Front-end

JavaScript

JS. Hoisting

- Поднятие

JS. Hoisting

Поднятие или hoisting — это механизм в JavaScript, в котором переменные и объявления функций, передвигаются вверх своей области видимости перед тем, как код будет выполнен.

Как следствие, это означает то, что совершенно неважно где были объявлены функция или переменные, все они передвигаются вверх своей области видимости, вне зависимости от того локальная она или же глобальная.

Стоит отметить то, что механизм “поднятия” передвигает только объявления функции или переменной. Назначения переменным остаются на своих местах.

JS. Hoisting

Поднятие (hoisting) — это поведение, при котором функцию или переменную можно использовать до объявления.

Поднятие предполагает, что объявления переменных и функций физически перемещаются в начало кода — «поднимается».

JS. Hoisting

Например:

```
console.log(test);   // вывод: undefined

// объявляем переменную test

var test;
```

Эта программа выведет в консоль значение undefined.

Аналогично ведет себя такая программа:

```
var test;

// используем переменную

console.log(test); // вывод: undefined
```

Поскольку переменная test объявлена, но не инициализирована (не присвоенозначение), JavaScript определяет ее как undefined.

JS. Hoisting

Не поддерживает инициализацию

JavaScript не поддерживает поднятие с инициализацией. То есть так hoisting работать не будет:

```
console.log(a); // =>  undefined

var a = 5;
```

JS. Hoisting

С let нельзя

Если переменная объявлена с помощью ключевого слова let, применить к ней hoisting нельзя.

```
a = 5;

console.log(a);

let a; // => ошибка
```

Вывод: Uncaught ReferenceError: Cannot access 'a' before initialization

При использовании let сначала объявление, потом использование.

JS. Hoisting

Внутри функции — поднятие только до начала функции

Когда переменная используется внутри функции, ее можно поднять только в начало функции

```
var a = 4;

function greet() {
    b = 'привет';
    console.log(b);
    var b;
}

greet(); // => привет
console.log(b); // => Uncaught ReferenceError: b is not defined
```

При объявлении переменной с поднятием переменная доступна только в той области видимости, в которой она объявлена.

JS. Hoisting

Поднятие функций

Функцию можно также вызвать до ее объявления.

```
greet();

function greet() {
    console.log('Приветики');
}
```

JS. Hoisting

С функциональными выражениями нельзя

Если определить функцию в виде выражения при поднятии, возникнет ошибка. Причина: поднимаются только объявления.

```
greet();

let greet = function () {
    console.log('Приветики'); // => Uncaught ReferenceError: greet is not defined

};
```

Использование var здесь не поможет. Все равно будет ошибка

JS. Hoisting

Важно учитывать, что hoisting может привести к нежелательным последствиям в вашей программе. Поэтому лучше сначала объявлять переменные и функции перед использованием — избегать поднятий.

А в случае с переменными лучше использовать let, а не var.

JS. Hoisting

Не стоит привыкать к var. Использование этого ключевого слова считается плохой практикой по нескольким причинам.

Отсутствие блочной области видимости. Var-переменная, созданная в блоке if-else или цикле, доступна за пределами своего блока. Например, в C++ или Java не получится получить доступ к счётчику цикла нигде, кроме тела этого цикла, а вот в JavaScript, если счётчик объявлен как var, — получится легко. Это практически всегда приводит к труднораспознаваемым логическим ошибкам.

Повторное объявление. Переменные var можно создавать повторно, и компилятор не будет ругаться. Почему это плохо? Допустим, в вашем JS-файле 4000 строк кода. Вы объявили на 5-й строке переменную books, которая хранит количество книг на сайте. А затем, добравшись до 3005-й строки, забыли об этом и создали другую переменную books — только теперь она хранит количество книг в корзине покупателя. Даже если вы уверены, что всегда сможете отличить одну переменную от другой, то программисты, которые будут работать с вашим кодом, точно не скажут вам спасибо.

«Поднятие» переменной, или hoisting. Все переменные var считаются объявленными перед запуском скрипта. При этом они остаются undefined до тех пор, пока не выполнится код инициализации.

JS. Hoisting

Вот как различается поведение счётчика цикла, если его создавать с помощью var и с помощью let:

var	let

for(var i = 0; i < 5; i++) {}
 	
//Переменная i доступна за пределами цикла
 	
console.log(i);
	for(let i = 0; i < 5; i++) {}
	
	//Переменная i доступна только внутри цикла
	//Попытка использовать её приведёт к ошибке
	
	console.log(i);

В остальном поведение этих переменных идентично.