Кодстайл — общие принципы

Предисловие

Это внутренние правила Яндекс Практикума, которые касаются форматирования и оформления HTML- и CSS-кода. Соблюдение этих правил:

- позволит избежать ошибок, обычных на первом этапе обучения;
- даст навык грамотного оформления кода;
- приучит к соблюдению внутренних стандартов — а этого от вас потребуют в любой IT-компании;
- облегчит работу студентам и преподавателям Яндекс Практикума.

Эти правила применимы к рабочим файлам, содержащим HTML- и CSS-код.

Общие правила оформления кода

Используйте два пробела для отступа

Применяйте для отступов пробелы, не используйте табуляцию или сочетания пробелов и табуляции.

```
<div class="parent">
  <div class="child">
  </div>
</div>
```

Отступы в два пробела делают код компактнее. Такое форматирование чаще используется в стандартах IT-компаний.

В большинстве редакторов кода удобно нажимать клавишу Tab для создания вложенности. Этот символ интерпретируется как табуляция.

Чтобы изменить эту настройку и установить для клавиши Tab отступ в два пробела, в редакторе Visual Studio Code можно использовать специальные плагины. Самый популярный из них — Editorconfig.

Процесс настройки параметров Visual Studio Code описан в официальном справочнике. На практике вам достаточно установить сам плагин. Найти его можно в разделе с плагинами по имени Editor Config for VS Code. Или на маркетплейсе по этой ссылке.

После установки плагина поместите в корень проекта файл конфигурации с именем `.editorconfig` и таким содержимым:

```
# http://editorconfig.org


# A special property that should be specified at the top of the file outside of

# any sections. Set to true to stop .editor config file search on current file

root = true

[*]
# Indentation style

# Possible values - tab, space

indent_style = space

# Indentation size in single-spaced characters

# Possible values - an integer, tab
```

```
indent_size = 2

# Line ending file format

# Possible values - lf, crlf, cr

end_of_line = lf

# File character encoding

# Possible values - latin1, utf-8, utf-16be, utf-16le

charset = utf-8

# Denotes whether to trim whitespace at the end of lines

# Possible values - true, false

trim_trailing_whitespace = true

# Denotes whether file should end with a newline

# Possible values - true, false

insert_final_newline = true
```

Применяйте кодировку UTF-8

Установите кодировку UTF-8 в HTML-шаблонах и документах — `<meta charset="utf-8">`. Сохраняйте файлы шаблонов и документов в той же кодировке (чаще всего она устанавливается автоматически).

Не нужно специально прописывать кодировку для таблиц стилей. Им по умолчанию присваивается UTF-8. Это универсальная кодировка, её поддерживают все современные браузеры и приложения.

Используйте только строчные символы

Весь код, как в HTML, так и в CSS, должен быть написан строчными символами.

Это касается имён HTML-элементов, атрибутов и их значений за исключением text/CDATA, CSS-селекторов, свойств и значений свойств за исключением строк, например, в значениях атрибутов alt и title.

Заглавными буквами писали, чтобы элементы разметки выделялись в коде. Когда появились редакторы кода с подсветкой синтаксиса, надобность в заглавных буквах отпала. Их использование воспринимается как "old school".

```
<!-- Не рекомендуется -->
<A HREF="/">Home</A>

<!-- Рекомендуется -->

<img src="yandex.png" alt="Yandex">
```

```
/* Не рекомендуется */
color: #E5E5E5;

/* Рекомендуется */
color: #e5e5e5;
```

Удаляйте пробелы в конце строк

Пробелы в конце строк не нужны и могут усложнить операции с кодом.

```
<!-- Не рекомендуется -->

<p>Пробел в конце строки? </p>
```

```
<!-- Рекомендуется -->

<p>Спасибо, не нужно.</p>
```

Отделяйте оформление от разметки

Не используйте теги для оформления элементов. Не полагайтесь на центрирование через тег `<center>` или изменение веса шрифта через тег `<b>`. Пусть стили определяются через CSS. Вторая буква ‘S’ в аббревиатуре CSS значит ‘Style’, а многие декоративные теги устарели.

```
<!-- Не рекомендуется -->

<p>Текст с <b>акцентом</b></p>

<!-- Рекомендуется -->

<p>Текст с <span class="accent">акцентом</span></p>
```

Не пишите стили внутри HTML, если можете обойтись без этого.

```
<!-- Не рекомендуется -->

<p>Текст с <span style="font-weight: bold;">акцентом</span></p>

<!-- Рекомендуется -->

<p>Текст с <span class="accent">акцентом</span></p>
```

Не используйте атрибут type для файлов CSS

Прописывать атрибуты type для связанных таблиц стилей нет необходимости, так как HTML5 присваивает типы text/css по умолчанию. Это работает даже в старых версиях браузеров.

```
<!-- Не рекомендуется: использован атрибут type -->

<link rel="stylesheet" href="https://www.google.com/css/maia.css" type="text/css">

<!-- Рекомендуется -->

<link rel="stylesheet" href="https://www.google.com/css/maia.css">
```

Пишите с новой строки и с отступами

Пишите с новой строки каждый новый элемент: блок, список или таблицу. Делайте отступ перед каждым новым дочерним элементом. Это упростит чтение кода

Если много атрибутов в HTML, переносите их на новые строки

Хотя в языке HTML нет рекомендации, ограничивающей длину строк, длинные строки стоит разбивать — такой код станет проще читать.

При переносе строк их продолжения должны иметь отступ минимум в два пробела относительно начальной строки.

```
<a
  href="https://yandex.ru"
  class="my-wonderful-link"
  target="_blank"
  id="myId"
> Ссылка с большим количеством атрибутов</a>
```

Используйте в HTML двойные кавычки

Заключайте значения атрибутов в двойные (""), а не одинарные (') кавычки. Это упростит взаимодействие с CSS и JS, улучшит читаемость кода. Когда в HTML-разметке кавычки двойные, а в коде CSS и скриптов одинарные, легче различать области разных языков.

```
<!-- Не рекомендуется: использованы одинарные кавычки -->

<a class='maia-button maia-button-secondary'>Войти </a>

<!-- Рекомендуется -->

<a class="maia-button maia-button-secondary">Войти</a>
```

CSS

Разделение правил

Всегда оставляйте пустую строку (двойной переход на новую строку) между правилами.

```
html {
  background: #fff;
}

body {
  margin: auto;
  width: 50%;
}
```

Ставьте пробел перед открывающей фигурной скобкой

Всегда ставьте пробел между селектором и открывающей фигурной скобкой перед блоком объявления.

В каждом правиле открывающая фигурная скобка должна стоять на той же строке, что и селектор:

```
/* Не рекомендуется: не хватает пробела */

.video{
  margin-top: 1em;
}
/* Рекомендуется */

.video {
  margin-top: 1em;
}
```

Разделяйте пробелом свойство и значение

Всегда ставьте одиночный пробел между свойством и значением, но не между свойством и двоеточием.

```
/* Не рекомендуется: нет пробела после двоеточия */

h3 {
  font-weight:bold;
}

/* Рекомендуется */

h3 {
  font-weight: bold;
}
```

Отделяйте объявления стилей

Каждое объявление стилей пишите на отдельной строке:

```
/* Не рекомендуется: два объявления стилей — на одной строке */

a:focus {
  position: relative; top: 1px;
}

/* Рекомендуется */

h1 {
  font-weight: normal;
  line-height: 1.2;
}
```

Соблюдайте пунктуацию

После каждого объявления ставьте точку с запятой. Если что, валидатор вас поправит. Отсутствие пунктуации — нарушение стандартов W3.org, техническая ошибка.

```
/* Не рекомендуется: забыли точку с запятой */

.test {
  display: block;
  height: 100px
}

/* Рекомендуется */

.test {
  display: block;
  height: 100px;
}
```

Давайте классам и идентификаторам осмысленные названия

Используйте имена, отражающие назначение блоков и элементов. Это делает код более читаемым и понятным. Для работы с CSS используйте селекторы классов.

```
/* Не рекомендуется: бессмысленное имя */

.abyrvalg {}

/* Не рекомендуется: описательное имя */
```

```
.button-green {}

/* Рекомендуется: конкретное описательное имя */

.gallery {}
```

Не назначайте стили элементам и ID, используйте классы

Не задавайте стили для имён элементов HTML или сочетаний этих имён с идентификаторами и классами. Это излишне расширяет или излишне ограничивает область применения стилей.

```
/* Не рекомендуется (установлен стиль для элемента): */

div {}
/* Не рекомендуется (установлен стиль для элемента с классом): */

div.error {}

/* Рекомендуется: */

.error {}
```

Без исключительной необходимости не назначайте стили элементам по ID, избегайте конструкции `#id_name {}`, используйте `.class_name {}`

Не ставьте единицы измерения для нулевых значений

Не указывайте единицы измерения после нулевых значений.

```
/* Не рекомендуется: можно написать короче */

margin: 0px;
padding: 0px;

/* Рекомендуется */

margin: 0;
padding: 0;
```

Сокращайте запись шестнадцатеричных чисел

По возможности используйте трёхсимвольную шестнадцатеричную систему для указания цветовых кодов — так лаконичнее.

```
/* Не рекомендуется: можно написать короче */

color: #eebbcc;

/* Рекомендуется */

color: #ebc;
```

Используйте в CSS одинарные кавычки

Значения атрибутов и свойств, в том числе в селекторах, пишите с одинарными кавычками (').

Исключение — правило `@charset`. Для него применяйте двойные кавычки. Одинарные недопустимы. URL указывайте без кавычек.

```
/* Не рекомендуется: кавычки в URL */

@import url("https://www.google.com/css/maia.css");

/* Не рекомендуется: двойные кавычки */

html {
    font-family: "open sans", arial, sans-serif;
}

/* Рекомендуется */

@import url(https://www.google.com/css/maia.css);

html {
    font-family: 'open sans', arial, sans-serif;
}
```

Послесловие

Обязательность приведенных выше правил

Описанные правила — не технические ограничения и не стандарты. Их нарушение не ведёт к ошибке интерпретации. Это наши внутренние правила оформления кода.

К окончанию курса вы начнёте нарушать эти правила осознанно. Усвоенные навыки позволят вам определять, где, когда и зачем следует отходить от правил.

Работа с чужим кодом

Сохраняйте стиль исходного кода.

Перед редактированием определите формат кода, с которым вам предстоит работать.

Изучите принципы, которыми руководствовался автор, и следуйте им, даже если они расходятся с вашими. Так вы сохраните логику и читаемость кода.

Ваш код не должен значительно отличаться от оригинального по форматированию. Различия усложнят понимание кода для любого, кто будет читать его после вас.

Мы написали правила форматирования кода для того, чтобы наши разработчики и студенты говорили на одном диалекте языка HTML. Стандарты других команд могут значительно отличаться от наших.