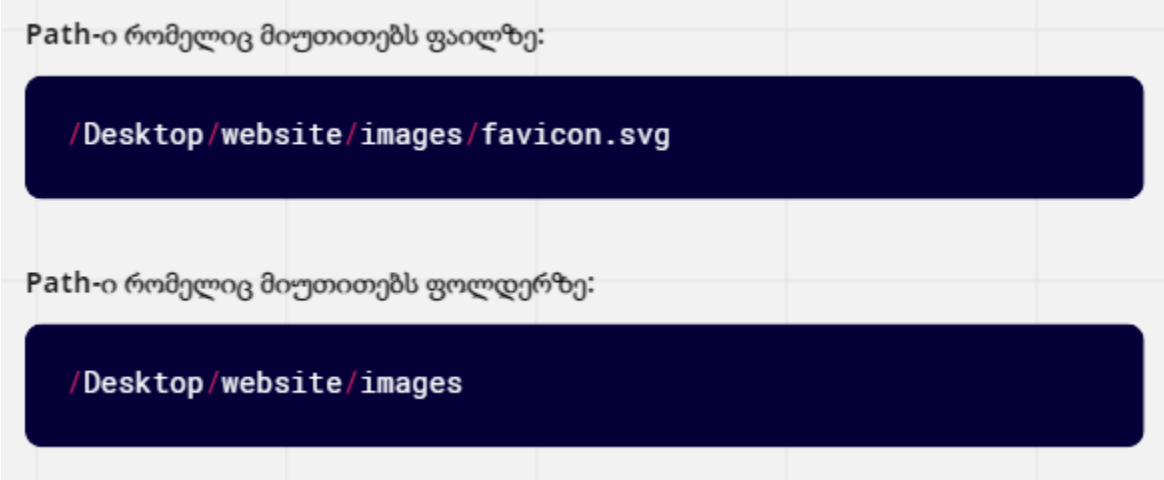
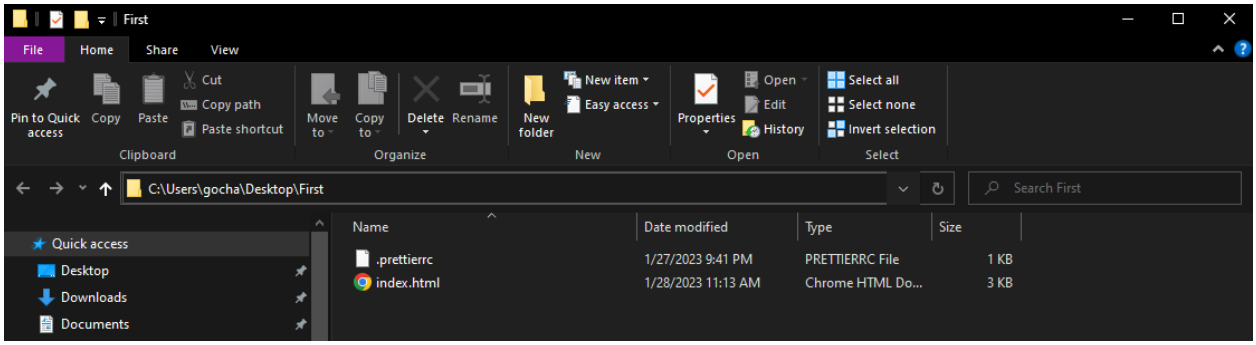
ლინკები, URL-ები და Path-ები

გამარჯობა, მიუხედავად იმისა რომ დღევანდელი ბლოგი უშუალოდ HTML-ს არ ეხება და ვისაუბრებთ ლინკებზე, URL-ზე და Path-ზე, მაინც ძალიან ყურადღებით უნდა იყო და ყველაფერი კარგად გაიაზრო,რადგან მსგავსი ფუნდამენტური საკითხების ცოდნა მომავალში თუნდაც html-ში აგვარიდებს ისეთ პრობლემებს,როგორიცაა რომელიმე ფაილისა თუ ფოლდერის მისამართის ატრიბუტისთვის არასწორად "მიმაგრება" და ა.შ. წარმოგიდგენია ქმნიდე ვებ-საიტს და არ იცოდე მისი სრული მისამართი როგორ გამოიყურება ან რა ნაწილებისგან შედგება, ან როგორ შეგვიძლია პროგრამისტობის გარეშეც,მისამართზე მანიპულირება.საიდან და როგორ მოდის მთელი ეს ინფორმაცია ჩვენს ვებ-გვერდებზე.ლინკი ალბათ დღესაც კი გაგზავნე რომელიმე მეგობართან,ან youtube-დან თუ Instagram-დან გააზიარე რომელიმე სხვა სოციალურ ქსელში რაიმე ვიდეო ან ფოტო ანუ ამით იმის თქმა მინდა,რომ დღევანდელ ბლოგში განვიხილავ საკითხებს,რომელსაც თავადაც ხშირად ეხები,თუმცა დღეიდან გააზრებული გექნება,რაც მომავალი პროგრამისტისთვის აუცილებელია.

დავიწყოთ Path-ით. Path არის ფაილის ან ფოლდერის ადგილმდებარეობის აღმნიშვნელი ტექსტური ინფორმაცია,რომელიც იერარქიის დაცვით ფოლდერებს მოუყვება სულ თავიდან ან იმ ფოლდერიდან,სადაც ვიმყოფებით → სასურველ ფოლდერამდე ან ფაილამდე.ისინი ერთმანეთისგან გამიჯნულია ყველაზე ხშირ შემთხვევაში "/" დახრილი ხაზით იგივე slash-ით, windows ფოლდერებში ეს გამიჯვნა ხდება "\" backslash-ით.ასევე Path-ში,იგივე მისამართში შევხვდებით **:** -ს, **.** -ს, **..** -ს და ამ ყველა სიმბოლოს თავისი დანიშნულება აქვს,რომელსაც გასწავლი.გარდა იმისა რომ Path გამოიყენება ფაილის ურთიერთობების წარმოსაჩენად და მისი მისამართის ვიზუალურ ვერსიას წარმოადგენს,ის ასევე აუცილებელია URL კონსტრუქციაში,რომელსაც მოგვიანებით შევეხებით ფართოდ.



მოდი გავშალოთ პირველი მაგალითი რომ პრინციპი სწორად გაიგო, არსებობს ფოლდერი Desktop,სადაც არის ასევე ფოლდერი website, რომელშიც არის ფოლდერი images და მასში ფაილი favicon.svg ,სწორედ ამ უკანასკნელის მისამართს წარმოადგენს ეს კონკრეტული მაგალითი.



ეს არის Windows ფოლდერის შემთხვევაში როგორ ვხედავთ path-ს ანუ დააკვირდი "\" backslash-ს,რითიც არის გამოყოფილი ფოლდერები ერთმანეთისგან.
ალბათ ზუსტად თუ არ იცი რას წარმოადგენს,ისე მაინც გაგიგია,რომ Windows-ის შემთხვევაში როგორც წესი გვაქვს ხოლმე C დისკი და ასევე D დისკი (Windows-ის გადამყენებლებს რომ ვეხვეწებით ხოლმე,მასზე შენახული ფოლდერები არ წაგვიშალონ 😀),შეიძლება ჰქონდეს E,F და ა.შ დისკებიც ჩვენი საჭიროებიდან გამომდინარე (ნუ ჩავულრმავდებით). აქ მთავარია დავიმახსოვროთ ის რომ Windows-ის შემთხვევაში,C დისკი არის მთავარი დისკი,რომელიც შეიცავს Windows ოპერაციულ სისტემას,რომელზეც კომპიუტერი მუშაობს და ის კომპიუტერული დამისამართების სისტემისთვის მნიშვნელობით დაახლოებით იგივეა,როგორც ჩვენი მისამართისთვის დედამიწა (არ დაიბნე, ცოტა ხანში უკეთ გაიაზრებ რასაც ვგულისხმობ). რაც შეეხება macOS-სა და Linux-ს,რადგან მათ არ გააჩნიათ C დისკისნაირი ე.წ "ფოლდერი", დამისამართებისას მათ მივმართავთ არა C:-ით,არამედ პირდაპირ /-ით ვიწყებთ,როგორც ზემოთ აღწერილ მაგალითებშია.

არსებობს path-ის 2 ტიპი: **absolute path** და **relative path**.
Absolute path ანუ იგივე სრული მისამართი არის მთლიანი (სრული) გზა, ფაილური სისტემის დასაწყისიდან იმ ფოლდერამდე ან ფაილამდე რომელზეც ვდგავართ. კომპიუტერებში ეს არის ან **C:** ან **/**-ით დაწყებული მისამართი (როცა საქმე ეხება მაგალითად macOS) ხოლო ზოგადად სერვერებზე **/**-ით დაწყებული,რომელიც აღნიშნავს ე.წ root (ფესვი) ანუ ფაილური სისტემის დასაწყისს.რა არის ფაილური სისტემის დასაწყისი და რა შუაშია დედამიწა? Windows-ის შემთხვევაში C დისკი და დანარჩენ შემთხვევებში root ფოლდერები არიან დედამიწა. წარმოიდგინე რომ ჩვენს მისამართს ასე აღვწერდეთ: ჩვენ ვცხოვრობთ **დედამიწაზე -> ევროპაში -> საქართველოში -> თბილისში -> საბურთალოზე -> ვაჟა-ფშაველას 45-ში**. იგივენაირად გამოდის მისამართის აღწერა absolute path-ის შემთხვევაში.
დაიმახსოვრე რომ თუ **/**-ით დაწყებულ მისამართს დაინახავ ანუ საქმე გაქვს absolute path-თან და არა relative path-თან (C: შემთხვევაში უფრო ადვილია არ დაგავიწყდეს,რადგან პირდაპირ გახსენებს რომ C დისკიდან იწყება).
ახლა განვიხილოთ რას წარმოადგენს relative path.ამ შემთხვევაში ათვლა იწყება იმ ფოლდერიდან,რომელშიც ვიმყოფებით და მთავრდება იმ ფაილზე ან ფოლდერზე,რომელზეც ვდგავართ (ან რომელსაც გვინდა მივმართოდ მაგალითად html-ში) .წინა მაგალითი რომ გავიხსენოთ,relative path-ის შემთხვევაში ჩვენი დამისამართება იქნებოდა შემდეგნაირი : **თბილისი -> საბურთალო -> ვაჟა-ფშაველას 45.**

```
images/favicon.svg
```

ეს არის **relative path**-ის მაგალითი,დაიმახსოვრე რომ ზუსტად იგივე შინაარსისაა ქვემოთ წარმოდგენილი მაგალითიც,რადგან წერტილი **.** აღნიშნავს იმ ფოლდერს,სადაც იმ მომენტში იმყოფები

```
./images/favicon.svg
```

თუკი დიდ პროექტზე ვმუშაობთ და პროექტის მთავარ ფოლდერში შექმნილი გვაქვს სხვა ბევრი ფოლდერი და მასში სათითაოდ კიდევ ბევრი ფოლდერი ან ფაილია განთავსებული და ჩვენი საწყისი ფოლდერისგან შორს ვიმყოფებით და გვინდა მივწვდეთ ჩვენთან ახლოსმდებარე სხვა ფოლდერსა თუ ფაილს,relative path გვაძლევს ამის საშუალებას ერთი ფოლდერით უკან გასვლის ბრძანებით ანუ იგულისხმება სადაც ვდგავართ უშუალოდ იმ ფაილის ან ფოლდერის მშობელ ფოლდერში გასვლა, რასაც ვერ გვთავაზობს absolute path და მისი გამოყენების შემთხვევაში მოგვიწევდა საწყისი ფოლდერიდან გრძელი გზის გავლა დანიშნულების ადგილამდე,როცა relative path-ით შეიძლება ერთ გადამისამართებაში მივწვდეთ იგივეს. ოგორც ხედავ ეს ბრძანება ორი წერტილით **..** - ით იწერება და მისი გამოყენება შეგიძლია იმდენჯერ,რამდენი ფოლდერითაც ხარ გასასვლელი სასურველ ფაილამდე (ან ფოლდერამდე).

```
../courses.html
```

```
../images/favicon.svg
```

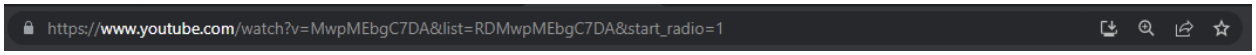
```
../../courses.html
```

დაიმახსოვრე რომ სერვერებზე,სადაც ვებ-საიტებს ანთავსებენ,როგორც წესი იყენებენ relative path-ს და არა absolute path-ს ინფორმაციის დაცულობის მიზნით (და კიდევ ბევრი სხვა მიზეზით) და სერვერს წვდომა აქვს არა უშუალოდ ჩვენი კომპიუტერის ფაილური სისტემის დასაწყისზე,არამედ კონკრეტულად ჩვენი სამუშაო ფოლდერი წარმოადგენს მისთვის ამ ფაილურ დასაწყისს.

უკვე ბევრი რამ ისწავლე დამისამართების სისტემაზე და მისი მუშაობის პრინციპებზე კომპიუტერებში ანუ path-ზე ,ახლა განვიხილოთ, როგორ მუშაობს ეს სისტემა **ინტერნეტში** ფაილების,ფოლდერებისა და ვებ-საიტების დასამისამართებისას ანუ **URL (Uniform Resource Locator)** მეთოდი.URL არის მისამართი სხვადასხვა ვებ-რესურსამდე მისასვლელად,იქნება ეს უშუალოდ ვებსაიტი,ფოტო,ვიდეო თუ სხვა ტიპის ფაილი.

ეს არის URL-ის მაგალითი,სადაც დანიშნულების წერტილი არის კონკრეტული ვებ-საიტი,ამ შემთხვევაში Facebook.com

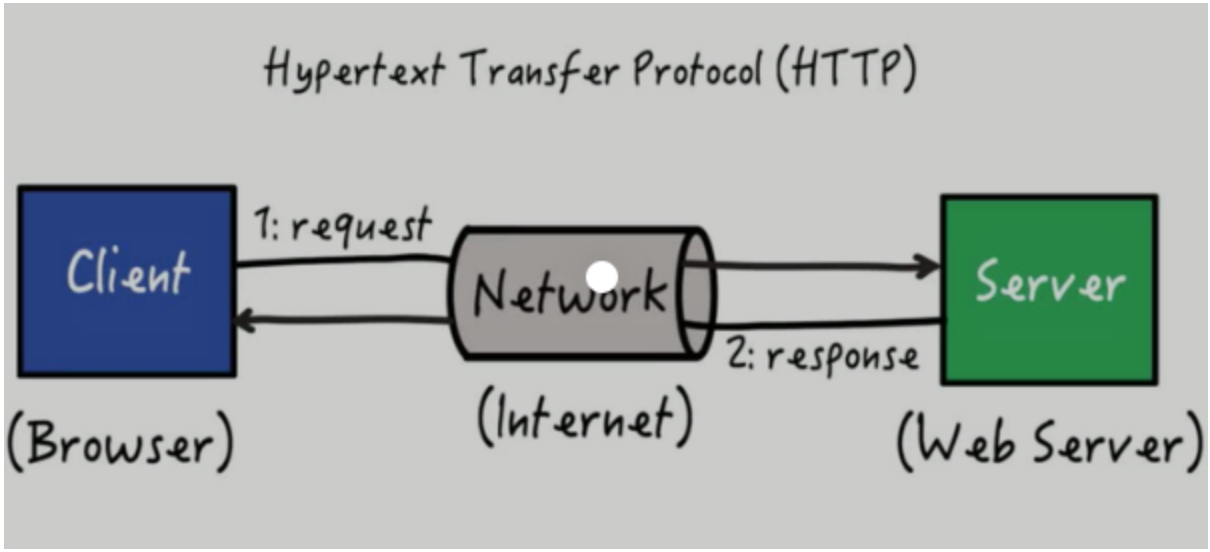
თუმცა ჩვენ უნდა ვიცოდეთ რა ნაწილებისგან შედგება ის,რა ქვია და რა დანიშნულება აქვს თითოეულ მათგანს და როდესაც დავუშვათ ასეთ URL-ს შევხვდებით,არ დავიბნეთ (არ შეგეშინდეს,არც ისე ჩახლართულია,როგორც ერთი შეხედვით ჩანს).



როგორც ხედავ URL-ის პირველი ელემენტია **https://** პროტოკოლი.როგორც ადამიანები ვამყარებთ კომუნიკაციას და ენის საშუალებით ვცვლით ინფორმაციებს,ასევე კომპიუტერები ამყარებენ კომუნიკაციას ინფორმაციის მიმოცვლის მიზნით კომპიუტერული ენების საშუალებით.პროტოკოლი არის სწორედ ასეთი,კომპიუტერული ენების სახეობა.ჩვენ ბრაუზერს **https://** სიმბოლოების გამოყენებით ვეუბნებით რომ ვაპირებთ სხვა კომპიუტერს(სერვერს) ვესაუბროთ https პროტოკოლის მეშვეობით.ამ კონკრეტულ მაგალითში ეს სხვა კომპიუტერი არის სერვერი,რომელზეც განთავსებულია novatori.ge-ს ფაილები.

უფრო ვრცლად პროტოკოლზე - იმისათვის რომ მოწყობილობებს შორის ინფორმაციის გაცვლა (კომუნიკაციის დამყარება) მოხდეს,მოწყობილობები უნდა შეთანხმდნენ მონაცემთა გაცვლის ფორმატზე.პროტოკოლი არის სწორედ ის წესების სისტემა (ერთობლიობა),რომელიც განსაზღვრავს თუ როგორ ხდება მონაცემების გაცვლა კომპიუტერებს შორის ანუ უშუალოდ პროტოკოლი განსაზღვრავს ფორმატს.

ფოტოზე წარმოდგენილია HTTP პროტოკოლის მუშაობის პრინციპი.ის გამოიყენება ისეთი რესურსების მიმოცვლისთვის,როგორიცაა html დოკუმენტები.ჩვენ კონკრეტული ბრძანების გამოყენებით ვგზავნით მოთხოვნას სერვერზე თუ რა სახის ინფორმაცია გვსურს დაგვიბრუნდეს,იქნება ეს ფოტო,ვიდეო,რეკლამა თუ ა.შ და სერვერი ამ ინფორმაციას აბრუნებს და ასახავს მომხმარებლის ბრაუზერში.

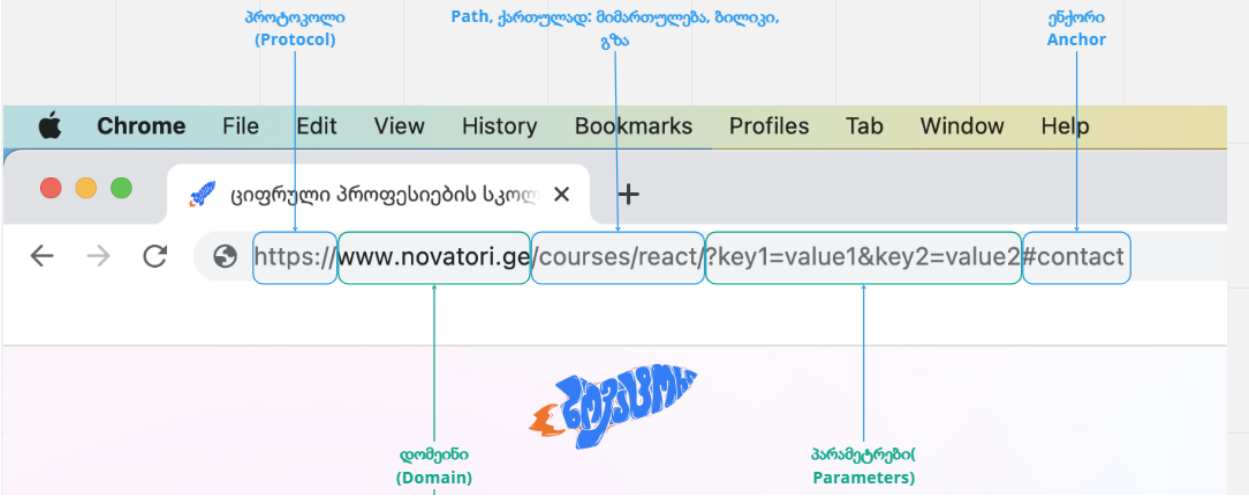


ამ ეტაპზე საიტების უმეტესობა იყენებს არა http,არამედ **https (Hypertext Transfer Protocol Secure)** პროტოკოლს,რადგან http-ს პროტოკოლით ბრაუზერიდან სერვერამდე გაგზავნილი ნებისმიერი ინფორმაცია,იქნება ეს პაროლი თუ სხვა,არის დაუშიფრავი და მისი წაკითხვა "გზაში" ადვილი იქნება ჰაკერებისთვის.https პროტოკოლით გაგზავნილი ინფორმაცია კი ჯერ იშიფრება და მხოლოდ ამის შემდეგ იგზავნება სერვერზე. არ ენდო საიტს,რომელიც დღემდე http პროტოკოლს იყენებს.

http

https

მოდი დანარჩენი URL ელემენტები ამ კონკრეტული მაგალითით გავიაროთ ⬇️



როგორც ხედავ პროტოკოლის შემდეგ მოდის დომეინი,რომლის ცნობაც ყველაზე მარტივია ხოლმე.დომეინს ჩვენ ვიცნობთ როგორც საიტის დასახელებას,სინამდვილეში ამ ჩანაწერით ბრაუზერს ვეუბნებით თუ რომელ კომპიუტერს (იგივე სერვერს) უნდა დაუკავშირდეს და უკავშირდება იმ სერვერს,რომელზეც განთავსებულია ამ შემთხვევაში www.novatori.ge-ის ფაილები.დომეინს ვებსაიტის მფლობელი ყიდულობს.როგორც წესი ძალიან დიდ კომპანიებს უკვე ნაყინი აქვთ მათ სახელთან მიმსგავსებული მრავალრიცხოვანი დომეინები,რაზეც რა თქმა უნდა არაფერია განთავსებული.მარტივ ენაზე რომ გითხრა,ვებსაიტის მფლობელმა რომ მოინდომოს თავისი საიტისთვის facebook-ის,youtube-ს და ა.შ მიახლოვებული სახელის დარქმევა,ეს არ გამოუვა და საიტს facebook17.com-ს ვერ დაარქმევს 😀

URL-ში დომეინის შემდეგ მოდის ჩვენთვის უკვე ნაცნობი path,რითიც ვიგებთ რომელ ფოლდერში იმყოფება ის კონკრეტული ვებგვერდი,რისი ჩვენებაც გვინდა. იმისათვის რომ არ დავიბნეთ და გავიგოთ სად მთავრდება მისამართი და სად იწყება პარამეტრები (ან სად მთავრდება რეალობა და იწყება ოცნება),მათ შორის გამოიყენება **?** სიმბოლო, დააკვირდი ფოტოს /courses/react/ არის path რომელიც გვეუბნება რომ ის კონკრეტული ვებ-გვერდი რისი ჩვენებაც გინდა არის courses ფოლდერში არსებულ react ფოლდერში,შემდეგ სიმბოლო ?,რომელიც გვეუბნება რომ აქ მორჩა მისამართი და გადადიხარ პარამეტრებზე.პარამეტრები როგორც ხედავ ჩაწერილია შენთვის უკვე ნაცნობი key=value-ს პრინციპით.ალბათ გინახავს youtube-ს უზარმაზარი URL (კიდევ განახებ რასაც ვგულისხმობ),ამით იმის თქმა მინდა რომ ჩვენ URL-ს შეგვიძლია ნებისმიერი რაოდენობის ინფორმაცია გავატანოთ სწორედ პარამეტრებით.იმისათვის რომ არ დავიბნეთ აქაც გვაქვს პარამეტრების ერთმანეთისგან გასარჩევად სიმბოლო **&** ანუ პარამეტრები ერთმანეთისგან გამოიყოფა end-ის აღმნიშნველი სიმბოლოთი.ჩვენს მაგალითში URL-ის ბოლო ნაწილია ენქორი - anchor,დააკვირდი,ის იწყება **#**-ით (ეს სიმბოლო გვეუბნება რომ საქმე გვაქვს ID-თან) და ამ ნაწილით url-ს ვასწავლით სადამდე უნდა ჩამოისქროლოს ჩვენი ვებ-გვერდი.ჩვენ ახლო მომავალში ვისწავლით თუ რა არის ID,რა პრინციპით ვანიჭებთ ელემენტს და როგორ ვიყენებთ მას (ეს ერთ-ერთი

მაგალითია,რასაც შემდეგ დავუბრუნდები და უფრო მარტივად გაიგებ,ახლა უბრალოდ დაიმახსოვრე რომ ეს ნაწილი არის anchor).

მოდი კიდევ ერთი მაგალითი ვნახოთ და "დავშალოთ" URL მარტივ მამრავლებად და შემდეგ თავადაც შეგიძლია გაერთო,გასწავლი როგორ.

https:// - პროტოკოლი
www.youtube.com - დომეინი
/watch - path (მიანიშნებს რომელ ფოლდერში ინახება სერვერზე ის კონკრეტული ვიდეო,რაზეც ჩვენ ვდგავართ)
? - path-ისა და პარამეტრების ერთმანეთისგან გამომყოფი სიმბოლო.
v=rUDMo7hwihs - პირველი პარამეტრი, სადაც key არის v, = ბუნებრივია არის = და rUDMo7hwihs არის უკვე value.
& - პარამეტრის მეორე პარამეტრისგან გამომყოფი სიმბოლო
list=PLKxVoO5jUTlvsVtDcqrVn0ybqBVILj2z8 - ესეც მეორე პარამეტრი.
რადგან პარამეტრის ზოგიერთი key ჩვენთვის სახელიდან გამომდინარე შინაარსობრივად გასაგებია,შეგიძლია თავად სცადო მასზე მანიპულაციები.ჩართე youtube-ზე შენი საყვარელი სიმღერების list,დადექი მისამართზე ისე რომ არა მხოლოდ youtube.com-ს არამედ სრულ URL-ს ხედავდე და მაგალითად list პარამეტრი მონიშნე და წაშალე,ნახავ რომ შენი საყვარელი სიმღერა კვლავ ჩართული დარჩება,თუმცა გაქრება playlist,რის ჩვენებაზეც პასუხისმგებელი იყო ეს კონკრეტული list პარამეტრი. გაერთე, შენ დიდებული პროგრესი გაქვს.