებული ვარიანტიც - მისი გამოძახება მოხდეს მხოლოდ მაშინ, როდესაც, მაგალითად, არეზე თაგვით დაწკაპუნებისას ან კლავიშზე ხელის დაჭერის მომენტში ასევე .alt, .ctrl, .shift ან .meta კლავიშებზეც არის

ხელი დაჭერილი (ბოლო მათგანი Windows ოპერაციულ სისტემაში შესაბამისი გრაფიკული გამოსახულების მქონე კლავიშითაა წარმოდგენილი, Apple-სგან განსხვავებით).

კლავიშ-მოდიფიკატორი	ფსევდონიმები უმთავრესი კლავიშებისათვის
.[*Vue key alias*]	• .enter • .tab • .delete • .esc • .space • .up • .down • .left • .right
.[*letter*]	უჩვენეთ სიმბოლო, რომელიც გამოვა კლავიშზე დაჭერისას. მაგალითად, გამოიყენეთ .s კლავიშის მოდიფიკატორი, S-კლავიშზე ზემოქმედების გასაკონტროლებლად.
.[*system modifier key*]	.alt, .ctrl, .shift ან .meta კლავიშები გამოიყენება როგორც სხვა კლავიშებთან წყვილში, ასევე არეზე თაგვით დაწკაპუნებასთან ერთად.

ჯერ განვიხილოთ ის მარტივი ამოცანა, როდესაც <textarea>-ელემენტის შიგნით ასოს შეტანით ხდება .s-მოდიფიკატორისადმი მიმართვა და შედეგად - მონიტორზე შეტყობინების გამოტანა:

მაგალითი

```
<!DOCTYPE html>
<html>
<head>
  <title>კლავიშზე ზემოქმედების მაგალითი</title>
  <style>
    #app {
      border: black dashed 1px;
      width: 200px;
      padding: 0 20px 20px 20px;
    }
    #app > div {
      width: 160px;
      background-color: lightcoral;
      padding: 20px;
    }
    #app span {
      color: black;
      font-weight: bold;
      font-family: 'Courier New', Courier, monospace;
    }
  </style>
</head>
<body>
<h1>ვაჩვენოთ 'S'-კლავიშზე ხელის დაჭერაზე რეაგირების მაგალითი</h1>
<div id="app">
  <p>The 'S' key generates an alert:</p>
    <textarea cols="20" rows="3" v-on:keydown.s="createAlert" placeholder
          = "Start writing..">
      </textarea>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

```
<script>
  const app = Vue.createApp({
    methods: {
      createAlert(evt) {
        alert("თქვენ ხელი დააჭირეთ 'S'-კლავიშს !");
      }
    }
  })
 app.mount('#app')
</script>
</body>
</html>
```

შესაძლებელია ხდომილობის მოდიფიკატორების ერთდროულად გამოყენებაც, ანუ ხდომილობა გამოიძახება მაშინ, თუ ადგილი ექნება ერთზე მეტ ასეთ მოვლენას.

ქვემოთ ნაჩვენებია .s და .ctrl მოდიფიკატორების ერთდროულად გამოყენების მაგალითი, როდესაც ამ მოდიფიკატორებისადმი მიმართვა ხდება «s» და «ctrl» კლავიშებზე ერთდროულად ხელის დაჭერისას, რის შედეგადაც <textarea> ელემენტის შიგნით მონიტორზე აისახება შესაბამისი შეტყობინება.

მაგალითი_3

```
<!DOCTYPE html>
<html>
<head>
  <title> კომბინირებული ზემოქმედების მაგალითი </title>
  <style>
    #app {
      border: black dashed 1px;
      width: 200px;
      padding: 0 20px 20px 20px;
    }
```

```
    #app > div {
      width: 160px;
      background-color: lightcoral;
      padding: 20px;
    }
    #app span {
      color: black;
      font-weight: bold;
      font-family: 'Courier New', Courier, monospace;
    }
  </style>
</head>
<body>
<h3> წარმოგიდგენთ 'S' და 'Ctrl' კლავიშებით ერთად სარგებლობის შედეგს!</h3>
<div id="app">
  <p> ერთდროულად დააჭირეთ ხელი 'Ctrl' და 'S' კლავიშებს:</p>
  <textarea cols="20" rows="3" v-on:keydown.ctrl.s="createAlert" placeholder="Start writing.."></textarea>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    methods: {
      createAlert(evt) {
        alert(" თქვენ ერთდროულად ხელი დააჭირეთ 'S' და 'Ctrl' კლავიშებს!")
      }
    }
  })
          app.mount('#app')
</script>
</body></html>
```

ამჯერად განვიხილოთ თაგვის მეშვეობით გამოძახებული ხდომილობების მოდიფიკატორები და წარმოვადგინოთ მათთან მუშაობის მაგალითები.

პირველ რიგში აღვნიშნოთ ის გარემოება. რომ თაგვით დაწკაპუნებით გამოწვეულ ხდომილობაზე რეაგირება შესაძლებელია **v-on:click** ნოტაციით, მაგრამ ხშირად მიმართავენ დაზუსტებასაც, რომელი ღილაკით - მარჯვენათი თუ მარცხენათი - განხორციელდა ეს ქმედება, რისთვისაც გამოიყენება დამატებითი სახის ინფორმაცია .left, .center ან .right-ის სახით.

მაგალითი_4

```
<!DOCTYPE html>
<html>
<head>
 <title>თაგვის მარჯვენა ღილაკზე დაწკაპუნებით ხდომილობის გამოწვევა</title>
 <style>
  #app {
   border: black dashed 1px;
   width: 200px;
   padding: 20px;
  }
  #app > div {
   width: 160px;
   padding: 20px;
   cursor: default;
   font-weight: bold;
  }
 </style>
</head>
<body>
```

```
   <h1>>თაგვის მარჯვენა ღილაკზე დაწკაპუნებით ხდომილობის
                                    გამოწვევა</h1>
<div id="app">
  <div v-on:click.right="changeColor" v-bind:style = "{
          backgroundColor:'hsl('+bgColor+',80%,80%)' }">
    <p>დააწკაპუნეთ თაგვის მარჯვენა ღილაკზე!</p>
  </div>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        bgColor : 0
      }
    },
    methods: {
      changeColor() {
        this.bgColor+=50;
      }
    }
  })
       app.mount('#app')
</script>

</body>
</html>
```

მაგალითი_5

```
<!DOCTYPE html>
<html>
<head>
  <title>
```

ფონის ფერის შეცვლა თაგვის მარჯვენა ღილაკის და ctrl-კლავიშის კომბინაციით

```
 </title>
 <style>
  #app {
   border: black dashed 1px;
   width: 200px;
   padding: 20px;
  }
  #app > div {
   width: 160px;
   padding: 20px;
   cursor: default;
   font-weight: bold;
  }
 </style>
</head>
<body>
<center>
 <h3>DIV-ელემენტში ფონის ფერის შესაცვლელად დააწკაპუნეთ
     არეზე თაგვის მარჯვენა ღილაკით და ერთდროულად ხელი
     დააჭირეთ ctrl-კლავიშს!
 </h3>
 <div id="app">
  <div v-on:click.right.ctrl="changeColor" v-bind:style=
        "{ backgroundColor:'hsl('+bgColor+',80%,80%)' }">
   <p> აბა ჰე, გელოდებით! </p>
  </div>
 </div>
</center>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
 const app = Vue.createApp({
```

```
  data() {
    return {
      bgColor : 0
    }
  },
  methods: {
    changeColor() {
      this.bgColor+=50;
    }
  }
 })
       app.mount('#app')
</script>
</body>
</html>
```

შემდეგ, თუ .prevent ხდომილობის მოდიფიკატორს დავუმატებთ .right მოდიფიკატორს, ამ თანამშრომლობით თავიდან იქნება აცილებული თაგვის მარჯვენა ღილაკით დაწკაპუნებისას ჩამოშლადი სიის მონიტორზე დუმილით ასახვა.

მაგალითი 6:

```
<!DOCTYPE html>
<html>
<head>
  <title> ხდომილობა თაგვის მარჯვენა ღილაკით  </title>
  <style>
    #app {
      border: black dashed 1px;
      width: 200px;
      padding: 20px;
    }
```

```
    #app > div {
      width: 160px;
      padding: 20px;
      cursor: default;
      font-weight: bold;
    }
  </style>
</head>
<body>
<h1>Example</h1>
<p>დუმილით, თაგვის მარჯვენა ღილაკით DIV-ელენტზე
  დაწკაპუნებისას drop-down (ანუ ჩამოშლადი) მენიუ
  მონიტორზე არ აისახება!
</p>
<div id="app">
  <div v-on:click.right.prevent="changeColor" v-bind:style = "{
    backgroundColor:'hsl ('+bgColor+',80%,80%)' }">
    <p> დააჭირეთ ხელის თაგვის მარჯვენა ღილაკზე! </p>
  </div>
</div>
<script src="https://unpkg.com/vue@3/dist/vue.global.js">
</script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        bgColor : 0
      }
    },
```

```
    methods: {
      changeColor() {
        this.bgColor+=50;
      }
    }
  })
  app.mount('#app')
</script>
</body>
</html>
```

თაგვის მარჯვენა ან მარცხენა ღილაკზე დაწკაპუნება შესაძლებელია კომბინაციაში შეათავსოთ სხვა ხდომილებათა მოდიფიკატორებთან.

დავალება: დაწერეთ კოდები ისეთი სცენარებისათვის, როგორითაც მოხდება ასეთი კომბინაციებით, მაგალითად, მონიტორზე სურათების ცვლა.

დანართი

Event	Occurs When	Belongs To
abort	The loading of a media is aborted შეწყვეტილია ინფორმაციის ჩამოტვირთვა Загрузка носителя прервана	UiEvent, Event
afterprint	A page has started printing დაიწყო ფურცლის ბეჭდვა Страница начала печататься	Event
animationend	A CSS animation has completed დამთავრდა CSS-ანიმაციის პროცესი CSS-анимация завершена.	AnimationEvent
animationiteration	A CSS animation is repeated მეორდება CSS-ანიმაციის პროცესი CSS-анимация повторяется	AnimationEvent
animationstart	A CSS animation has started დაიწყო CSS-ანიმაციის პროცესი CSS-анимация началась.	AnimationEvent
beforeprint	A page is about to be printed დასაბეჭდად მზადდება ფურცელი Страница готовится к печати	Event
beforeunload	Before a document is about to be unloaded სანამ დოკუმენტი ჩამოიტვირთება Прежде чем документ будет выгружен	UiEvent, Event
blur	An element loses focus	FocusEvent

	ელემენტი კარგავს ფოკუსს Элемент теряет фокус	
change	The content of a form element has changed ფორმის ელემენტის შემცველობა შეიცვალა Содержимое элемента формы изменилось	Event
click	An element is clicked on ელემენტზე დაწკაპუნება Щелчок по элементу	MouseEvent
contextmenu	An element is right-clicked to open a context menu თაგვის მარჯვენა ღილაკზე დაწკაპუნებით კონტექტური მენიუს გაღება Щелчок правой кнопкой мыши по элементу открывает контекстное меню	MouseEvent
copy	The content of an element is copied ელემენტის შემცველობის კოპირება Содержимое элемента копируется	ClipboardEvent
cut	The content of an element is cut ელემენტის შემცველობის ამოჭრა Содержимое элемента вырезано	ClipboardEvent
dblclick	An element is double-clicked ელემენტზე ორჯერ დაწკაპუნება Элемент дважды щелкнут	MouseEvent
drag	An element is being dragged	DragEvent

	ხდება ელემენტის გადათრევა Элемент перетаскивается	
dragend	Dragging of an element has ended ელემენტის გადათრევა დამთავრდა Перетаскивание элемента завершилось	DragEvent
dragenter	A dragged element enters the drop target ელემენტის გადათრევის მიზანი მიღწეულია Перетаскиваемый элемент попадает в цель перетаскивания	DragEvent
dragleave	A dragged element leaves the drop target გადათრევადი ელემენტი ტოვებს სამიზნეს Перетаскиваемый элемент покидает цель перетаскивания.	DragEvent
dragover	A dragged element is over the drop target გადათრევადი ელემენტი სამიზნის თავზეა Перетаскиваемый элемент находится над целью перетаскивания.	DragEvent
dragstart	Dragging of an element has started დაიწყო ელემენტის გადათრევა Перетаскивание элемента началось	DragEvent
drop	A dragged element is dropped on the target გადათრევადმა ელემენტმა მიაღწია სამიზნეს Перетаскиваемый элемент попадает в цель	DragEvent

durationchange	The duration of a media is changed მედიის ხანგრძლიობა შეიცვალა Продолжительность медиа изменена	Event
ended	A media has reach the end ("thanks for listening") ინფორმაცია სრულად გადაიცა („მადლობა მოსმენისათვის“) СМИ дошли до конца («спасибо, что выслушали»)	Event
error	An error has occurred while loading a file ფაილის ჩატვირთვისას მოხდა შეცდომა Произошла ошибка при загрузке файла	ProgressEvent, UiEvent, Event
focus	An element gets focus ელემენტი ღებულობს ფოკუსს Элемент получает фокус	FocusEvent
hashchange	There has been changes to the anchor part of a URL URL-მისამართის ბირთვში შეტანილია ცვლილებები Были внесены изменения в якорную часть URL-адреса.	HashChangeEvent
input	An element gets user input ელემენტს მომხმარებლისგან მიეწოდება ინფორმაცია Элемент получает пользовательский ввод	InputEvent, Event
invalid	An element is invalid ელემენტი არამოქმედია	Event

	Элемент недействителен	
keydown	A key is down კლავიში დაწეულია Ключ опущен	KeyboardEvent
keypress	A key is pressed კლავიში დაჭერილია Нажата клавиша	KeyboardEvent
keyup	A key is released კლავიში აშვებულია Ключ отпущен	KeyboardEvent
load	An object has loaded ობიექტი ჩაიტვირთა Объект загрузился	UiEvent, Event
loadeddata	Media data is loaded მედიამონაცემები ჩატვირთულია Медиаданные загружены	Event
loadedmetadata	Meta data (like dimensions and duration) are loaded მედიამონაცემები (მაგ., ზომები და ხანგრძლიობა) იტვირთება Метаданные (например, размеры и продолжительность) загружаются.	Event
loadstart	The browser starts looking for the specified media ბროუზერი იწყებს მითითებული მედიაწყაროს ძებნას	ProgressEvent

	Браузер начинает поиск указанного носителя	
message	A message is received through the event source შეტყობინება მიღებული იქნა ხდომილობის წყაროდან Сообщение получено через источник событий	Event
mousedown	The mouse button is pressed over an element ელემენტზე დაჭერილია თაგვის ღილაკი Кнопка мыши нажата на элемент	MouseEvent
mouseenter	The pointer is moved onto an element თაგვის მაჩვენებელი გადაადგილდება ელემენტზე Указатель перемещается на элемент	MouseEvent
mouseleave	The pointer is moved out of an element თაგვის მაჩვენებელი სცდება ელემენტის ფარგლებს Указатель перемещается за пределы элемента	MouseEvent
mousemove	The pointer is moved over an element თაგვის მაჩვენებელი დაყენებულია ელემენტზე Указатель наведен на элемент	MouseEvent
mouseover	The pointer is moved onto an element	MouseEvent

	თაგვის მაჩვენებელი გადაადგილდება ელემენტზე Указатель перемещается на элемент	
mouseout	The pointer is moved out of an element თაგვის მაჩვენებელი სცდება ელემენტის ფარგლებს Указатель перемещается за пределы элемента	MouseEvent
mouseup	A user releases a mouse button over an element მომხმარებელი ათავისუფლებს ელემენტზე დაჭერილ თაგვის ღილაკს Пользователь отпускает кнопку мыши над элементом	MouseEvent
		WheelEvent
open	A connection with the event source is opened ვუკავშირდებით ხდომილობის წყაროს Открывается соединение с источником события	Event
pagehide	User navigates away from a webpage მომხმარებელი მიდის ვებფურცლიდან Пользователь уходит с веб-страницы	PageTransitionEvent
pageshow	User navigates to a webpage მომხმარებელი გადადის ვებფურცელზე Пользователь переходит на веб-страницу	PageTransitionEvent
paste	Some content is pasted in an element	ClipboardEvent

	ელემენტში რაიმე კონტენტის ჩასმა Некоторый контент вставлен в элемент	
pause	A media is paused პაუზა Медиа приостановлено	Event
play	The media has started or is no longer paused მედია გაშვებულია ან აღარ არის დაპაუზებული Медиа запущено или больше не приостановлено	Event
progress	The browser is downloading media data ბროუზერი გადმოიწერს მედიამონაცემებს Браузер загружает медиаданные	Event
reset	A form is reset ფორმა სუფთავდება Форма сбрасывается	Event
select	User selects some text მომხმარებელი ირჩევს ტექსტს Пользователь выбирает текст	UiEvent, Event
show	A <menu> element is shown as a context menu <menu>- ელემენტი აისახება, როგორც კონტექსტური მენიუ Элемент <menu> отображается как контекстное меню.	Event

submit	A form is submitted ფორმა გაიგზავნა Форма отправлена	Event
unload	A page has unloaded ფურცელი ამოღებული იქნა Страница выгрузилась	UiEvent, Event
volumechange	The volume of a media is changed (includes muting) შეიცვალა ხმიანობის დონე (გამორთვის ჩათვლით) Громкость мультимедиа изменена (включая отключение звука)	Event
waiting	A media is paused but is expected to resume (e.g. buffering) მედია-ფაილი დაპაუზებულია (შესაძლოა, ბუფერიზაციის გამო) Медиа-файл приостановлен, но ожидается его возобновление (напр. буферизация)	Event
wheel	The mouse wheel rolls up or down over an element თაგვის ბორბალი ელემენტზე ზემოთ-ქვემოთ მიგორავს Колесо мыши катится вверх или вниз по элементу	WheelEvent

თავი 3. Kotlin

3.1. შესავალი

Kotlin არის თანამედროვე დაპროგრამების ენა. ის ასპარეზზე გამოვიდა 2016 წელს და სწრაფად მოიპოვა პოპულაროზა. გამოიყენება მრავალი დანიშნულებით, რომელთა შორის განსაკუთრებით აღსანიშნავია Android ოპერაციული სისტემისა და სერვერებისთვის ვებდანართების შექმნა.

აქვე შევნიშნავთ, რომ Android ოპერაციული სისტემა ფართოდ გამოიყენება სმარტფონებსა და პლანშეტებში, ელექტრონულ წიგნებსა და ნოუტბუკებში, ათასგვარ სხვადასხვა დანიშნულების მოწყობილობებში, იქნება ეს მაჯის საათები, ფიტნეს-სამაჯურები თუ Google Glass სათვალეები და სხვ.

შემდეგ, ეს ენა მთლიანად თავსებადია Java-სთან, რაც ნიშნავს, რომ Kotlin-ზე დაწერილ პროგრამებში შეიძლება გამოყენებული იქნას Java-ზე დაწერილი კოდი და ამ ენისათვის შექმნილი ბიბლიოთეკები.

ჩამოვთვალოთ Kotlin ენის სხვა ღირსებებიც:

- Kotlin მუშაობს ყველა ცნობილ პლატფორმაზე (Windows, Mac, Linux, Raspberry Pi და სხვ.)
- ენის სინტაქსი ლაკონიურია;
- ხასიათდება მაღალი უსაფრთხოებით;
- ვრცელდება უფასოდ;
- მრავალი დეველოპერის მოღვაწეობის შედეგად სწრაფად ვითარდება.
- მისი შესწავლისათვის აუცილებელი არაა დაპროგრამების სფეროში გამოცდილების ფლობა.

Kotlin IDE

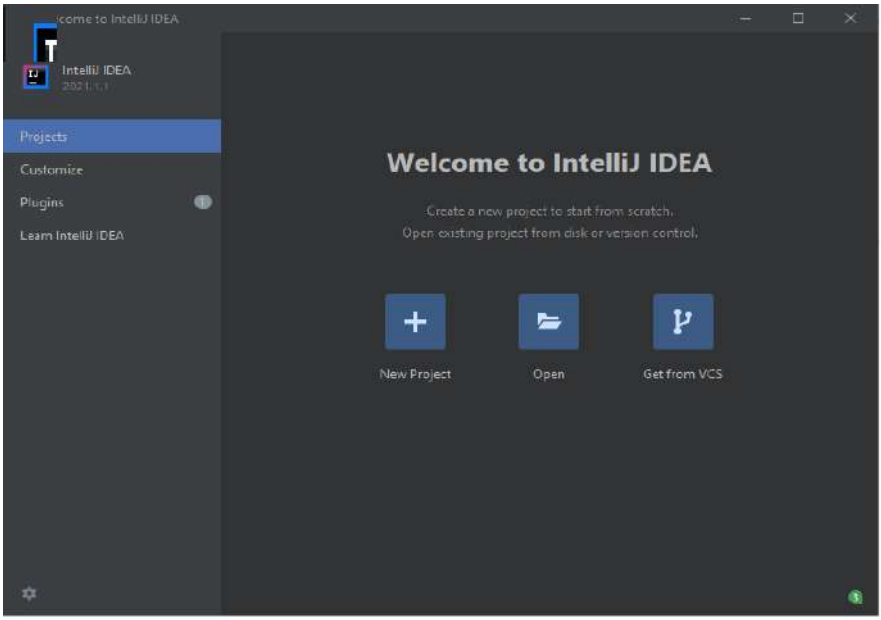
IDE (აბრევიატურა აღნიშნავს კოდის შექმნისათვის განკუთვნილ ინტეგრალურ გარემოს) ამ ენაზე კოდის დაწერისა-თვის ყველაზე უფრო მარტივი და მოხერხებული საშუალებაა.

Kotlin-ისათვის დამუშავებული IntelliJ-ის სახელის მქონე IDE შეიძლება ჩამოვტვირთოთ საიტიდან:

https://www.jetbrains.com/idea/download/

3.2. ინსტრუქცია მუშაობისათვის

IntelliJ-ის ჩამოტვირთვის და დაყენების შემდეგ მასთან მუშაობას ვიწყებთ **New Project** ღილაკზე დაწკაპუნებით:

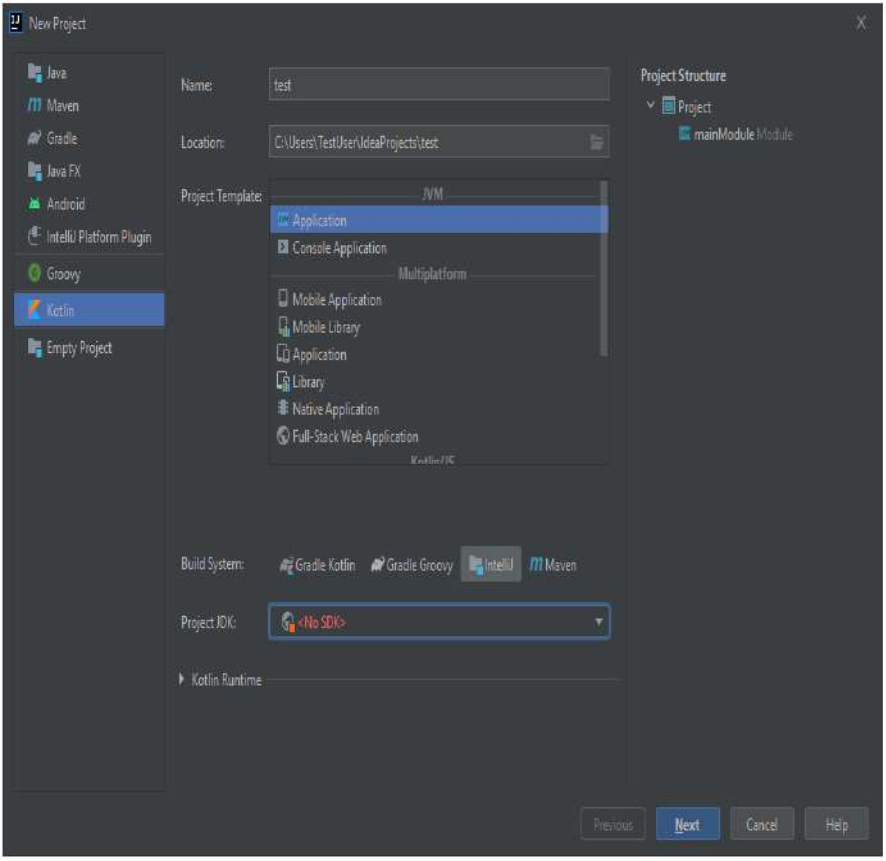


სურ. 3.1

ამის შემდეგ შემდეგ მარცხენა მენიუში ვაწკაპუნებთ **Kotlin**-ზე და შეგვყავს სახელი პროექტისთვის.

Next და Kotlin პროექტის გასაშვებად ვატარებთ მოსამზადებელ სამუშაოებს:

უნდა დავაინსტალიროთ რაიმე სახელწოდებით JDK (Java Development Kit), რისთვისაც ვაწკაპუნებთ **Project JDK**-ზე, მენიუში ვირჩევთ "Download JDK " პუნქტს, მის ვერსიას და გამყიდველს (მაგალითად, AdoptOpenJDK 11-ს) და ვაწკაპუნებთ **Download** ღილაკზე:



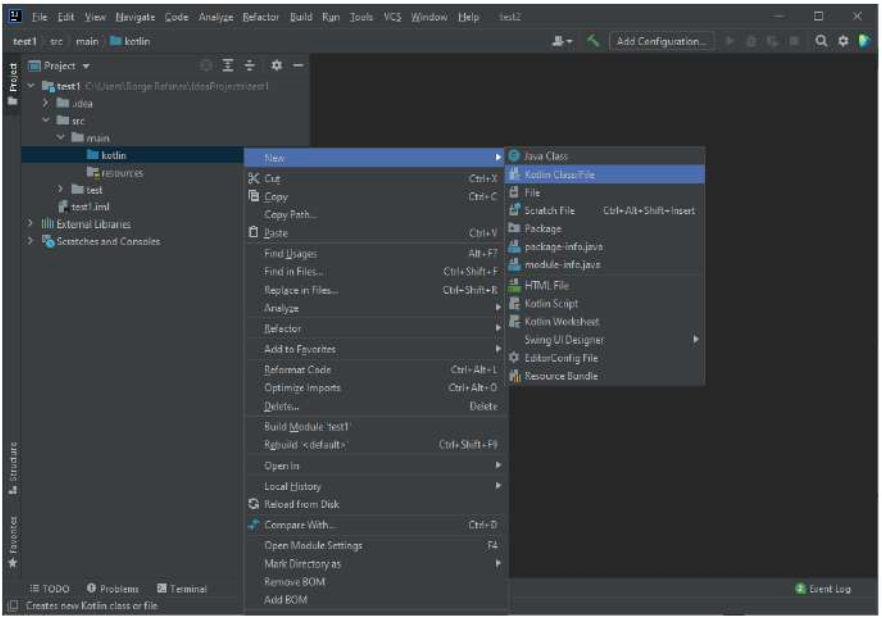
სურ. 3.2

ამრიგად, JDK გადმოწერილი და დაინსტალირებულია.

ამის შემდეგ ვირჩევთ მას მენიუდან და ვაწკაპუნებთ ჯერ Next-სა და ბოლოს Finish-ზე.

3.3. ვიწყებთ Kotlin პროექტთან მუშაობას:

ვხსნით **src** საქაღალდეს და Kotlin ფაილის შექმნის მიზნით ვიმეორებთ ქვემოთ სურათზე მითითებულ ნაბიჯებს:



სურ. 3.3

ვაწკაპუნებთ File ოფციაზე და ვქმნით Main სახელის მქონე Kotlin ფაილს - Kotlin ფაილს ვუმატებთ Main სახელს.

შედეგად იქმნება ჩვენი პირველი Kotlin ფაილი!

ამის შემდეგ ამ Main.kt ფაილში შეგვაქვს კოდი:

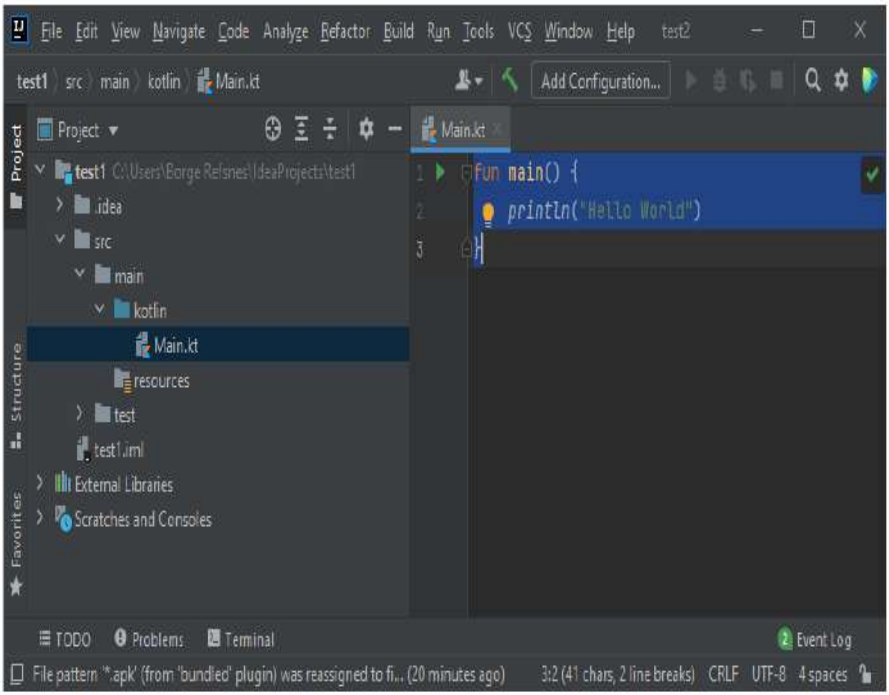
Main.kt

```
fun main() {
  println("Hello World")
}
```

და პროგრამას ვუშვებთ შესრულებაზე:

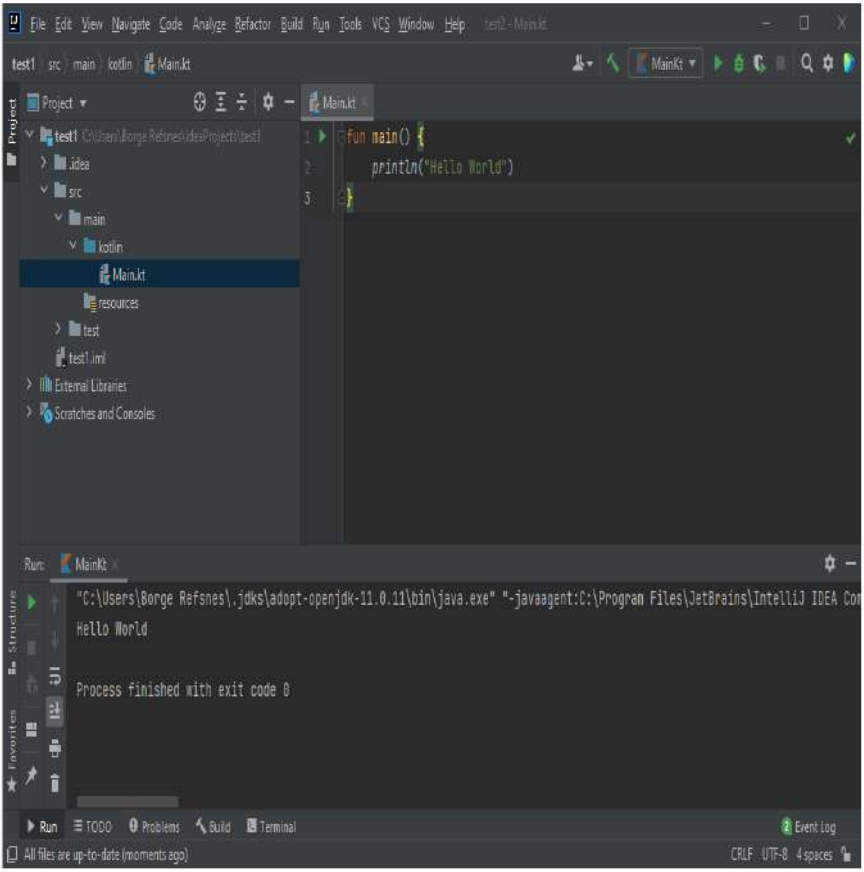
ნავიგაციის ზედა პანელზე ვაწკაპუნებთ Run ღილაკზე, შემდეგ “Run”-ზე და ვირჩევთ "Main.kt"-ს.

IntelliJ ქმნის პროექტს და შესრულებაზე უშვებს Kotlin ფაილს.



სურ. 3.4

შედეგს ექნება ასეთი სახე:


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
test1 src main kotlin Main.kt
MainKt
Project
test1 C:\Users\Borge Refsnes\IdeaProjects\test1
.idea
src
main
kotlin
Main.kt
resources
test
test1.iml
External Libraries
Scratches and Consoles
Main.kt
fun main() {
println("Hello World")
}
Run: MainKt
"C:\Users\Borge Refsnes\.jdks\adopt-openjdk-11.0.11\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Co
Hello World
Process finished with exit code 0
Run TODO Problems Build Terminal
Event Log
All files are up-to-date (moments ago)
CRLF UTF-8 4 spaces

სურ. 3.5

3.4. ფუნქციები, ცვლადები, კონსტანტები

ზემოთ მოყვანილ მაგალითში ფიგურირებს ფუნქცია:

```
fun main() {
  println("Hello World")
}
```

მისი შესრულების შემდეგ მონიტორზე აისახება Kotlin-ენაზე ჩვენ მიერ შექმნილი პირველი პროგრამის რეზულტატი:

"Hello World"

შენიშვნა: სხვა ენებისაგან განსხვავებით, Kotlin-ში აუცილებელი არაა ოპერატორი წერტილ-მძიმით დავაბოლოვოთ. ასევე, უნდა გავითვალისწინოთ, რომ ოპერატორებში გამოყენებული გასაღებური სიტყვები პატარა ასოებით იწერება.

წინა მაგალითში მოყვანილ ფუნქციაში შევიტანოთ მცირე კორექტივი:

```
fun main() {
  println(3 + 3)
}
```

შედეგი იქნება: 6.

ავხსნათ ამ მაგალითებთან დაკავშირებით რამდენი მომენტი:

1. ვხედავთ, რომ Kotlin-ში ფუნქციის აღმნიშვნელად გამოიყენება Fun გასაღებური სიტყვა;
2. main() ფუნქცია ფიგურირებს ამ ენაზე დაწერილ ყოველ პროგრამაში;

3. println() ფუნქციის მიერ რაიმე ინფორმაციის გამოტანის შემდეგ კურსორი გადადის მომდევნო სტრიქონის დასაწყისში (შეამოწმეთ);
4. print () ფუნქციის შესრულების შემდეგ კი ის ადგილზე რჩება (შეამოწმეთ).

აქვე შევნიშნოთ, რომ ქვემოთ მოყვანილ მაგალითში მონიტორზე გამოტანისას ტექსტები ერთმანეთს რომ არ გადაებას, მათ ბოლოებში ვსვამთ დამატებითი შუალედების სიმბოლოებს.

```
fun main() {
  print("Hello World! ")
  print("I am learning Kotlin. ")
  print("It is awesome!")
}
```

ცხადია, კომენტარები ამ ენაშიც გამოიყენება (შევნიშნავთ, რომ ისინი ჩვენთვის უკვე ნაცნობია სხვა ენებიდან):

```
// ეს კომენტარია
/* ეს არის რამდენიმესტრიქონიანი
   კომენტარის მაგალითი */
```

გავეცნოთ Kotlin-ში ცვლადების გამოცხადებისა და სარგებლობის წესებსაც:

```
fun main() {
  var name = "John"
  val birthyear = 1975

  println(name)
  println(birthyear)
}
```

val გასაღებური სიტყვის სახით ვამჩნევთ სიახლეს - ეს სპეციფიკური ცვლადი php ენაში უკვე ნასწავლი define()-ით გამოცხადებული კონსტანტის მსგავსია, რომელსაც მნიშვნელობა ენიჭება მხოლოდ ერთხელ და შემდეგ იგი მთელი პროგრამის შესრულების მანძილზე აღარ იცვლება.

აქედან გამომდინარე, ქვემოთ მოყვანილი კოდი გამოიტანს შეტყობინებას შეცდომის შესახებ:

```
val name = "John"
name = "Robert"  // შეცდომაა - Val-ის მნიშვნელობის შეცვლა აკრძალულია!
println(name)
```

განსხვავებით შემდეგი კოდისაგან:

```
var name: String
name = "John"
println(name)
```

შენიშვნა: *(ამონარიდი უკვე ნასწავლი მასალიდან)* ზოგჯერ საჭიროა, რომ პროგრამაში თავიდან ავიცილოთ ცვლადის მნიშვნელობის და ტიპის შეცვლა. მაშინ ცვლადს მხოლოდ ერთხელ, პროგრამის დასაწყისში მიენიჭება მნიშვნელობა (ამ მნიშვნელობის მიხედვით - ტიპიც) და მერე ის აღარ იცვლება. ცვლადის აღწერილი ნაირსახეობა, ფაქტობრივად, კონსტანტაა. PHP-ში მას ასეც უწოდებენ. მისი განსაზღვრისას $ სიმბოლოს არ იყენებენ და სახელს დიდი ასოებისაგან ადგენენ.

მოვიყვანოთ PHP ენაში ამგვარი სახის კონსტანტის გამოცხადების მაგალითი:

```
<html>
<head>
<title>კონსტანტის შექმნა</title>
</head>
```

```
<body>
<?php
  define ( “USER”, “Gerald” );
  print “Welcome ”.USER;
?>
</body>
</html>
```

Kotlin-ში აუცილებელი არაა ცვლადის გამოცხადებისას მისი ტიპიც იყოს ნაჩვენები (თუმცა ეს შესაძლებელია) - ტიპი ცვლადს განესაზღვრება მისთვის მინიჭებული მნიშვნელობიდან გამომდინარე.

ამასთან, დასაშვებია, ცვლადისათვის მნიშვნელობის მინიჭება მოხდეს არა მისი გამოცხადებისთანავე, არამედ - მოგვიანებითაც, მაგრამ მაშინ უკვე აუცილებელია ცვლადის ტიპის ჩვენება ცვლადის გამოცხადებისთანავე.

ამრიგად, გვაქვს ცვლადის გამოცხადების შემდეგი ვარიანტები:

```
fun main() {
  var asaki = 25                // int
  var saxeli: String = "John" // string
  var info: string
}
```

მონიტორზე გამოსატანი ტექსტისა და ცვლადის გადაბმა Kotlin-ში ხდება + (პლუს) სიმბოლოს დახმარებით:

```
val name = "John"
println("Hello " + name)
```

ცვლადების სახელდების და ზოგიერთი სხვა წესი ასეთია:

- სახელები შეიძლება შეიცავდნენ დიდ და პატარა ლათინურ ასოებს, ციფრებს, ქვედა ტირეს და დოლარის ნიშანს;
- სახელი უნდა დაიწყოს პ ა ტ ა რ ა ასოთი (დასაშვებია $ სიმბოლოთითაც, მაგრამ ეს რეკომენდებულა არაა) და ისინი არ უნდა შეიცავდნენ შუალედებს;
- სახელები რეგისტრის მიმართ მგრძნობიარე არიან (უკვე ვიცით, ეს რას ნიშნავს);
- აკრძალულია სახელებად დარეზერვირებული სიტყვების გამოყენება, როგორიცაა, მაგ., Kotlin, var , String და სხვ.

მოვიყვანოთ ამ წესების დაცვით ცვლადების სახელდებისა და გამოცხადების მაგალითები:

```
var myNum = 5               // Int
var myDoubleNum = 5.99   // Double
var myLetter = 'D'           // Char
var myBoolean = true        // Boolean
var myText = "Hello"         // String
```

ამგვარადვე ცხადდება კონსტანტებიც:

```
val myNum = 5               // Int
val myDoubleNum = 5.99   // Double
val myLetter = 'D'           // Char
val myBoolean = true        // Boolean
val myText = "Hello"         // String
```

აქვე შევნიშნოთ, რომ მონიტორზე გამოსატანი ტექსტისა და ცვლადის გადაბმისათვის, გარდა ზემოთ მოყვანილისა, Kotlin-ში შესაძლებელია სხვა გზასაც მივმართოთ:

```
fun main() {
  var firstName = "John"
  var lastName = "Doe"
  println("My name is $firstName $lastName")
```

}

როგორც ზემოთ აღინიშნა, ცვლადების გამოცხადებისას დასაშვებია მათი ტიპის ჩვენებაც:

```
val myNum: Int = 5                  // Int
val myDoubleNum: Double = 5.99      // Double
val myLetter: Char = 'D'            // Char
val myBoolean: Boolean = true       // Boolean
val myText: String = "Hello"        // String
```

მონაცემთა ტიპები შემდეგ ჯგუფებად იყოფა:

- რიცხვითი - Numbers
- ბულის - Booleans
- სიმბოლოს ტიპის - Characters
- სტრიქონული - Strings
- მასივები - Arrays

1. რიცხვითი

1.1. მთელრიცხვა:

1.1.1. Byte

ვსარგებლობთ მაშინ, თუ დარწმუნებული ვართ, რომ ჩვენთვის საჭირო ცვლადის მნიშვნელობა (-128 -127) დიაპაზონს არ გასცდება, ასეთი ცვლადები გამოიყენება მეხსიერებაში ადგილის ეკონომიის მიზნით.

1.1.2. Short

გამოიყენება (-32768 - 32767) დიაპაზონის რიცხვებისათვის.

1.1.3. Int

(-2147483648 - 2147483647) დიაპაზონის რიცხვებისათვის.

1.1.4. Long

(-9223372036854775808 – 9223372036854775808) დიაპაზონის რიცხვებისათვის.

1.2. მცურავი მძიმის ტიპის:

1.2.1. Float

(3,4e-038 - 3,4e+038) დიაპაზონში მყოფი რიცხვები უნდა დაბოლოვდნენ F სიმბოლოთი, მაგ.: val myNum: Float = 5.75F

1.2.2. Double

რიცხვები იმყოფება (1,7e-308 -ი 1,7e+038) დიაპაზონში, მაგ.: val myNum: Double = 19.99

2. Boolean

ბულის ტიპის რიცხვებმა შეიძლება მიიღონ მხოლოდ true ან false მნიშვნელობა, მაგ.: val isKotlinFun: Boolean = true

3. Characters (Char)

სიმბოლოს ტიპის მონაცემები გამოიყენება ერთად-ერთი სიმბოლოს შესანახავად. მაგ.: val myGrade: Char = 'B'

4. String

სტრიქონული ტიპის ცვლადების მნიშვნელობა არის ცალმაგ ან ორმაგ ბრჭყალებში მოთავსებული ტექსტი.

(დავალება: ეს საკითხი, ინტერნეტში მოძიებულ მასალებზე დაყრდნობით, განიხილეთ უფრო ვრცლად და წარმოადგინეთ ერთ-ერთი შესრულებული ლაბორა-ტორიული სამუშაოს სახით).

5. Arrays

მასივებში ერთი ცვლადის სახელით ინახება რამდენიმე მნიშვნელობა (ამ საკითხს დავუბრუნდებით).

3.5. მონაცემთა ტიპების გარდაქმნა

ამ თემის განხილვამდე ჯერ მოვიყვანოთ Kotlin ენაში გამოყენებული ოპერატორების სია (ისინი ნაცნობია სხვა ენებიდანაც):

ოპერატორი	მაგალითი	ტოლფასი გამოსახულება
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3

შედარების ოპერატორი	შედარების ფორმა	მაგალითი
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

ლოგიკური	ფუნქცია	აღწერა	მაგალითი
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	

დავუბრუნდეთ მონაცემთა ტიპების გარდაქმნის საკითხს:

Kotlin ენის თავისებურება, მაგალითად, Java ენისაგან განსხვავებით, არის ის, რომ მონაცემთა ტიპების გარდაქმნა ხორციელდება სპეციალური ფუნქციების მეშვეობით, კერძოდ, რიცხვითი ტიპის მონაცემების სხვა ტიპში გადასაყვანად გამოიყენება შემდეგი ფუნქციები:

toByte(), toShort(), toInt(), toLong(), toFloat(), toDouble() ანდა toChar().

მოვიყვანოთ მაგალითი:

```
val x: Int = 5
val y: Long = x.toLong()
println(y)
```

Kotlin ენაში სიახლეა ის, რომ იგი შესაძლებლობას იძლევა სტრიქონული ტიპის მონაცემი აღქმული იქნას სიმბოლოების მასივად და თითოეულ სიმბოლოს ამგვარი წესით მივადგეთ:

```
var txt = "Hello World"

println(txt[0])   // პირველი ელემენტია H.
println(txt[2])   // მესამე ელემენტია l.
```

შენიშვნა: ისევე, როგორც მრავალ სხვა ენაში, აქაც მასივის პირველი ელემენტის ნომერია 0. მასივების თემას დაწვრილებით შემდგომ განვიხილავთ.

ქვემოთ მოგვყავს კიდევ რამდენიმე მაგალითი ენაში ჩაშენებული ზოგიერთი ხშირად გამოყენებადი ფუნქციით სარგებლობისა. გავეცნოთ მათ დანიშნულებას მიღებულ შედეგებზე დაკვირვების შედეგად:

1.
```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
println("ტექსტური სტრიქონის სიგრძეა: " + txt.length)
```
2.
```
var txt = "Hello World"
println(txt.toUpperCase())   // გამოდის "HELLO WORLD"
println(txt.toLowerCase())   // გამოდის "hello world"
```
4.
```
var txt1 = "Hello World"
var txt2 = "Hello World"
println(txt1.compareTo(txt2))  // გამოდის 0 (ადგილი აქვს ტოლობას)
```
5.
```
var firstName = "John "
var lastName = "Doe"
```
6.
```
println(firstName.plus(lastName))   //  John Doe7.
// სტრიქონში ცვლადების მნიშვნელობების ჩამატება
```

შესაძლებელია ასეთი წესითაც:

```
    fun main() {
      var firstName = "John"
      var lastName = "Doe"
  println("My name is $firstName $lastName")   // ჩემი სახელია
John Doe
// აქ სიტყვებს შორის შუალედების ჩამატება საჭირო აღარაა.
    }
8.  fun main() {
    var txt = "განსაზღვრეთ 'locate' სიტყვის
   ადგილსამყოფელი!"
    println(txt.indexOf("locate"))      //   7  განმეორება იგივე
   შედეგს იძლევა!
    }
9.  fun main() {                  //  ტექსტი უნდა ჩაისვას მხოლოდ
ორმაგ ბრჭყალებში.
     var txt1 = "It's alright"    // დასაშვებია მასში ცალმაგი
    ბრჭყალების ჩასმაც.
     println(txt1)
    }
10.  fun main() {
    val x = 10
    val y = 9
    println(x > y) // აბრუნებს true-ს, რადგანაც 10 > 9
}
11.  fun main() {
      val time = 22
      if (time < 10) {
        println("Good morning.")
      }
```

```
    else if (time < 20) {
      println("Good day.")
    }
    // დასაშვებია რამდენიმე else if -ბლოკის გამოყენებაც.
    თუ მათი რიცხვი დიდია, მაშინ ვიყენებთ
    when გამოსახულებას (იხ. ქვემოთ).
     else {
      println("Good evening.")
    // დაიბეჭდება "Good evening."
    }
    }
12. fun main() {
      val x = 10;
      println(x == 10); // ხდება ტოლობაზე შემოწმება.
    დაიბეჭდება true.
     }
13. fun main() {
        val time = 20
       val greeting = if (time < 18) "Good day." else "Good
    evening."
// ზედა სახის გამოსახულებაში if-ის მომდევნოდ
აუცილებელია else  ფრაგმენტის ჩართვაც!
თუ ეს კოდები მხოლოდ თითო ოპერატორს მოიცავს,
მაშინ, როგორც ვხედავთ,  შესაძლებელია ფიგურული
ფრჩხილები არც გამოვიყენოთ.
        println(greeting)
    }
14. fun main() {
        val time = 20
```

```
  val greeting = if (time < 18) "Good day." else "Good
evening."
// ზედა სახის გამოსახულებაში if-ის მომდევნოდ
აუცილებელია else  ფრაგმენტის ჩართვაც!
  println(greeting)
}
```

15. გავეცნოთ დაპირებულ when გამოსახულებასაც:

```
fun main() {
val day = 4
val result = when (day) {
  1 -> "Monday"
  2 -> "Tuesday"
  3 -> "Wednesday"
  4 -> "Thursday"
  5 -> "Friday"
  6 -> "Saturday"
  7 -> "Sunday"
  else -> "Invalid day."
}
println(result)  // გამოდის "Thursday"
}
```

ვხედავთ, რომ ზემოთ მოყვანილ ოპერატორებს აქვს როგორც მსგავსება, ისე განსხვავება სხვა ენებში გამოყენებული იმავე დანიშნულების ოპერატორებთან.

3.6. ისევ ფუნქციების შესახებ

უფრო დაწვრილებით განვიხილოთ ეს საკითხი:

ფუნქციებთან უკვე გვქონდა შეხება, მაგალითად, println("Hello World")-ის სახით.

ეს და აქამდე განხილული სხვა ფუნქციები იყვნენ (და არიან) წინასწარ განსაზღვრული, ენაში ჩაშენებული ფუნქციები.

ამჯერად კი მიზნად დავისახოთ ამა თუ იმ დანიშნულების მქონე საკუთარი ფუნქციების შექმნა:

1. პირველ ყოვლისა, უნდა აღვნიშნოთ ის, რომ ფუნქციის სახელწოდებას წინ უნდა უძღოდეს **fun** გასაღებური სიტყვა;
2. მის დასახელებას უნდა მოსდევდეს მრგვალი ფრჩხილები, თუნდაც მათში პარამეტრი (პარამეტრები) არც იყოს მითითებული. მოვიყვანოთ რაიმე ასეთი ფუნქციის გამოცხადების მაგალითი:

```
fun myFunction() {
  println("I just got executed!")
```

3. ფუნქციის გამოძახება კი ხდება ზემოთ არაერთხელ ნაჩვენები ხერხით, მაგალითად, პუნქტ 2-ში გამოცხადებული ფუნქციისათვის ამგვარად:

```
fun main() {
  myFunction()
  myFunction()
  myFunction()
}
// I just got executed!
// I just got executed!
// I just got executed!
```

4 გამოძახებისას ფუნქციებს ხშირად პარამეტრებიც გადაეცემა. რამდენიმე პარამეტრის არსებობისას, ისინი ერთმანეთისაგან მძიმით უნდა განცალკევდნენ. გამოძახებისას აუცილებელია პარამეტრების ტიპის ჩვენებაც, მაგალითად:

```
fun myFunction(fname: String) {
  println(fname + " Doe")
}

fun main() {
  myFunction("John")
  myFunction("Jane")
  myFunction("George")
}
// John Doe
// Jane Doe
// George Doe
```

ქვემო მაგალითში fname პარამეტრს გადაეცემა John, Jane და George არგუმენტები:

```
fun myFunction(fname: String, age: Int) {
  println(fname + " is " + age)
}

fun main() {
  myFunction("John", 35)
  myFunction("Jane", 32)
  myFunction("George", 15)
}
```

```
// John is 35
// Jane is 32
// George is 15
```

შესაძლებელია ფუნქციას დავაბრუნებინოთ პარამეტრის მნიშვნელობა და იგი ცვლადს მივანიჭოთ.

ამ მიზნით გამოიყენება return გასაღებური სიტყვა, ამასთან, აუცილებელია მითითებული იქნას დასაბრუნებელი პარამეტრის და შესაბამისად მისი მნიშვნელობის ტიპიც:

```
fun myFunction(x: Int): Int {
  return (x + 5)
}

fun main() {
  var result = myFunction(3)
  println(result)       // 8 (3 + 5)
}
```

მეორე მაგალითი (რამდენადმე სახეცვლილი):

```
fun myFunction(x: Int, y: Int): Int {
  return (x + y)
}

fun main() {
  var result = myFunction(3, 5)
  println(result)       // 8 (3 + 5)
}
```

Kotlin ენა საშუალებას იძლევა, გასაღებური სიტყვის ნაცვლად იმავე მიზნის მისაღწევად გამოყენებული იქნას ასეთი მიდგომა:

```
fun myFunction(x: Int, y: Int) = x + y

fun main() {
  var result = myFunction(3, 5)
  println(result)       // 8 (3 + 5)
}
```

3.7. ციკლები

1. do – while

ეს არის ციკლი, რომელიც ნებისმიერ შემთხვევაში ერთხელ მაინც შესრულდება.

მოვიყვანოთ მაგალითი:

```
fun main() {
  var i = 0;
  do {
    println(i)
    i++
  }
  while (i < 5)
}
```

ამ ციკლის შესრულების შედეგი იქნება:

```
0
1
2
3
4
```

კიდევ ერთხელ აღვნიშნოთ, რომ ეს ციკლი იმუშავებს მაშინაც, თუ while პირობა არ სრულდება, მაგრამ ეს მოხდება მხოლოდ ერთხელ:

```
fun main() {
 var i = 9;
 do {
  println(i)    //  9
  i++
 }
 while (i < 5)
}
```

ანუ მხოლოდ ერთხელ დაიბეჭდება შედეგი - რიცხვი 9.

2. While

წინა ციკლისგან განსხვავებით, ეს ციკლი პირობის შეუსრულებლობის შემთხვევაში ერთხელაც არ იმუშავებს. ამასთან, მისი გამოყენების დროს უნდა დავრწმუნდეთ, რომ ციკლის ტანში მოთავსებული კოდი ისეა შედგენილი, რომ არ მოხდეს ე. წ. **ჩაციკლვა.**

ქვემოთ მოყვანილ კონკრეტულ შემთხვევაში ასეთ შედეგს უზრუნველყოფს i პარამეტრის ზრდა:

```
fun main() {
  var i = 0;
  do {
    println(i)
    i++
```

```
  }
  while (i < 5)
 }
```

შედეგი ამ შემთხვევაშიც იქნება ზემოთ **do–while** ციკლისათვის განხილული მაგალითის ანალოგიური:

```
0
1
2
3
4
```

3. break ინსტრუქცია

არცთუ იშვიათად რაიმე პირობის შესრულების შემოწმების შემდეგ საჭირო ხდება ციკლიდან გამოსვლა, რაც ხდება break ინსტრუქციის მეშვეობით:

```
fun main() {
  var i = 0

  while (i < 10) {
    println(i)
    i++
    if (i == 4) break
  }
  println ("თავისუფალი ვარ!")

}
```

4. continue ინსტრუქცია

break ინსტრუქციისაგან განსხვავებით, ასევე რაიმე პირობის შესრულებისას მოცემული ადგილიდან გადასვლა ხდება არა ციკლის გარეთ, არამედ მის თავში, ანუ ციკლი გრძელდება. მაგრამ აღარ სრულდება მიმდინარე იტერაციაში continue ინსტრუქციის შემდეგ არსებული კოდი.

დავალება: წინა ფაილის მოდიფიცირება-შესრულების შედეგად დარწმუნდით ზემოთ ნათქვამის მართებულობაში.

```
fun main() {
  var i = 0
  while (i < 9) {
    println(i)
    i++
    if (i == 4) {
    i++
      continue
    }
  }
}
```

შენიშვნა: for ციკლს ქვემოთ გავეცნობით.

3.8. მასივები

Kotlin-ში მასივის შექმნა ხდება arrayOf() ფუნქციის დახმარებით.

მოვიყვანოთ მაგალითი:

```
val cars = arrayOf("Volvo", "BMW", "Ford", "Mazda")
```

ელემენტისადმი მიდგომა კი ხორციელდება ინდექსის ჩვენებით. პირველი ელემენტის ინდექსი ამ ენაშიც არის 0:

```
fun main() {
  val cars = arrayOf("Volvo", "BMW", "Ford", "Mazda")
  println(cars[0])    // დაიბეჭდება Volvo
}
```

მასივის სიგრძის შესახებ ინფორმაცია ინახება size თვისებაში:

```
fun main() {
  val cars = arrayOf("Volvo", "BMW", "Ford", "Mazda")
  println(cars.size)     //  4
}
```

შესამოწმებლად არსებობს თუ არა მოცემული ელემენტი მასივში, გამოიყენება in ოპერატორი:

```
fun main() {
  val cars = arrayOf("Volvo", "BMW", "Ford", "Mazda")
  if ("Volvo" in cars) {
    println("It exists!")  // მოცემულ შემთხვევაში დაიბეჭდება ეს შეტყობინება.
   } else {
    println("It does not exist.")
   }
}
```

მასივის ელემენტები შესაძლებელია ჩავათვალიეროთ ციკლში.

ამ მიზნით გამოვიყენოთ (ზემოთ შეპირებული) for-ციკლი, რომელშიც ელემენტებთან მისადგომად უნდა ჩავრთოთ უკვე ნაცნობი in ოპერატორი:

```
fun main() {
  val cars = arrayOf("Volvo", "BMW", "Ford", "Mazda")
for (x in cars) {
   println(x)    // გამოვა მასივის ელემენტები  სიის სახით.
  }
}
```

for-ციკლი Kotlin ენაში, სხვა ენებისგან განსხვავებით, გარკვეული თავისებურებებით ხასიათდება. მაგალითად, მისი მეშვეობით შესაძლებელი არის სხვადასხვა სახის დიაპაზონების ჩათვალიერება:

```
fun main() {
// ქვემოთ ოპერატორის მეშვეობით ამოირჩევა და  მონიტორზე
   დაიბეჭდება ლათინური ალფაბეტის მიხედვით
   დაიბეჭდება სია მითითებულ დიაპაზონში.
  for (chars in 'a'..'x') {
   println(chars)
   }
 }
```

მოვიყვანოთ მეორე მაგალითი:

```
fun main() {
  for (nums in 5..12) {
    println(nums)      // დაიბეჭდება სია 5-დან 12-ის ჩათვლით.
  }
}
```

in ოპერატორით შესაძლებელია მასივში რაიმე ელემენტის არსებობის შემოწმებაც:

```
val nums = arrayOf(2, 4, 6, 8)
if (2 in nums) {
  println("It exists!")      // გამოვა ეს შეტყობინება.
} else {
  println("It does not exist.")
}
```

for-ციკლში ასევე შესაძლებელია უკვე ნაცნობი break და continue გასაღებური სიტყვების გამოყენება:

```
for (nums in 5..15) {
  if (nums == 10) {
    break
  }
  println(nums)
}
fun main() {
  for (nums in 5..15) {
    if (nums == 10) {
      continue
    }
    println(nums)
  }
}
```

3.9. ობიექტზე ორიენტირებული დაპროგრამება

თავდაპირველად განვმარტოთ, თუ რა მსგავსება და განსხვავებაა პროცედურულ და ობიექტზე ორიენტირებულ დაპროგრამებებს შორის:

პირველი მიდგომა გულისხმობს მონაცემების დამმუშავებელი პროცედურების და მეთოდებს შექმნას, ხოლო მეორე - ობიექტების შექმნას, რომლებიც მოიცავენ როგორც მონაცემებს, ასევე მეთოდებსაც.

ობიექტ-ორიენტირებულ დაპროგრამებას პროცედურულთან შედარებით გააჩნია რიგი უპირატესობებისა:

1. მეტი სისწრაფე და სიმარტივე;
2. პროგრამის სტრუქტურის გაცილებით უკეთესი აღქმადობა;
3. კოდის მოდიფიცირების და გაწყობის გამარტივება;
4. კოდის აგების დაფუძნება DRY (Don't Repeat Yourself - განმეორებების გარეშე) პრინციპზე, რაც გულისხმობს დანართის მუშაობისათვის საჭირო კოდების ერთ ადგილზე შენახვასა და განმეორებებით გამოყენებას.

ობიექტ-ორიენტირებული დაპროგრამებისათვის უმთავრესი ცნებებია **კლასი** და **ობიექტი.**

კლასი ობიექტების შესაქმნელად განკუთვნილი შაბლონია, **ობიექტი** კი - მის ბაზაზე ფორმირებული კონკრეტული ერთეული - ე. წ. კლასის ეგზემპლარი.

მოვიყვანოთ კლასის გამოცხადების მაგალითი:

```
class Car {
  var brand = ""
  var model = ""
  var year = 0
```

```
}
```

ამ კლასის ბაზაზე შევქმნათ კონკრეტული ობიექტი - კლასის ეგზემპლარი:

```
val c1 = Car()     //  ვქმნით Car კლასის ეგზემპლარს, c1 სახელის მქონეს.
c1.brand = "Ford" // აქ და ქვემოთ თვისებებს ვანიჭებთ მნიშვნელობებს.
c1.model = "Mustang"
c1.year = 1969
println(c1.brand)   // Ford
println(c1.model)   // Mustang
println(c1.year)    // 1969
```

იმავე კლასის ბაზაზე შევქმნათ რამდენიმე ობიექტი:

```
val c1 = Car()
c1.brand = "Ford"
c1.model = "Mustang"
c1.year = 1969

val c2 = Car()
c2.brand = "BMW"
c2.model = "X5"
c2.year = 1999

println(c1.brand)  // Ford
println(c2.brand)  // BMW
```

საერთოდ, ობიექტ-ორიენტირებული დაპროგრამების პარადიგმა კლასის ეგზემპლარების ფორმირებისათვის გვთავაზობს უფრო გამარტივებულ ხერხსაც - ესაა კლასის გამოცხადებისას მასში კონსტრუქტორი-ფუნქციის ჩართვა და ამის შემდეგ

ობიექტების შექმნისას მისადმი მიმართვის გამარტივებული წესების გამოყენება.

ამასთან, საქმის გასაადვილებლად ეს მიდგომა Kotlin ენაში სპეციფიკური, უფრო მარტივი სახით რეალიზდება და ზემოთ განხილული მაგალითისათვის ამგვარი სახით წარმოგვიდგება:

```
class Car(var brand: String, var model: String, var year: Int)
```

კლასის ეგზემპლარების შექმნა და მათი თვისებებისთვის შესაბამისი მნიშვნელობის მინიჭება მართლაც უფრო მარტივი გზით ხდება:

```
fun main() {
  val c1 = Car("Ford", "Mustang", 1969)
  val c2 = Car("BMW", "X5", 1999)
  val c3 = Car("Tesla", "Model S", 2020)
}
```

შემდეგ, კლასში მისი გამოცხადებისას, როგორც წესი, ქმნიან ერთი ან მეტი რაოდენობის ფუნქციას, რომლებთანაც მიმართვა შეუძლიათ კლასის ეგზემპლარებს:

```
class Car(var brand: String, var model: String, var year: Int) {
  // კლასის ფუნქცია
  fun drive() {
    println("მოგესალმებით!") // ენაში ჩაშენებული ფუნქციის გამოძახება
  }
}
fun main() {
  val c1 = Car("Ford", "Mustang", 1969)
  // ფუნქციის გამოძახება
  c1.drive()      // კლასში შექმნილ ფუნქციისადმი მიმართვა. მოგესალმებით!
```

ცხადია, კლასში გამოცხადებულ ფუნქციას შეიძლება გააჩნდეს პარამეტრებიც, (წინა შემთხვევისაგან განსხვავებით), რომელთა მნიშვნელობაც დაკონკრეტდება კლასის ეგზემპლარიდან ამ ფუნქციისადმი მიმართვისას:

```
class Car(var brand: String, var model: String, var year: Int) {
  // კლასის ფუნქცია
  fun drive() {
    println("მოგესალმებით!")
  }

  // პარამეტრებიანი კლასის ფუნქცია
  fun speed(maxSpeed: Int) {
    println("მაქს. სიჩქარე არის: " + maxSpeed)     }
    }

fun main() {
  val c1 = Car("Ford", "Mustang", 1969)

  // ქვემოთ დაიბეჭდება Ford Mustang 1969
  println(c1.brand + " " + c1.model + " " + c1.year)

  // Call the functions
  c1.drive()                          //  მოგესალმებით!
  c1.speed(200)                      // მაქს. სიჩქარეა: 200
}
```

ობიექტ-ორიენტირებული დაპროგრამების პარადიგმის ერთ-ერთი უმნიშვნელოვანესი ნიშან-თვისება არის მოცემული კლასის ბაზაზე, რომელსაც **სუპერკლასს** უწოდებენ, მისი

მემკვიდრე **ქვეკლასის** ფორმირების შესაძლებლობა ანუ მოკლედ - **მემკვიდრეობითობა.**

ქვეკლასს შეუძლია ისარგებლოს წინაპრისაგან მემკვიდრეობით გადმოცემული ყველა თვისებითა და ფუნქციით.

მოვიყვანოთ მაგალითი:

```
// სუპერკლასი
open class MyParentClass {
  val x = 5
}

// ქვეკლასი
class MyChildClass: MyParentClass() {
  fun myFunction() {
    println(x) // x ცვლადი მემკვიდრეობითაა გადმოცემული სუპერკლასიდან superclass
  }
}
// MyChildClass ობიექტის შექმნა და myFunction-ის გამოძახება
fun main() {
  val myObj = MyChildClass()
  myObj.myFunction()
}
```

ლიტერატურა

1. https://www.w3schools.com/kotlin/index.php
2. https://developer.android.com/kotlin?gclid=Cj0KCQjw5oiMBhDtARIsAJi0qk1zx2Ak2Erc4_p0d1efwgG26ziqiHuvUrU2AoiINDpTx9mFNhIHhpUaAh8zEALw_wcB&gclsrc=aw.ds

https://www.w3schools.com/w3js/default.asp
3. https://www.w3schools.com/lib/w3.js
4. https://www.w3schools.com/vue/index.php
5. ღვინეფაძე გ. Javascript და მისი შესაძლებლობების განმავითარებელი თანამედროვე ტექნოლოგიები. სახელმძღვანელო. სტუ, საგამომცემლო სახლი: „ტექნიკური უნივერსიტეტი“. თბილისი, 2018.

Gela Ghvinepadze, Nino Chorkhauli

Web technologies (W3.JS, Vue.JS, Kotlin). Textbook

ISBN 978-9941-8-6745-3

გადაეცა წარმოებას 1.06 2024. ოფსეტური ქაღალდის ზომა 60X84 1/16. პირობითი ნაბეჭდი თაბახი 16,25. ტირაჟი 50 ეგზ.

(იბეჭდება ავტორის ხარჯით)

სტუ-ს „IT კონსალტინგის სამეცნიერო ცენტრი~,
თბილისი, მ.კოსტავას 77