

Яндекс



# Code Style

Михаил Давыдов  
Разработчик JavaScript

# История про разработчиков

В ролях

- разработчик А
- разработчик Б (новичек)

Эта история вымышленная  
и к Яндексу не относится

Жил-был один разработчик  
и писал вот так:

# Код разработчика А

```
function name(value)
  -tab->{
  -tab->-tab->var a = 0, b = 42;
  -tab->-tab->if (value) return a
  -tab->}
```

**В проект пришел еще один и  
стал писать так:**

# Код разработчика Б

```
var name = function (value) {  
  . . . var a = 0,  
  . . . . . b = 42;  
  
  . . . if (value) {  
  . . . . . return a;  
  . . . }  
};
```



И общего Code Style у них  
не было...



# Код разработчика Б

```
var name = function (value) {
```

```
    ... var a = 0,
```

```
    ... if (value) {
```

```
        ... return a;
```

```
    ... }
```

```
};
```

# Код разработчика А

```
function name(value)
```

```
-tab->{
```

**Б: У самого говнокод!**

```
-tab->-tab->if (value) return a
```

```
-tab->}
```

# Код разработчика Б

```
var name = function (value) {
```

```
    ...var a = 0,
```

```
    ...b = 42;
```

**А: Почему так ф-ю**

```
    ...if (value) {
```

**написал?**

```
    ...return a;
```

```
    ...}
```

```
};
```

# Код разработчика А

```
function name(value)
  -tab->{
    -tab->-tab->var a = 0, b = 42;
    -tab->-tab->::(value) return a
  -tab->}
```

**Б: А ты вообще табы используешь!**

# Код разработчика Б

# А: Зачем тут скобка?

```

    ... if (value) {
    ...     return a;
    ... }
} ;

```



# Код разработчика А

```
function name(value)
  -tab->{
  -tab->-tab->var a = 0, b = 42;
  -tab->-tab->if (value) return a
  -tab->}
```

**Б: Уайтсмитс? Ты из  
какой пещеры вылез?**



# Код разработчика Б

**А: Зачем тут точка с запятой?**

```
... if (value) {  
... return a;  
... }  
};
```

# Код разработчика А

```
function name(value)
  -tab->{
  -tab->-tab->var a = 0, b = 42;
  -tab->-tab->if (value) return a
  -tab->}
```

**Б: А ты зачем перенос  
не поставил?**

# Код разработчика Б

```
var name = function (value) {
```

```
... var a = 0,
```

```
... b = 42;
```

**A: RTFM or GTFO!**

```
... if (value) {
```

```
... return a;
```

```
... }
```

```
} ;
```

# Код разработчика А

```
function name(value)
- tab->{
- tab->var a = 42;
- tab->- tab->if (value) return a
- tab->}
```

Б: А в бубен?



Нужна договоренность в  
стиле кода

# Организационный профит Code Style

- Это закон
  - Должна быть отдельная страница о которой все знают
  - Страницу просто найти
- Закон разрешает конфликты
  - Не знание закона не освобождает от ответственности
- Решает большинство проблем
- Эту страницу нужно дать прочитать перед началом работы

Если не понятно, то нужно  
дать понять причину ввода  
такого Code Style



# Применение CS

Для быстрого восприятия кода

Для предотвращения ошибок в коде

Для быстрого написания кода

Каждый из следующих  
примеров зависит от  
восприятия конкретного  
человека

# Обобщение элементов

# Обобщение соседних элементов

```
var name = function (value) {  
  . . . var a = 0,  
  . . . . . b = 42;  
  . . . if (value) {  
  . . . . . return a;  
  . . . }  
};
```

# Обобщение соседних элементов

Перенос строки разрывает обобщение

```
var name = function (value) {  
  . . . var a = 0,  
  . . . . . b = 42;  
  
  . . . if (value) {  
  . . . . . return a;  
  . . . }  
};
```

# Обобщение соседних элементов

Перенос строки разделяет разные типы CSS свойств и обобщает соседние элементы

```
.b-form {  
    font-family: Arial;  
    font-size: 2em;  
  
    padding: 0 10px;  
    min-height: 100px;  
}
```

# Вертикаль – список

# Вертикаль – список

**var** - список переменных

```
var name = function (value) {  
  . . . var a = 0,  
  . . . . . b = 42;  
  
  . . . if (value) {  
  . . . . . return a;  
  . . . }  
};
```



# Вертикаль – список

Coma-first style – его цель уменьшить число ошибок.

```
var name = function (value) {  
  . . var a = 0  
  . . . , b = 42  
  . . . , c = 42  
  . . . , d = 42  
  . . . ;  
  
} ;
```

# Вертикаль – список

Отсутствие запятых легко заметить

```
var name = function (value) {  
  . . var a = 0  
  . . . , b = 42  
  . . . c = 42  
  . . . , d = 42  
  . . . ;  
  
} ;
```

# Вертикаль – список

Coma-first style для литералов Object

```
var a =  
  . { b: 2  
  . , c: 3  
  . , d: 4  
  . }  
  . ;
```

<https://gist.github.com/357981>

ALL code is ugly.  
Yours, mine, everyone's.  
Code Is Ugly. Just face it.

# Вертикаль – список

## Выравнивание по свойству

```
.b-form {  
    -webkit-transition: color;  
    -moz-transition: color;  
    -ms-transition: color;  
    -o-transition: color;  
    transition: color;  
}
```

# Вертикаль – список

Выравнивание по значению  
Так проще прочитать

```
.b-form {  
    -webkit-transition: color;  
    -moz-transition: color;  
    -ms-transition: color;  
    -o-transition: color;  
    transition: color;  
}
```

# Вертикаль – список

```
<ol class="b-list">  
... <li class="b-item">Abc</li>  
... <li class="b-item">Abc</li>  
... <li class="b-item">Abc</li>  
</ol>
```

# Вертикаль – список

## Вызов функций по цепочке – список

```
jQuery( '.clickable' )  
  .filter( 'a' )  
  .click( handler )  
  .end()   
  .show( 'slow' )  
  ;
```



# Вертикаль – список

## Несколько условий – список

```
if (typeof a !== "undefined" &&  
    typeof b !== "undefined" &&  
    typeof c === "string") {  
  
    // your stuff  
  
}
```

# Единообразие

# Единообразие

Похожий код быстрее воспринять

```
var name = function (value) {  
    . . . var a = 0,  
    . . . . . b = 42;  
  
    . . . if (value) {  
    . . . . . return a;  
    . . . }  
};
```

# Единообразие

Похожий код быстрее воспринять

```
// Этот пробел обязательный  
function A(value) {  
  
}  
  
// Этот пробел для единообразия  
var A = function (value) {  
  
};
```

# Разрядка

# Разрядка

БуряМглоюНебоКроет vs  
Буря мглою небо кроет

```
var name = function (value) {  
    . . . var a = 0,  
    . . . . . b = 42;  
  
    . . . if (value) {  
    . . . . . return a;  
    . . . }  
};
```

# Разрядка, но без фанатизма

Буря          Мглою          Небо          Кроет vs  
Буря мглою небо кроет

```
for (var i = 0; i < 100; i++) {  
  
}
```

```
for (var i = 0; i < 100; i++) {  
  
}
```

**Сильно разряженный код  
сложно воспринять**



Однако не везде она работает


В HTML перед и после равно пробел как правило не ставят

```
<div class="b-header"></div>
```

**Начало - конец**

# Начало блока "видит" конец блока

Код представляет из себя "параграфы текста"

```
var name = function (value) {  
  . . . var a = 0,  
  . . . . . b = 42;  
   f (value) {  
    . . . return a;  
  }  
};
```

Начало блока "видит" конец блока


Код представляет из себя "параграфы текста"

```
var name = function (value) {  
  . . . var a = 0,  
  . . . . . b = 42;  
  . . . if (value) {  
    . . . return a;  
  }  
};
```

The diagram illustrates the block structure of the code. A large blue arrow on the left points from the opening curly brace of the function to the closing curly brace and semicolon, representing the function's scope. A smaller blue arrow points from the opening curly brace of the 'if' statement to its closing curly brace, representing the 'if' block's scope. Blue dots are placed at the start of each line of code to indicate the beginning of a new line.

# Начало блока "видит" конец блока

```
<div class="b-header">  
... <div class="b-item">Abc</div>  
... <div class="b-item">Abc</div>  
... <div class="b-item">Abc</div>  
</div>
```



# Начало блока "видит" конец блока

Не везде это применимо: много писать, мельтешение, баги с пробелами

```
<div class="b-header">
  . . . . <div class="b-item">
  . . . . ↓ . . . . Abc
  . . . . </div>
  . . . . <div class="b-item">
  . . . . ↓ . . . . Abc
  . . . . </div>
</div>
```

# Предотвращение ошибок





# Лишние пробелы

Переносы и пробел = +2 лишних пробельных символа

```
var item = $( '.b-item' ) [ 0 ] ;  
  
if ( item.innerHTML = "Abc" ) {  
    // Недостижим  
}
```

Точка с запятой в FE

# Точка с запятой в Function Expression

```
var name = function (value) {  
  . . . var a = 0,  
  . . . . . b = 42;  
  
  . . . if (value) {  
  . . . . . return a;  
  . . . }  
}
```

```
cat **/*.js > all.js
```


# Точка с запятой в Function Expression

SyntaxError: missing ; before statement

```
var name = function (value) {  
  
} var name2 = function (value) {  
  
}
```

# Точка с запятой в Function Expression

```
var name = function (value) {  
  . . . var a = 0,  
  . . . . . b = 42;  
  
  . . . if (value) {  
  . . . . . return a;  
  . . . }  
}
```



# Скобки у циклов

# Скобки у циклов и условий

Лень писать скобки

```
var name = function (value) {  
    . . . var a = 0,  
    . . . . . b = 42;  
  
    . . . if (value)  
    . . . . . return a;  
};
```



# Скобки у циклов и условий

## Редкая логическая ошибка


```
var name = function (value) {  
    . . . var a = 0,  
    . . . . . b = 42;  
  
    . . . if (value)  
    . . . . . a = 42;  
    . . . . . return a;  
};
```

# Скобки у циклов и условий

```
var name = function (value) {  
    . . . var a = 0,  
    . . . . . b = 42;  
  
    . . . if (value) {  
    . . . . . a = 42;  
    . . . . . return a;  
    . . . }  
};
```


# Скобки у циклов и условий

Много скобок создает много шума

```
var name = function (value) {  
    . . . . var a = 0,  
    . . . . . . . . b = 42;  
  
    . . . . if (value)  return a;  
};
```

# Auto semicolon insertion

# Auto semicolon insertion

```
var name = function (value) {  
  . . . var a = 0,  
  . . . . . b = 42;  
  
  . . . if (value) {  
    . . . . . return   
    . . . . . {pewpew: a};  
  . . . }  
};
```

# Auto semicolon insertion

После return будет поставлена ;

```
var name = function (value) {  
  . . . var a = 0,  
  . . . . . b = 42;  
  
  . . . if (value) {  
  . . . . . return;  
  . . . . . {pewpew: a};  
  . . . }  
};
```

<http://es5.github.com/#x7.9>

# Свойства с префиксами в CSS

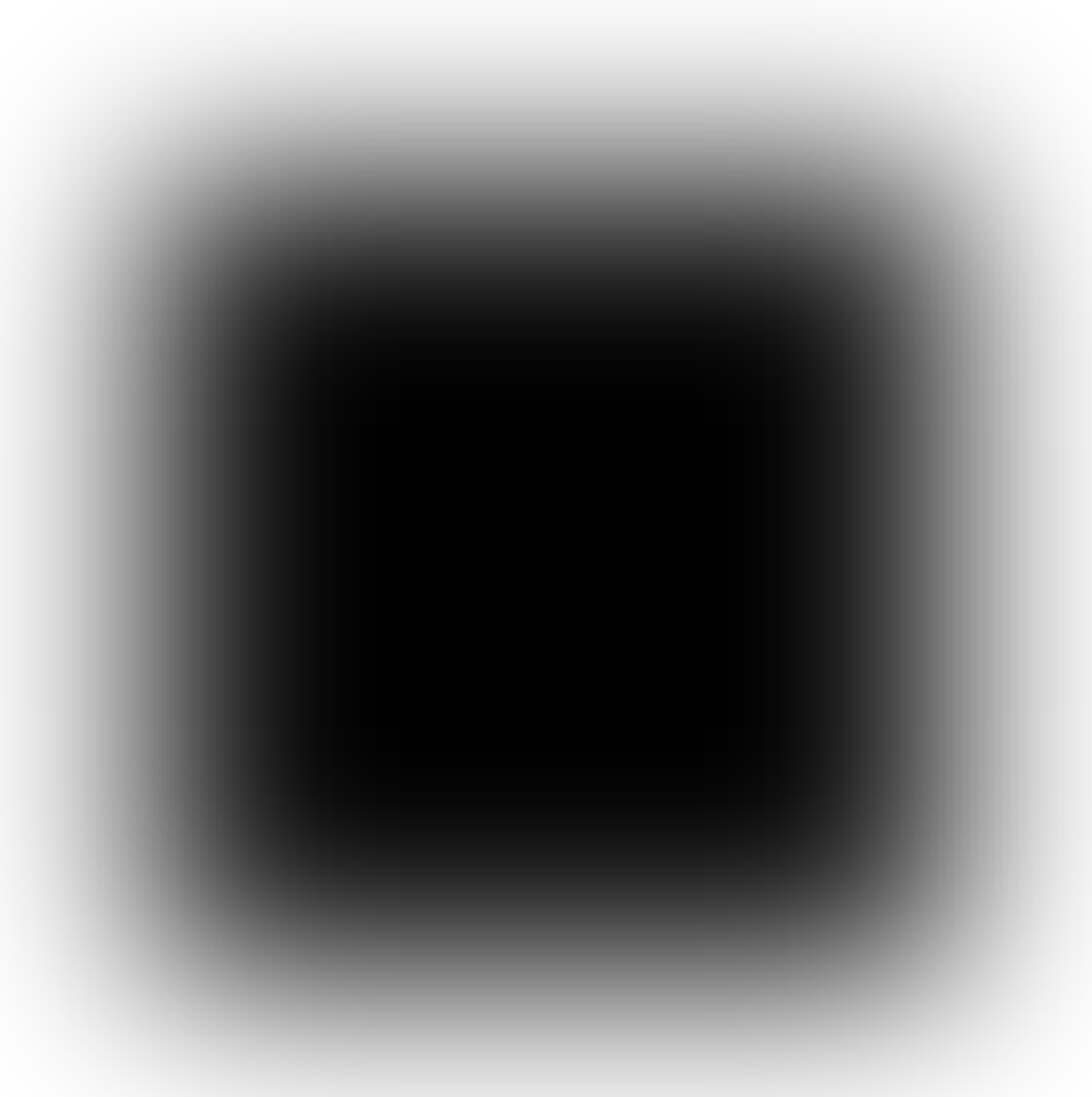
Последнее свойство перекрывает  
предшествующее

# Свойства с префиксами в CSS

```
.b-form {  
    box-shadow: 0 0 400px 200px #000;  
    -webkit-box-shadow: 0 0 400px 200px #000;  
}
```



# Свойства с префиксами в CSS



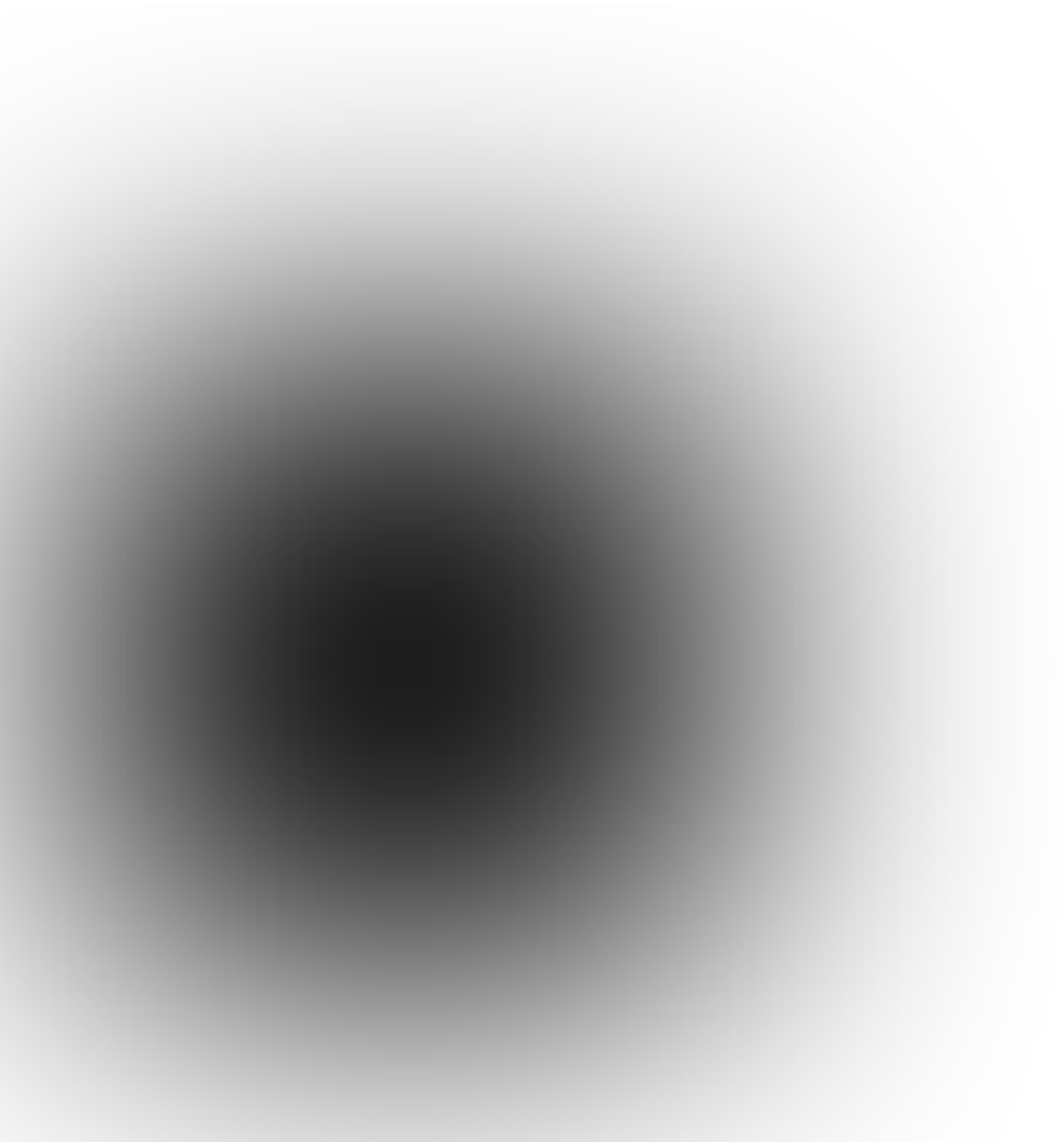
# Свойства с префиксами в CSS

Последним должен идти свойство без префикса

```
.b-form {  
    -webkit-box-shadow: 0 0 400px 200px #000;  
    box-shadow: 0 0 400px 200px #000;  
}
```

<http://pepelsbey.net/pres/pre-fixes/>

# Свойства с префиксами в CSS



# Еще несколько слов...

- **Документирование неявного кода**
  - Код должен быть максимально самодокументируемым
- **Написание примеров использования**
  - Чтобы не рыться в коде
- **Сокращение длины строки**
  - Чтобы не вертеть головой и не скролить
- **Уменьшение уровней вложенности**
  - Быстрее читать код
- **Единообразие в модулях и именах**
- **Автоматизированная проверка кода** ♥
  - IDE
  - Pre Commit Hook

**Код должен выглядеть так  
как будто его пишет один  
человек**

А как в Яндексе?

# Как мы делаем в Яндексе

- Love & Pease
- У каждого проекта свой Code Style
  - Страница в Вики
  - Или CONTRIBUTE.md файл в корне проекта
- Есть стиль по умолчанию
  - Страница в Вики со стилей по языкам
  - Не обязательный – на случай если лень
- Стараемся использовать общепринятый
  - Так проще новым разработчикам
  - Привычней читать сторонний код
- Code Review
  - Зависит от размера проекта
  - Как правило новички проходят обязательно

# Заключение

- Соглашение Code Style - **Must have!**
- Нужно знать меру
  - Разрядка пробелами
  - Фигурные скобки
- Все правила имеют на то причину



01 WINS

81

00 WINS

CAGE

RAIDEN

FRIENDSHIP





Михаил Давыдов

Разработчик JavaScript

[azproduction@yandex-team.ru](mailto:azproduction@yandex-team.ru)



azproduction

# Спасибо