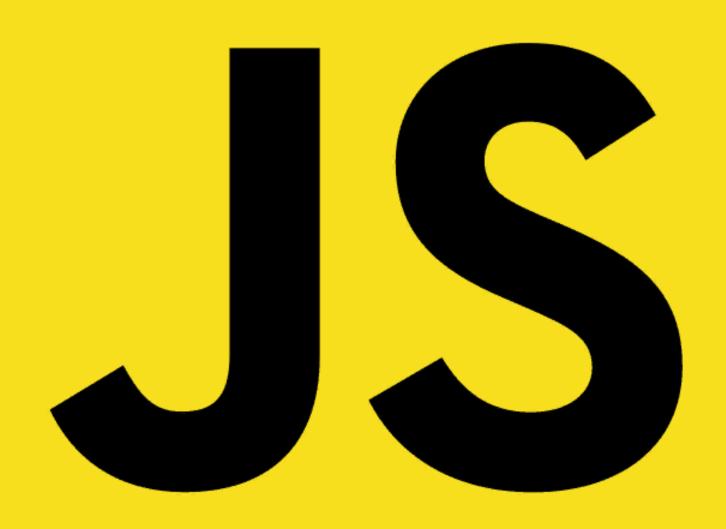
Язык JavaScript (функцииобёртки, декораторы)



Декоратор – приём программирования, который позволяет взять существующую функцию и изменить/расширить ее поведение.

Декоратор получает функцию и возвращает обертку, которая делает что-то своё «вокруг» вызова основной функции:

```
function decorator(func){
    function wrapper(...args){
        //do something
        return func(...args);
    }
    return wrapper;
}
```

```
Один простой декоратор вы уже видели ранее – это функция bind:
function bind(func, context) {
      return function() {
             return func.apply(context, arguments);
      };
let obj = {
      name: 'petr',
      age: 18
function hello(){
   console.log(`Меня зовут ${this.name}. Мне ${this.age} лет`);
let bound = bind(hello, obj);
bound();
```

Другой пример - декоратор логирующий время выполнения функции:

```
function benchmark(func) {
     return function() {
           let started = Date.now();
           let result = func(...arguments);
           let finished = Date.now();
           console.log(`Функция "${func.name}"
выполнилась за ${finished-started} микросекунд`);
           return result;
     };
```

Другой пример - декоратор кэширующий результат выполнения функции:

```
function cache(func){
     let memory = {};
     return function(...args){
           key = args.toString();
           if (!(key in memory))
                memory[key] = func(...args);
           return memory[key];
```

ПОЛЕЗНЫЕ ССЫЛКИ

https://learn.javascript.ru/decorators - Функции-обёртки, декораторы