Դաս 01: JS Ներածություն var, let, const

JavaScript կոդը կարող ենք ավելացնել HTML-ում **<script>** թեգի միջոցով: Երբ բրաուզերը հանդիպում է **<script>** թեգին, այն դադարեցնում է HTML-ի կոդի ընթերցումը և անմիջապես կատարում է **JavaScript** կոդը: Դա նշանակում է, որ, եթե կոդը շատ երկար ժամանակ պահանջի աշխատանքի համար կամ ունենա սխալներ, բրաուզերը չի շարունակի էջի բեռնավորումը, մինչև **JavaScript**-ը ամբողջությամբ չկատարվի:
Եթե **JavaScript**-ը ցանկանում ենք կցել **<head>** բաժնում (էջի սկզբում), կարող ենք օգտագործել **defer** ատրիբուտը՝ ծրագրի կատարման հետ կապված խնդիրներ չունենալու համար:

```
<head>
  <script src="script.js" defer></script>
</head>
```

Փոփոխականները համակարգչի հիշողության մեջ որոշակի տեղ զբաղեցնող տարրեր են, որոնք բնութագրվում են հստակ **անունով**, **տիպով** և **արժեքով:** Փոփոխականներն օգտագործվում են ինֆորմացիա պահելու համար:
JavaScript-ում փոփոխականները հայտարարելու երեք եղանակ կա՝ **var**, **let** և **const**:

var-ը փոփոխականներ հայտարարելու հնացած եղանակ է, որը գոյություն ունի **JavaScript**-ի ամենավաղ տարբերակներից ի վեր: Այն ունի մի շարք առանձնահատկություններ, որոնք կարող են հանգեցնել կոդի սխալների և դժվարացնել դրանց հայտնաբերումը: **Var**-ի հիմնական խնդիրը բլոկային տեսանելիության բացակայությունն է: **Var**-ով հայտարարված փոփոխականները հասանելի են կոդի բլոկի սահմաններից դուրս, ինչը կարող է հանգեցնել անկանխատեսելի արդյունքների:

let-ը փոփոխականներ հայտարարելու ավելի ժամանակակից եղանակ է, որը ներդրվել է **ES6-ում** (**ECMAScript 2015**): **Let**-ի հիմնական առանձնահատկությունը բլոկային տեսանելիությունն է: Բացի այդ տվյալ փոփոխականի տիպը չի թույլատրում, որ միևնույն բլոկում հայտարարվի նույն անունով մեկից ավելի փոփոխական:

const-ը նույնպես փոփոխականներ նոր եղանակ է: Ինչպես **let**-ը, այնպես նաև **const**-ը ունի բլոկային տեսանելիություն և տվյալ փոփոխականի տիպը չի թույլատրում, որ միևնույն բլոկում հայտարարվի նույն անունով մեկից ավելի փոփոխական: Բացի այդ, **const** փոփոխականները չեն կարող փոփոխվել հայտարարելուց հետո:

JavaScript-ում տվյալների տիպերը բաժանվում են երկու հիմնական խմբի՝ **պրիմիտիվ (primitive)** և **ոչ պրիմիտիվ (non-primitive)** տիպեր:

Console **մեթոդներ**

console.log() – տպում է սովորական տեղեկություններ:
console.warn() – ցուցադրում է նախազգուշացում:
console.error() – ցուցադրում է սխալ:

Պրիմիտիվ տիպեր

Պրիմիտիվ տիպերն այն տիպերն են, որոնք անմիջապես պարունակում են իրենց արժեքը և փոփոխելի չեն:

1. **Number** – թվեր, օրինակ՝ **let age = 25;**
2. **String** – տեքստային տիպ, օրինակ՝ **let name = "Aram";**
3. **Boolean** – երկու արժեք՝ **true** կամ **false**, օրինակ՝ **let isActive = true;**

4. **Null** – հատուկ տիպ, որն ունի միայն մեկ արժեք՝ **null**, որը ցույց է տալիս, որ փոփոխականը դատարկ է, օրինակ՝ **let result = null;**

5. **Undefined** – նշանակում է, որ փոփոխականը հայտարարվել է, բայց արժեք չի ստացել, օրինակ՝ **let data;**

6. **Symbol** – եզակի արժեքով տվյալ տիպ, որն օգտագործվում է հիմնականում օբյեկտներում հատկությունների իդենտիֆիկացիայի համար, օրինակ՝ **let id = Symbol("id");**

7. **BigInt** – մեծ ամբողջ թվերի համար նախատեսված տիպ, օրինակ՝ **let bigNumber = 1234567890123456789012345678 90n;**

Ոչ պրիմիտիվ տիպեր (Օբյեկտներ)

JavaScript-ում բոլոր ոչ պրիմիտիվ տիպերը օբյեկտներ են, ինչը նշանակում է, որ դրանք պահպանում են ոչ թե անմիջական արժեքը, այլ հղում (**reference**):

Object – JavaScript-ում հիմնական ոչ պրիմիտիվ տիպն է, որն օգտագործվում է հավաքական տվյալներ պահելու համար:
Array – օբյեկտի հատուկ տեսակ, որը թույլ է տալիս պահել տվյալների ցուցակ:
Function – օբյեկտի տեսակ, որը թույլ է տալիս կատարելու գործողություններ:

Տրամաբանական բլոկներ (if, else if, else)

JavaScript-ում պայմանի ստուգումներն արվում են **if**, **else if** և **else** բլոկներով:

```
let score = 85;

if (score >= 90) {
  console.log("Excellent!");
} else if (score >= 75) {
  console.log("Good Job!");
```

```
} else {
  console.log("Keep Trying!");
}
```

Տրամաբանական օպերատորներ (&&, ||, ??, >, <, >=, == ...)

JavaScript-ում կան մի քանի լոգիկական օպերատորներ, որոնք օգտագործվում են պայմաններ ստուգելու համար:

&& օպերատորը վերադառնում է **`true`** միայն այն դեպքում, երբ երկու արտահայտությունները ճիշտ են: Եթե առնվազն մեկը **`false`** է, ապա արդյունքը **`false`** է:

|| օպերատորը վերադարձնում է **`true`**, եթե առնվազն մեկը ճիշտ է: Եթե երկու արտահայտությունները **`false`** են, ապա արդյունքը `false` է:

?? օպերատորը վերադարձնում է աջ կողմի արժեքը միայն այն դեպքում, եթե ձախ կողմի արժեքը **null** կամ **undefined** է: Այն թույլ է տալիս սահմանել **default** արժեքներ:

! օպերատորը օգտագործվում է արտահայտության արժեքը հակադարձելու համար: Եթե արտահայտությունը ճիշտ է, ապա այն դառնում է սխալ և հակառակը:

ES6-ում (ECMAScript 2022)

??= օպերատորը նշանակում է ձախ կողմի փոփոխականին նոր արժեքը միայն այն դեպքում, երբ տվյալ փոփոխականը **`null`** կամ `undefined` է: Այն միավորում է **??** օպերատորը նշանակման հետ:

&&= օպերատորը նշանակում է փոփոխականին նոր արժեքը միայն այն դեպքում, երբ այն ճշմարիտ է (**true**): Եթե փոփոխականը սխալ է (**false**), ապա նշանակումը չի կատարվում:

`||=` օպերատորը նշանակում է փոփոխականին նոր արժեքը միայն այն դեպքում, երբ այն սխալ է (**false**): Եթե փոփոխականը ճշմարիտ է (**true**), ապա նշանակումը չի կատարվում:

Կարող ենք ստեղծել սեփական սխալները, նշելով սխալի տեքստը և առաջացման պատճառը:

```
throw new Error("Something went wrong", {cause:
"Network issue"})
```