**Вопрос 1** **Выбрать правильные утверждения об области видимости объекта:**

Область видимости определяется положением переменной в коде программы

Область видимости объект можно присвоить

Переменная может быть одновременно и локальной и глобальной

Область видимости для переменной может быть присвоена ключевыми словами

**Ответ:** Область видимости определяется положением переменной в коде программы

**Вопрос 2** **Что такое глобальный объект?**

Свойство объекта window

Область видимости, которого ограничена телом всего скрипта

Область видимости, которого ограничена циклом

Область видимости, которого ограничена одном методом

**Ответ:** Свойство объекта window  
Область видимости, которого ограничена телом всего скрипта

**Вопрос 3** **Что вернет свойство length:**

**var items = ['shoes', 'shirts', 'socks', 'sweaters'];**

**items.length;**

Количество символов массива

Ошибку, так как такая инициализация массива недопустима

4

Ничего, так как в массиве находиться не числа

**Ответ:** 4

**Вопрос 4** **Что можно делать с функцией?**

Создавать анонимно и присваивать в качестве значений переменных или свойств объектов.

Наследовать одни функции другими.

Передавать аргументы в функцию.

Передавать несколько параметров.

Передавать как аргумент и возвращать в качестве результата при вызове других функций.

**Ответ:** Создавать анонимно и присваивать в качестве значений переменных или свойств объектов.   
Передавать аргументы в функцию.   
Передавать несколько параметров.   
Передавать как аргумент и возвращать в качестве результата при вызове других функций.

**Вопрос 5** **Что будет записано в переменную, если в ее параметр, который принимает функция, ничего не передали?**

Переменная не будет создана

null

0

undefined

**Ответ:** undefined

**Вопрос 6** **Что такое замыкание?**

Замыкание — это комбинация функции и лексического окружения, в котором эта функция была определена.

Замыкание — это инструмент, который используется для передачи неопределенного количества параметров в функцию.

Замыкание — это вложенность одного метода в другой, при котором вложенный метод наследует всю область видимости. При этом переменные продолжают существовать даже при множественном вызове одной и той же функции.

Замыкание — это инструмент, который используется для указания значения «по умолчанию» параметрам функции.

**Ответ:** Замыкание — это комбинация функции и лексического окружения, в котором эта функция была определена.   
Замыкание — это вложенность одного метода в другой, при котором вложенный метод наследует всю область видимости. При этом переменные продолжают существовать даже при множественном вызове одной и той же функции.

**Вопрос 7** **Что будет передано в метод showname:**

**function() {**

**showName(this);**

**}**

Объект целиком

Ошибка, так как не указано конкретное поле

Первое поле класса, на который ссылается this

Все поля в виде массива

**Ответ:** Объект целиком

**Вопрос 8** **Для чего в данном случае служит переменная square:**

**var square = function(x) {**

**return x \* x;**

**};**

Переменная хранит в себе значение x

Переменная принимает значение, которое возвращает функция

Переменная хранит название функции

Переменная ссылается на функцию function

**Ответ:** Переменная ссылается на функцию function

**Вопрос 9** **Что будет возвращено из функции, если возвращаемое выражение не указано?**

undefined

null

NaN

Код не отработает

**Ответ:** undefined

**Вопрос 10** **Как правильно вызвать метод showMessage:**

**function showMessage() {**

**alert( 'Привет всем присутствующим!' );**

**}**

showMessage;

showMessage[];

showMessage();

showMessage.Invoke()

**Ответ:** showMessage()

**Вопрос 11** **Безопасно ли обращение к полям объекта в его контексте с помощью непосредственно имени, или предпочтительнее использовать this:**

**sayHi: function() {**

**alert( user.name );**

**}**

**sayHi: function() {**

**alert( this.name );**

**}**

Безопаснее использовать через имя объекта

Оба варианта полностью безопасны

Безопаснее использовать через this

Оба варианта не гарантируют, что объект не будет изменен до обращения к его полям

**Ответ:** Безопаснее использовать через this

**Вопрос 12** **Что будет выведено в i:**

**function count() {**

**for (var i = 0; i < 3; i++) {}**

**alert( i );**

**}**

Вывод 3

Вывод null

Вывод 2

Ошибка, сообщающая о том, что переменная i не инициализирована

**Ответ:** Вывод 2

**Вопрос 13** **Какая конструкция позволит нам привязать к функции данные, сохраняющиеся между ее вызовами?**

Наследование

Рекурсия

Задержки

Замыкания

**Ответ:** Замыкания

**Вопрос 14** **Известно, что this представляет собой связку «base-name-strict», где:**

name - имя свойства

strict - вспомогательный флаг для передачи use strict

base – объект

base – имя свойства

name - объект

strict - имя свойства

**Ответ:** name - имя свойства   
strict - вспомогательный флаг для передачи use strict   
base – объект

**Вопрос 15** **Что будет выведено при вызове переменной a:**

**function foo() {**

**var a = 2; }**

**alert(a);**

2

Ошибка, сообщающая о том, что переменная a не инициализирована

0

undefined

**Ответ:** 2

**Вопрос 16** **Для чего может использоваться рекурсия?**

Для избегания дублирования кода

Для вызова функции самой себя

Для полной замены функциональности цикла

Использование рекурсии не имеет видимых преимуществ

Для подсчета факториала числа

**Ответ:** Для избегания дублирования кода   
Для вызова функции самой себя   
Для подсчета факториала числа

**Вопрос 17** **Выбрать правильные утверждения про объект arguments:**

Объект, представляющий аргументы для выполняемых в данный момент функций и функции, вызвавшие его

Доступ к отдельным аргументам осуществляется тем же способом, каким обращаются к элементам массива

Объект arguments — содержит ссылки, а не значения

Объект arguments нужно создавать явно для передачи в функцию

Для указания аргументов по умолчанию можно использовать оператор ||

Arguments поможет скопировать в существующий объект свойства из одного или нескольких других

**Ответ:** Объект, представляющий аргументы для выполняемых в данный момент функций и функции, вызвавшие его   
Доступ к отдельным аргументам осуществляется тем же способом, каким обращаются к элементам массива   
Объект arguments — содержит ссылки, а не значения   
Для указания аргументов по умолчанию можно использовать оператор ||   
Arguments поможет скопировать в существующий объект свойства из одного или нескольких других

**Вопрос 18** **Выбрать правильные утверждения об операторе return:**

Указанное значение возвращается туда, где была вызвана функция.

При вызове оператора return в функции, выполнение этой функции прекращается.

Если значение не указано, возвращается undefined вместо него.

Ключевое слово stop прерывает выполнение функции но не возвращает значение в от личии от return.

**Ответ:** Указанное значение возвращается туда, где была вызвана функция.   
При вызове оператора return в функции, выполнение этой функции прекращается.   
Если значение не указано, возвращается undefined вместо него.

**Вопрос 19** **Определить истину про свойство length:**

Возвращает размер объекта в байтах

Свойство length принимает массив

Представляет собой число, равное индексу последнего определенного элемента

Получает или задает длину массива

**Ответ:** Получает или задает длину массива

**Вопрос 20** **Что такое Контекст выполнения, стек?**

Контекст выполнения – это служебная информация, которая соответствует текущему запуску функции.

Контекст выполнения – инструмент, который хранит значение текущей функции только в одном экземпляре, после выхода из функции очищается.

Контекст выполнения – инструмент, который записывает в стек текущую информацию о методе такую как значения переменных, после завершения работы текущего контекста из стека достается новый.

Контекст выполнения - программный инструмент, где хранятся все переменные и методы текущей программы.

**Ответ:** Контекст выполнения – это служебная информация, которая соответствует текущему запуску функции.   
Контекст выполнения – инструмент, который записывает в стек текущую информацию о методе такую как значения переменных, после завершения работы текущего контекста из стека достается новый.

**Вопрос 21** **Что будет выведено на экран, если fn() будет вызван 3 раза подряд:**

**var fn = (function() {**

**var numberOfCalls = 0;**

**return function() {**

**return ++ numberOfCalls;**

**}})();**

1, 2, 3

0, 0, 0

Такая функция не отработает

1, 1, 1

**Ответ:** 1, 2, 3

**Вопрос 22** **Определите преимущества или недостатки рекурсии:**

Легкость проверки корректности значений

Облегчение отладки

Громоздкость записи

Неизменяемость переменных

Рекурсия увеличивает производительность

**Ответ:** Легкость проверки корректности значений   
Облегчение отладки   
Неизменяемость переменных

**Вопрос 23** **Возможно ли использовать функции внутри других функций и как при этом будет вести себя область видимости (функция a содержит в себе функцию b)?**

Область видимости b будет наследоваться a

Область видимости a будет наследоваться b

Область видимости у каждой из функций будет своя уникальная

Вложенные функции использовать запрещено

**Ответ:** Область видимости a будет наследоваться b

**Вопрос 24** **Какую задачу выполняет данная функция рекурсивно:**

**function fun(x, n) {**

**if (n != 1) {**

**return x \* fun(x, n - 1);**

**} else {**

**return x;**

**}**

**}**

Вычисление факториала числа

Инкрементация переменной на определенное значение

Возведение числа в степень

Умножение двух чисел

**Ответ:** Возведение числа в степень

**Вопрос 25** **Что включает в себя контекст выполнения?**

Значения переменных

Название метода, который отработал перед текущей функцией

Выполненные методы

Текущее положение в коде

**Ответ:** Значения переменных  
Текущее положение в коде

**Вопрос 26** **Для чего нужно использовать функции?**

Для структурирования больших программ.

Для увеличения быстродействия.

Использование функций не имеют преимуществ.

Для избегания дублирования кода.

Для повторения одного и того же действия во многих частях программы.

**Ответ:** Для структурирования больших программ.   
Для увеличения быстродействия.   
Для избегания дублирования кода.   
Для повторения одного и того же действия во многих частях программы.

**Вопрос 27** **Определите верные утверждения о this:**

Для получения доступа к полям объекта нужно написать this.

Это ключевое слово используется для созданий ссылок на функцию

Это ссылка на текущий объект, в контексте которого был вызван this.

Использование this не гарантирует, что функция работает именно с тем объектом, в контексте которого вызвана.

**Ответ:** Для получения доступа к полям объекта нужно написать this.   
Это ссылка на текущий объект, в контексте которого был вызван this.

**Вопрос 28** **Допустимо ли писать код после ключевого слова return и отработает ли он:**

**function square(x) {**

**return x \* x;**

**var a = 2;**

**alert(a);**

**}**

Да, но такой код не всегда отработает корректно

Нет, будет выведено сообщение об ошибке

Нет, будет возвращено первое значение справа от return то есть x\*x, дальнейший код проигнорирован

Да, будет возвращено значение после отработки выражения и выведен код

**Ответ:** Нет, будет возвращено первое значение справа от return то есть x\*x, дальнейший код проигнорирован

**Вопрос 29** **Как обратиться к глобальной переменной, если есть локальная с таким же именем?**

Локальные переменные не должны иметь имен глобальных

Обратиться к ней как к свойству window

Заключить переменную в круглые скобки()

Заключить переменную в квадратные скобки[]

**Ответ:** Обратиться к ней как к свойству window

**Вопрос 30** **Выбрать правильно объявленные функции, которые отработают корректно:**

🞎 function fun1(a, b, c) {

return b\*b - 4\*a\*c;

}

🞎 var square = function(x) {

return x \* x;

};

🞎 function init() {}

🞎 function(x+y) {

return this;

}

**Ответ:** 🞎 function fun1(a, b, c) {  
 return b\*b - 4\*a\*c;   
}   
🞎 var square = function(x) {  
 return x \* x;   
};   
🞎 function init() {}