Замыкания JS.

В JavaScript функции могут находиться внутри других функций. Когда одна функция находится внутри другой, то внутренняя функция имеет доступ к переменным внешней функции.

**Замыкания** — это функции, ссылающиеся на независимые (свободные) переменные. Другими словами, функция, определённая в замыкании, «запоминает» окружение, в котором она была создана.

**Замыкание** — это функция, у которой есть доступ к области видимости, сформированной внешней по отношению к ней функции даже после того, как эта внешняя функция завершила работу.

**Лексическое окружение** - это скрытый объект, который связан с функцией и создаётся при её запуске. В нём находятся все локальные переменные этой функции, ссылка на внешнее лексическое окружение, а также некоторая другая информация.

Понятие **«лексическое окружение»** или «статическое окружение» в JavaScript относится к возможности доступа к переменным, функциям и объектам на основе их физического расположения в исходном коде.

Пример:

let a = 'global';

function outer() {

let b = 'outer';

function inner() {

let c = 'inner'

console.log(c); // 'inner'

console.log(b); // 'outer'

console.log(a); // 'global'

}

console.log(a); // 'global'

console.log(b); // 'outer'

inner();

}

outer();

console.log(a); // 'global'

Для того чтобы понять замыкания, нам нужно разобраться с двумя важнейшими концепциями JavaScript. Это — контекст выполнения (Execution Context) и лексическое окружение (Lexical Environment).

**Контекст выполнения** — это абстрактное окружение, в котором вычисляется и выполняется JavaScript-код. Когда выполняется глобальный код, это происходит внутри глобального контекста выполнения. Код функции выполняется внутри контекста выполнения функции.

Каждый раз, когда JS-движок создаёт контекст выполнения для выполнения функции или глобального кода, он создаёт и новое лексическое окружение для хранения переменных, объявляемых в этой функции в процессе её выполнения.

**Лексическое окружение** — это структура данных, которая хранит сведения о соответствии идентификаторов и переменных. Здесь «идентификатор» — это имя переменной или функции, а «переменная» — это ссылка на объект (сюда входят и функции) или значение примитивного типа.

**Лексическое окружение содержит два компонента:**

**Запись окружения** (environment record) — место, где хранятся объявления переменных и функций.

**Ссылка на внешнее окружение** (reference to the outer environment) — ссылка, позволяющая обращаться к внешнему (родительскому) лексическому окружению. Это — самый важный компонент, с которым нужно разобраться для того, чтобы понять замыкания.

Когда JS-движок создаёт глобальный контекст выполнения для выполнения глобального кода, он создаёт и новое лексическое окружение для хранения переменных и функций, объявленных в глобальной области видимости. В результате **лексическое окружение глобальной области видимости будет выглядеть так:**

globalLexicalEnvironment = {

environmentRecord: {

a : 'global',

outer: < reference to function object >

}

Внешнее лексическое окружение: null

}

ссылка на внешнее лексическое окружение (outer) установлена в значение null, так как у глобальной области видимости нет внешнего лексического окружения.

**Когда движок создаёт контекст выполнения для функции outer(),** он создаёт и лексическое окружение для хранения переменных, объявленных в этой функции в ходе её выполнения. В результате лексическое окружение функции будет выглядеть так:

functionLexicalEnvironment = {

environmentRecord: {

b : 'outer',

}

Внешнее лексическое окружение: <globalLexicalEnvironment>

}

|  |  |
| --- | --- |
| **Замыкания JS.**  В JavaScript функции могут находиться внутри других функций. Когда одна функция находится внутри другой, то внутренняя функция имеет доступ к переменным внешней функции.  **Замыкания** — это функции, ссылающиеся на независимые (свободные) переменные. Другими словами, функция, определённая в замыкании, «запоминает» окружение, в котором она была создана.  **Замыкание** — это функция, у которой есть доступ к области видимости, сформированной внешней по отношению к ней функции даже после того, как эта внешняя функция завершила работу.  **Лексическое окружение** - это скрытый объект, который связан с функцией и создаётся при её запуске. В нём находятся все локальные переменные этой функции, ссылка на внешнее лексическое окружение, а также некоторая другая информация.  Понятие **«лексическое окружение»** или «статическое окружение» в JavaScript относится к возможности доступа к переменным, функциям и объектам на основе их физического расположения в исходном коде.  Пример:  let a = 'global';  function outer() {  let b = 'outer';  function inner() {  let c = 'inner'  console.log(c); // 'inner'  console.log(b); // 'outer'  console.log(a); // 'global'  }  console.log(a); // 'global'  console.log(b); // 'outer'  inner();  }  outer();  console.log(a); // 'global'  Для того чтобы понять замыкания, нам нужно разобраться с двумя важнейшими концепциями JavaScript. Это — контекст выполнения (Execution Context) и лексическое окружение (Lexical Environment).  **Контекст выполнения** — это абстрактное окружение, в котором вычисляется и выполняется JavaScript-код. Когда выполняется глобальный код, это происходит внутри глобального контекста выполнения. Код функции выполняется внутри контекста выполнения функции.  Каждый раз, когда JS-движок создаёт контекст выполнения для выполнения функции или глобального кода, он создаёт и новое лексическое окружение для хранения переменных, объявляемых в этой функции в процессе её выполнения. | **Лексическое окружение** — это структура данных, которая хранит сведения о соответствии идентификаторов и переменных. Здесь «идентификатор» — это имя переменной или функции, а «переменная» — это ссылка на объект (сюда входят и функции) или значение примитивного типа.  **Лексическое окружение содержит два компонента:**  **Запись окружения** (environment record) — место, где хранятся объявления переменных и функций.  **Ссылка на внешнее окружение** (reference to the outer environment) — ссылка, позволяющая обращаться к внешнему (родительскому) лексическому окружению. Это — самый важный компонент, с которым нужно разобраться для того, чтобы понять замыкания.  Когда JS-движок создаёт глобальный контекст выполнения для выполнения глобального кода, он создаёт и новое лексическое окружение для хранения переменных и функций, объявленных в глобальной области видимости. В результате **лексическое окружение глобальной области видимости будет выглядеть так:**  globalLexicalEnvironment = {  environmentRecord: {  a : 'global',  outer: < reference to function object >  }  Внешнее лексическое окружение: null  }  ссылка на внешнее лексическое окружение (outer) установлена в значение null, так как у глобальной области видимости нет внешнего лексического окружения.  **Когда движок создаёт контекст выполнения для функции outer(),** он создаёт и лексическое окружение для хранения переменных, объявленных в этой функции в ходе её выполнения. В результате лексическое окружение функции будет выглядеть так:  functionLexicalEnvironment = {  environmentRecord: {  b : 'outer',  }  Внешнее лексическое окружение: <globalLexicalEnvironment>  } |