Практическое занятие № 16
"Функции в языке JavaScript. Работа с функциями"

Цель работы:
1. Изучение особенностей функций в языке JavaScript и работы с ними, а также возможностей данной модели программирования.
2. Получение практических навыков в задании (объявлении) функций в JavaScript.

Оборудование и программное обеспечение:
1. ПК, локальная сеть.
2. ОС MS Windows 7 (10), офисный пакет MS Office, браузер MS Internet Explorer.

План занятия:
1. Освоение теоретической части занятия в соответствии с разделами.
2. Выполнение практической части занятия в соответствии с заданиями.
3. Повторение и закрепление основных изученных понятий и терминов.
4. Оформление отчета.

ОБЩИЕ СВЕДЕНИЯ

Функции - ключевая концепция в JavaScript. Важнейшей особенностью языка является первоклассная поддержка функций (functions as first-class citizen).

Любая функция - это объект и, следовательно, ею можно манипулировать как объектом, в частности:

- передавать как аргумент и возвращать в качестве результата при вызове других функций (функций высшего порядка).
- создавать анонимно и присваивать в качестве значений переменных или свойств объектов

Это определяет высокую выразительную мощность JavaScript и позволяет относить его к числу языков, реализующих функциональную парадигму программирования (что само по себе есть очень круто по многим соображениям).

Функция в JavaScript – это специальный тип объектов, позволяющий формализовать средствами языка определённую логику поведения и обработки данных.

Для понимания работы функций необходимо (и достаточно) иметь представление о следующих моментах:

- способы объявления
- способы вызова
- параметры и аргументы вызова (arguments)
- область данных (Scope) и замыкания (Closures)
- объект привязки (this)
- возвращаемое значение (return)
- исключения (throw)
- использование в качестве конструктора объектов
- сборщик мусора (garbage collector)

ОБЪЯВЛЕНИЕ ФУНКЦИЙ

Функция (как и всякий объект) должна быть объявлена (определена, define) перед её использованием.

Объявление (определение) функции - указание сигнатуры и тела функции:

- сигнатура - имя (необязательно) и список входных формальных параметров
- тело функции - комбинация управляющих конструкций и выражений языка над внешними и локальными данными

Объявление пользовательской функции всегда находится в теле некоторой внешней функции-контейнера, которая, в свою очередь, возможно также вложена в некоторую функцию. Цепочка всех таких внешних функций, вложенных друг в друга, образует лексический диапазон функции.

Во избежание оговорок о глобальных переменных и функциях, удобно полагать, что программа на языке JavaScript представляет собой тело неявной функции [[main]]().

Существует три способа объявить функцию:

В ДЕКЛАРАТИВНОМ СТИЛЕ

Для декларативного объявления функции используется синтаксическая конструкция

```
function идентификатор(параметры) {
  инструкции
  return выражение
}
```

Ключевое слово function.

Идентификатор (обязательно).

Список имён формальных параметров (и значений по умолчанию) в круглых скобках, разделенных запятыми.

Тело функции в фигурных скобках вида {}.

Пример. Следующий код объявляет функцию с именем square и параметром number; тело состоит из инструкции return и выражения, которое дословно формализуют следующую логику: "вернуть результат произведения аргумента number на самого себя".

```
function square(number) {
  return number * number;
}
```

Особенностью декларативного объявления является его "всплытие"(hoisting) в начало функции, независимо от того в каком месте контейнера оно находится.

В примере ниже декларативное объявление функции находится после вызова:

```
{
  print(square(5));
  // инициализация "всплывает" вместе с декларацией переменной square
  // Аналогичный код в функциональном стиле работать не будет
  function square(n){return n*n}
}
```

В ФУНКЦИОНАЛЬНОМ СТИЛЕ

Функции также могут быть созданы внутри выражения. Такие функции, как правило, анонимны:

```
var square = function(number) {
  return number * number;
}
```

Но могут иметь определённое имя. (это имя удобно использовать для рекурсии, или в отладчике(debugger)):

```
var factorial = function fac(n) {return n<2 ? 1 : n*fac(n-1)};
print(factorial(3));
```

СТРЕЛОЧНЫЕ ФУНКЦИИ

Современный стандарт языка поддерживает анонимные стрелочные функции (fat arrow function).

```
(param1, param2, …, paramN) => { statements }
(param1, param2, …, paramN) => expression
```

Они особенно удобны, когда надо задать аргумент для функции высшего порядка:

```
[0, 1, 2, 5, 10].map((x) => x * x * x); // вернет [0, 1, 8, 125, 1000].
```

Помимо упрощённого синтаксиса, такие функции всегда неявно привязываются в МОМЕНТ ОБЪЯВЛЕНИЯ к текущему лексическому контексту выполнения:

```
function Person(){
  this.age = 0;
  setInterval(() => {
    this.age++; // в данном случае this будет ссылаться на создаваемый объект obj, а не на window
  }, 1000);
}
var obj = new Person();
```

В СТИЛЕ ООП

Учитывая то, что функция по сути является объектом, можно использовать оператор new и Function конструктор чтобы создавать функции динамически во время выполнения (подобно тому как это делает eval()).

Однако, такого подхода следует избегать из соображений производительности и безопасности.

```
var powerOfFive = new Function('x', 'return ' + Array(5).map(()=>'x').join('*'));
```

МЕТОДЫ

Функции очень часто используются как методы объектов, реализующих ООП.

Метод - это функция, заданная как значение свойства объекта.

Специальный синтаксис вызова методов позволяет неявно передавать объект в качестве привязки (this).

```
class Greeting{
 constructor(prefix){
  this.prefix = prefix;
 }
 // это метод:
 hello(name){
  return `${this.prefix}, ${name}`;
 }
}
var obj = new Greeting("Привет");
// вызов метода (obj передаётся в качестве контекста `this`)
obj.hello('БОРЯ');
```

ВЫЗОВ (CALL) ФУНКЦИИ

Помимо манипуляций с ними как с обычными объектами, функции можно вызывать - запустить процесс вычисления выражений над данными.

Вызов функций – это последовательное выполнение управляющих инструкций и выражений из тела функции применительно к входным данным и контексту.

Выражение (expression) - комбинация математических и специальных операций, а также вызовов функций, которые описывают способ вычисления результирующего значения(result) в зависимости от входящих данных(input).

В момент вызова фунции могут быть переданы:

входные данные в виде списка аргументов-значений,
а также объект привязки - специальный аргумент, именуемый ключевым словом this.

Кроме этого, выражения в функции могут адресоваться:

к константам (строковые литералы, числа и т.д.),
к локальным переменным,
а также к внешним свободным переменным по цепочке замыкания.

В результате выполнения функция возвращает некоторое значение (явно с помощью return или неявно) или бросает исключение.

Например, вызвать square() можно следующим образом:

```
// Функция выполняет свои инструкции над аргументом 5
// и возвращает значение-результат 25
var result = square(5);
```

Вызов функции должен быть выполнен в пределах её области видимости после того как функция объявлена. Область видимости функции определяется точно также, как и для переменной любого другого типа.

РЕКУРСИЯ

Рекурсивная функция - функция, содержащая вызов самой себя непосредственно в своём теле либо через другие функции.

Классический пример рекурсивной функции, вычисляющей факториал:

```
function factorial(n) {
 if ((n == 0) || (n == 1))
  return 1;
 else
```

```
    return (n * factorial(n - 1));
}
//вычислить факториал пяти:
var e = factorial(5); // e будет равно 120
```

Вызов с помощью .apply() и .call()

Существуют и другие способы вызывать функции. Поскольку функции сами являются объектами, они содержат собственные методы. В частности, метод apply(), может использоваться, например, когда нужно адресоваться к функции динамически, или передать различное количество аргументов, или явно указать контекст.

```
var fn = resolveAction();
fn.apply(context, [number1, number2]);
```

Смотрите Function объект для более детального понимания.

Выполнение практической части в соответствии с разделами

Номер варианта	*Содержание задания*
Вариант 1	1. Используя материал из раздела «Общие сведения», выполните пример с объявлением функции по способу «В декларативном стиле». 2. Выполните пример по вызову функции с любым объектом на Ваш выбор.
Вариант 2	1. Используя материал из раздела «Общие сведения», выполните пример с объявлением функции по способу «В функциональном стиле». 2. Выполните пример с использованием специального синтаксиса вызова методов с любым объектом на Ваш выбор.
Вариант 3	1. Используя материал из раздела «Общие сведения», выполните пример с объявлением «Стрелочной функции». 2. Выполните пример с использованием рекурсивной функции с любым объектом на Ваш выбор.

ОФОРМЛЕНИЕ ОТЧЕТА

Содержание отчета:

1. Наименование и цель практической работы.
2. Результаты выполненных действий практической части в соответствии с заданием.
3. Ответы на контрольные вопросы.

Контрольные вопросы к практической работе № 16:

1. Дайте понятие функции в понимании действий с объектами.
2. Дайте понятие объявления функции и перечислите известные способы объявления функций.
3. Дайте понятие рекурсивной функции.
4. Что предполагают понятия - «Вызов функции» и «Выражение»?
5. Что означает понятие «Метод»?