**ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ**

**ՀԱՄԱԼՍԱՐԱՆ**

**Կուրսային աշխատանք**

Առարկա՝ Ծրագրային համակարգերի պաշտպանություն

Դասախոս՝ Ա. Մարտիրոսյան

Ուսանող՝ Շարմազանովա Նարե

Ակադեմիական խումբ՝ ՏՏ019

***Ներածություն***

Գաղտնագրություն (կամ ծածկագիտությունը (Հունարեն κρυπτός, kryptos, «գաղտնի, թաքնված»; և γράφειν, gráphin, «գրել», կամ -λογία, -logia, «սովորել») գիտություն է ինֆորմացիայի գաղտնիության ապահովման և իսկությունը, լրիվությունը ստուգելու և ապահովելու, նաև հեղինակներին պարզելու ու հաստատելու մասին։

Ինֆորմացիան կարող է պահվել (օրինակ՝ ֆայլ կոշտ սկավառակի վրա), փոխանցվել (օրինակ՝ էլեկտրոնային հաղորդակցություն երկու կամ ավելի կողմերի միջև) կամ օգտագործվել (երբ օգտագործվում է հաշվարկի համար): Կրիպտոգրաֆիան ունի չորս հիմնական նպատակ.

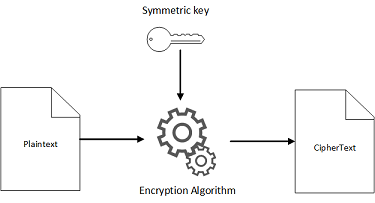
* Գաղտնիություն - տրամադրում է տեղեկատվության հասանելիություն միայն լիազորված օգտվողներին:
* Ամբողջականություն - ապահովում է, որ տեղեկատվությունը չի շահարկվել:
* Հավաստիություն - հաստատում է տեղեկատվության իսկությունը կամ օգտագործողի ինքնությունը:
* Մերժման անհնարինության ապահովում - անհնարին է դարձնում նախկին պարտավորությունների կամ գործողությունների ժխտումը:

Գաղտնագրությունը ժամանակակից հասարակության հիմքն է: Դրա վրա են հիմնված անթիվ ինտերնետային հավելվածներ, որոնք աշխատում են Hypertext Transfer Protocol Secure (HTTPS), անվտանգ տեքստային և ձայնային հաղորդակցություններով և նույնիսկ թվային արժույթներով:

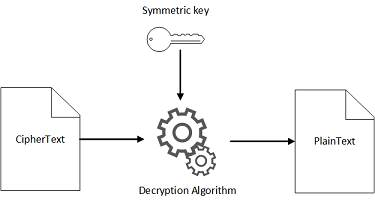
*Գաղտնագրում սիմետրիկ բանալու միջոցով*

Սիմետրիկ բանալիների կոդավորման ալգորիթմներն օգտագործում են նույն գաղտնագրային բանալիները՝ պարզ տեքստը գաղտնագրելու և գաղտնագրված տեքստը վերծանելու համար: Սիմետրիկ գաղտնագրման դեպքում հաղորդագրության բոլոր ստացողները պետք է մուտք ունենան ընդհանուր բանալին:Նկարներում ցույց է տրված, թե ինչպես են գաղտնագրումը և ապակոդավորումը աշխատում սիմետրիկ բանալիների և ալգորիթմների հետ, երբ բոլոր կողմերն օգտագործում են *նույն ընդհանուր* բանալին:

Առաջին նկարում սիմետրիկ բանալին և ալգորիթմն օգտագործվում են հաղորդագրությունը պարզ տեքստից կոդավորվածի փոխակերպելու համար:



Երկրորդ նկարը ցույց է տալիս նույն սիմետրիկ ստեղնը և սիմետրիկ ալգորիթմը, որոնք օգտագործվում են գաղտնագրված տեքստը պարզ տեքստի վերածելու համար:



*Գաղտնագրումը ասիմետրիկ բանալու միջոցով*

Ասիմետրիկ (բաց բանալիով) ծածկագրությունը ներառում է ալգորիթմների լայն շրջանակ: Դրանք հիմնված են մաթեմատիկական խնդիրների վրա:

Նման խնդիրների ամենահայտնի օրինակներից մեկը թվի գործակցման խնդիրն է՝ լավ ընտրված p և q պարզ թվերի համար մենք կարող ենք արագ հաշվարկել N=p\*q արտադրյալը։ Բայց եթե տրվի միայն N, ապա շատ դժվար է վերականգնել p և q:

Լայնորեն օգտագործվող հանրային բանալիների գաղտնագրման ալգորիթմը, որը հիմնված է ֆակտորացման խնդրի վրա:

Գաղտնագրման սխեման կոչվում է ասիմետրիկ, եթե այն օգտագործում է մեկ բանալի (բաց)՝ տվյալների գաղտնագրման համար, և մեկ այլ, բայց մաթեմատիկորեն կապված (փակ)՝ դրանք վերծանելու համար։

Անհրաժեշտ է, որ անհնար լինի հաշվարկել փակ բանալին, եթե հայտնի է միայն բաց բանալին: Հետևաբար, բաց բանալին կարող է փոխանցվել, բայց փակ բանալին կարող է գաղտնի և ապահով պահվել: Այս ստեղները կոչվում են բանալիների զույգ:

Կուրսային աշխատանքի ուսումնասիրման ընթացքում ծրագրային մասը կտրոհվի երկու հիմնական մասերի՝

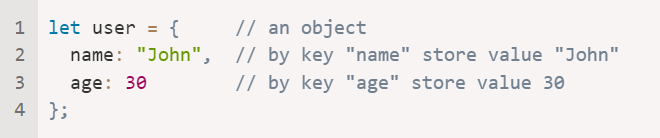
* Ստանալ Դիֆի-Հելմանի ալգորիթմը ՝ JavaScript ծրագրավորման լեզվի միջոցով
* Ապահովել կայքի անվտանգային միջավայրը՝ ստուգելով մուտք գործողի username և password –ը նախապես ներդրված տվյալների օբյեկտում

*Առաջին հատվածի իրագործումը հակիրճ*

Առաջին հատվածը իրագործելու համար պետք է կապենք մեր կայքի մարմինը՝ HTML-ը,ֆունկցիոնալ մասի հետ: Ինչպես գիտենք՝ կայքերի ինտերֆեյսի հետ աշխատանքը հիմնականում իրականացվում է JavaScript ծրագրավորման լեզվով:



Նախ ներմուծում ենք տվյալների բազա,դա կարող է լինել ցանկացած array իր objectներով,որն ունի login,email,password keyեր իրենց համարժեք արժեքներով/value-ներով:



Սեղման պրոցեսին գործառույթ տալու համար ավելացնում ենք addEventListener և աշխատանք տանում event loop-երի:

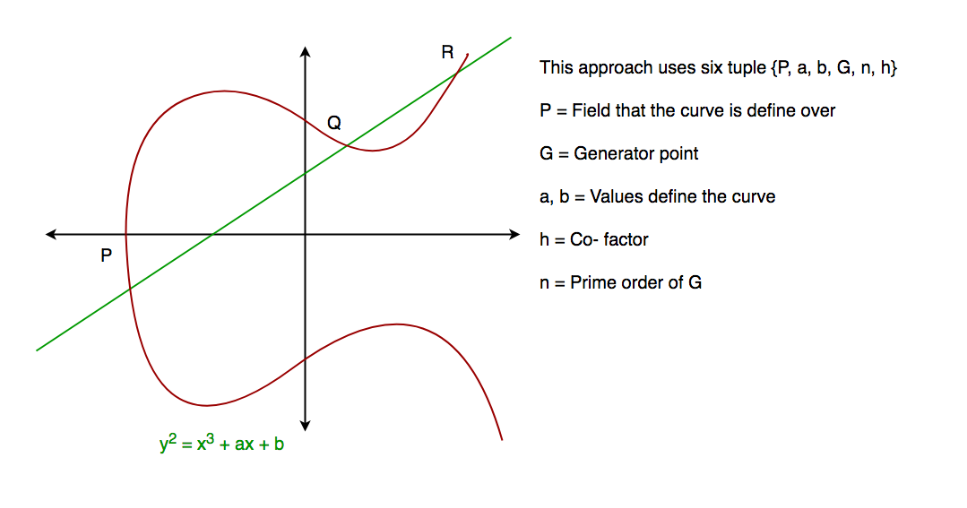


Ստուգում ենք մեր event օբյեկտի target արժեքի համապատասխանությունը վերոնշյալ գրված տվյալների բազայում:

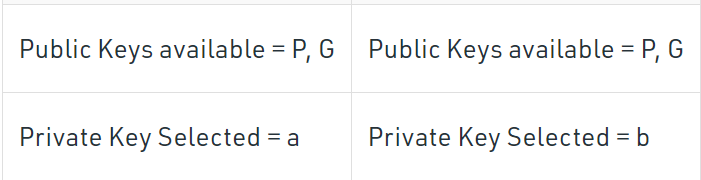
*Երկրորդ հատվածի իրագործումը հակիրճ*

Երկրորդ հատվածը իրագործելու համար պետք է JavaScript դինամիկ ծրագրավորման լեզվի շնորհիվ աշխատեցնենք Դիֆի-Հելմանի ալգորիթմը:

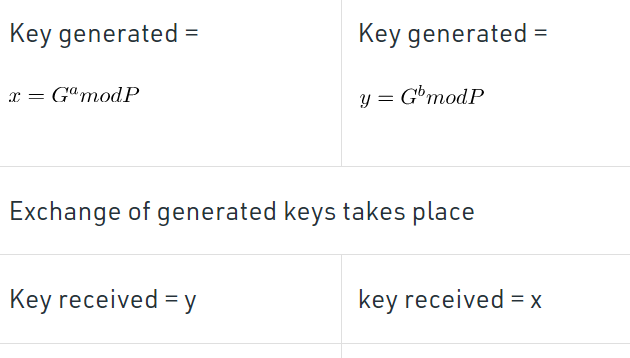
Ալգորիթմի հիմքում ընկած է բանալիների փոխանակման պրոտոկոլ: Diffie-Hellman ալգորիթմն օգտագործվում է գաղտնի հաղորդակցության համար՝ հանրային ցանցի միջոցով տվյալների փոխանակման ժամանակ՝ օգտագործելով էլիպսային կորը՝ միավորներ ստեղծելու և գաղտնի բանալին պարամետրերի միջոցով ստանալու համար:



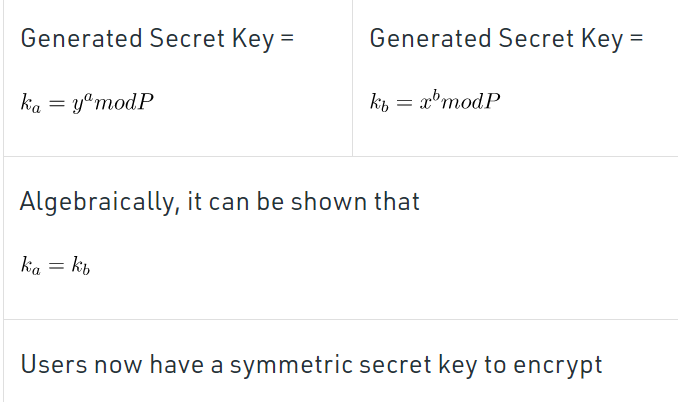
Ալգորիթմը գրելու համար մեզ հարկավոր են 4 փոփոխականներ,երկու հայտնի՝ P,G բանալիներ և երկու կողմերը վերցնում են մեկական անհայտ արժեք ՝ a,b:



Բանաձևի համաձայն, կողմերից յուրաքանչյուրը գեներացնում է բանալի,որոնք փոխանակվում են երկուսի միջև:

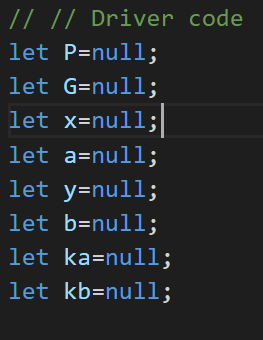


Այնուհետև մեկ այլ բանաձևի համաձայն՝ ստացված բաց բանալիներով երկու կողմերը գեներացնում են փակ բանալին,որոնց արժեքը (եթե տվյալները ճիշտ են փոխանցված) հավասար են:

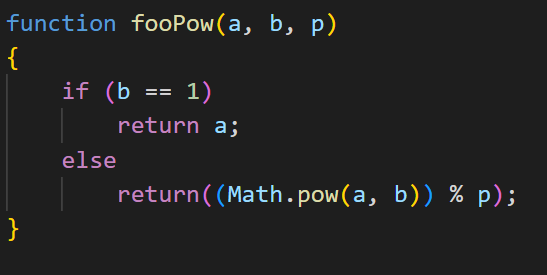


**Առաջին ծրագրի կատարում**

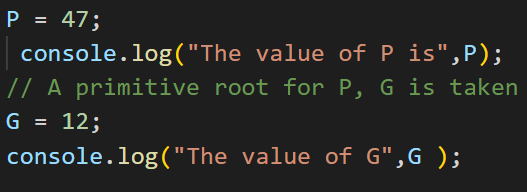
JavaScript ծրագրավորման լեզվի ինտերպրետացիան կատարվում է տող առ տող,հետևաբար սկզբում մենք ներմուծում ենք փոփոխականները: Լեզուն փոփոխականների հայտարարման 3 ձև ունի՝ let,var և const: Let հայտարարման ձևը ներմուծվել է ES6-ում,ծրագրի շրջանակներում կօգտվենք հայտարարման այդ ձևից:

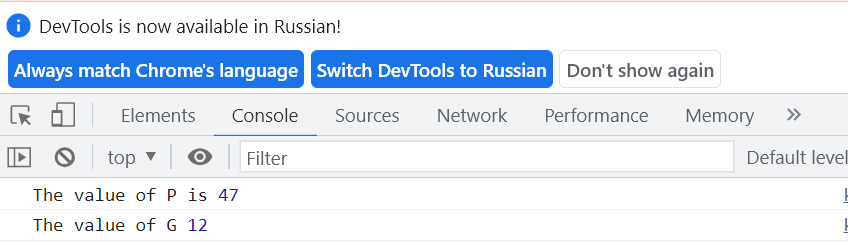


Քանի որ ի սկզբանե փոփոխականները չունեն արժեքներ,նրանց կտանք null: Function Declaration-ի միջոցով գրում ենք մեր fooPow ֆունկցիան:

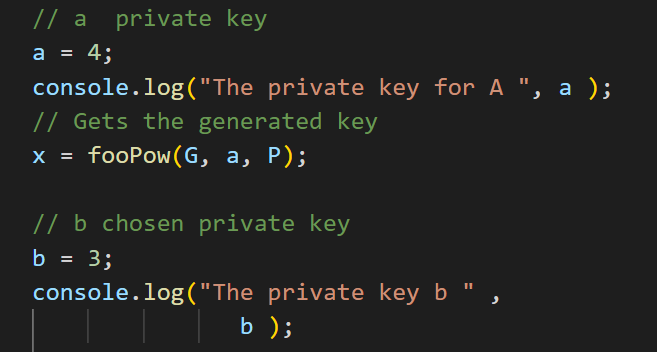


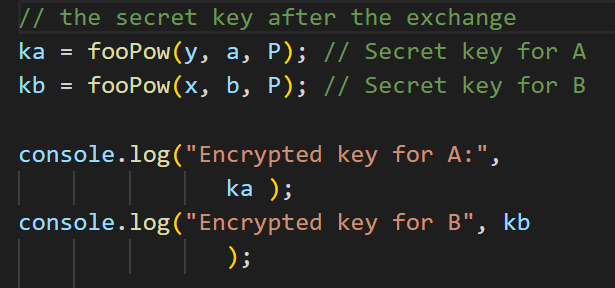
P և G public բանալիներին տալիս ենք արժեքներ և արտածում դրանք կոնսոլի վրա:



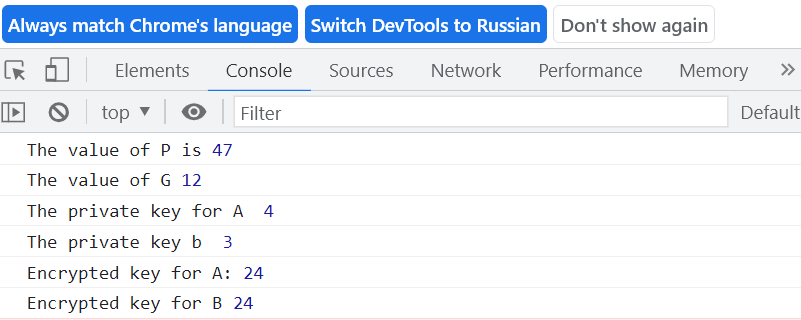


Այնուհետև տալիս ենք private a և b բանալիների արժեքները և fooPow ֆունկցիայի միջոցով ստանում x և y գեներացվող բանալիների արժեքները:



Բանալիների փոխանակումը կատարելուց հետո հաշվում ենք <<գաղտնի բանալիները>>՝ ka,kb: 

Կոնսոլում ստանում ենք հետևյալ պատկերը՝

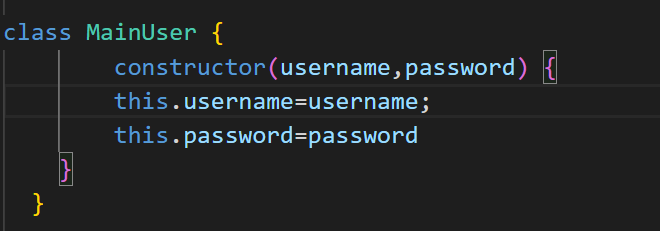


**Երկրորդ հատված**

Երկրորդ հատվածում,օգտագործելով Hyper Text Markup Language և CSS,ստանում ենք մեր կայքի Sign In կայքի արտաքին տեսքը,որն ունի username,password լրացման դաշտեր և Sign Up կոճակ: Ճիշտ տվյալներ մուտքագրելու դեպքում կայքը մեսիջ է ուղարկում,որ տվյալը true է և տպում է այն կոնսոլում:

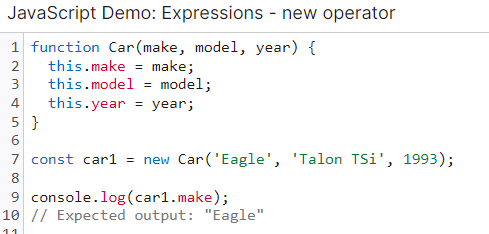
Սխալ մուտքագրման դեպքում alert գործողությունն է կատարվում,նշվում է,որ տվյալը սխալ է և տեղափոխում է մեզ մեկ այլ էջ՝ Error4o4 not found,որը ևս ստեղծվել է HTML/CSS-ի միջոցով:

Ստուգման ֆունկցիոնալ մասն ապահովվելու համար,օգտագործել եմ JavaScript ծրագրավորման լեզուն: Ստեղծում ենք MainUser կլասս,որի կոնստրուկտոր ֆունկցիան ունի 2 արգումենտ:

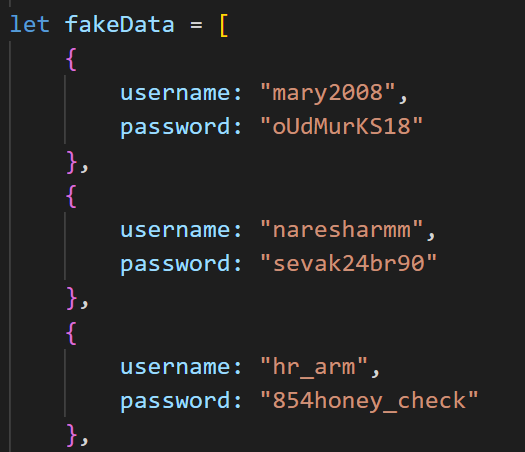


Կլասսի միջոցով մենք կարող ենք <<կանչել>> նույն օբյեկտը՝ տարբեր value-ներով,դա արվում է new օպերատորի միջոցով:

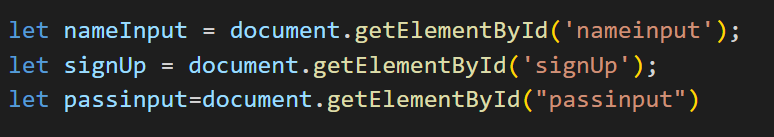
Օրինակ՝



Ստեղծում ենք fakeData անվամբ array,որն իր մեջ ունի ևս մի քանի օբյեկտներ՝ 2 key` username/password: Array ստեղծելը կանխամտածված է,չնայած որ array-ը ևս օբյեկտ է համարվում,սակայն հիմնական տարբերությունը կայանում է նրանում,որ Array-ն ունի ինդեքսացիա,հետևաբար նրա վրա կարող ենք ցիկլ կիրառել:



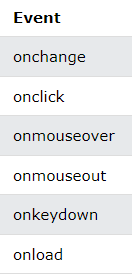
Ծրագրի ֆունկցիոնալ մասը և տեսքը,այլ կերպ ասած՝ JavaScript-ը և HTML/CSS-ը կապելու համար բախվում ենք DOM (Document Object Model)-ի գաղափարին: DOM-ն ունի իր ֆունկցիաները,որոնց միջոցով փոփոխականը կապվում է html էլեմենտին:



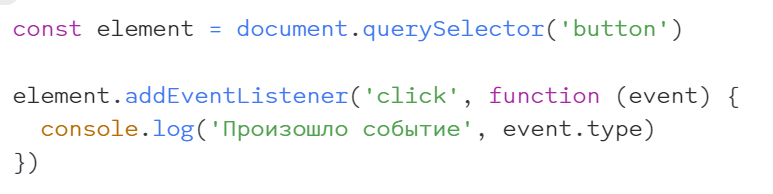


Ստեղծում ենք դեռևս դատարկ արժեքներով օբյեկտ:

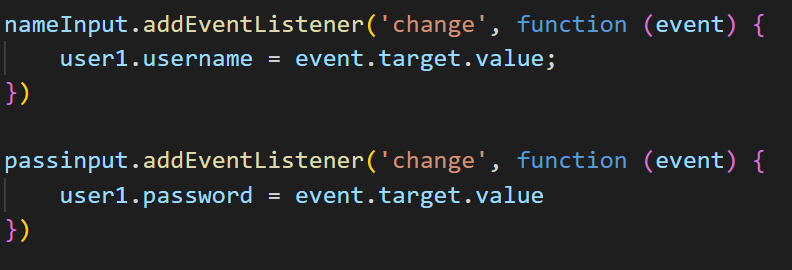
Կայքում յուրաքանչյուր գործողություն՝ մկնիկի շարժում,հաստատում,տեքստի հավաքագրում գործողություն է,JavaScript-ում այդ գործողությունները կոչվում են event-ներ և eventListener-ների միջոցով մենք կարող ենք այս կամ այն գործողության կատարման ժամանակ փոփոխություն կատարել ծրագրում:



EventListener-ի օրինակ

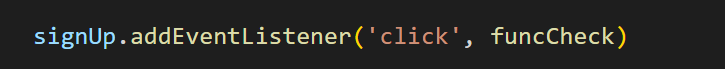


Event target-ը այն էլեմենտն է,որի վրա տեղի է ունեցել event-ը,այն ունի իր value-ները: Event.target.value-ի միջոցով մեր դատարկ user1 օբյեկտին տալիս ենք արժեքներ:

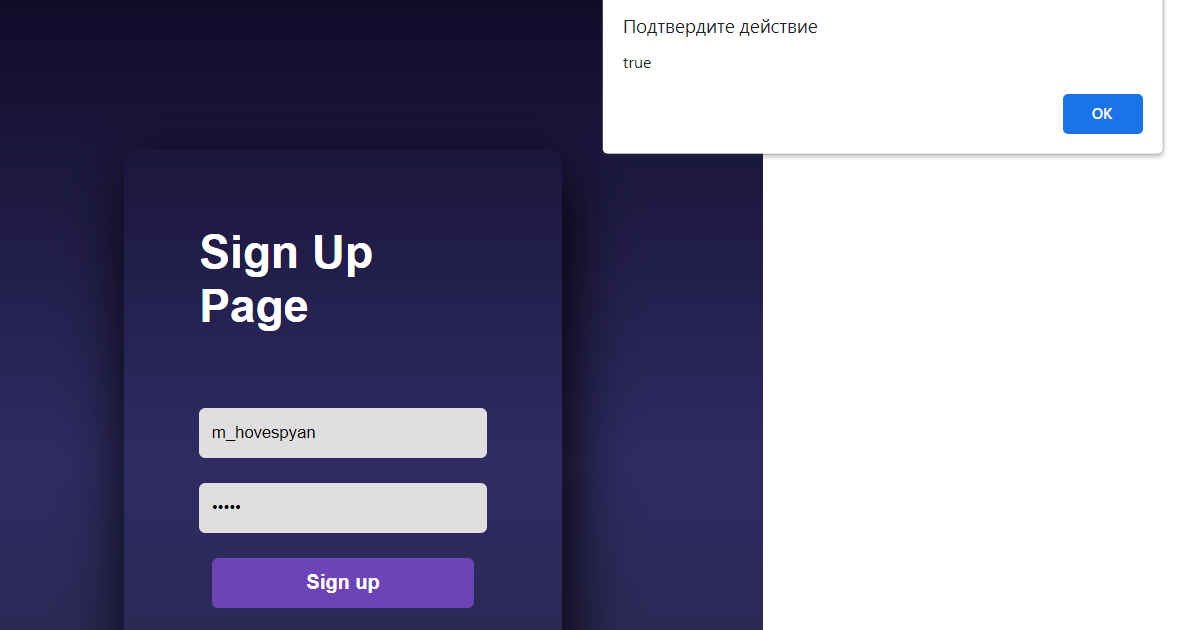


Գրում ֆունկցիա,ըստ որի եթե user1-ի data-ն (մեր կողմից հավաքվածը՝ այսինքն event.target.value-ի արժեքը) հավասար է նախապես ներմուծված fakeData օբյեկտի username/password զույգերից որևէ մեկին,ապա true է,հակառակ դեպքում՝ կայքը տեղափոխվում է error404 not found կայքէջ:

Կանչում ենք մեր ֆունկցիան



Ճիշտ մուտքագրման դեպքում ստանում ենք՝



Սխալ մուտքագրման դեպքում՝

