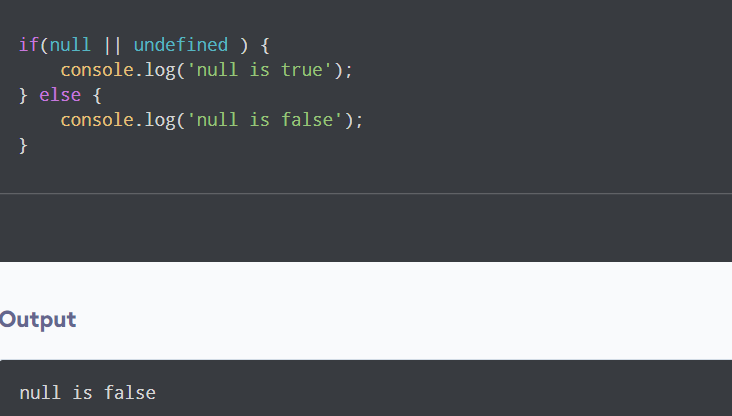
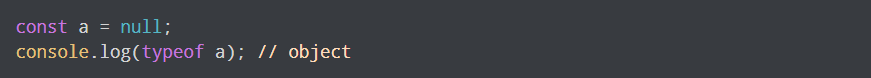
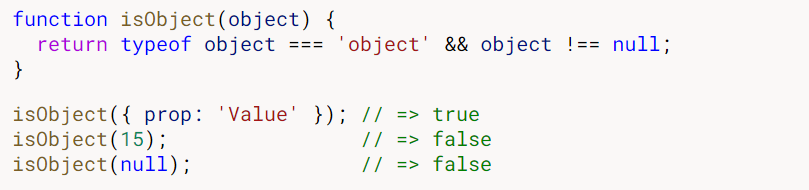
**NULL**

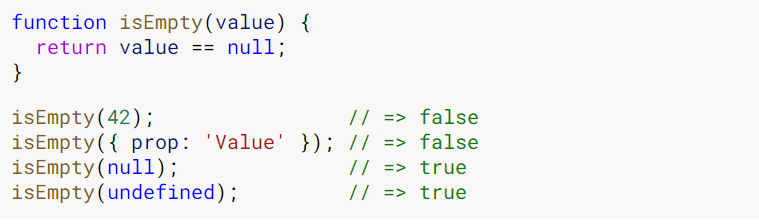
* JavaScript-ում null-ը հատուկ արժեք է, որը ներկայացնում է դատարկ կամ անհայտ արժեք:
* JavaScript-ում undefined-ը և null-ը համարվում են կեղծ արժեքներ:



* JavaScript-ում null-ը դիտվում է որպես օբյեկտ: Այն կարելի է ստուգել՝ օգտագործելով typeof օպերատորի:







* Բոլոր JavaScript ծրագրավորողները վաղուց սովոր են այն փաստին, որ typeof null === 'object'; // true, թեև null-ը իրականում պարզունակ արժեք է: Շատերը գիտեն, որ սա վրիպակ է, և Բրենդան Այքն անձամբ է դա ընդունում: Այս վրիպակը, հավանաբար, երբեք չի շտկվի՝ գոյություն ունեցող ծածկագիրը լեզվի նոր տարբերակների հետ համատեղելի պահելու անհրաժեշտության պատճառով:

Հետաքրքիր պատմություն այն մասին, թե ինչպես է դա տեղի ունեցել. Այն ունի իր արմատները լեզվի առաջին տարբերակում, մասնավորապես այն փաստը, որ փոփոխականների արժեքները պահվում էին 32-բիթանոց բջիջներում հետևյալ ձևաչափով.

29-31 բիթ. արժեքը ինքնին;

1-3 բիթ. տվյալների տիպի պիտակ;

Տիպի պիտակի ընդամենը հինգ տարբերակ կար.

000: object;

1: integer;

010: double;

100: string

110: boolean;

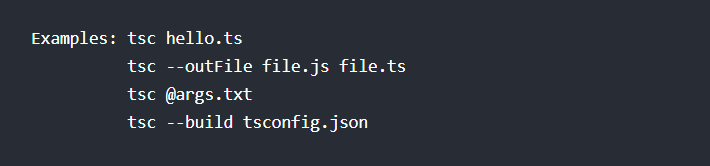
Համապատասխանաբար, եթե ամենաքիչ նշանակալից բիթը հավասար էր մեկի, ապա մնացած 31 բիթերը մեկնաբանվում էին որպես ամբողջ թիվ։ Եթե ​​0, ապա տեսակը որոշվել է կախված հաջորդ երկու բիթերի արժեքից։

Եվ այսպես եղավ, որ typeof-ը սկսեց սահմանել null-ը որպես օբյեկտ. այն ստուգեց տիպի պիտակը, որը նրան ասաց, որ null-ը ոչ այլ ինչ է, քան օբյեկտ:

**TYPESCRIPT**

TypeScript կոմպիլյատորը (նաև հայտնի է որպես «tsc») շատ փաթեթների, պլագինների և գործիքների հիմքն է, որոնք աշխատում են TypeScript-ի հետ: Այնուամենայնիվ, մենք հաճախ չենք կանչում tsc-ն ուղղակիորեն, քանի որ մենք հարմարեցնում ենք tsc-ի վարքագիծը մեր օգտագործած փաթեթավորող սարքերում:

Երբ մենք տեղադրում ենք TypeScript-ը գլոբալ npm i -g, մենք ստանում ենք գործիք, որը կոչվում է tsc: Եթե ​​մենք գործարկենք այս հրամանը help tsc --help ֆլագով, մենք կտեսնենք մեծ թվով տարբերակներ, որոնք գալիս են դրա հետ: Հիմնականները, որոնց վրա արժե ուշադրություն դարձնել, օրինակների



Եթե ​​մենք ստեղծենք hello.ts ֆայլ, ներսում գրենք const four:number = 2 + 2, ապա գործարկենք tsc hello.ts, ապա կստանանք կոմպիլացված hello.js ֆայլ՝ var four = 2 + 2; արժեքով:

Մենք հազվադեպ կօգտագործենք tsc-ն այս կերպ, քանի որ tsc-ն ներառված է պլագիններում, ինչպիսիք են Webpack, Parcel և Angular CLI:

Փոխարենը մենք օգտագործում ենք tsc --init: Այն ստեղծում է ֆայլ գրացուցակի արմատում, որը կոչվում է tsconfig.json: Այն պարունակում է բոլոր լռելյայն արժեքները: Սա շատ հարմար է, քանի որ հրամանի տողում այս բոլոր տարբերակները սահմանելու փոխարեն, երբ մենք այն կանչում ենք, մենք դրանք սահմանում ենք այս TS կազմաձևման ֆայլում: Երբ մենք գործարկում ենք tsc հրամանը, կոմպիլյատորը կվերցնի կարգավորումները tsconfig.json-ից:

Նաև մենք կարող ենք փոխել կազմաձևման կարգավորումները, օրինակ՝ փոխել ourDir-ի արժեքը ./dist-ի: Այժմ բոլոր կազմված ֆայլերը պետք է պահվեն արտաքին գրացուցակում ./dist:

Այս ֆայլում սահմանված ցանկացած բան կնշի, թե ինչ տարբերակներ կօգտագործի TypeScript կոմպիլյատորը կոմպիլյացիայի ժամանակ: Երբ TypeScript-ը սկսում է բողոքել բավարար տեսակների չօգտագործման մասին, կարող եք սկսել անջատել այդ նախազգուշացումներից կամ սխալներից մի քանիսը: