Правила оформления js-кода

Редактировать эту страницу

Самое главное

Как писать неподдерживаемый код?

Эта статья в полной мере отражает своё название, в ней описаны основные ошибки принаписании js кода и дано разъяснение "почему это плохо" по каждой из них.

Комментирование кода

> Если вам кажется, что нужно добавить комментарий для улучшения понимания, это значит, что ваш код недостаточно прост, и, может, стоит переписать его.
>
> — Р.Мартин, Чистый код

При комментировании кода следует использовать JSDoc комментарии. Такие комментарии позволяют сразу понять, что принимает и что делает функция:

```
/**
 * Возвращает x в степени n, только для натуральных n
 *
 * @param {number} x Число для возведения в степень.
 * @param {number} n Показатель степени, натуральное число.
 * @return {number} x в степени n.
 */
function pow(x, n) {
    ...
}
```

Для однострочных поясняющих комментариев в коде следует использовать обычные комментарии, начинающиеся с `//`.

Очень важны комментарии, которые объясняют не ***что***, а ***почему*** в коде происходит именно это.

Без поясняющих комментариев возможна, например, такая ситуация:

- Вы открываете код, который был написан какое-то время назад, и видите, что он неоптимален/говнокод/так_никто_не_делает.

- Переписываете под «более очевидный и правильный» вариант.
- Порыв, конечно, хороший, да только этот вариант вы, или кто-то другой уже обдумали раньше. И отказались, а почему — забыли. И если бы был поясняющий комментарий — вы бы не потеряли время впустую.

Оформление JavaScript кода

- Используем строгий режим `'use strict';` там, где это возможно.
- В качестве отступов используем табуляцию, а не пробелы.
- После названия функции ставим пробел.
- Между скобкой и первым параметром, между последним параметром и скобкой пробелы не ставятся.
- Между параметрами и вокруг операторов ставим пробел.
- Перед открывающей фигурной скобкой ставим пробел (египетский стиль).
- Открывающая фигурная скобка находится на одной строке.
- Закрывающая фигурная скобка располагается на следующей строке.
- Между логическими блоками ставим пустую строку.
- Точки с запятой нужно ставить, даже если их, казалось бы, можно пропустить.

Используйте одинарные кавычки для строк. Тогда html-код не будет различаться в js и html.

```
// Плохо
var html = "<div class='my-class'></div>";

// Хорошо
var html = '<div class="my-class"></div>';
```

Используйте шорткаты.

```
// плохо
if (name !== '') {
    // делаем что-то
}

// хорошо
if (name) {
    // делаем что-то
}

// плохо
if (collection.length > 0) {
    // делаем что-то
}

// хорошо
if (collection.length) {
    // делаем что-то
}
```

Уровни вложенности

- Уровней вложенности должно быть немного.
- Лучше быстро обработать простые случаи, вернуть результат, а дальше разбираться со сложным, без дополнительного уровня вложенности.
- Главное — не краткость кода, а его простота и читаемость. Совсем не всегда более короткий код проще для понимания, чем более развёрнутый.

Например функцию:

```
function isEven(n) { // проверка чётности
    if (n % 2 == 0) {
        return true;
    }
    else {
        return false;
    }
}
```

можно упростить избавившись от лишней вложенности:

```
function isEven(n) { // проверка чётности
    if (n % 2 == 0) {
        return true;
    }

    return false;
}
```

Переменные

- **Имя переменной** — существительное.
- **Имена переменных не должны начинаться с подчеркивания.**
- Имя переменной пишем в верблюжей нотации (camelCase).

Именуем переменные в стиле camelCase.

```
var myName = 'Фёдор';
```

Несколько переменных подряд перечисляем через запятую, при этом следует перечислять переменные по одной на строку и выравнивать вертикально.

```
var myName = 'Фёдор',
        myGender = 'мужик',
        myAge = '35';
```

Допускается использование короткого синтаксиса условий ifelse при назначении переменных, но лишь в тех случаях, когда условия просты

```
```

```
// Плохо
var i = i ? i < 0 ? Math.max(0, len + i) : i : 0;

// Хорошо
var foo = bar || 'not set',
        baz = (foo == 150) ? 'wow' : 'sad';
```

Для упрощения восприятия кода следует ставить префикс `$` для переменных, являющихся JQuery объектом.

```
var $this = $(this),
        $data = $this.data(),
        id = $data.id;
```

Функции

- **Имя функции — глагол** или начинается с глагола.
- Имя функции должно **понятно и чётко отражать, что она делает**.
- Имя функции пишем в верблюжей нотации (camelCase).
- Обязательно использование `"use strict";` в функциях!
- Функции пишем в конец файла (по фозможности).
- Функции должны быть небольшими.
- Если функция большая — желательно разбить её на несколько.
- **Одна функция — одно действие**

Функции, которые начинаются с "show" — что-то показывают:

```
showMessage(...)     // префикс show, "показать" сообщение
```

Функции, начинающиеся с "get" — получают, и т.п.:

```
getAge(...)          // get, "получает" возраст
calcD(...)           // calc, "вычисляет" дискриминант
createForm(...)      // create, "создает" форму
checkPermission(...) // check, "проверяет" разрешение, возвращает true/false
```

Условия

Всегда пишем в нормальном виде, с открывающими и закрывающими скобками, даже если условие в одну строку.

```
// плохо
if(foo = bar) console.log('foo = bar');

// хорошо
if (foo = bar) {
        console.log('foo = bar');
}
```

Инструменты для автоформатирования кода, настройка параметров

Плагины для SublimeText

- jsFormat — плагин для форматирования js-кода.
- jQuery bundle — подсветка синтаксиса и сниппеты для jQuery.
- JavaScript Completions — Автокомплит для js-функций и методов.
- JSHint — Удобный и наглядный плагин для проверки корректности js-кода.
- Minify — Плагин для минификации js/html/css кода.

Для подготовки раздела использовалась так же статья Советы по стилю кода