**ПЕРЕМЕННЫЕ**

Ключевые слова var, let и const

Переменная – это идентификатор для хранения данных. С помощью переменной можно получить значение по ее имени (идентификатору).  
Ключевые слова для объявления переменных:  
*var  
let  
const*  
Другими словами, переменные являются «контейнерами», в которые помещается информация. На самом деле, переменная являет собой выделенную область памяти программы, в которой хранятся данные. Объявление переменной:

var message;

Значения в переменную присваиваются при помощи оператора присваивания (равно =):

message = "Hello world!";

При обращении к переменной по имени мы получаем ее значение.

console.log(message); //"Hello world!"

После объявления переменной можно одновременно присвоить ей значение:

const message = "Hello world!";

const (или var или let) – ключевое слово для определения переменной.  
message – имя переменной.  
«Hello world!» – значение переменной.  
Переменные можно объявлять последовательно:

let user = "John";

let age = 25;

Либо с помощью одного ключевого слова, определить несколько переменных:

let user = "John", age = 25;

Именование переменных

* • имя может состоять из буквы, цифры, символов $ и \_;
* • первый символ не должен быть цифрой;
* • регистр букв имеет значение (message и Message разные переменные);
* • нельзя использовать зарезервированные имена (class, delete, return и т.д.);
* • кириллица не запрещена, но не рекомендуется.

Используйте английский, никакого транслита:

var spisokZakazovPolzovatelya; // DO NOT USE THIS

var userOrders;

Используйте camelCase:

var user-settigs; //DO NOT USE THIS

var userSettings;

Не включайте в имя тип данных и случайные символы. Имя должно описывать что хранится в переменной:

var listOfObjects; //DO NOT USE THIS

var string11, asd, qwe; //DO NOT USE THIS

var itemsList, userName, currentDate;

Hoisting (поднятие переменных)

Hoisting означает, что перед тем как выполнится, JavaScript код анализируется (инициализация JavaScript) и все найденные переменные поднимаются на верх без своих значений. Таким образом, в начале фазы выполнения все переменные уже инициализированы, но значения им будут присвоены позже.  
На самом деле переменные физически не перемещаются в начало кода, они фиксируются в памяти программы, и она (программа) уже знает, что такие переменные существуют.  
Этот код

var x = 1;

console.log(x + " " + y); // "1 undefined"

var y = 2;

интерпретируется следующим образом

var x = 1;

var y; // "поднятая" переменная

console.log(x + " " + y); // "1 undefined"

y = 2;

При обращении к переменной до ее определения с помощью var ее значением будет undefined.

console.log(data); //undefined

var data = 10;

Но переменные определенные с помощью let и const не «поднимаются».

console.log(newData); //Uncaught ReferenceError: newData is not defined

console.log(PI); // Uncaught ReferenceError: PI is not defined

let newData = 15;

const PI = 3.14;

Область видимости

Область видимости – это та часть приложения, в которой «действует» или «видна» объявленная переменная. То есть, часть кода в которой, при обращении к переменной, получим ее корректное значение. Детальнее об этом в теме «Функции».  
Объявленные с помощью var («функциональная» область видимости):

var framework = "Angular";

if (true) {

var framework = "React";

console.log(framework); //"React"

}

console.log(framework); //"React"

Объявленные с помощью let (блочная область видимости):

let framework = "Angular";

if (true) {

let framework = "React";

console.log(framework); // "React"

}

console.log(framework); // "Angular"

Переменные, объявленные при помощи var «видны» везде, а переменные объявленные при помощи let или const видны в середине блока кода, который создается при помощи фигурных скобок ( {} ), за пределами блочной области такие переменные не «видно».

Особенности

Если объявить переменную без ключевого слова, то она попадет в глобальную область видимости. Глобальная область видимости – область или место в памяти JavaScript, к которой имеют доступ любые части приложения. Этого делать не рекомендуется. Детальнее об этом в теме «Функции».  
Работая в режиме «use strict», объявить переменную без ключевого слова не получится, будет ошибка. «use strict» – строчка, которая обычно указывается в самом начале js файла. Она дает команду интерпретатору, что код написанный в этом файле полностью соответствует стандарту ES5.

"use strict";

name = "John"; // ReferenceError

Значения переменных можно изменять:

var message = "Hello world!";

message = "Hello JavaScript!";

Значения переменных можно копировать из одной в другую:

let message, greetings;

message = "Hello world!";

greetings = message;

Особенности var

* • способ объявления существовал всегда (этот способ существует с первой версии JavaScript);
* • поднимаются (функционирует механизм hoisting);
* • функциональная область видимости;
* • можно объявить переменную с таким же именем (переопределить) (но не в режиме “use strict”).

var storage = 10;

var storage = "This is text";

Особенности let

* • способ объявления был добавлен в спецификацию ES6;
* • не поднимаются (отсутствует понятие hoisting);
* • блочная область видимости (внутри фигурных скобок);
* • нельзя объявить переменную с таким же именем (переопределить).

let totalCount = 10;

let totalCount = 15; // SyntaxError: redeclaration of let newData

Особенности const

Переменные при помощи const определяются, когда необходимо определить константу – переменную, значение которой никогда не меняется. До введение ключевого слова const в ES6 переменных, которые нельзя изменить, не существовало. Поэтому использовалось ключевое слово var, но имя записывалось заглавными буквами и с нижним подчеркиванием, что бы другие разработчики знали, что это константа и ее «нельзя» изменять.

var PI\_NUMBER = 3.14;

Константы:

* • не поднимаются (отсутствует понятие hoisting);
* • нельзя объявить переменную с таким же именем;
* • нельзя изменить значение переменной;

const framework = "Angular";

framework = "React"; // Uncaught TypeError: Assignment to constant variable.

* • нельзя объявить переменную без значения (инициализировать).

const framework; // Uncaught SyntaxError: Missing initializer in const declarati