hillel IT school

Lesson 2

План заняття

- Оголошення змінних.
- Сплив змінних (hoisting).
- Порівняння var, let, const.
- Типи даних.
- Шаблонні рядки.

Зарезервовані слова

- let
- const
- break
- continue
- for
- function
- case
- default
- delete
- return
- switch
- try
- true
- false
- null
- new
- if
- else
- while

Області видимості

Область видимості - важлива концепція, що визначає доступність змінних. Ця концепція є основою замикань, поділяючи змінні на глобальні і локальні.

- Глобальна - змінна доступна у будь якому місті коду;
- Локальна - змінна доступна тільки у своєму блоці (scope);

Області діляться на:

Блочна - блок коду JavaScript визначає область видимості змінних, оголошених за допомогою ключових слів const і let;

Область функцій - функції JavaScript створюють область видимості для всіх змінних, незалежно від того, за допомогою якого ключового слова вони оголошені (var, const або let);

Область модуля - модулі ES6 також створюють область видимості для змінних, функцій та класів;

Вложена область - ви можете створювати одну область в іншій;

Глобальна - глобальна область видимості є зовнішньою областю. Вона доступна для будь-якої внутрішньої чи локальної області видимості. У глобальному браузері є область видимості, створювана при завантаженні JavaScript-файлу, вказаного в атрибуті src тега script;

Лексична - лексична область видимості означає, що доступність змінних визначається статично положенням даних змінних всередині області видимості вкладеної функції: змінні з області видимості зовнішньої функції доступні області видимості вкладеної функції.

Hosting - підняття

По суті, коли JavaScript компілює весь код, всі оголошення змінних, що використовують **var**, піднімаються/hoisted у верхню частину їхньої функціональної/локальної області видимості (якщо оголошується всередині функції) або в глобальну область видимості (якщо оголошується поза функцією) незалежно від того, де було зроблено фактичну декларацію.

Оголошення змінних і функцій розміщуються на етапі компіляції, але вони залишаються саме там, де ми ввели їх у свій код.

Отже, під капотом відбувається таке: на етапі створення, двигун JavaScript переглядає код і, як тільки він бачить ключове слово var або ключове слово function, він виділяє деяку пам'ять для них.

var - let - const

var:

- var має сферу дії функції;
- оголошення var піднімаються, але не ініціалізуються.
- Також, якщо ти створиш var на верхньому рівні (глобальний рівень), буде створено властивість для глобального об'єкта; у випадку з браузером – це об'єкт window.

let & const - мають область видимості блоку.

Різниця між оголошеннями var/function та оголошеннями let/const/class полягає в ініціалізації. Перші ініціалізуються з невизначеним значенням невизначеного. Однак другі, лексично оголошені змінні, залишаються не ініціалізованими. Це означає, що ReferenceError викидається під час спроби доступу до них. Вони будуть ініціалізовані лише після того, як оператори let/const/class будуть визначені. Все що до, називається тимчасовою мертвою зоною.

Тимчасова мертва зона (Temporal dead zone) - це не синтаксичне місце розташування, а час між створенням змінної (області) і ініціалізацією.

Типи данних

1. Число - Number;
2. BigInt - число вільної довжини;
3. Строка - String;
4. Логічний тип - Boolean;
5. null;
6. undefined;
7. Symbol;
8. Об`єкт - Object;

Перші сім це прімітиви, передаються за знченням.

Об'єкт це особий тип який передається за посиланням.

Підвиди об’єкту:

1. Масив - Array
2. Функція - Function

Number

Представляє як ціле число так і число з плавоючьою точкою.

До числа відносять також:

- Infinity - є математичною нескінченністю ∞.
- -Infinity
- NaN - означає обчислювальну помилку. Це результат неправильної чи невизначеної математичної операції.

Звичайні числа JavaScript зберігаються в 64-бітному форматі IEEE-754, який також називають «числа з плаваючою точкою подвійної точності» (double precision floating point numbers).

BigInt числа дозволяють працювати з цілими числами довільної довжини. Вони потрібні досить рідко і використовуються у випадках, коли необхідно працювати зі значеннями більш ніж (2n53-1) або менш ніж (2n53-1).

String

String у JavaScript повинна бути укладена в лапки:

- ''
- ""
- `` - string themplate. Вираз усередині ${…} обчислюється, і його результат стає частиною string;

Boolean

Булевий тип (boolean) може приймати лише два значення: true (істина) та false (брехня).

null

Спеціальне значення null не відноситься до жодного з типів, описаних вище. Це просто спеціальне значення, яке є «нічим», «порожнім» або «значенням невідомим». Але воно резервує ячейку пам'яті у системі.

undeined

Воно означає, що значення не було присвоєно. Пам'ять не резервується у системі.

Якщо змінна оголошена, але їй не надано жодного значення, то її значенням буде undefined.

Object – reference type

Всі інші типи називаються «примітивними», тому що їх значеннями можуть бути лише прості значення (чи то рядок, чи число, чи щось ще).

У об'єктах зберігають колекції даних чи складніші структури.

Symbol

Тип символу (symbol) використовується для створення унікальних ідентифікаторів в об'єктах.

Оператор typeof

typeof() - повертає тип аргументу. Це корисно, коли хочемо обробляти значення різних типів по-різному чи просто хочемо зробити перевірку.

Використання:

- typeof variable; // return type
- typeof(variable); // return type

Запам'ятайте:

1. typeof null є "object". Це офіційно визнана помилка в typeof, що веде початок з часів створення JavaScript і збережена для сумісності.
2. typeof function() буде function, хоча такого типу даних не існує.

Дякую за увагу