[ITMO 2024] Прототипы

Ограничение времени 10 с

Ограничение памяти 64.0 Мб

Ввод	Вывод
input.json	output.json

Что требуется сделать

Нужно решить 9 подзадач. В конце страницы приведен шаблон файла для решения.

1. Функция `getNewObjWithPrototype` должна возвращать новый объект, где аргумент `obj` является его прототипом.

```
const getNewObjWithPrototype = (obj) => {
    // code
}```
```

2. Функция `getEmptyObj` возвращает пустой объект без прототипов.

```
const getEmptyObj = () => {
    // code
}```
```

3. Функция `setPrototypeChain` задает цепочку прототипов таким образом, чтобы поиск нужного свойства выполнялся по цепочке `programmer -> student -> teacher -> person`.
   На вход функция принимает объект вида `{ programmer, student, teacher, person }`.

```
const setPrototypeChain = ({ programmer, student, teacher, person }) => {
    // code
}```
```

4. Функция `getObjWithEnumerableProperty` возвращает объект со свойствами `name: 'Alex'`, `age: 18`, `work: 'empty'`, где перечисляемым свойством является только свойство `age`.

```
const getObjWithEnumerableProperty = () => {
    // code
}```
```

5. Функция `getWelcomeObject` на вход получает объект вида `{ name: 'Alex', age: 18 }`. Возвращается объект с 1 методом `voice` (входной объект является прототипом для возвращаемого объекта). Метод возвращает строку `"Hello, my name is Alex. I am 18."` (значения зависит от входного объекта).

```
const getWelcomeObject = (person) => {
    // code
}```
```

6. Сделать класс `Singleton` так, чтобы он принимал на вход любое число и записывал его в свойство `id`. (Напоминание: класс `Signleton` возвращает один и тот же созданный экземпляр класса).

```
class Singleton {
    // code
}```
```

7. Функция `defineTimes` должна расширять класс Number функцией `times`, которая принимает callback и вызывает ее заданное количество раз.
   `callback` принимает на вход 2 аргумента: порядковый индекс вызова и исходное количество вызовов.

```
const defineTimes = () => {
    // code
}```

// Пример использования defineTimes(); const count = 5;

count.times((index, value) => console.log(index, value)); // 1, 5 // 2, 5 // 3, 5 // 4, 5 // 5, 5```

8. Добавить всем массивам геттер `uniq`, который возвращает новый массив уникальных значений.

```
const defineUniq = () => {
    // code
}```

// Пример использования defineUniq(); const arr = [1,2,2];

console.log(arr.uniq); // [1,2]; console.log(arr); // [1,2,2];```

9. Добавить всем массивам геттер `uniqSelf`, который меняет сам массив и возвращает уникальные значения.

```
const defineUniqSelf = () => {
    // code
}
```

```
// Пример использования
defineUniqSelf();
const arr = [1,2,2];

console.log(arr.uniqSelf); // [1,2];
console.log(arr); // [1,2];
```