

Code Styles

Качество кода

Руководства по стилю

Прочитать:

<http://learn.javascript.ru/coding-style>

[Google JavaScript Style Guide](#)

[Airbnb JavaScript Style Guide](#)

[jQuery JavaScript Style Guide](#)

Пробел между параметрами

Между именем функции и скобкой (между скобкой и первым параметром нет пробела)

Фигурная скобка { на той же строке, через пробел

Отступ 2 пробела

Пробел после for

Пробелы вокруг операторов

точка с запятой; обязательна

Пробел между параметрами

Пустая строка между логическими блоками

Длина строки не более 80 символов

} else { без перевода строки

Пробелы вокруг вложенного вызова

```
function pow(x, n) {
  var result = 1;
  for (var i = 0; i < n; i++) {
    result *= x;
  }
  return result;
}

var x = prompt("x?", "");
var n = prompt("n?", "");
if (n < 0) {
  alert('Степень ' + n +
    'не поддерживается, введите целую степень, большую 0');
} else {
  alert( pow(x, n) );
}
```

Самые известные – это:

АВТОМАТИЗИРОВАННЫЕ средства проверки

[JSLint](#) – проверяет код на соответствие [стилю JSLint](#), в онлайн-интерфейсе вверху можно ввести код, а внизу различные настройки проверки, чтобы сделать её более мягкой.

[JSHint](#) – вариант JSLint с большим количеством настроек.

[ESLint](#) - вариант для проверки современного стандарта ECMAScript

[Closure Linter](#) – проверка на соответствие [Google JavaScript Style Guide](#).

Как писать неподдерживаемый код?

- **Однобуквенные переменные**

Называйте переменные коротко: `a`, `b` или `c`.

- **Русские слова и сокращения**

В одном месте напишите `var ssilka`, в другом `var ssylka`, в третьем `var link`, в четвёртом – `var lnk...` Это действительно великолепно работает и очень креативно!

- **Будьте абстрактны при выборе имени**

При выборе имени старайтесь применить максимально абстрактное слово, например `obj`, `data`, `value`, `item`, `elem` и т.п.

Как писать неподдерживаемый код?

- **Повторно используйте имена**

По возможности, повторно используйте имена переменных, функций и свойств. Просто записывайте в них новые значения.

- **Перекрывайте внешние переменные**

```
1 var user = authenticateUser();
2
3 function render() {
4   var user = anotherValue();
5   ...
6   ...многобукв...
7   ...
8   ... // <-- программист захочет внести исправления сюда, и..
9   ...
10 }
```

Задание

Продолжить работу над заданием поле и превратить его в игру “Сапер”

Описание задания: <https://goo.gl/LsyfYL>

Прочитать: <http://learn.javascript.ru/coding-style>

Прочитать: <http://learn.javascript.ru/write-unmain-code>

Проверить свой код на наличие стилистических ошибок по Google Style

User Story Mapping

Понять, концепцию story mapping

Опишите действия которые вы совершили, чтобы оказаться здесь

- начните с того, что вы проснулись, заканчивая прибытием сюда
- каждое действие на отдельный стикер



User Story Mapping



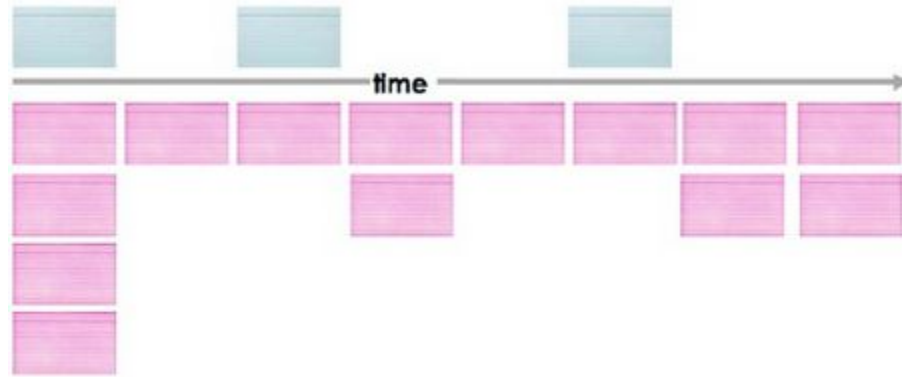
- Определить ключевые виды деятельности(активности) персон, которые должен поддерживать продукт, каждый вид деятельности записать на отдельной карточке.
- Расположить их по порядку использования слева направо.

User Story Mapping



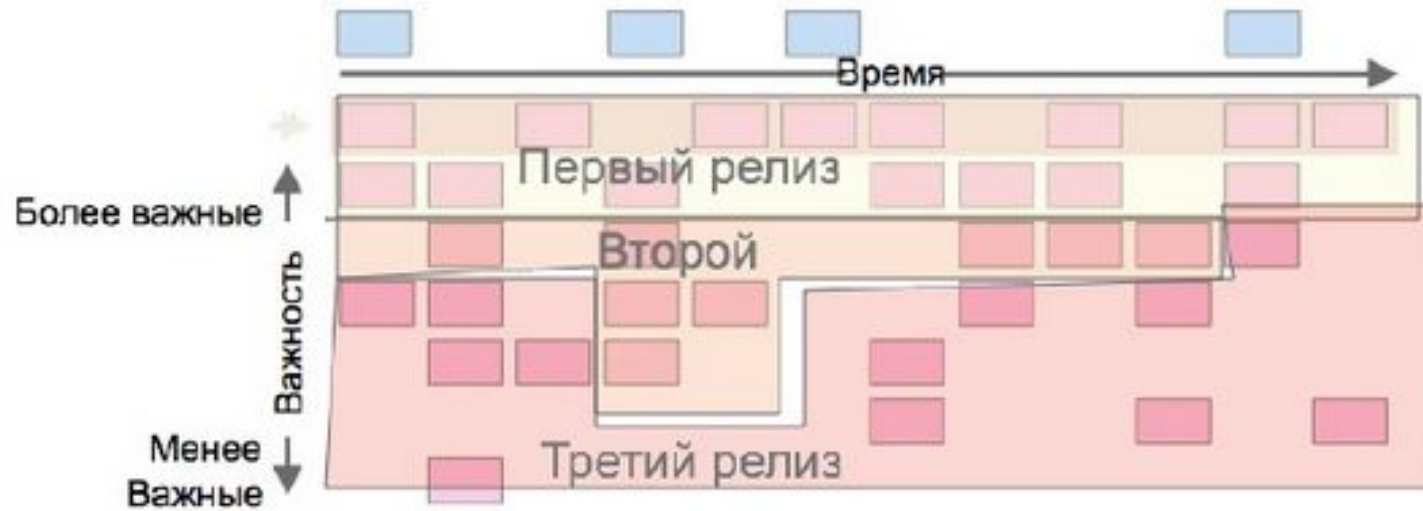
- Определить отдельные задачи, которые составляют каждую активность.
- Расположить задачи в одной строке в логическом, последовательном порядке под соответствующим видом деятельности.
- Проверить активности и задачи

User Story Mapping



Добавьте ниже подзадачи, дополнения или улучшения, основываясь на приоритете пользователя

User Story Mapping



Вопросы



Dmitry Anikin

CTO of Roonyx

E-mail: dima@roonyx.tech

GitHub: <https://github.com/d-anikin>