

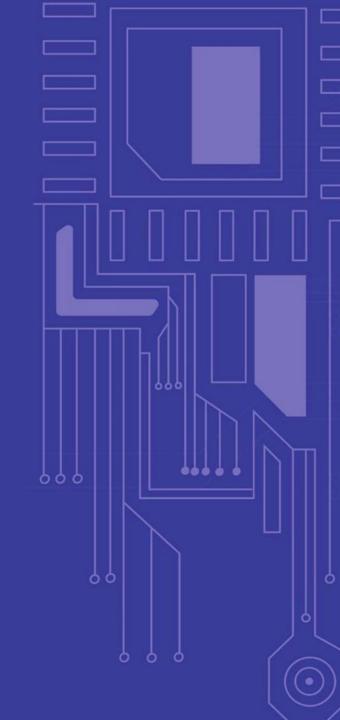




Занятие 5

Контекст. Область видимости. Замыкание





План занятия



- 1. Назначение функций
- 2. Контекст
- 3. Область видимости
- 4. Замыкание

Назначение функций



Функции – важный инструмент в JS

Оператор () – вызов функции

Если назначить функцию переменной, указав ее имя, за которым следует символ ()

Такая функция будет вызвана, и будет присвоено значение, возвращаемое этой функцией

const newValue = setValue('string')

Поднятие

Код читается сверху вниз интерпретатором JS, но выполняется в два этапа:

- 1. Ищутся объявления функций запоминается все, что находится в процессе, т.е. поднятие.
- 2. Когда код фактически выполняется интерпретатором.

Однако на 1-м этапе не распознаются функции, которые были присвоены переменным с помощью ключевых слов let или const

Назначение функций



Анонимные функции

При присвоении функции переменной имя функции можно не указывать, т.к. ее можно вызвать в операторе, указав имя переменной и оператор (). Такие функции называются анонимными функциональными выражениями. Их синтаксис выглядит:

```
let переменная = function (параметры) {
    операторы
    return значение
}
```

Анонимные функции можно сделать самовызывающимися, заключив функцию целиком в круглые скобки () и добавить в конце выражения оператор ().

```
( function () { операторы: return значение } ) ()
```

Область видимости



Все переменные в JS имеют определенную область видимости, в пределах которой они могут действовать. Областей видимости две: глобальная и локальная.

Глобальная

Переменная и функция созданная в этой области доступна из любой точки программы. Это означает, что переменные существуют постоянно и доступны для функций.

Локальная

Переменные созданные внутри функциональных блоков, доступны локально на протяжении всего цикла функции. Они существуют только во время ее выполнения и потом удаляются. Область видимости ограничена { }.

Рекомендуется объявлять переменные в начале функционального блока, чтобы их лексическая область видимости соответствовала времени жизни функции. Это означает, что переменные с одинаковыми именами могут существовать внутри отдельных функций без создания конфликта.

Область видимости



Все переменные в JS имеют определенную область видимости, в пределах которой они могут действовать. Областей видимости две: глобальная и локальная.

Глобальная

Переменная и функция созданная в этой области доступна из любой точки программы. Это означает, что переменные существуют постоянно и доступны для функций.

Локальная

Переменные созданные внутри функциональных блоков, доступны локально на протяжении всего цикла функции. Они существуют только во время ее выполнения и потом удаляются. Область видимости ограничена { }.

Рекомендуется объявлять переменные в начале функционального блока, чтобы их лексическая область видимости соответствовала времени жизни функции. Это означает, что переменные с одинаковыми именами могут существовать внутри отдельных функций без создания конфликта.

Замыкание



Возникает необходимость сохранять значения, которые остаются постоянно доступными.

Как пример запомнить значение счетчика в ходе выполнения программы

Без использования глобальных переменных это можно реализовать используя замыкания

Замыкание (**closure**) – это комбинация функции и лексического окружения вместе со всеми доступными внешними переменными.

В JS замыкания создаются каждый раз при создании функции, а именно во время ее создания



Замыкание

```
function getCounter() {
    let counter = 0;
    return function () {
         return counter++
let count = getCounter()
console.log(count())
                       // print 0
console.log(count())
                       // print 1
console.log(count())
                       // print 2
```











Спасибо за внимание



