**Вопрос 1** **Что будет выведено после отработки данной части кода?**

**User.prototype.maxage = 110;**

**Employee.prototype = Object.create(User.prototype);**

**Employee.prototype.maxage = 120;**

**var tom = new User("Том", 26);**

**console.log(tom.maxage);**

26

120

null

110

**Ответ:** 110

**Вопрос 2** **При вызове return перед this в данном случае, будет возвращен:**

**function BigAnimal() {**

**this.name = "mouse";**

**return "BigMouse";**

**}**

this

Объект

Будет ошибка так как конструкторы ничего не возвращают

Будет возвращен null

**Ответ:** this

**Вопрос 3** **Указать верные утверждения о \_\_proto\_\_**

Поиск свойства происходит в свойстве \_\_proto\_\_ если в объекте оно не было найдено

Это свойство полностью доступно пользователю для работы

Это скрытое системное свойство

Абсолютно любой объект в JavaScript имеет свойство \_\_proto\_\_

**Ответ:** Поиск свойства происходит в свойстве \_\_proto\_\_ если в объекте оно не было найдено   
Это скрытое системное свойство   
Абсолютно любой объект в JavaScript имеет свойство \_\_proto\_\_

**Вопрос 4** **Что означает данная часть кода?**

**var animal = new function() {**

**this.name = "Joe";**

**this.canWalk = true;**

**};**

Создание конструктора с именем animal

Только определение функции конструктора

Вызов функции конструктора и создание объекта сразу

Такой код не отработает

**Ответ:** Вызов функции конструктора и создание объекта сразу

**Вопрос 5** **Что делает данный код?**

**function User(name) {**

**this.name = name;**

**this.sayHi = function() {**

**alert( "myname: " + this.name );**

**};**

**}**

Создает метод который выводит myname

Создает конструктор без параметров

Создать метод в методе невозможно

Создает конструктор который во внутреннем методе выводит переданное имя

**Ответ:** Создает конструктор который во внутреннем методе выводит переданное имя

**Вопрос 6** **Как правильно реализовать наследование класса Rabbit от Animal?**

Rabbit.prototype.prototype = Animal.\_\_proto\_\_;

Rabbit.prototype.\_\_proto\_\_ = Animal.prototype;

Rabbit:Animal

Rabbit.\_\_proto\_\_ = Animal.\_\_proto\_\_;

Rabbit.prototype = Object.create(Animal.prototype);

**Ответ:** Rabbit.prototype.\_\_proto\_\_ = Animal.prototype;   
Rabbit.prototype = Object.create(Animal.prototype);

**Вопрос 7** **Можно ли создавать методы в конструкторе?**

Да, только если конструктор не принимает параметры

Да

Невозможно

Да, только если конструктор принимает параметры

**Ответ:** Да

**Вопрос 8** **Укажите верные утверждения про данный код:**

**var animal = new BigAnimal**

Не отработает так как скобки не указаны()

Будет вызван конструктор, но код не отработает так как параметры не переданы

Вызывает конструктор без параметров

Идентичен var animal = new BigAnimal();

**Ответ:** Не отработает так как скобки не указаны()   
Вызывает конструктор без параметров

**Вопрос 9** **Что будет выведено в результате выполнения этого кода?**

**var obj = {};**

**function A() { return obj; }**

**function B() { return obj; }**

**var a = new A;**

**var b = new B;**

**alert( a == b )**

Ошибка, ничего не будет выведено

undefined

true

false

**Ответ:** false

**Вопрос 10** **Свойство prototype это:**

Ссылка на объект, который хранит свойства и методы

Любое свойство в функции называется prototype

Модификатор доступа

Свойство для записи в него числовых значений

**Ответ:** Ссылка на объект, который хранит свойства и методы

**Вопрос 11** **Имеет ли смысл указывать \_\_proto\_\_ для без оператора new?**

Да, для изменения свойств объекта

Нет, указывать нужно только при создании новых объектов

Да, для реализации наследования

Указывать можно и с новыми и с существующими

**Ответ:** Нет, указывать нужно только при создании новых объектов

**Вопрос 12** **Будет ли данное сравнение правдивым?**

**Number.prototype.\_\_proto\_\_ === Object.prototype**

Да, поскольку все типі данных наследуются от object

Нет

Будет возвращен null

Да, если переменные \_\_proto\_\_ и prototype хранят одинаковые значения

**Ответ:** Да, поскольку все типі данных наследуются от object

**Вопрос 13** **Выберите истинные утверждения о возможностях оператора new:**

Устанавливает прототип \_\_proto\_\_ объекту ссылкой на prototype

Оператор создает пустой объект

Применяет функцию-класс к новосозданному объекту

Оператор передается в конструктор объекта для установки заданных свойств

**Ответ:** Устанавливает прототип \_\_proto\_\_ объекту ссылкой на prototype   
Оператор создает пустой объект   
Применяет функцию-класс к новосозданному объекту

**Вопрос 14** **Укажите верные утверждения о свойстве prototype:**

Свойство имеет всегда одно свойство constructor

Свойство используется при создании новых объектов оператором new

Свойство имеет всегда одно свойство \_\_proto\_\_

Свойство имеет два свойства \_\_proto\_\_ и constructor

**Ответ:** Свойство имеет всегда одно свойство constructor   
Свойство используется при создании новых объектов оператором new

**Вопрос 15** **Указать верные утверждения о массивах Array:**

Object.prototype имеет прототипом Array.prototype.

Методы массивов Array хранятся в Object.prototype.

Методы массивов Array хранятся в Array.prototype.

Array.prototype имеет прототипом Object.prototype.

**Ответ:** Методы массивов Array хранятся в Array.prototype.   
Array.prototype имеет прототипом Object.prototype.

**Вопрос 16** **Какие функции позволяют манипулировать свойством \_\_proto\_\_?**

Object.create, Object.getPrototypeOf

Только Object.getPrototypeOf

Только Object.getPrototype

Только Object.create

**Ответ:** Object.create, Object.getPrototypeOf

**Вопрос 17** **Указать верные утверждения о свойстве prototype:**

Будет создан у объекта, только после добавления туда какого-либо свойства или метода

Свойство всегда возвращает true

По умолчанию объект prototype «пустой»

Свойство всегда возвращает false

**Ответ:** По умолчанию объект prototype «пустой»

**Вопрос 18** **При вызове return перед this в данном случае, будет возвращен:**

**function BigAnimal() {**

**this.name = "mouse";**

**return { name: "BigMouse" };**

**}**

Объект

Будет возвращен null

Будет ошибка так как конструкторы ничего не возвращают

This

**Ответ:** Объект

**Вопрос 19** **Если создан класс Rabbit, будет ли равно:**

**Rabbit.prototype.constructor == Rabbit**

Будет возвращен false

Ошибка, ничего не будет выведено

Будет возвращен undefined

Да, будет возвращен true

**Ответ:** Да, будет возвращен true

**Вопрос 20** **Метод который меняет внутреннее свойство [[Prototype]] объекта obj на proto имеет сигнатуру:**

Object.definePrototypeOf(obj, proto)

Object.PrototypeOf(obj, proto)

Object.create(obj, proto)

Object.setPrototypeOf(obj, proto)

**Ответ:** Object.setPrototypeOf(obj, proto)

**Вопрос 21** **Как реализовано наследование в JavaScript?**

Нужно записать наследуемый класс справа от родительского через двоеточие

Требуется передать класс который требуется наследовать в конструктор родительского класса

Через объект prototype

Реализуется через ссылку \_\_proto\_\_

**Ответ:** Реализуется через ссылку \_\_proto\_\_

**Вопрос 22** **Выбрать пример фабричного метода:**

Object.getPrototypeOf

Object.assign

Object.create

Object.setPrototypeOf

**Ответ:** Object.create

**Вопрос 23** **Указать верные утверждения про функции конструкторы:**

При вызове с ключевым словом new, функция получает новый пустой объект в качестве this

Любая функция может быть вызвана с new

Названия таких функций обязательно писать с маленькой буквы

Названия таких функций обязательно писать с большой буквы

**Ответ:** При вызове с ключевым словом new, функция получает новый пустой объект в качестве this   
Любая функция может быть вызвана с new

**Вопрос 24** **Определить верные утверждения о цепочке прототипов (prototype chain) это:**

Обозначение того, что прототип имеет во вложенности еще 2 прототипа

Это вложенный прототип у прототипа

Это все прототипы которые доступны в программе

null не имеет прототипа и является завершающим звеном в цепочке прототипов

**Ответ:** Это все прототипы которые доступны в программе   
null не имеет прототипа и является завершающим звеном в цепочке прототипов

**Вопрос 25** **Для чего используется конструктор в JavaScript?**

Для создания нескольких однотипных объектов

Для сокрытия объектов

Для инициализации полей объектов

Для приведения типов

**Ответ:** Для создания нескольких однотипных объектов   
Для инициализации полей объектов

**Вопрос 26** **Что произойдет, если указать свойство prototype для строки?**

Указать такое свойство строке невозможно

Можно получить возможность наследования от строки

Строка будет проигнорирована, так как это примитивное значение

В свойство будет записано null

**Ответ:** Строка будет проигнорирована, так как это примитивное значение

**Вопрос 27** **Выбрать верные утверждения о конструкторах в JavaScript:**

Для вызова функции как конструктор, нужно вызвать у нее метод Invoke

Конструктор - это обычная функция вызванная при помощи new

Для задания функции поведения конструктора, нужно перед названием написать ключевое слово const

Для выделения функции как конструктора, ее название пишут с большой буквы, как того требует венгерская нотация

**Ответ:** Конструктор - это обычная функция вызванная при помощи new   
Для выделения функции как конструктора, ее название пишут с большой буквы, как того требует венгерская нотация

**Вопрос 28** **Что делает метод Object.assign(a, b)?**

Копирует все перечислимые (enumerable) свойства объекта b в объект a, а затем возвращает объект a

Такой метод не отработает так как не все параметры переданы

Копирует все перечислимые (enumerable) свойства объекта b в объект a, не возвращая результат

Копирует все перечислимые (enumerable) свойства объекта a в объект b, а затем возвращает объект b

**Ответ:** Копирует все перечислимые (enumerable) свойства объекта b в объект a, а затем возвращает объект a