

Метод 5 - Однострочное форматирование

- | | |
|---------|----------|
| 1. b | 14. b |
| 2. α | 15. 1) b |
| 3. α | 2) c |
| 4. b | 3) α |
| 5. c | 4) d |
| 6. 1) b | 16. b |
| 2) c | 17. b |
| 3) d | 18. b |
| 4) α | 19. c |
| 17. b | 20. b |
| 8. α | 21. d |
| 9. α | 22. α |
| 10. α | 23. 1) α |
| 11. α | 2) b |
| 12. α | 3) c |
| 13. α | 4) d |
| | 24. α |
| | 25. α |

Оценки на основании ответов

26. Асинхронное программирование — это способ написания кода, при котором выполнение длительных задач не блокирует основной поток выполнения. Когда будет продолжаться выполнение утяжеленной кода, пока не будет завершено асинхронной операции.

27. let: старый способ, область видимости ограничена функцией, возвращает "вместимо".

let: область видимости ограничена блоком {...}, не может переопределить в том же блоке

const: то же, что let, но значение переменной не может измениться после присвоения.

28. Promise — это объект, представляющий результат асинхронной операции, который будет доступен

"в Будущем". У них есть 3 состояния: pending (ожидание), fulfilled (выполнено успешно), rejected (ошибка).

29. Event Loop - механизм, который координирует выполнение кода. Он следит за стеком вызовов и очередь задач. Если стек пуст, Event Loop берет задачу из очереди и отправляет ее в стек на выполнение.

30. JS использует прототипное наследование. Объекты могут наследовать свойства других объектов. С точки зрения классов в ES6 синтаксис стал похож на классические ООП, но, по сути, это все равно прототипы.

31. Объект this - это ссылка на контекст выполнения функции. Его значение зависит от того, как вызвана функция.

- В методе объекта - это сам объект

- В обычной функции - window

- В стрелочных функциях - this.

32. Собака - это классы JavaScript о действиях персонажей.

33. Call: вызывает функцию с заданным this и аргументами переданную.

Apply: тоже самое, но аргументы переданные массивом.

Bind: не вызывает функцию сразу, а возвращает новую функцию с "привязанным" this.

34. JSON - это текстовый формат обмена данными.

35. Деструктуризация - удобный способ извлечения данных из массивов и объектов в отдельные переменные.

Пример:

- Массив: `const [a, b] = [10, 20];`
- Объект: `const { name, age } = { name:
: 'Ivan', age: 25 };`