ლექცია 13-15. JSON

* სინტაქსისი და მონაცემთა ტიპები
* როგორ წავიკითხოთ მონაცემები
* როგორ ჩავწეროთ მონაცემები JSON ფორმატში
* როგორ მოვახდინოთ JSON -ის Srting-ის კონვერტირება

***მცირე ისტორიული ექსკურსი***

როგორც ბოლო ხანს გახდა ტრენდული - ფორმატი JSON-იც შექმნილ იქნა ფარდულში 2001 წელს. ამის შესახებ იტყობინება ფორმატის ავტორი დუგლას კროკფორდი (***Douglas Crockford***) თავის წიგნში "როგორ მუშაობს javascript-ი"  (Douglas Crockford**,** How JavaScript Works**,** Publisher : Virgule-Solidus October 18, 2018, 279 pages, ISBN-10 : 1949815005, ISBN-13 : 978-1949815009). დუგლას კროკფორდის თქმით მას სჭირდებოდა მონაცემების გადაგზავნა სერვერსა და კლიენტს შორის. ამ მიმართებით დომინანტი იყო XML ფორმატი, მაგრამ კროკფორდს ის არ აკმაყოფილებდა, რადგანაც მის საჭიროებას წარმოადგენდა ასევე მონაცემების გაცვლა ორ სხვადასხვა ენაზე დაწერილ პროგრამებს შორის. ამ ამოცანის გადასაწყვეტად დუგლას კროკფორდმა გადაწყვიტა გამოეყენებინა JavaScript-ის ობიექტის ლიტერები. ამ ფორმატს ეწოდა JSON (იკითხევა "ჯეისან")- *JavaScript* *Object Notation* - ჯავასკრიპტის ობიექტის ჩაწერა. ამასთან ავღნიშნოთ, რომ *JavaScript* დაპროგრამების ენის გარდა JSON გასაგებია სხვა ენებისთვისაც - Python, Ruby, PHP და Java.

პირველი JSON ჩანაწერი რომელიც სერვერზე გასაგზავნად შეიქმნა, იყო:

<html><head><script>

document.domain = "fudco.com";

parent.session.receive({to: "session", do: "test", text: "Hello world"});

</script></head></html>

მაგრამ იგი სხვადასხვა მიზეზის გამო ვერ გაიგზავნა, ამ პრობლების გადასაწყვეტად დუგლას კროკფორდს ეხმარებოდა მისი კომპანიონი **ჩიპ მორნინგსტარი** (Chip Morningstar). დადებითი შედეგი მალე დადგა. JSON და XML-ის შედარებითი ანალიზი მოგვიანებისთვის გადავდოთ. შევთანხმდეთ, რომ ფორმატი გამოიყენება სტრუქტურირებული მონაცემების შესაქმნელად და გადასაგზავნათ.

## *სინტაქსისი და მონაცემთა ტიპები*

JSON დოკუმენტის ასაგებად გამოიყენება ტექსტი, ფიგურული და კვადრატული ფრჩხილები, ორწერტილი, მძიმე, ორმაგი ბრჭყალები და შესაძლებელია დამატებით კიდევ რაიმე სიმბოლო.

1. "**:**"- (ორწერტილი) გამოიყენება მონაცემის სახელისა და მისი მნიშვნელობის გამოსაყოფად;

2. "**,**" - (მძიმე) გამოიყენება წყვილების - მონაცემის სახელი:მნიშვნელობის გამოცალკევებისთვის.

3. "**{**" და "**}**" (ფიგურული ფრჩხილები) - ობიექტის ასაგებად;

4. "**[**" და "**]**" (კვადრატული ფრჩხილები) - მასივის ასაგებად.

**JSON-ის მონაცემები** ჩაიწერება ისევე როგორც JavaScript ობიექტის თვისებები, მონაცემის სახელის და მისი მნიშვნელობის წყვილის სახით, მაგალითად:

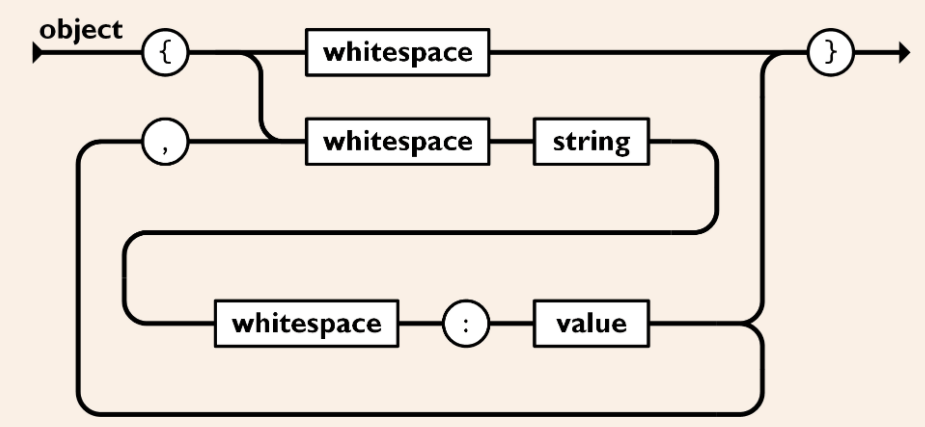
**"saxeli":"სანდრო"**

JSON-ის მონაცემების სახელი აუცილებლად უნდა მოვათავსოთ ორმაგ ბრჭყალებში, რაც JavaScript-ში არ მოითხოვება. მსგავსება ასოცირებულ მასივთან გადაღები/ მნიშვნელობის ჩაწერასთან თვალშისაცემია.

**JSON-ის ობიექტი** ჩაიწერება ფიგურულ ფრჩხილებში, ისევე როგორც JavaScript-ს ობიექტში, შესაძლებელია ჩაწერილ იქნას რამდენიმე წყვილით - მონაცემის სახელი/მნიშვნელობა, რომელიც მძიმითაა გამოყოფილი. მაგალითად:

**{"saxel":"სანდრო","gvari":"ჯიბუტი","mamis\_saxeli":"აკაკი"}**

ზოგადად -



whitespace - შორისი, string - სტრიქონული, value - მნიშვნელობა.

**JSON-ის მასივი** ჩაიწერება კვადრატულ ფრჩხილებში, ისევე როგორც JavaScript-ს მასივში, შესაძლებელია ჩაიწეროს ობიექტები, მაგალითად თანამშრომლების მასივი:

"**TanamSromlebi":[**

**{"saxel":"სანდრო","gvari":"ჯიბუტი","mamis\_saxeli":"აკაკი"},**

**{"saxel":"ანდრო","gvari":"კვაშალი","mamis\_saxeli":"გიორგი"},**

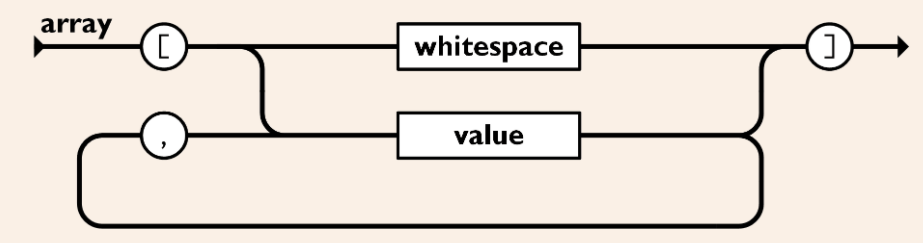
**{"saxel":"ანა","gvari":"ბრეგვაძე","mamis\_saxeli":"ნოდარი"}**

**]**

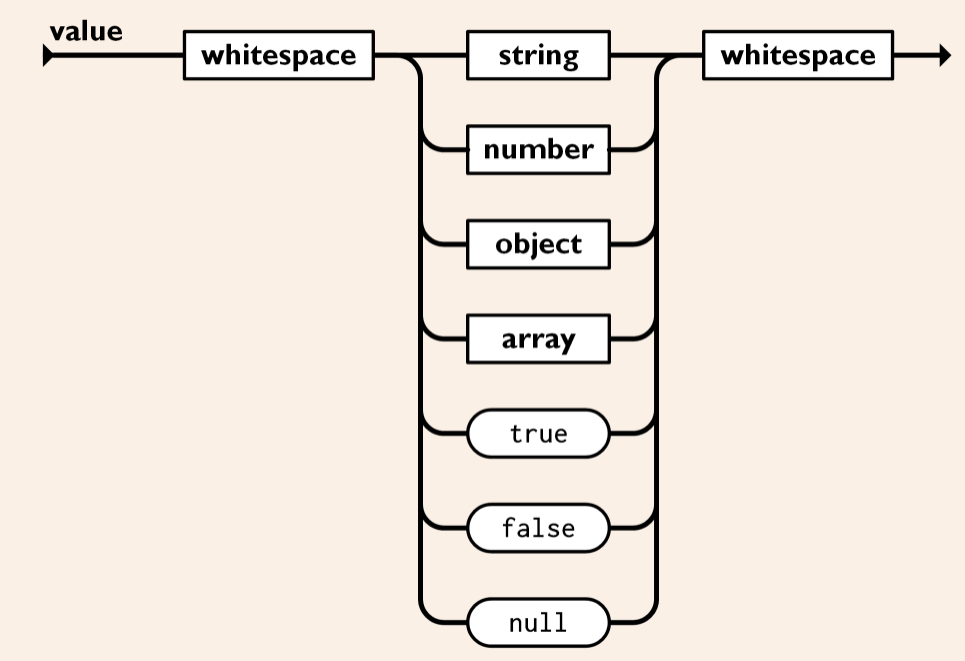
ამ მაგალითში, მასივი "თანამშრომლები" შედგება სამი ობიექტისგან. ამასთან, ყოველი ობიექტი არის ჩანაწერი კონკრეტული პიროვნების მონაცემების შესახებ.

ან განვიხილოთ ასეთი მასივი - [7,-2,10]. აქ მონაცემები (მნიშვნელობები) არის ერთი და იგივე ტიპის და მკაცრად სრტუქტურირებულია, განსხვავებით ობიექტისგან, სადაც წყვილის - გასაღები/მნიშვნელობის მდებარეობას არსებითი მნიშვნელობა არა აქვს.

ზოგადად -



განვიხილოთ თუ რა სინტაქსისი აქვს მნიშვნელობას, მონაცემს (value), და რა ტიპისაა იგი. მონაცემი შეიძლება იყოს შემდეგი ტიპის: სტრიქონული, რიცხვითი, ლოგიკური (true -კი, false - არა მნიშვნელობებით), null, ობიექტი, მასივი. აქედან პირველი ოთხი არის მარტივი ტიპის მონაცემი; ობიექტი და მასივი კი რთული ტიპის მონაცემია.



**სტრიქონული** მონაცემი - უნიკოდის ნული ან მეტი სიმბოლოების ერთობლიობა, რომელიც ორმაგ ბრჭყალებშია მოთავსებული, ამასთან შესაძლებელი არის შებრუნებული სლეშის (\) გამოყენება, როგორც ეკრანირების სიმბოლოსი.

**რიცხვითი** მონაცემი - ათობითი რიცხვი, მთელი ან წილადი, ფიქსირებული მძიმით (წერტილით), ან ექსპონენციალური სახით, მაგალითად - 3.7E+6.

**null** მონაცემი - არარსებული მნიშვნელობაც დასაშვებია.

როგორც ვხედავთ JSON ფორმატის მონაცემების შექმნისას ძალზე ყურადღებით უნდა ვიყოთ რათა არ დავუშვათ შეცდომა, განსაკუთრებულად სინტაქსიკური, რამეთუ მაგალითად ზედმეტი ან დაკლებული მძიმეც კი შეცდომას გამოიწვევს. ამიტომ სასარებლოა JSON კოდის შექმნისას გამოვიყენოთ რომელიმე სპეციალური რედაქტორი, რომელსაც კოდის შემოწმების საშუალება აქვს. ასე მაგალითად ონლაინ რედაქტორი ამ მისამართზე - https://jsonlint.com/.

***JSON სქემა***

როგორც გვახსოვს JSON მონაცემების გადასაგზავნად გამოიყენება. მონაცემები კი შეიძლება იყოს სტრუქტურირებული და რაც მთავარია მონაცემთა ველებში შეიძლება იყოს გარკვეული მოთხოვნები ჩაწერილი. მაგალითად, თუ გვაქვს პენსიონერთა მონაცემთა ბანკი, ცხადია, რომ პერსონათა მონაცემებში, ასაკის ველში უნდა იყოს ჩაწერილი რიცხვი მეტი ან ტოლი 65-ისა. ან, თუკი გვაქვს ადმინისტრატორთა მონაცემები, ყოველი ადნიმისტრატორის მონაცემებში ასაკი უნდა იყოს 65-ზე ნაკლები. ზოგადად, ჩანაწერის ველი შეიძლება აკმაყოფილებდეს გარკვეულ პირობას, ასევე მონაცემებიც შეიძლება იქნენ გარკვეული წესით სტრუქტურირებული. როცა საჭიროა გადაიგზავნოს ასეთი ტიპის მონაცემები, სასურველია არ გაიპაროს შეცდომა. ამიტომაც იქმნება JSON სქემა, რომელიც ამ საკითხს არეგულირებს. ამისთვის კი ისევ JSON კოდით იწერება პირობები. არსებობს სქემების შაბლონები, ან სქემა თავადაც შეიძლება შექმნათ და მერე ამ სქემის მიხედვით ხდება მონაცემების ე.წ. "ვალიდაცია" - კოდის შემოწმება სისწორეზე. კონკრეტული JSON კოდის რაიმე სქემის მიხედვით ვალიდაცია შესაძლებელია გავიაროთ ონლაინ მეთოდებით, მაგალითად შემდეგ მისამართებზე არსებულიე.წ. "ვალიდატორებით": https://www.jsonschemavalidator.net/ , https://tools.icoder.uz/json-validator.php .

***JSON ფაილი***

JSON ფორმატის მონაცემები შეიძლება ჩაწერილი იყოს .html, .php და .js გაფართოების ფაილებში, სადაც ისინი უნდა იქნენ წარმოდგენილი სტრიქონის სახით - ანუ ბრჭყალებში მოთავსებული ტექსტის სახით. JSON ფორმატის მონაცემები შესაძლებელია ასევე ჩაწერილ იქნეს ცალკე ფაილად .json გაფართოებით.

ფაილი .json გაფართოებით შეიძლება შეიქმნას notepad (ბლოკნოტის), notepad++ -ის საშუალებით. ხშირად გამოიყენება სპეციალური რედაქტორები, რომლებშიც შეიძლება სინტაქსიკური შეცდომების ნახვა, გასწორება და კოდის ფაილის (.json გაფართოებით) სახით, მაგალითად ამ მისამართზე - <http://www.jsoneditoronline.org/> .

განვიხილოთ მეთოდი, რომელიც მოგვცემთ საშუალებას გამოვიყენოთ ფაილი .json გაფართოებით იმ ფაილში, რომელსაც აქვს .js გაფართოება.

JavaScript– ის მიერ JSON ფაილის ჩატვირთვა მარტივია. უფრო მეტიც, ეს შეიძლება გაკეთდეს JQuery– ს გარეშე. აქ განვიხილავთ თუ როგორ შეგიძლიათ მარტივად შევასრულო ეს.

დავიწყოთ შემდეგი შინაარსის JSON ფაილის შექმნით. შეინახეთ ფაილის სახელი როგორც "monacemi.json", ხოლო monacemi ჯეისანის ფაილში მონაცემები იყოს შემდეგნაირი:

data = '[{"name" : "გიორგი", "secondname" : "პირველი", "age" : "32"}]';

შევქმნათ ჯავასკრიპტის ფაილი, რომელიც წაიკითხავს ჯეისან monacemi.json ფაილს, დავარქვათ ამ ჯავასკრიპტ ფაილს "myscript.js", ხოლო მისი კოდია:

function load() {

var mydata = JSON.parse(data);

alert(mydata[0].name);

alert(mydata[0].secondname);

alert(mydata[0].age);

}

მაინც უნდა ჩაწეროთ ორივე: ჯეისან და ჯავასკრიპტ ფაილები htm ფაილში, რომელიც უნდა შევქმნათ. ამ HTML ფაილს ვუწოდოთ "test.htm" სახელი, რომლის კოდია:

<!doctype html>

<html>

<head>

<title>მიზანი - JSON ფაილის ლოკალური ჩამოტვირთვა JQuery-ის გარეშე</title>

<script type="text/javascript" src="file.json"></script>

<script type="text/javascript" src="myscript.js"></script>

</head>

<body onload="load()">

მოხდა JSON ფაილის ლოკალური ჩამოტვირთვა JQuery-ის გარეშე

</body>

</html>

monacemi.json ფაილში data ცვლადი არის სტრინგის ტიპის, ხოლო myscript.js ფაილში ხდება ამ სტრინგის ცვლადის გარდაქმნა ჯავასკრიპტის მასივში და ობიექტში, რომ მერე მათზე გამოყენებულ იქნეს ჯავასკრიპტის შესაძლებლობები, კერძოდ მასივი ელემენტებზე და ობიექტების თვისებზე წვდომა. დაწვრილებით განვიხილოთ ეს გარდაქმნა.

***JSON -ის Srting-ის კონვერტირება და პირიქით***

შევნიშნო, რომ JSON ფუნქციები განსაზღვრული არაა, ამიტომ JSON ფორმის - სტრინგზე, მონაცემებზე არსებითი ქმედებების განსახორციებლად ის უნდა გარდავქმნათ javascript ელემენტად. ამისთვის კი გამოიყენება javascript-ის მეთოდი მისადაგებული JSON ობიექტზე.

მეთოდი JSON.parse() ფრჩხილებში მოთავსებულ სტრინგს გარდაქმნის javascript ელემენტად. მაგალითად :

ლისტინგი 1.

<!DOCTYPE html>

<html>

<body>

<p id="user"></p>

<script>

var s = '{"first\_name" : "Sammy", "last\_name" : "Shark", "location" : "Ocean"}';

var obj = JSON.parse(s);

document.getElementById("user").innerHTML =

"Name: " + obj.first\_name + " " + obj.last\_name + "<br>" +

"Location: " + obj.location;

</script>

</body>

</html>

ლისტინგ 1-ში მოთავსებული კოდის შესრულებისას ვიღებთ:

Name: Sammy Shark  
 Location: Ocean

ლისტინგი 2.

<!DOCTYPE html>

<html>

<body>

<p id="user"></p>

<script>

var s = '{"first\_name" : "Sammy", "last\_name" : "Shark", "location" : "Ocean"}';

var obj = JSON.parse(s);

document.getElementById("user").innerHTML =

"Name: " + obj.first\_name + " " + obj.last\_name + "<br>" +

"Location: " + obj.location;

document.write("Name:"+obj.first\_name+" ");

document.write(obj.last\_name+"<br>");

document.write("Location:"+obj.location);

document.write("<br>"+"<br>"+s);

</script>

</body>

</html>

ლისტიმგ 2-ში მოთავსებული კოდის შესრულებისას ვრებულობთ

Name: Sammy Shark  
Location: Ocean

Name:Sammy Shark  
Location:Ocean  
  
{"first\_name" : "Sammy", "last\_name" : "Shark", "location" : "Ocean"}

შედეგი იმის დემონსტრირებაა, რომ ერთ შემთხვევაში ჩვენ გვაქვს სტრიქონული ცვლადი (s), ხოლო სხვა შემთხვევაში ობიექტი (obj).

JSON.parse()-ის შებრუნებული ქმედებაა მეთოდი JSON.stringify() - ჩვენ ვღებულობ სტრიქონულ ვცლადს.

ლისტინგი 3

<!DOCTYPE html>

<html>

<body>

<p id="user"></p>

<script>

var s = '{"first\_name" : "Sammy", "last\_name" : "Shark", "location" : "Ocean"}';

var obj = JSON.parse(s);

document.getElementById("user").innerHTML =

"Name: " + obj.first\_name + " " + obj.last\_name + "<br>" +

"Location: " + obj.location;

document.write("Name:"+obj.first\_name+" ");

document.write(obj.last\_name+"<br>");

document.write("Location:"+obj.location);

document.write("<br>"+"<br>"+"s="+s);

var newS=JSON.stringify(obj);

document.write("<br>"+"<br>"+"newS="+newS);

</script>

</body>

</html>

ლისტინ 3-ი თავიდან მოცემულია s სტრინგი, რომელიც გარდაიქმნება obj ობიექტად, ხოლო შემდეგ obj ობიექტი ისევ გარდაიქმნება იგივე სტრინგად, ოღონდ სხვა სახელით - newS.

Name: Sammy Shark  
Location: Ocean

Name:Sammy Shark  
Location:Ocean  
  
s={"first\_name" : "Sammy", "last\_name" : "Shark", "location" : "Ocean"}  
  
newS={"first\_name":"Sammy","last\_name":"Shark","location":"Ocean"}