**1)**

**насчет bind(this)**

**0**

[Данияр](https://www.udemy.com/user/daniiar-mataev/) ·

Lecture 132

 · 4 месяца назад

метод bind() меняет контекст вызова на то что в ее парметрах, что бы не связывался с window, тогда почему тут bind(this) заменяют на тот же this? можете объяснить просто не понял.

**2 replies**

Follow replies

Иван Петриченко

[Иван](https://www.udemy.com/user/yan-kovalenko-2/) — Instructor  
4 месяца назад

**0**

Добрый день. Я так понимаю, что эти операции находятся внутри constructor'а класса.

Если дословно, то:

- мы берем метод, который будет внутри экземпляра объекта: this.nextOne. Мы его конструируем. Тут this ссылается на новосозданный экземпляр.

- в него мы помещаем метод, объявленный внутри класса. Класс - это прототип, чертеж для создания экземпляров.

- и при помощи bind мы этот метод "подвязываем" к новосозданному экземпляру.

Получается, что каждый из этих методов будет работать внутри объекта и ссылаться (работать) на него.

**2)**

Иван Масенко

**Контекст для функции внутри функции**

**0**

[Иван](https://www.udemy.com/user/ivan-masenko/) ·

Lecture 132

 · 5 месяцев назад

Как по мне нелогично конечно, что функция которая создана внутри функции ссылается на контекст окна. Ваш пример с бомжом и котом можно интерпретировать таким образом: кот живет благодаря бомжу, который дарит коту свою доброту и еду для существования, т.е. "мир" дочерней функции составлен при помощи родительской. Он вызывается из контекста родителя. И если родитель имеет свой контекст, то и дочерний должен иметь тоже самое. Вам это не кажется?  
  
Ну ок. Возьмем это как обычно за аксиому :) но вопрос у меня такой. Если я попаду на практике в ситуацию, когда у меня будет функция внутри функции и мне потребуется тот же самый контекст, то тогда придется постоянно жестко привязывать через bind, или есть другие более гибкие варианты?  
  
На первый вопрос тоже пожалуйста ответьте :)

**4 replies**

Follow replies

K

[Konstantin](https://www.udemy.com/user/konstantin-1383/)  
5 месяцев назад

**0**

Покажу на примере, как я во внутренней функции получаю контекст

1. let btn = document.querySelector('button');
3. btn.addEventListener('click', function () {
4. let that = this; // кладу контекст в переменную
5. function showThis(that) {
6. console.log(that);
7. }
8. showThis(that); // передаю контекст во вложенную функцию
9. });

  Может это костыльный способ, но нагляден и понятен. Как это сделать через call, apply, bind - пока не знаю ))

K

[Konstantin](https://www.udemy.com/user/konstantin-1383/)  
5 месяцев назад

**1**

Посмотрел урок 30 про стрелочные функции. Открыл для себя другой способ ))

1. let btn = document.querySelector('button');
3. btn.addEventListener('click', function () {
4. let showThis = () => {
5. console.log(this);
6. };
7. showThis();
8. });

Иван Масенко

[Иван](https://www.udemy.com/user/ivan-masenko/)  
5 месяцев назад

**0**

Ну вот я рефакторил код с таймером. Перевел все в классы и конструкторы. С интервалом пришлось помучится. Получилось пока добиться при помощи bind. Он вроде и рабочий, но тоже больно костыльный)  
Константин, благодарю за ваши способы. Опробую. Жду ответа от автора урока.

1. setClock(id, endtime) {
2. let timer = document.getElementById(id),
3. hours = timer.querySelector('.hours'),
4. minutes = timer.querySelector('.minutes'),
5. seconds = timer.querySelector('.seconds'),
7. timeInterval = setInterval(updateClock.bind(this), 1000);
9. function updateClock() {
10. let t = this.getTimeRemaining(endtime);
11. hours.textContent = t.hours;
12. minutes.textContent = t.minutes;
13. seconds.textContent = t.seconds;
15. if (t.total <= 0) {
16. clearInterval(this.timeInterval);
17. }
18. }

Иван Петриченко

[Иван](https://www.udemy.com/user/yan-kovalenko-2/) — Instructor  
5 месяцев назад

**1**

Добрый день. Увы, но это действительно аксиома)

Вариант Константина тоже применяется, но сейчас намного реже. С появлением стрелочных функций, которые берут контекст родителя, предпочитают использовать их.

Вот пример из моего недавнего ролика на ютубе:

1. const recursy = (element) => {
2. element.childNodes.forEach(node => {
3. if (node.nodeName === '#text' && node.nodeValue.replace(/\s+/g, "").length > 0){
5. if (!node.parentNode.getAttribute('data-fz')){
6. let value = window.getComputedStyle(node.parentNode, null).fontSize;
7. node.parentNode.setAttribute('data-fz', +value.replace(/px/g, ""))
8. node.parentNode.style.fontSize = node.parentNode.getAttribute('data-fz') \* this.scale + "px";
9. } else {
10. node.parentNode.style.fontSize = node.parentNode.getAttribute('data-fz') \* this.scale + "px";}
12. } else {
13. recursy(node);
14. }
15. })
16. }

(Редактор кода тут немного сбивает табуляцию, но я думаю смысл понятен)

Здесь используется this.scale, и чтобы во всех вложенностях не терять контекст мы и используем стрелочные функции.