Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ**

**Кафедра автоматизации обработки информации (АОИ)**

**Э. К. Ахтямов**

# ОСНОВЫ ГИПЕРТЕКСТОВОГО ПРЕДСТАВЛЕНИЯ ИНТЕРНЕТ-КОНТЕНТА

**Методические указания по выполнению лабораторных работ для студентов ФДО, обучающихся по направлениям подготовки 38.03.05 «Бизнес-информатика» и 09.03.04 «Программная инженерия» (уровень бакалавриата)**

**Томск 2017**

Корректор: А. Н. Миронова

**Ахтямов Э. К.**

Основы гипертекстового представления интернет-контента : методические указания по выполнению лабораторных работ для студентов ФДО, обучающихся по направлениям подготовки 38.03.05 «Бизнес-информатика» и 09.03.04 «Программная инженерия» (уровень бакалавриата) / Э. К. Ахтямов. – Томск : ФДО, ТУСУР, 2017. – 25 с.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Данное методическое пособие содержит указания по выполнению лабораторных работ, варианты заданий, требования к оформлению отчета. При организации и проведении лабораторных работ решаются следующие задачи:

– получение первичных навыков работы на веб-языке;

– изучение особенностей и областей применения языков JavaScript, HTML, CSS;

– получение навыков применения языков веб-разработки.

Выбор варианта лабораторной работы осуществляется по общим правилам с использованием следующей формулы:

$$V = (N \times K) \text{ div } 100,$$

где $V$ – искомый номер варианта,

$N$ – общее количество вариантов,

div – целочисленное деление,

при $V = 0$ выбирается максимальный вариант,

$K$ – код варианта.

### *Требования к оформлению отчета*

При оформлении отчетов по лабораторным работам следует руководствоваться требованиями образовательного стандарта вуза: ОС ТУСУР 01–2013. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления. Режим доступа: https://storage.tusur.ru/files/40668/rules_tech_01-2013.pdf

# 1 ЛАБОРАТОРНАЯ РАБОТА № 1 «ВВЕДЕНИЕ В ЯЗЫК HTML. СОЗДАНИЕ ПРОСТОЙ ВЕБ-СТРАНИЦЫ»

***Цель работы:*** изучение семантических особенностей элементов языка HTML и получение навыков семантической разметки.

## *Методические указания*

Для разметки текста с использованием языка HTML первоначально необходимо создать текстовый файл с расширением *.html в любом текстовом редакторе. При этом обязательно нужно обратить внимание, чтобы создаваемый документ имел кодировку UTF-8. В качестве текстового редактора рекомендуется использовать Notepad++ или Sublime Text 3. Скачать данное ПО можно по ссылкам: https://notepad-plus-plus.org/download/v7.3.3.html (Notepad++), https://www.sublimetext.com/3 (Sublime Text 3).

В созданном текстовом документе необходимо последовательно разместить блок с директивой версии используемого языка HTML, блок HEAD с технической информацией о странице (кодировка UTF-8, заголовок страницы) и блок BODY, в котором будет описана разметка текста:

```
<!Doctype HTML>
<html>
    <head>
    <meta charset=”UTF-8”>
        <title>Заголовок страницы</title>
    </head>
    <body>
        ….
    </body>
</html>
```

При разметке текста необходимо помнить, что каждый элемент в HTML несет некоторую семантическую нагрузку (кроме декоративных элементов, использование которых на данный момент не рекомендуется). Так как в отличие от человека семантические сети, используемые в современных поисковых системах, не могут распознать, чем по смыслу является тот или иной блок (заголовок, абзац, определение, формула), то это необходимо указывать явно в виде соответствующего HTML-элемента. Ранее помимо семантической нагрузки HTML-элементы несли еще и декоративную функцию, однако с развитием CSS эта функция отошла на задний план.

Для вставки символов, отсутствующих в раскладке клавиатуры, применяются символы-мнемоники или же «Таблица символов Unicode», доступная по ссылке https://unicode-table.com/ru/.

В каждом варианте задания размещена часть текста или формула, которую необходимо разметить с помощью HTML-элементов.

### *Требования к содержанию отчета*

Отчет по лабораторной работе должен содержать:

1. Титульный лист.

2. Задание.

3. HTML-код с необходимыми комментариями и пояснениями предназначения используемых элементов (какую семантическую нагрузку несет тот или иной тег). Код должен быть оформлен в соответствии с требованиями (см. приложение А).

4. Выводы по лабораторной работе.

### *Задание*

1. Согласно варианту задания разметить представленный текст с помощью HTML-элементов. Описать в комментариях.

2. Провести тестирование кода используя ресурс https://validator.w3.org/. Исправить полученные ошибки.

### *Варианты*

**Вариант 1**

Как же эффективно привнести технологии в такой рабочий процесс? Есть один простой способ – вспомнить о Wolfram|Alpha. Если ввести 2+2, то Wolfram|Alpha, точно так же, как и Mathematica, выдаст в ответе 4. Но если ввести, скажем, «new york», «2.3363636» или «cos(x) log(x)», то простого ответа, который можно посчитать, не будет. Вместо него Wolfram|Alpha сгенерирует отчет, содержащий набор «интересных фактов» касательно введенных вами данных.

**Вариант 2**

Бесчисленное множество статей по чистой математике начинаются со слов: «Положим, что F – поле с такими-то и такими-то свойствами». Поэтому нам нужно иметь возможность вводить нечто подобное, чтобы потом система автоматически выдавала нам теоремы и факты о поле F, в сущности, самостоятельно создавая полноценную статью, посвященную полю F.

**Вариант 3**

Компьютеры и люди

Но в современном мире Wolfram Language также имеется свободная форма ввода на естественном языке. Ключевой момент здесь заключается в том, что, используя её, можно эффективно использовать все многообразие удобных (хотя и некрасивых) вариантов записи, которые понимают и используют только настоящие математики. Например, «L2» может в соответствующем контексте интерпретироваться как «Лебегово пространство

второй степени». Система распознавания естественного языка позаботится о том, чтобы разрешить неоднозначность трактовки такого запроса и найдёт каноническую символьную форму для него.

**Вариант 4**

Спецслужбы нанимают на работу много математиков, но потенциальные сотрудники должны понимать, что их работа будет заключаться в «подглядывании» за всеми, утверждает Том Лейнстер.

За последние 10 месяцев крупный международный скандал охватил некоторых крупнейших в мире работодателей. Эти организации обвиняют в нарушении закона в промышленных масштабах, что в настоящее время является предметом широкого возмущения. Каким образом отреагировало сообщество математиков? Во многом они игнорируют данный факт.

**Вариант 5**

Этапы отрисовки любой страницы:

* подготовка к запросу на сервер;
* запрос данных с сервера;
* шаблонизация;
* обновление DOM.

– «Ок, теперь у нас есть метрики, мы можем отправить их на сервер», – говорим мы.

– «Что же дальше?» – вопрошаете вы.

– «А давай построим график!» – отвечаем мы.

– «А что будем считать?» – уточняете вы.

**Вариант 6**

В случае ускорения или замедления медиана, конечно, изменится. Но она не может рассказать, сколько пользователей ускорилось, а сколько замедлилось. APDEX – метрика, которая сразу говорит: хорошо или плохо. Метрика работает очень просто. Мы выбираем временной интервал [0; t], такой, что если время показа страницы попало в него, то пользователь счастлив.

**Вариант 7**

Сейчас модуль обновления сам логирует все свои стадии, и можно легко понять причину замедления: медленнее стал отвечать сервер либо слишком долго выполняется JavaScript. Выглядит это примерно так:

```
this.timings['look-ma-im-start']      =      Date.now();
this.timings['look-ma-finish'] = Date.now();
```

**Вариант 8**

Для окончательного выбора библиотеки нам нужно сравнить:

| Библиотека | IE 9 | Opera 12 |
|---|---|---|
| vcdiff | 8 | 5 |
| google diff | 1363 | 76 |

**Вариант 9**

То есть это обычный массив из объектов. Каждый объект – отдельный ресурс. У каждого объекта есть три свойства. k – названия ключа в localStorage для этого ресурса. p – патч для ресурса, который сгенерировал vcdiff. s – чексумма для ресурса актуальной версии, чтобы потом можно было проверить правильность наложения патча на клиенте. Чексумма вычисляется по алгоритму Флетчера.

**Вариант 10**

Фактически мы экономим 80-90% трафика. Размер загружаемой статистики в байтах:

| Релиз | С патчем | Без патча |
|---|---|---|
| 7.7.20 | 397 | 174 549 |
| 7.7.21 | 383 | 53 995 |
| 7.7.22 | 483 | 3 995 |

**Вариант 11**

Автор: @doochik C++-разработчик. Электронная почта: (doochik@yandex-team.ru). Компания: Яндекс.

**Вариант 12**

Берем еще один интервал, (t; 4t] (в четыре раза больше первого), и считаем, что если страница показана за это время, то пользователь в целом удовлетворен скоростью работы, но уже не настолько счастлив. И применяем формулу: (кол-во счастливых пользователей + кол-во удовлетворенных / 2) / (общее кол-во). Получается значение от нуля до единицы, которое, видимо, лучше всего показывает, хорошо или плохо работает почта.

**Вариант 13**

Комментарии (3):

– Mogaika (mogaika@yandex-team.ru) 30 ноября 2014 в 17:05

А можете привести сравнение, насколько быстрее грузится lite-версия?

**Вариант 14**

Комментарии (3):

– alexeimois (test@yandex.ru) 22 ноября 2014 в 17:35

Мы измеряем скорость загрузки с помощью Яндекс.Метрики: help.yandex.ru/metrika/reports/monitoring_timing.xml

**Вариант 15**

И для множества других областей это только вершина айсберга. В невидимой части айсберга находится новый очень мощный механизм изменения «графических тем» – когда вместо множества отдельных опций Вам надо только определить общую тему для графика, например, ”web” (веб), ”minimal” (минимальный) или ”scientific” (научный).

**Вариант 16**

Подавляющую часть времени программист на языке Haskell имеет дело с так называемыми чистыми функциями (все, конечно, зависит от программиста, но мы здесь говорим о том, как должно быть). Вообще-то, «чистыми» эти функции называют для того, чтобы их не путали с тем, что подразумевают под термином «функция» в императивном программировании. На самом деле это самые обычные функции в математическом понимании этого термина. Вот простейший пример такой функции, складывающей три числа:

```
addThreeNumbers x y z = x + y + z
```

**Вариант 17**

Заметили общий паттерн? На псевдокоде его можно записать примерно так:

```
функция (аргументы и/или иногда что-то ещё)
{      // сделай чистые вычисления
и/или      // сделай что-то ещё
return (результат чистых вычислений и/или что-то
ещё)
}
```

**Вариант 18**

11 декабря 1998 г. для исследования Марса был запущен космический аппарат Mars Climate Orbiter. После того как аппарат достиг Марса, он был потерян. После расследования выяснилось, что в управляющей программе одни дистанции считались в дюймах, а другие – в метрах. И в одном, и в другом случае эти значения были представлены типом Double. В результате функции, считающей в дюймах, были переданы аргументы, выраженные в метрах, что закономерно привело к ошибке в расчётах.

**Вариант 19**

До ее написания я сформулировал такие требования к будущей программе:

– моя программа не должна быть программой под DOS. Слишком много примеров ориентировано на нее в связи с простым API. Моя программа обязательно должна была запускаться на современных ОС;

– программа должна использовать кучу – получать в свое распоряжение динамически распределяемую память;

– чтобы не быть слишком сложной, программа должна работать с целыми беззнаковыми числами без использования переносов.

**Вариант 20**

После того как мы определились с библиотекой для диффа, нужно определиться с тем, где и как хранить статику на клиенте. Формат файла с патчами для проекта выглядит так:

```
[
{ "k": "jane.css",    "p": [patch],    "s": 4554 },
{ "k": "jane.css",    "p": [patch],    "s": 4554 }
]
```

# 2 ЛАБОРАТОРНАЯ РАБОТА № 2 «АНИМАЦИИ»

***Цель работы:*** изучение особенностей создания анимации на веб-страницах, а также графических преобразований элементов при помощи механизмов трансформаций по спецификации CSS Transform Module Level 2.

## *Методические указания*

В лабораторной работе изучаются визуальная модель форматирования и присущие ей преобразования. Все координаты, которые описывают положение и размеры элементов в локальной системе координат, могут быть изменены с помощью свойства transform. Для преобразований используются матрицы трансформаций или соответствующие им функции преобразований: rotate, scale, skew, translate и т. д.

Чтобы получить возможность анимирования преобразований следует воспользоваться двумя методами: использовать свойство `transition` или свойство `animation`. В первом случае анимация ограничена двумя кадрами и необходимо вызвать событие, инициализирующее начало анимации. Как правило, для этого используется динамический псевдокласс `:hover`. В случае со свойством animation разработчику предоставляется больший простор действий: анимацию можно усложнить, используя покадровый сценарий в правиле `@keyframes`. Для инициации анимации свойством animation событие указывать не надо: анимация начнется сразу после загрузки страницы, если не указана задержка.

Чтобы указать несколько преобразований, необходимо перечислить их через запятую:

```
div {transition: background 0.3s ease, color 0.2s linear;}
```

Для всех преобразований применяется временная функция linear. Если не указано количество итераций, то принимается значение по умолчанию – 1. Время выполнения преобразований – 5 с.

### *Требования к содержанию отчета*

Отчет по лабораторной работе должен содержать:

1. Титульный лист.

2. Задание.

3. Листинг HTML-кода, оформленный в соответствии с приложением А.

4. Листинг CSS-кода, оформленный в соответствии с приложением Б и снабженный комментариями к использованным свойства.

5. Фрагменты результатов тестирования HTML-кода с помощью ресурса https://validator.w3.org/

6. Сравнение выполнения свойств transition и animation на основе варианта задания.

7. Выводы, сформулированные в виде обоснования использования того или иного свойства в рамках варианта задания.

### *Задание*

1. Выполнить преобразования и анимацию в соответствии с вариантом двумя способами: с помощью свойств `transition` и `animation`.

2. Определить наиболее оптимальный вариант с точки зрения кода и целесообразности использования свойства.

### *Варианты*

**Вариант 1**

Объект: круг. При наведении происходит плавное увеличение, изменение цвета и поворот элемента по оси *X*.

**Вариант 2**

Объект: квадрат. Происходит наклон по оси *Y* по направлению к зрителю, квадрат растягивается в параллелограмм. После окончания анимация повторяется еще 4 раза.

**Вариант 3**

Объект: прямоугольник. При наведении изменяется прозрачность от полностью невидимого элемента до полностью видимого, вместе с этим ширина прямоугольника увеличивается в 4 раза, длина – в 3.

**Вариант 4**

Объект: пятиугольник. Цвет постепенно меняется в последовательности: красный – синий – зеленый – желтый – фиолетовый – красный. Во второй половине анимации добавляется поворот против часовой стрелки по оси *Y* и положительный наклон по оси *Z*.

**Вариант 5**

Объект: шестиугольник. При наведении элемент приближается к зрителю и 4 раза прокручивается против часовой стрелки.

**Вариант 6**

Объект: треугольник. Элемент «пульсирует»: быстро приближается и отдаляется от зрителя несколько раз.

**Вариант 7**

Объект: круг. При наведении элемент перемещается вдоль оси *X* вправо до края экрана и при этом увеличивается.

**Вариант 8**

Объект: квадрат. Цвет постепенно меняется в последовательности: красный – синий – зеленый – коричневый – фиолетовый – красный. Во второй половине анимации добавляется поворот против часовой стрелки по оси *Y* и положительный наклон по оси *Z*.

**Вариант 9**

Объект: прямоугольник. При наведении происходит плавное увеличение, изменение цвета и поворот элемента по оси *Z*.

**Вариант 10**

Объект: пятиугольник. Элемент «пульсирует»: быстро приближается и отдаляется от зрителя несколько раз.

**Вариант 11**

Объект: шестиугольник. Происходит наклон по оси *Y* по направлению к зрителю. После окончания анимация повторяется еще 4 раза.

**Вариант 12**

Объект: треугольник. При наведении элемент приближается к зрителю и 4 раза прокручивается против часовой стрелки.

**Вариант 13**

Объект: круг. Элемент «пульсирует»: быстро приближается и отдаляется от зрителя несколько раз. Анимация повторяется 5 раз.

**Вариант 14**

Объект: квадрат. При наведении происходит плавное уменьшение, изменение цвета и поворот элемента по оси *Y*.

**Вариант 15**

Объект: прямоугольник. Цвет постепенно меняется в последовательности: красный – синий – зеленый – желтый – фиолетовый – красный. Во второй половине анимации добавляется поворот против часовой стрелки по оси *Y* и положительный наклон по оси *Z*. По окончании анимации все возвращается в исходное состояние в обратном порядке.

**Вариант 16**

Объект: пятиугольник. При наведении элемент перемещается вдоль оси *Y* вправо до края экрана и при этом уменьшается.

**Вариант 17**

Объект: шестиугольник. Происходит наклон по оси *X*, при этом меняется прозрачность. После окончания анимация повторяется еще 3 раза.

**Вариант 18**

Объект: треугольник. При наведении курсора элемент отдаляется от зрителя и 4 раза прокручивается против часовой стрелки.

**Вариант 19**

Объект: пятиугольник. На первом кадре элемент увеличивается в 4 раза, затем растягивается по оси *Y*, перемещается на 100px вниз, поворачивается на 45 градусов и перемещается на 100px вверх.

**Вариант 20**

Объект: шестиугольник. Элемент перемещается по экрану, описывая квадрат. Анимация повторяется 10 раз.

# 3 РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Лебедев А. Ководство / А. Лебедев. – М. : Изд-во Студии Артемия Лебедева, 2014. – 536 с.

2. Макконнелл С. Совершенный код / С. Макконнелл ; пер. В. Вшивцева. – М. : Русская Редакция, Microsoft Press, 2017. – 896 с.

3. Мержевич В. Вёрстка веб-страниц / В. Мержевич. – М. : htmlbook.ru, 2011.

4. Флэнаган Д. JavaScript. Подробное руководство / Д. Флэнаган ; пер. А. Киселева. – М. : Символ-Плюс, 2013. – 1080 с.

5. Фаулер М. Рефакторинг. Улучшение существующего кода / М. Фаулер, К. Бек ; пер. С. Маккавеева. – М. : Символ-Плюс, 2008. – 432 с.

6. Фаулер М. Шаблоны корпоративных приложений / М. Фаулер, Д. Райс, М. Фоммел, Э. Хайет, Р. Ми, Р. Стаффорд. – М. : Вильямс, 2016.

7. Шикин Е. Компьютерная графика. Полигональные модели / Е. Шикин, А. Боресков. – М. : Диалог-МИФИ, 2005.

8. Скотт Б. Проектирование веб-интерфейсов / Б. Скотт, Т. Нейл. – М. : Символ-Плюс, 2010. – 352 с.

9. Купер А. Об интерфейсе / А. Купер, Р. М. Рейманн, Д. Кронин, К. Носсел. – СПб. : Питер, 2017. – 720 с.

10. Нильсен Я. Веб-дизайн / Я. Нильсен. – М. : Символ-Плюс, 2006.

# ПРИЛОЖЕНИЕ А

## Требования к оформлению документа

### *Отступы и длина строки*

- Длина строки не должна превышать 110 символов.
- В качестве отступов используется 4 пробела.
- Не должно быть лишних пустых строк, если это не обусловлено лучшей читаемостью:

```
<!-- Плохо -->
    <p>Абзац текст</p>

    <p>Ещё один</p>
    <ul>
        <li>Cats</li>
        <li>Not cats</li>
    </ul>
<!-- Хорошо -->
    <p>Абзац текст</p>
    <p>Ещё один</p>
    <ul>
        <li>Cats</li>
        <li>Not cats</li>
    </ul>
```

- Не должно быть лишних пробелов:

```
<!-- Плохо -->
<ul>
    <li> Cats </li>
    <li> <strong>Not cats</strong></li>
</ul>
<abbr title = "Health Points">HP</abbr>
<!-- Хорошо -->
<ul>
    <li>Cats</li>
    <li><strong>Not cats</strong></li>
</ul>
<abbr title="Health Points">HP</abbr>
```

• Необходимо соблюдать правила вложенности согласно примерам:

```
<div>Здесь мало текста, он влезает в 110 символов, не переносим</div>
<div>
    А здесь уже более длинный текст, не влезает в 110 символов, переносим его
    на новую строку внутри элемента и делаем отступ в 4 пробела.
</div>
<div>
    <div>
    Вложенные элементы (кроме текстовых) переносим внутрь родительского
    элемента на новую строку и делаем отступ в 4 пробела.
    </div>
    Ещё немного текста
    <div>Ещё один элемент</div>
</div>
<div>
    <div>
    Текстовые <em>элементы ведут себя как текст, каким бы длинным не было
    их содержимое</em>, вот так.
    </div>
</div>
```

### *Именование тегов и атрибутов*

- Пишем теги строчными буквами:

```
<!-- Неправильно -->
<TITLE>Паравеб</tItLe>
<!-- Правильно -->
<title>Паравеб</title>
```

- Void elements (одиночные) теги не закрываем, normal elements (парные) закрываем всегда:

```
<!-- Неправильно -->
<img src="cats.jpeg" />
<meta charset="utf-8"/>
<ul>
    <li>Cats
    <li>Not cats
</ul>

<!-- Правильно -->
<img src="cats.jpeg">
<meta charset="utf-8">
<ul>
    <li>Cats</li>
    <li>Not cats</li>
</ul>
```

- Пишем атрибуты строчными буквами:

```
<!-- Плохо -->
<abbr TITLE="Hell Points">HP</abbr>

<!-- Хорошо -->
<abbr title="Health Points">HP</abbr>
```

- Одиночные атрибуты пишем без значений, а остальные со значениями в двойных кавычках:

```
<!-- Неправильно -->
<input type=button disabled>
<input type="button" disabled="">
<input type="button" disabled="disabled">
<input type='button' disabled>

<!-- Правильно -->
<input type="button" disabled>
```

- Зарезервированные значения атрибутов пишем строчными буквами:

```
<!-- Непрвильно -->
<input type="BUTTON">

<!-- Правильно -->
<input type="button">
```

***Обязательные элементы и атрибуты***

- Всегда указываем:

<!DOCTYPE html>

<html lang="ru">

<head>

<body>

<title>

<meta charset="utf-8">

```
<!-- Правильно -->
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Лабораторная работа</title>
    </head>
    <body>
    </body>
</html>
```

- У изображений всегда указываем атрибут alt="":

```
<!-- Хорошо -->
<img src="logo.jpg" alt="Логотип">
```

### *Запрещенные элементы и атрибуты*

- Устаревшие элементы: , <font>, <marquee> и др.
- Элементы визуальной разметки: <b>, <u>, <i>, <s> и др.
- Устаревшие атрибуты: border для таблиц, type для списков, align для выравнивания и др.

### *Экранирование символов*

- В тексте элементов всегда заменяем символы < и > на < и >

```
<!-- Неправильно -->
<div>Текст с <угловыми> скобками</div>

<!-- Правильно -->
<div>Текст с &lt;угловыми&gt; скобками</div>
```

# ПРИЛОЖЕНИЕ Б

## Требования к оформлению CSS

- Длина строки не должна превышать **80 символов**.
- В качестве отступов используем **4 пробела**.
- В конце файла оставляем пустую строку.
- Имена селекторов должны отражать смысл:

```
/* Неправильно */
 .a, .foo, .red, {}

 /* Правильно */
 .logo {}
```

- В именах селекторов можно использовать только a–z, –, -- и __:

```
/* Неправильно */
.username,
.user-1,
.LOGO {}

/* Правильно */
.user-name,
.first-user,
.logo {}
```

- Каждый новый селектор пишем с новой строки:

```
 /* Неправильно */
 .user-name, .first-user, .logo
 {}


 /* Правильно */
 .user-name,
 .first-user,
 .logo
 {}
```

- Закрывающую скобку пишем с новой строки:

```
/* Неправильно */
.user-name {
    font-weight: bold; }

 /* Правильно */
 .user-name {
    font-weight: bold;
}
```

- Между наборами правил размещаем одну пустую строку:

```
/* Неправильно */
.user-name {
    font-weight: bold;
}
.logo {
    float: left;
}

/* Правильно */
.user-name {
    font-weight: bold;
}

.logo {
    float: left;
}
```

- Каждое правило размещаем в новой строке:

```
/* Неправильно */
.user-name {
    font-weight: bold; color: #f00;
}

/* Правильно */
.user-name {
    font-weight: bold;
    color: #f00
}
```

- Каждое правило обязательно заканчиваем знаком `;` (даже последнее):

```
/* Неправильно */
.user-name {
    font-weight: bold;
    color: #f00
}
/* Правильно */
.user-name {
    font-weight: bold;
    color: #f00; /* <--- */
}
```

- Между правилом и значением ставим `:` и один пробел:

```
/* Неправильно */
.user-name {
    font-weight : bold;
    color:#f00;
}
/* Правильно */
.user-name {
    font-weight: bold;
    color: #f00;
}
```

- Не используем `id` в качестве селекторов в CSS:

```
/* Неправильно */
#user-name {
    color: #f00;
}

/* Правильно */
.user-name {
    color: #f00;
}
```

### *Требования к оформлению HTML*

Не используем атрибут `style`, вместо него указываем атрибут `class`, стили пишем в CSS в отдельном файле:

```
<!-- Неправильно -->
<div style="color: red;">Hello World</div>

<!-- Правильно -->
<div class="user-name">Hello World</div>
```