

[🏠](#) → [Язык JavaScript](#) → [Качество кода](#) 12 июля 2022 г.

## 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     }
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 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 числом.
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: <https://jsdoc.app>.

### 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».
- 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).

 [Предыдущий урок](#)

[Следующий урок](#) 

Поделиться   

 [Карта учебника](#)

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



## Комментарии

- Если вам кажется, что в статье что-то не так - вместо комментария напишите [на GitHub](#).
- Для одной строки кода используйте тег `<code>` , для нескольких строк кода — тег `<pre>` , если больше 10 строк — ссылку на песочницу ([plnkr](#), [JSBin](#), [codepen...](#))
- Если что-то непонятно в статье — пишите, что именно и с какого места.

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS ?

Имя



14

Поделиться

Лучшие

Новые

Старые

**X****хлорка**

2 месяца назад



13.12.2022 9:31

0

1

Ответить • Поделиться ›

**Д****Дарья Янукович**

3 месяца назад



Привет!С ребятами создали сообщество и помогаем себе и другим с изучением javascript. Будем рады если присоединитесь к нам - ТЛГ deveveloper\_house\_jun\_front

0

3

Ответить • Поделиться ›

**Д****Дарья Янукович**

5 месяцев назад



Привет!С ребятами создали сообщество и помогаем себе и другим с изучением javascript. Будем рады если присоединитесь к нам - ТЛГ deveveloper\_house\_jun\_front

0

2

Ответить • Поделиться ›

**I****Илья**

6 месяцев назад



Ктонибудь может объяснить как работает данный код?

```
// здесь мы добавляем виски
for(let i = 0; i < 10; i++) {
  let drop = getWhiskey();
  smell(drop);
  add(drop, glass);
}
```

```
// здесь мы добавляем сок
for(let t = 0; t < 3; t++) {
  let tomato = getTomato();
  examine(tomato);
  let juice = press(tomato);
  add(juice, glass);
}
```

■ 6 месяцев назад

Ну я вижу 2 цикла, несколько разных функций и переменных. Код "сложный", поэтому перед циклами комментарии поясняющие. Но чтобы избавиться от них, автор выносит циклы в отдельные функции

1 0 Ответить • Поделиться ›

**iWerton**

→ Ilya

6 месяцев назад

— 🚩

NaN

0 0 Ответить • Поделиться ›

**H****Никита Самарин**

6 месяцев назад

— 🚩

13.08.22

0 0 Ответить • Поделиться ›

**N****Nait**

7 месяцев назад

— 🚩

Всем привет! Не понял, для чего тут вторая строчка (nextPrime:), буду благодарен, если объясните, гугл не помог.

```
function showPrimes(n) {
  nextPrime:
  for (let i = 2; i < n; i++) {
    // проверяем, является ли i простым числом
    for (let j = 2; j < i; j++) {
      if (i % j == 0) continue nextPrime;
    }
    alert(i);
  }
}
```

1 0 Ответить • Поделиться ›

**Валерий Языков**

→ Nait

2 месяца назад

— 🚩

Если условие не выполняется, то возврат идёт к метке (2 строка), если бы этой ссылки не было то continue бы работало только внутри второго цикла.

1 0 Ответить • Поделиться ›

**Илья Александров**

→ Nait

7 месяцев назад edited

— 🚩

Это метка для цикла. она используется для циклов в цикле, чтобы дать название определенному циклу, чтобы можно было к нему обратиться. Это для того нужно, чтобы если во вложенном внутри цикле не соблюдается условие, то break/continue применялся на нужный цикл, а не только на тот, где требование не соблюдается. Вроде так, подробнее в разделе о

циклах найдете

4 0 Ответить • Поделиться ›



**Favet**

→ Nait

7 месяцев назад



Если я не ошибаюсь, если условие не выполняется, всё сводится к nextPrime и код будет повторяться, пока условия алгоритма не будут выполнены

0 0 Ответить • Поделиться ›

**F**

**friz**

7 месяцев назад



А вы тоже все ссылки игнорируете?

2 0 Ответить • Поделиться ›

**V**

**Viktor**

7 месяцев назад



22.07.2022 10:04 На ураааа)

0 6 Ответить • Поделиться ›



**Uliwe**

→ Viktor

7 месяцев назад edited



Здесь люди в комментарии заходят чтобы посмотреть какие то вопросы по теме или разные решения задач, но никак не для того, чтобы узнать во сколько ты, идиот тупоголовый прошел этот урок, debil блядь

4 0 Ответить • Поделиться ›

**M**

**Макс Крутой**

→ Uliwe

4 месяца назад



Как лайкнуть?

0 0 Ответить • Поделиться ›

**D**

**Дима Василенко**

→ Viktor

7 месяцев назад



Да всем пофигу на твои "На урааа", если уж на то пошло, на рабочем столе текстовый файл создай и там отмечай свои "успехи"

6 0 Ответить • Поделиться ›

**V**

**Viktor**

→ Дима Василенко

7 месяцев назад



Ну мне комфортнее здесь, так что если тебе не нравятся мои комментарии, заблокируй меня и учи себе дальше. Тут много кто отмечает дату прохождения тем, так что я такой не один

0 5 Ответить • Поделиться ›

**V**

**Valentin**

→ Viktor

час назад





ты прост реально либо малолетка тупая либо идиот, ты вбросы делаешь, к которым не вернешься, твои комменты в пустую, ты про них даже не вспомнишь, лучше води дневник, записывай заметки

0 0 Ответить • Поделиться ›

**C**

**Слава**

→ Viktor

7 месяцев назад

— 🚩

Перестаньте изучать Джаваскрипт. Оставьте в покое этот язык, его не для вас делали!

2 0 Ответить • Поделиться ›

**V**

**Viktor**

→ Слава

7 месяцев назад

— 🚩

Вижу у тебя проблему с половой жизнью, тебе срочно надо найти девушку, хотя нет, ее у тебя не будет как и мозгов

0 0 Ответить • Поделиться ›

**Загрузить ещё комментарии**

Подписаться

Privacy

Не продавайте мои данные