Advanced Functions

Назад към каталога Частичен тест

1. Как можем да достъпим контекста на изпълнение на дадена функция?

- ○ Чрез `arguments`
- ○ Чрез `prototype`
- ◉ Чрез `this`
- ○ Чрез `that`

2. Arrow-функциите запазват контекста на изпълнение, в който са декларирани, независимо от къде се изпълняват.

- ◉ Вярно
- ○ Невярно

3. Отбележете вариантите за контекст на изпълнение:

(изберете всички подходящи отговори)

- ☑ Като метод на обект
- ☐ Като виртуален метод
- ☐ Overload на фунцкия
- ☑ Глобално изпълнение
- ☑ Като слушател на събитие в DOM

4. Кой метод за посочване на контекста на изпълнение приема аргументите като масив?

- ◉ `apply()`
- ○ `bind()`

- `call()`

5. **Как се подават аргументите чрез метода `call()`?**

- [x] Като отделни елементи
- [ ] Не можем да подаваме аргументи
- [ ] Като JSON
- [ ] Като масив

6. **Функциите като първокласни елементи на езика означава:**

- [ ] Можем да ги заменим с резултата от тяхното изпълнение
- [ ] Винаги връщат стойност
- [ ] Приемат само един аргумент
- [x] Можем да ги третираме като стойности

7. **Функциите могат да връщат като резултат други функции.**

- [x] Вярно
- [ ] Невярно

8. **Кой термин отговаря на определението "Функция, която приема като аргумент друга функция, или връща като резултат друга функция"?**

- [x] Функция от по-висок ред
- [ ] Не съществува еквивалент в JavaScript
- [ ] Функкция на втора степен
- [ ] Сложна функция

9. **Кои са свойствата на "чистите" функции (pure functions)?**

(изберете всички подходящи отговори)

- [x] При едни и същи аргументи връщат еднакъв резултат
- [ ] Не са част от обект
- [ ] Не съдържат променливи
- [x] Нямат странични ефекти
- [ ] Част са от обект

10. **Как наричаме свойството на вложена функция да вижда обхвата (scope), в който е декларирана?**

- ○ Inheritance
- ○ Overloading
- ◉ Closure
- ○ Dereference

11. **Какъв ще е резултата от изпълнението на следния код:**

```
1. function outer() {
2.   let a = 5;
3.
4.   function innerA() {
5.     a = 6;
6.   }
7.   function innerB() {
8.     console.log(a);
9.   }
10.
11.   return {
12.     innerA,
13.     innerB
14.   };
15. }
16.
17. const result = outer();
18. result.innerA();
19. result.innerB();
```

- ○ Числото 5 в конзолата

- Грешка при изпълнение
- undefined в конзолата
- Грешка при компилация
- Числото 6 в конзолата

12. **Как наричаме следния израз:**

```
1. (function() => {
2.   let name = 'Peter';
3.   return name;
4. })();
```

- IIFE
- Closure
- Higher-order function
- Predicate

13. **На кой ред се извършва частично прилагане (partial application) на аргументи:**

```
1. const person = { name: 'Peter' };
2.
3. function sayHi() {
4.   console.log(this.name + ' says hi');
5. }
6.
7. function sum(a, b) {
8.   return a + b;
9. }
10.
11. const increment = sum.bind(null, 1);
12. const greet = sayHi.bind(person);
```

- 1
- 7-9
- 11
- 12

14. **Currying и Partial Application са едно и също нещо.**

- Вярно
- Невярно

0 / 14 верни отговора