

Раздел

[Качество кода](#)

Навигация по уроку


Плохие комментарии

Хорошие комментарии

Итого

Комментарии

Поделиться

[Редактировать на GitHub](#)[🏠](#) → [Язык программирования JavaScript](#)  
→ [Качество кода](#) 8-го августа 2019

## Комментарии

Как мы знаем из главы [Структура кода](#), комментарии могут быть однострочными, начинающимися с `//`, и многострочными: `/* ... */`.

Обычно мы их используем, чтобы описать, как и почему работает код.

На первый взгляд, в комментариях нет ничего сложного, но новички в программировании часто применяют их неправильно.

### Плохие комментарии

Новички склонны использовать комментарии, чтобы объяснять, «что делает код». Например, так:

```
1 // Этот код делает это (...) и вот это (...)  
2 // ...и кто знает, что ещё...  
3 очень;  
4 сложный;  
5 код;
```

Но в хорошем коде количество «объясняющих» комментариев должно быть минимальным. Seriously, код должен быть таким, чтобы его можно было понять без комментариев.

Про это есть хорошее правило: «Если код настолько запутанный, что требует комментариев, то, может быть, его стоит переделать?»

### Рецепт: выносите код в функции

Иногда выгодно заменить часть кода функцией, например, в таком случае:

```
1 function showPrimes(n) {  
2   nextPrime:  
3   for (let i = 2; i < n; i++) {  
4  
5     // проверяем, является ли i простым числом  
6     for (let j = 2; j < i; j++) {  
7       if (i % j == 0) continue nextPrime;  
8     }  
9  
10    alert(i);  
11  }  
12 }
```

Лучший вариант – использовать отдельную функцию `isPrime`:

```
1 function showPrimes(n) {  
2  
3   for (let i = 2; i < n; i++) {  
4     if (!isPrime(i)) continue;  
5  
6     alert(i);  
7   }  
8 }  
9  
10 function isPrime(n) {  
11   for (let i = 2; i < n; i++) {  
12     if (n % i == 0) return false;  
13   }
```

Раздел

[Качество кода](#)

Навигация по уроку

Плохие комментарии

Хорошие комментарии

Итого

Комментарии

Поделиться



[Редактировать на GitHub](#)



```
14
15     return true;
16 }
```

Теперь код легче понять. Функция сама становится комментарием. Такой код называется *самодокументированным*.

## Рецепт: создавайте функции

И если мы имеем такой длинный кусок кода:

```
1  // здесь мы добавляем виски
2  for(let i = 0; i < 10; i++) {
3      let drop = getWhiskey();
4      smell(drop);
5      add(drop, glass);
6  }
7
8  // здесь мы добавляем сок
9  for(let t = 0; t < 3; t++) {
10     let tomato = getTomato();
11     examine(tomato);
12     let juice = press(tomato);
13     add(juice, glass);
14 }
15
16 // ...
```

То будет лучше отрефакторить его с использованием функций:

```
1  addWhiskey(glass);
2  addJuice(glass);
3
4  function addWhiskey(container) {
5      for(let i = 0; i < 10; i++) {
6          let drop = getWhiskey();
7          //...
8      }
9  }
10
11 function addJuice(container) {
12     for(let t = 0; t < 3; t++) {
13         let tomato = getTomato();
14         //...
15     }
16 }
```

Здесь комментарии тоже не нужны: функции сами говорят, что делают (если вы понимаете английский язык). И ещё, структура кода лучше, когда он разделён на части. Понятно, что делает каждая функция, что она принимает и что возвращает.

В реальности мы не можем полностью избежать «объясняющих» комментариев. Существуют сложные алгоритмы. И есть хитрые уловки для оптимизации. Но в целом мы должны стараться писать простой и самодокументированный код.

## Хорошие комментарии

Итак, обычно «объясняющие» комментарии – это плохо. Но тогда какой комментарий считается хорошим?

### Описывайте архитектуру

Сделайте высокоуровневый обзор компонентов, того, как они взаимодействуют, каков поток управления в различных ситуациях... Если вкратце – обзор кода с высоты птичьего полёта. Существует специальный

Раздел

Качество кода

Навигация по уроку

Плохие комментарии

Хорошие комментарии

Итого

Комментарии

Поделиться



Редактировать на GitHub



язык [UML](#) для создания диаграмм, разъясняющих архитектуру кода. Его определённо стоит изучить.

### Документируйте параметры и использование функций

Есть специальный синтаксис [JSDoc](#) для документирования функций: использование, параметры, возвращаемое значение.

Например:

```
1  /**
2   * Возвращает x, возведённое в n-ную степень.
3   *
4   * @param {number} x Возводимое в степень число.
5   * @param {number} n Степень, должна быть натуральным ч
6   * @return {number} x, возведённое в n-ную степень.
7   */
8  function pow(x, n) {
9    ...
10 }
```

Подобные комментарии позволяют нам понимать назначение функции и правильно её использовать без необходимости заглядывать в код.

Кстати, многие редакторы, такие как [WebStorm](#), прекрасно их распознают для того, чтобы выполнить автодополнение ввода и различные автоматические проверки кода.

Также существуют инструменты, например, [JSDoc 3](#), которые умеют генерировать HTML-документацию из комментариев. Получить больше информации о JSDoc вы можете здесь: <http://usejsdoc.org/>.

### Почему задача решена именно таким способом?

Важно то, что написано. Но то, что не написано, может быть даже более важным, чтобы понимать происходящее. Почему задача решена именно этим способом? Код не даёт ответа.

Если есть несколько способов решить задачу, то почему вы выбрали именно этот? Особенно если ваш способ – не самый очевидный.

Без подобных комментариев возможна следующая ситуация:

1. Вы (или ваш коллега) открываете написанный некоторое время назад код и видите, что в нём есть, что улучшить.
2. Вы думаете: «Каким глупым я раньше был и насколько умнее стал сейчас», и переписываете его на «более правильный и оптимальный» вариант.
3. ...Желание переписать код – это хорошо. Но в процессе вы понимаете, что «оптимальное» решение на самом деле не такое уж и оптимальное. Вы даже смутно припоминаете, почему, так как в прошлый раз вы уже его пробовали. Вы возвращаетесь к правильному варианту, потратив время зря.

Комментарии, объясняющие решение, очень важны. Они помогают продолжать разработку в правильном направлении.

### В коде есть какие-то тонкости? Где они используются?

Если в коде есть какие-то тонкости и неочевидные вещи, его определённо нужно комментировать.

## Итого

Комментарии – важный признак хорошего разработчика, причём как их наличие, так и отсутствие.

Хорошие комментарии позволяют нам поддерживать код, дают возможность вернуться к нему после перерыва и эффективнее его использовать.

**Комментируйте:**

- Общую архитектуру, вид «с высоты птичьего полёта».

Раздел

[Качество кода](#)

Навигация по уроку

Плохие комментарии

Хорошие комментарии

Итого

Комментарии

Поделиться



[Редактировать на GitHub](#)



- Использование функций.
- Неочевидные решения, важные детали.

#### Избегайте комментариев:

- Которые объясняют, как работает код, и что он делает.
- Используйте их только в тех случаях, когда невозможно сделать настолько простой и самодокументированный код, что он не потребует комментариев.

Средства для генерации документации по коду, такие как JSDoc3, также используют комментарии: они их читают и генерируют HTML-документацию (или документацию в другом формате).

Проводим [курсы по JavaScript и фреймворкам](#). ✕



## Комментарии

перед тем как писать...

