

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Магнитогорский государственный технический университет  
им. Г.И. Носова»  
Многопрофильный колледж



**МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ  
ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ РАБОТ**

**ПМ.08. РАЗРАБОТКА ДИЗАЙНА ВЕБ-ПРИЛОЖЕНИЙ**

**МДК. 08.01 Проектирование и разработка интерфейсов пользователя**

для студентов специальности  
специальности 09.02.07 Информационные системы и программирование

**КВАЛИФИКАЦИЯ – РАЗРАБОТЧИК ВЕБ И МУЛЬТИМЕДИЙНЫХ ПРИЛОЖЕНИЙ**

Магнитогорск, 2019

**ОДОБРЕНО:**

Предметно-цикловой комиссией  
*Информатики и вычислительной техники*  
Председатель И.Г.Зорина  
Протокол № 6 от 20.02.2019

Методической комиссией МпК

Протокол №5 от «21» февраля 2019г

**Составитель:**

преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» МпК Ирина Геннадьевна Зорина  
преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» МпК Регина Артуровна Закирова

Методические указания по выполнению практических и лабораторных работ разработаны на основе рабочей программы ПМ.08 Разработка дизайна веб-приложений, МДК. 08.01 Проектирование и разработка интерфейсов пользователя.

Содержание практических и лабораторных работ ориентировано на формирование общих и профессиональных компетенций по программе подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование.

## СОДЕРЖАНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА.....	4
2 ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ ЗАНЯТИЙ .....	6
3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ .....	7
Практическое занятие № 1 Составление технического задания на разработку web-сайта .....	7
Лабораторное занятие № 1 Применение тегов HTML при создании web-страниц.....	12
Лабораторное занятие № 2 Создание формы на html-странице .....	25
Лабораторное занятие № 3 Форматирование web-страниц с использованием каскадных таблиц стилей .....	33
Лабораторное занятие № 4 Вёрстка .....	40
Лабораторное занятие № 5 Использование языка сценариев JavaScript при создании web-сайта .....	58
Лабораторное занятие № 6 Подготовка и оптимизация графики на web-странице .....	62
Лабораторное занятие № 7 Создание баннера для web-страницы.....	68
Лабораторное занятие № 8 Разработка эскизов веб-приложения .....	71
Лабораторное занятие № 9 Разработка прототипа дизайна веб-приложения.....	77
Лабораторное занятие № 10 Разработка схемы интерфейса веб-приложения .....	95

## 1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Состав и содержание практических и лабораторных занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью практических занятий является формирование профессиональных практических умений (умений выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности).

В соответствии с рабочей программой ПМ.08 Разработка дизайна веб-приложений, МДК.08.01 Проектирование и разработка интерфейсов пользователя, предусмотрено проведение практических и лабораторных занятий. В рамках практического/лабораторного занятия обучающиеся могут выполнять одну или несколько практических/лабораторных работ.

В результате их выполнения, обучающийся должен:

**уметь:**

У.3. Создавать дизайн с применением промежуточных эскизов, требований к эргономике и технической эстетике;

У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов;

У.5. Учитывать существующие правила корпоративного стиля;

У.6. Придерживаться оригинальной концепции дизайна проекта и улучшать его визуальную привлекательность;

У.02.2. Определять необходимые источники информации;

У.02.5. Выделять наиболее значимое в перечне информации;

У.04.2. Взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности;

У.09.2. Использовать современное программное обеспечение;

У.11.2. Выявлять достоинства и недостатки коммерческой идеи;

У.11.5. Определять инвестиционную привлекательность коммерческих идей в рамках профессиональной деятельности.

Содержание практических и лабораторных занятий ориентировано на формирование общих компетенций по профессиональному модулю программы подготовки специалистов среднего звена по специальности и овладению **профессиональными компетенциями:**

ПК 8.1 Разрабатывать дизайн-концепции веб-приложений в соответствии с корпоративным стилем заказчика

А также формированию **общих компетенций:**

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей.

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языке.

ОК 11. Планировать предпринимательскую деятельность в профессиональной сфере.

Выполнение обучающимися практических и лабораторных работ по ПМ.08 Разработка дизайна веб-приложений, МДК. 08.01 Проектирование и разработка интерфейсов пользователя, направлено на:

- обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам профессионального модуля;

- формирование умений применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;

- формирование и развитие умений: наблюдать, сравнивать, сопоставлять, анализировать, делать выводы и обобщения, самостоятельно вести исследования, оформлять результаты в виде таблиц, схем, графиков;

- развитие интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;

- выработку при решении поставленных задач профессионально значимых качеств, таких как самостоятельность, ответственность, точность, творческая инициатива.

Практические и лабораторные занятия проводятся после соответствующей темы, которая обеспечивает наличие знаний, необходимых для ее выполнения.

## 2 ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ ЗАНЯТИЙ

### *МДК. 08.01 Проектирование и разработка интерфейсов пользователя*

Разделы/темы	Темы практических/лабораторных занятий	Количество часов	Требования ФГОС СПО (уметь)
<b><i>Раздел 1. Технология проектирования и разработки интерфейсов пользователя</i></b>			
<b><i>Тема 08.01.01 Основы web-технологий.</i></b>	Практическая работа №1. «Составление технического задания на разработку web-сайта»	6	У.4, У.5 У.02.2, У.02.5, У.04.2, У.09.2, У.11.2, У.11.5
	Лабораторная работа № 1. Применение тегов HTML при создании web-страниц	4	
	Лабораторная работа № 2. Создание формы на html-странице.	4	
	Лабораторная работа № 3. Форматирование web-страниц с использованием каскадных таблиц стилей.	4	
	Лабораторная работа № 4. Вёрстка	8	
	Лабораторная работа № 5. Использование языка сценариев JavaScript при создании web-сайта	8	
	Лабораторная работа № 6. Подготовка и оптимизация графики на web-странице	4	
	Лабораторная работа № 7. Создание баннера для web-страницы	4	
<b><i>Тема 08.01.02 Web-дизайн</i></b>	Лабораторная работа № 8. Разработка эскизов веб-приложения	6	У.3, У.6 У.02.2, У.02.5, У.04.2, У.09.2, У.11.2, У.11.5
	Лабораторная работа № 9. Разработка прототипа дизайна веб-приложения	6	
	Лабораторная работа № 10. Разработка схемы интерфейса веб-приложения	6	
<b>ИТОГО</b>		<b>60</b>	

## 3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

### Тема 08.01.01 Основы web-технологий

#### Практическое занятие № 1

#### Составление технического задания на разработку web-сайта

**Цель:** получение практических навыков составления технического задания на разработку сайта

#### Выполнив работу, Вы будете:

##### *уметь:*

- У.02.2. Определять необходимые источники информации
- У.04.2. Взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности
- У.09.2. Использовать современное программное обеспечение

#### Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания.

#### Задание:

Составить техническое задание на разработку web-сайта по заданному варианту.

#### Краткие теоретические сведения:

**Техническое задание** — это документ, которым регламентируются технические, качественные и прочие характеристики готового изделия. Техническое задание составляется заказчиком изделия, его часто путают с проектной документацией, используемой при изготовлении и разрабатываемой производителем. Отсюда достаточно часто у заказчиков возникает некорректное убеждение, что техническое задание должен составлять непосредственно производитель.

Относительно сайта, техническое задание должно быть сформулировано полно, но в то же время достаточно лаконично. Это означает, что в техническом задании не должно быть избыточной «мусорной» информации, но вся информация, которая имеет непосредственное отношение к заказу, должна быть явно выражена. На практике же разработка ТЗ для сайта не представляет значительных сложностей.

#### Основные разделы ТЗ для сайта

##### *1 Информация о проекте*

Независимо от того, кто в конечном итоге будет работать с вашим техническим заданием, информация о компании и основных задачах сайта будет крайне полезной.

Постарайтесь в деталях описать направления деятельности вашей организации, а также целевую аудиторию и её потребности. Не лишним будет и указание проблем текущего сайта (если он имеется), а также цели и задачи, которые должен решить новый сайт.

##### *2 Технические особенности проекта*

Несмотря на то, что многие из пунктов могут вызвать у вас определённые трудности, информация о технических особенностях будущего сайта поможет разработать проект, отвечающий всем поставленным требованиям. Вот некоторые технические аспекты, на которые стоит обратить внимание:

- **Адаптивность.** Требуется ли вашему сайту отдельный вариант отображения на мобильных устройствах?
- **Кроссбраузерность.** Какие минимальные версии браузера должны отображать сайт? Помните, что старые браузеры (вроде Internet Explorer 7) существенно

урежают возможности разработки, занимая при этом не более 1% всех используемых в мире браузеров.

- **Система управления.** Если вы уже определились с тем, какую CMS выбрать для сайта, зафиксируйте это в ТЗ.

### *3 Структура сайта*

Описание основных элементов и страниц с использованием иерархической/древовидной модели позволит быстро определить главные модули сайта и взаимосвязи между ними.

### *4 Сквозные элементы*

Сквозными принято называть те блоки и конструкции, которые появляются в той или иной форме на всех страницах вашего сайта. В большинстве случаев, все сквозные элементы можно свести к четырём основным:

- **Шапка сайта** — верхняя часть, содержащая, как правило, логотип компании, навигацию по страницам, контактную информацию и дополнительные элементы.
- **Подвал сайта** — нижняя часть, являющаяся заключительной частью каждой страницы. Зачастую, может дублировать часть информации из шапки.
- **Боковые панели (сайдбары)** — вертикальные колонки, содержащие определённый набор функциональных блоков (виджетов). Пример: боковая панель на странице интернет-магазина, содержащая фильтры и навигацию по категориям.
- **Всплывающие окна и формы**, появляющиеся на страницах сайта при клике на кнопку или ином действии.

### *5 Уникальные страницы*

Как правило, объём работы дизайнера и разработчика зависит от количества уникальных разделов/страниц, на базе которых строится сайт. Именно поэтому каждую такую страницу, имеющую уникальные дизайн и структуру, необходимо зафиксировать и описать в ТЗ для сайта.

Уникальные страницы — своеобразные макеты, на базе которых будут создаваться и множиться страницы сайта, обладающие схожими характеристиками. Каждая такая страница требует затрат со стороны дизайнера и разработчика.

Для удобства, имеется для вас несколько примеров краткого описания таких страниц. Не забывайте, что вам необходимо использовать более развёрнутые и подробные формулировки.

- **Страница новости в блоге.** Содержит сквозные элементы в виде шапки и подвала, а также дополнительные блоки: заголовок новости, краткое описание, фотографию-обложку, дату публикации, текст новости и блок комментариев.
- **Страница товаров в каталоге.** Содержит сквозные элементы в виде шапки и подвала, а также дополнительные блоки: боковую колонку с фильтрацией товаров по заданным параметрам, список товаров в конкретной категории и блок персональных предложений.

В приведённых примерах, как видите, описаны уникальные страницы, на базе которых после интеграции с системой управления будут автоматически собираться похожие страницы.

### *6 Прочие страницы*

Зачастую заказчики забывают про описание дополнительных функциональных страниц в техническом задании для сайта. Вот краткий список того, что рекомендуется включить в ТЗ практически для любого сайта:

- **Типовая текстовая страница** — на базе неё будут создаваться все новые страницы, не попадающие под описанные уникальные страницы. Рекомендуется на этапе дизайна заложить в этот пункт все необходимые элементы для оформления текста: заголовки, параграфы, списки, таблицы, изображения, встраиваемые видео и так далее.

- **Страницы ошибок** — те самые небольшие странички на сайте, которые видит посетитель, когда что-то пошло не так. Не стоит недооценивать эти страницы — если подойти к их реализации с креативом, результат может удивить посетителей вашего сайта. Отличные примеры можно посмотреть здесь, здесь и здесь.
- **Страница результатов поиска** — один из важнейших функциональных блоков на сайте. От того, насколько удобно будут представлены результаты поиска иногда напрямую зависит конверсия в продажи.
- **Страницы входа и регистрации** — если на вашем сайте предполагается авторизация пользователей, позаботьтесь о том, чтобы формы были удобными.

### **ТЗ для сайта — важные моменты**

Помимо описания структуры страниц в тех. задании рекомендуется также указать другие моменты, определяющие скорее логику работы сайта, нежели его отображения.

#### *7 Детальное описание сущностей*

Для более чёткого понимания структуры при разработке сайта принято выделять сущности — определённые виды материалов, обладающие собственными характеристиками и свойствами. Поясним на примере:

1. Вы создаёте сайт-визитку, состоящий исключительно из нескольких страниц. В этом случае, сущностью будет «Страница», у каждой из которых есть свой заголовок, содержимое и другие опции.

2. Если вы захотите добавить на свой сайт раздел с новостями, то «Новость» будет новой сущностью. Помимо заголовка и содержимого эти материалы могут иметь, например, дату публикации или автора.

3. Кстати, «Автор» также является сущностью — у каждого из них может быть уникальная фотография и имя. В этом случае, сущности могут быть связаны друг с другом, как новость и её автор.

В общем виде, сущности — это своеобразные элементы в структуре вашего сайта, на основе которых создаются похожие.

#### *8 Функциональные особенности*

Все функциональные особенности будущего сайта, которые трудно отнести к какой-то конкретной странице, следует вынести в отдельный раздел, детально описав каждую из них. Например, одной из самых популярных функций на сайте является модуль комментирования. В этом случае при составлении ТЗ для сайта необходимо будет детально описать процесс публикации комментариев и их модерации.

Мы также рекомендуем вам описать в техническом задании процесс взаимодействия сайта со сторонними сервисами. Кнопки соцсетей, интеграция с CRM, отправка уведомлений на почту — всё, что выходит за рамки стандартного функционала должно быть закреплено в документации.

### **Порядок выполнения работы:**

Техническое задание должно содержать следующие пункты.

#### ***Цель проекта***

***Сайт должен обеспечивать реализацию следующих функций:***

1. Имиджевая.
2. Коммерческая
3. Рекламная
4. Информационная

#### ***1 Общие требования к сайту***

Общая структура сайта

Стилистическое оформление

Требования к отображению сайта в браузерах

Требования к верстке (адаптивность)  
 Требования к графическому и текстовому контенту  
**2 Общие требования к функционалу сайта**  
**Общий функционал**

Наименование функции	Описание функции
Форма обратной связи "Задать вопрос" Количество: форма	
Форма заказа обратного звонка Количество: форма	
Стандартный поиск по магазину Количество: 1 форма	
Стандартный слайдер Количество: 1 слайдер	
Яндекс карта	
Яндекс метрика Google Analytics	
Кнопки социальных сетей Количество: 5 кнопок	

**Функционал каталога товаров**

Наименование функции	Описание функции
Корзина товаров	
Личный кабинет	
После оформления заказа и отправки	
Карточка товара	
Возможность написания отзывов в карточке товара	
Фильтрация товаров	
Сортировка товаров	
«Новинки», «Топ продаж», «Акции»	
Сравнение товаров	
Выбор количества товаров на странице	
Изменение изображения товара при наведении	
Варианты расцветок	
Рекомендуемые товары	
Мониторинг заказов	
База данных заказов	
База данных товаров в магазине	

**3 Структура сайта и навигация**

**3.1 Header (Шапка) сайта**

**Меню сайта**

**3.2 Главная**

**3.3 Коллекции**

**3.3.1 Категории коллекции**

**3.4 Сотрудничество**

**3.4.1 Оптовая торговля**

**3.4.2 Индивидуальный заказ**

**3.5 Marhatter**

**3.5.1 О Бренде**

**3.5.2 Новости/Акции**

**3.6 Магазины**

**3.7 Контакты**

**3.8 Личный кабинет**

**Форма представления результата:**

Оформленное техническое задание.

**Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

## Тема 08.01.01 Основы web-технологий

### Лабораторное занятие № 1 Применение тегов HTML при создании web-страниц

**Цель:** получение практических навыков применения тегов HTML при создании web-страниц

**Выполнив работу, Вы будете:**

**уметь:**

- У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов
- У.02.2. Определять необходимые источники информации
- У.09.2. Использовать современное программное обеспечение

**Материальное обеспечение:**

Методические указания для выполнения практических работ, текстовый редактор: Atom, Sublime, Visual Studio Code, PHPStorm.

**Задание 1:** Создание домашней страницы, содержащей информацию о студенте.

**Краткие теоретические сведения:**

**Суть и составные части Web технологий**

Выделим базовые элементы технологии Web:

- Internet это всемирная сеть разнородных компьютерных сетей, взаимодействующих по протоколу TCP/IP.
- Web является одним из приложений Internet (наряду с электронной почтой, новостями и прочими электронными сервисами), предназначенным для массового распространения разнообразной информации.
- Носителями информации в Web служат Web-страницы, содержащие текст, графику, мультимедиа элементы и гиперссылки на другие ресурсы Web или Internet.
- Для передачи гипертекста Web-страниц в Internet используется специально разработанный протокол HTTP (Hyper Text Transfer Protocol).
- Для разработки Web-страниц используется специальный язык разметки гипер-текста HTML (Hyper Text Markup Language).
- Для просмотра Web-страниц используется специальная клиентская программа Web-браузер. В окне Web-браузера отображаются результаты интерпретации языка HTML с Web-страниц, полученных во время навигации по гиперссылкам.

**Основы языка разметки гипертекста - HTML**

Базовым элементом языка разметки гипертекста является - ТЕГ (дескриптор, маркер). Тег всегда заключен между скобками < > и имеет следующий вид: <ТЕГ параметр1="ЗНАЧЕНИЕ" ... параметрN="`ЗНАЧЕНИЕ">

Теги бывают одиночными и контейнерными. *Контейнером* называется пара: открывающий <ТЕГ> и закрывающий </ТЕГ>.

<ТЕГ> Содержимое контейнера </ТЕГ>

Открывающий тег служит для указания программе-браузеру начала какого-либо объекта или задания свойств объектов, помещенных в контейнер. Закрывающий тег служит для указания программе-браузеру о конце объекта или окончания применения свойств, заданных в открывающем теге. Параметры(атрибуты) тега задают значения свойств данного

объекта или объектов, помещенных в контейнер. Значения свойств, содержащие пробелы, берутся в кавычки, в остальных случаях кавычки можно опустить.

HTML документ представляет собой обычный текстовый файл, содержащий маркированный тегами форматирования текст, а также заданные специальными тегами ссылки на графические и прочие файлы мультимедиа, ссылки на другие документы HTML и ресурсы Internet.

Документ HTML начинается открывающим тегом <HTML> и заканчивается закрывающим тегом </HTML>. Между данной парой контейнерных тегов располагаются две другие основные части HTML документа: заголовок заключенный в контейнер <HEAD>...</HEAD> и тело документа в контейнере <BODY>...</BODY>. Таким образом структура простого HTML документа выглядит примерно так:

### **Структура HTML-документа**

```
<!DOCTYPE HTML PUBLIC "-//IETF/DTD HTML 3.2 // EN" >
<HTML>
<HEAD> Заголовок документа </HEAD>
<BODY> Тело документа </BODY>
</HTML>
```

### **Объявление <!DOCTYPE>**

Элемент <!DOCTYPE> должен первым указываться в документе HTML (теоретически). Он сообщает серверу WEB способ обработки документа и то, какие дескрипторы могут находиться на странице, хотя чаще всего он игнорируется браузерами. Поэтому его применение строго не обязательно.

Синтаксис: <!DOCTYPE HTML "текст""URL">

Здесь текст определяет версию HTML, а URL позволяет браузерам пользователей загрузить DTD.

### **Тэг <HTML>**

Тэг <HTML> определяет границы документа HTML, ему соответствует конечный тэг </HTML>. Между этими двумя тэгами располагается собственно весь документ. Как и <!DOCTYPE> тэги <HTML> и </HTML> - не являются строго обязательными. Но, все-таки, их использование является правилами хорошего тона т.к. браузеры у пользователей могут быть всякие и не известно - насколько корректно они визуализируют такой код.

В дополнение к обязательной структуре настоятельно рекомендуется вставлять различные структурные детали.

Каждый HTML документ должен содержать основную информацию о его происхождении.

Если Вы стремитесь к тому, чтобы люди отыскивали Ваш документ по соответствующим связям, важность предоставления информации о его происхождении становится очевидной. Когда пользователь найдет Ваш документ с помощью, например, поискового ресурса AltaVista, он, вероятнее всего, захочет узнать, к какому виду относится документ. Поэтому каждый файл HTML должен предоставлять самую основную информацию (или связи к информации) о его происхождении и природе. Например, в собрании книгоподобных документов, разделенных на малые файлы, каждый файл должен содержать, по крайней мере, связь к "первой странице" "книги" (home page).

О происхождении документа должна быть представлена, по крайней мере, следующая информация:

Автор документа, имеющий уникальное имя. При этом должна быть задана связь с домашней страницей автора. Если у документа несколько авторов, определите их всех, а также роль каждого из них; например, ведущего автора, редактора, действующего спонсора, а также лиц, формально отвечающих за документ. Дата создания документа или его

последней модификации, или и та и другая. Представляемая дата должна быть понятна во всем мире; в частности, название месяца лучше писать словом, а не цифрой. Контекст документа и его статус, например: часть официальной документации компании об одном из ее продуктов, или часть личной информации о хобби автора, или что-то другое. Адрес (URL) документа. Такая информация зачастую чрезмерна, однако она может быть очень полезной, когда кому-то нужна копия именно того документа, который он нашел. Лучше не полагаться на браузер (и пользователя), добавляющих такую информацию, когда сделана копия документа.

### Элемент <BODY>

Элемент <BODY> предназначается для выделения той части документа, которая будет визуализирована для пользователя. Он имеет как начальный, так и конечный теги. Начальный тег <BODY> может иметь несколько атрибутов.

Вложенные атрибуты элемента <BODY>:

- **BACKGROUND** Атрибут задает графическое изображение, которое как черепица заполнит фон документа. Синтаксис: <BODY BACKGROUND="(URL)(путь) имя файла">

- **BGCOLOR** Этот атрибут задает цвет фона документа при помощи шестнадцатеричных значений интенсивности цветов RGB, или при помощи строчного литерала, соответствующего названию цвета. Синтаксис: <BODY BGCOLOR="#ff0000"> или <BODY BGCOLOR="RED">

- **TEXT** Этот атрибут задает используемый по умолчанию цвет текста, который не является гиперссылкой. По умолчанию такой текст будет черным. Синтаксис: <BODY TEXT="цвет">

- **LINK** Этот атрибут задает цвет гиперссылки, в большинстве браузеров он задан по умолчанию темно синим. Синтаксис: <BODY LINK="цвет">

- **ALINK** Этот атрибут задает цвет активной гиперссылки, он меняет цвет гиперссылки в момент щелчка по ней мышью, не желательно задавать ему цвет фона по понятным причинам. Синтаксис: <BODY ALINK="цвет">

- **VLINK** Этот атрибут задает цвет посещенной гиперссылки, не желательно задавать ему цвет фона и цвет атрибута LINK по понятным причинам. Синтаксис: <BODY VLINK="цвет">

### Заголовочные тэги

Элемент <HEAD> </HEAD> определяет заголовок документа.

**BASE - базовый**, основной URL Цель Задание базового URL для относительных URL в документе (например, в атрибутах HREF элемента A). Этот элемент часто используется для отображения документов. Например: <BASE href="http://foo.com/index.html">.

Элемент BASE непосредственно не отображается в документе. Основной синтаксис <BASE HREF="URL">. Так как в документе допускается только один элемент BASE, Вы не можете иметь различные базовые URL в различных частях файла HTML. При отсутствии элемента BASE в документе URL самого документа становится базовым в пределах документа.

### МЕТА - метаинформация

Используется для задания метаинформации (информации о документе), т.е. пар имя/значение, описывающих свойства документа, например, авторство, истечение даты, список ключевых слов и т.д. Типичное отображение Элементы МЕТА не влияют на отображение самого документа. Они могут давать некоторый эффект при представлении информации о документе, например, в верхнем окне браузера или в ответе на запрос от поискового средства.

Синтаксис:

<META NAME=имя элемента метаинформации CONTENT=содержимое информации> или <META HTTP-EQUIV=имя элемента метаинформации CONTENT=содержимое информации>.

Примеры <META NAME=DESCRIPTION CONTENT= "An extensive guide to writing HTML 3.2 documents, with ../examples and practical advice.">

<META NAME=KEYWORDS CONTENT= "structural HTML, logical markup">.

Тег META влияет на индексирование документа, когда он включается в базу данных поискового сервера.

### **TITLE - "внешний" заголовок (титул)**

Используется для задания обязательного "внешнего" заголовка документа.

Титул может выводиться в окне заголовка программы просмотра; в списке результатов поиска, возвращаемых поисковым сервером; в горячем списке, определяемом пользователем; списке истории и т.д.

Основной синтаксис <TITLE>последовательность символов</TITLE>

В TITLE можно использовать escape последовательности, например, &lt; (для <) и &uml; (для д), но никакие теги HTML не разрешены, поэтому Вы не можете задавать в заголовке размеры шрифтов или выделения.

Примеры <TITLE>A study of population dynamics</TITLE> Примечания Написать хорошее заглавие - очень важно, так как списки результатов поиска, возвращаемые поисковым сервером, могут использовать его.

### **Комментарии**

Файл HTML может содержать комментарии, дающие пояснения для человека, читающего HTML код. Комментарии не влияют каким-либо образом на представление документа, т.е. они игнорируются браузером. Вы можете начать комментарии с четырехсимвольной последовательности <!-- (знак "меньше чем", восклицательный знак, два дефиса) и завершить его трехсимвольной последовательностью --> (два дефиса, знак "больше чем"). Например: <!-- Написано Иваном Ивановым -->.

### **Оформление текста**

**Элемент <EM>** Элемент <EM> используется с целью выделения особым шрифтом слова или текста. Синтаксис: <EM> Текст </EM>

**Элемент <CODE>** Элемент <CODE> используется с целью дополнительного выделения фрагментов программного кода. По умолчанию он отображается телетайпным шрифтом. Данный элемент предпочтительнее, чем элемент <TT> (телетайпный шрифт). Поскольку расположение пробелов существенно для чтения программного кода, элемент <CODE> целесообразно употреблять в сочетании с элементом <PRE>. Синтаксис: <CODE> листинг кода </CODE>

**Элемент <DFN>** Элемент <DFN> используется с целью обозначения терминов и определений по типу словарей или глоссариев. Синтаксис: <DFN> Текст </DFN>

**Элемент <CITE>** Элемент <CITE> используется с целью обозначения источника информации, из которого взята цитата. Синтаксис: <CITE> Текст </CITE>

**Элемент <STRONG>** Элемент <STRONG> используется с целью выделения особым шрифтом слова или текста. Синтаксис: <STRONG> Текст </STRONG>

**Элемент <I>** Элемент <I> используется с целью выделения курсивным шрифтом слова или текста. Синтаксис: <I> Текст </I>

**Элемент <B>** Элемент <B> используется с целью выделения полужирным шрифтом слова или текста. Синтаксис: <B> Текст </B>

**Элемент <U>** Элемент <U> используется с целью выделения подчеркиванием слова или текста. Синтаксис: <U> Текст </U>

**Элемент <SUP>** Элемент <SUP> используется с целью выделения надстрочных слова или текста. Синтаксис: <SUP> Текст </SUP>

**Элемент <SUB>** Элемент <SUB> используется с целью выделения подстрочных слова или текста. Синтаксис: <SUB> Текст </SUB>

**Элемент <BIG>** Элемент <BIG> используется с целью выделения крупным шрифтом слова или текста относительно основного текста. Синтаксис: <BIG> Текст </BIG>

**Элемент <SMALL>** Элемент <SMALL> используется с целью выделения мелким шрифтом слова или текста относительно основного текста. Синтаксис: <SMALL> Текст </SMALL>

**Элемент <FONT>** Элемент <FONT> используется с целью изменения выделения шрифтом слова или текста. С ним применяются два атрибута size и color. Некоторые браузеры поддерживают атрибут face, позволяющий задать любой из перечня шрифтов, если браузер не находит заданный шрифт - то используется шрифт, заданный по умолчанию. Синтаксис: <FONT size=n color="цвет"> Текст </FONT> или <FONT face="имя" color="цвет">Текст </FONT>

**Элемент <BASEFONT>** Элемент <BASEFONT> используется как альтернатива атрибуту size элемента <FONT>, он позволяет задать базовый размер шрифта во всем документе и не имеет конечного тега. По умолчанию значение его задается равным 3, значение size может выражаться так же и относительным размером, например, размер -1 означает размер на один меньший, чем по умолчанию. Синтаксис: <BASEFONT size=n>

**Шесть уровней заголовков <Hn>** Соответствующие каждому уровню гарнитура и размер шрифта зависят от браузера, стилю <H1> назначается самый большой и самый жирный шрифт, а стилю <H6> назначается самый маленький и самый невзрачный шрифт. Элемент может иметь атрибут align, который указывает отступ left, center или Right. Синтаксис: <Hn align=отступ> Текст заголовка </Hn>

**Разделительные линии <HR>** Элемент <HR> используется для проведения горизонтальной черты в документе, он может иметь атрибуты: color, задающий цвет линии, size высота в пикселях Width ширина в пикселях или процентах от ширины экрана, align режим выравнивания, и не имеет конечного тега. Синтаксис: <HR align="center" size=n Width=n color="цвет">

**Элемент <P>** Этот элемент задает один из способов разбиения текста на абзацы. Он может иметь вложенный атрибут align, который указывает отступ left, center, right, justify. Каждый следующий абзац игнорирует, заданное для предыдущего абзаца значение align. Синтаксис: <P align=отступ> Текст абзаца </P>

**Элемент <BR>** Этот элемент задает разрыв текста с переходом на новую строку. Он может иметь вложенный атрибут clear, который может принимать значения left, all или Right тем самым указывать обтекание текста в-круг плавающих изображений вставленных в текст нестандартным способом. Каждый следующий абзац игнорирует, заданное для предыдущего абзаца значение clear. Синтаксис: <BR clear=обтекание> Текст Может быть отменен тэгами <NOBR> и </NOBR>

**Элемент <PRE>** Весь текст, заключенный в тэги <PRE> и </PRE> будет визуализирован браузером точно так, как он визуализирован в исходном коде документа, кроме того текст выводится моноширинным шрифтом, что значительно упрощает задачу форматирования текста в колонки. Элемент поддерживается не всеми браузерами, он может иметь атрибут Width, который задает ширину отводимого пространства под текст в символах. Элемент сменил собой устаревшие элементы <XMP>, <LISTING> и <PLAINTEXT> Синтаксис: <PRE Width=число символов >...текст.. </PRE>

**Элемент <DIV>** Элемент <DIV> позволяет выделить в структуре документа несколько разделов. Он является блочным элементом, функционирующим во многом подобно элементу <P>. Если закрывающий тэг </P> опущен, то <DIV> эффективно заменяет его и начинает новый абзац. Он может иметь атрибут align, который указывает отступ left, center или Right.

Каждый следующий раздел игнорирует, заданное для предыдущего раздела, значение align. Синтаксис: <DIV align=отступ> Текст раздела </DIV>

**Элемент <ADDRESS>** Элемент <ADDRESS> используется для оформления контактной информации текущего документа, будь то адрес электронной почты или полный почтовый адрес с номером телефона. Синтаксис: < ADDRESS>контактная информация </ADDRESS>

**Элемент <BLOCKQUOTE>** Элемент <BLOCKQUOTE> позволяет выделить объемный текст-цитату из общего текста. Синтаксис: <BLOCKQUOTE> Текст </BLOCKQUOTE>

## Списки

**Элемент <OL>** используется с целью задания нумерованных списков, имеет атрибуты type=1, или A, или a, или I, или i для задания вида нумерации и staRt для указания, с какого индекса начинается нумерация списка.

Элемент <OL> включает в себя дополнительный элемент <LI>, который задает элементы списка. Синтаксис: <OL type=1 start=1 > <LI> элемент списка <LI> элемент списка </OL>

Пример:

1. элемент списка
2. элемент списка

**Элемент <UL>**, по сути, является аналогом <OL> без дополнительных элементов <LI>, он используется с целью задания нумерованных списков, имеет атрибут type=circle, square, или disc для задания вида маркера. Элемент <UL> включает в себя дополнительный элемент <LI>, который задает элементы списка. Синтаксис: <UL type=circle > <LI> элемент списка <LI> элемент списка </UL>

Пример:

- о элемент списка
- о элемент списка

**Элемент <DL>** используется с целью задания словарей, глоссариев и прочих перечней. Элемент <DL> включает в себя дополнительные элементы <DT> и <DD>, которые обозначают соответственно термин и определение. Синтаксис: <DL > <DT> термин 1 <DD>определение 1 <DT> термин 2 <DD>определение 2 </DL>

Пример:

термин 1 определение 1

## Порядок выполнения работы:

Создание домашней страницы, содержащей информацию о студенте.

Обязательное содержание:

- Информация о том, чей сайт и фото студента.
- Электронный почтовый адрес.
- Ссылка на рабочую страницу или сайт вуза студента.
- Мой ВУЗ - о МГТУ и МпК.
- Моя группа.
- Моя будущая профессия.
- Мои увлечения или хобби.
- Любая другая информация.

В оформлении страницы следует использовать максимальное количество вышеперечисленных тегов HTML.

Люди обмениваются визитными карточками. В виртуальном мире визитная карточка - ваша домашняя страничка. Как и визитки, личные сайты встречаются простые в оформлении, довольно вычурные.

Главное, чтобы оформление и содержание соответствовало цели, для которой ваш Home page появился на сетевых просторах.

Фирмы заводят себе сайты в Интернете, чтобы найти новых клиентов и рассказать о своем товаре. Через Интернет люди могут рекламировать и себя, то есть рассказывать потенциальному работодателю, какой вы замечательный.

Например, Артемий Лебедев, вебмастер, работы которого во многом определяют внешний вид всего русскоязычного Интернета (Рунета), поместил на свою домашнюю страничку (<http://www.tema.ru/>) длинный список сайтов, которые он оформлял, и прикольные графические работы.

"Некто" напишет, какой он замечательный физик, бухгалтер, переводчик или врач. Когда человек хочет найти новых друзей, единомышленников, он рассказывает о своем увлечении. При этом он не забудет поместить ссылку на свой e-mail или гостевую книгу, чтобы посетители могли оставить сообщение, комментарии. О чем бы ни была домашняя страница нельзя указывать ваш адрес, телефон, подробности биографии.

## **Задание 2: Создание таблиц в HTML.**

### **Краткие теоретические сведения:**

#### **Средства описания таблиц в HTML**

По мере развития WWW стало ясно, что средств, которые заложены в HTML, недостаточно для качественного отображения различного типа документов. Недостатком HTML было отсутствие в его составе средств отображения таблиц. Для этой цели обычно использовался предформатированный текст (тег <PRE>), в котором таблица обрисовывалась символами ASCII. Но такая форма представления таблиц была недостаточно высокого качества и выбивалась из общего стиля документа. После введения таблиц в HTML у Web-мастеров появился не просто инструмент для размещения текстовых и числовых данных, а мощное средство дизайна для размещения в нужном месте экрана графических образов и текста.

#### **Создание таблиц в HTML**

Для описания таблиц используется тег <TABLE>. Тег <TABLE>, как и многие другие, автоматически переводит строку до и после таблицы.

*Создание строки таблицы* — тег <TR> Тег <TR> (Table Row, строка таблицы) создает строку таблицы. Весь текст, другие теги и атрибуты, которые требуется поместить в одну строку, должны размещаться между тегами <TR></TR>.

*Определение ячеек таблицы* - тег <TD> Внутри строки таблицы обычно размещаются ячейки с данными. Каждая ячейка, содержащая текст или изображение, должна быть окружена тегами <TD></TD>. Число тегов <TD></TD> в строке определяет число ячеек (открыть)

```
<TABLE>
  <TR>
    <TD COLSPAN=3>Если в таблице два тега TR, то в ней две
строки.</TD>
  </TR>
  <TR>
    <TD>Если в строке три тега TD,</TD>
    <TD>то в ней</TD>
    <TD>три столбца.</TD>
  </TR>
</TABLE>
```

*Заголовки столбцов таблицы* — тег <TH>

Заголовки для столбцов и строк таблицы задаются с помощью тега заголовка <TH></TH> (Table Header, заголовок таблицы). Эти теги подобны <TD></TD>. Отличие состоит в том, что текст, заключенный между тегами <TH></TH>, автоматически записывается жирным шрифтом и по умолчанию располагается посередине ячейки. Центрирование можно отменить и выровнять текст по левому или правому краю. Если воспользоваться <TD></TD> с тегом <B> и атрибутом <ALIGN=center>, текст тоже будет выглядеть как заголовок. Однако следует иметь в виду, что не все браузеры поддерживают в таблицах жирный шрифт, поэтому лучше задавать заголовки таблиц с помощью <TH>.

```

<TABLE >
  <TR>
    <TH>Заголовок центрирован по умолчанию</TH>
    <TH COLSPAN=2>Заголовок может объединять столбцы</TH>
  </TR>
  <TR>
    <TH>Заголовок может быть расположен перед столбцами</TH>
    <TD>Текст или данные</TD>
    <TD>Текст или данные</TD>
  </TR>
  <TR>
    <TH ROWSPAN=3>Заголовок может объединять строки</TH>
    <TD>Текст или данные</TD>
    <TD>Текст или данные</TD>
  </TR>
  <TR>
    <TD>Текст или данные</TD>
    <TD>Текст или данные</TD>
  </TR>
  <TR>
    <TD>Текст или данные</TD>
    <TD>Текст или данные</TD>
  </TR>
</TABLE>

```

*Использование заголовков таблицы — тег <CAPTION>*

Тег <CAPTION> позволяет создавать заголовки таблицы. По умолчанию заголовки центрируются и размещаются либо над (<CAPTION ALIGN=top>), либо под таблицей (<CAPTION ALIGN=bottom>). Заголовок может состоять из любого текста и изображений. Текст будет разбит на строки, соответствующие ширине таблицы. Иногда тег <CAPTION> используется для подписи под рисунком. Для этого достаточно описать таблицу без границ.

```

<TABLE>
<CAPTION ALIGN=top>Заголовок над таблицей</CAPTION>
  <TR>
    <TD>Текст или данные</TD>
    <TD>Текст или данные</TD>
    <TD>Текст или данные</TD>
    <TD>Текст или данные</TD>
  </TR>
</TABLE>
<TABLE>
<CAPTION ALIGN=bottom>Заголовок под таблицей </CAPTION>
  <TR>
    <TD>Текст или данные</TD>
    <TD>Текст или данные</TD>

```

```
<TD>Текст или данные</TD>
</TR>
</TABLE>
```

### Атрибут NOWRAP

Обычно любой текст, не помещающийся в одну строку ячейки таблицы, переходит на следующую строку. Однако при использовании атрибута NOWRAP с тегами <TH> или <TD> длина ячейки расширяется настолько, чтобы заключенный в ней текст поместился в одну строку.

### Атрибут COLSPAN

Теги <TD> и <TH> модифицируются с помощью атрибута COLSPAN (Column Span, соединение столбцов). Если вы хотите сделать какую-нибудь ячейку шире, чем верхняя или нижняя, можно воспользоваться атрибутом COLSPAN, чтобы растянуть ее над любым количеством обычных ячеек.

```
<TABLE BORDER="3">
  <TR>
    <TD>Если вы хотите сделать какую-нибудь
      ячейку шире, чем верхняя или нижняя,
    </TD>
    <TD>можно воспользоваться атрибутом
      COLSPAN=, </TD>
  </TR>
  <TR>
    <TD bgcolor=white COLSPAN="2" > чтобы
      растянуть ее над любым количеством
      обычных ячеек.</TD>
  </TR>
</TABLE>
```

### Атрибут ROWSPAN

Атрибут ROWSPAN, используемый в тегах <TD> и <TH>, подобен атрибуту COLSPAN=, только он задает число строк, на которые растягивается ячейка. Если вы указали в атрибуте ROWSPAN=s число, большее единицы, то соответствующее количество строк должно находиться под растягиваемой ячейкой. Внизу таблицы ее поместить нельзя.

### Атрибут WIDTH

Атрибут WIDTH применяется в двух случаях. Можно поместить его в тег <TABLE>, чтобы дать ширину всей таблицы, а можно использовать в тегах <TR> или <TH>, чтобы задать ширину ячейки или группы ячеек. Ширину можно указывать в пикселах или в процентах. Например, если вы задали в теге <TABLE> WIDTH=250, вы получите таблицу шириной 250 пикселей независимо от размера страницы на мониторе. При задании WIDTH=50% в теге <TABLE> таблица будет занимать половину ширины страницы при любом размере изображения на экране. Так что, указывая ширину таблицы в пикселах, имейте в виду, что если у пользователя узкая область просмотра, ваша страница может выглядеть несколько странно. Если вы пользуетесь пикселями, и таблица оказывается шире области просмотра, внизу появится полоса прокрутки для перемещения вправо и влево по странице. В зависимости от поставленных задач и тот, и другой способ задания ширины таблицы может оказаться полезным.

### Атрибут WIDTH

Атрибут WIDTH применяется в двух случаях. Можно поместить его в тег <TABLE>, чтобы дать ширину всей таблицы, а можно использовать в тегах <TR> или <TH>, чтобы задать ширину ячейки или группы ячеек. Ширину можно указывать в пикселах или в процентах. Например, если вы задали в теге <TABLE> WIDTH=250, вы получите таблицу шириной 250 пикселей независимо от размера страницы на мониторе. При задании WIDTH=50% в теге <TABLE> таблица будет занимать половину ширины страницы при любом размере изображения на экране. Так что, указывая ширину таблицы в пикселах, имейте в виду, что если у пользователя узкая область просмотра, ваша страница может выглядеть несколько странно. Если вы пользуетесь пикселями, и таблица оказывается шире области просмотра, внизу появится полоса прокрутки для перемещения вправо и влево по странице. В зависимости от поставленных задач и тот, и другой способ задания ширины таблицы может оказаться полезным.

### **Атрибут CELLPADDING**

Данный атрибут определяет ширину пустого пространства между содержимым ячейки и ее границами, то есть задает поля внутри ячейки.

### **Атрибут CELLSPACING**

Атрибут CELLSPACING определяет ширину промежутков между ячейками в пикселах. Если этот атрибут не указан, по умолчанию задается величина, равная двум пикселям. С помощью атрибута CELLSPACING= можно размещать текст и графику там, где вам нужно. Если вы хотите оставить пустое место, можно вписать в ячейку пробел.

### **Атрибуты ALIGN и VALIGN**

Теги <TR>, <TD> и <TH> можно модифицировать с помощью атрибутов ALIGN и VALIGN.

- Атрибут ALIGN определяет выравнивание текста и графики по горизонтали, то есть по левому или правому краю, либо по центру. Горизонтальное выравнивание может быть задано несколькими способами:
  - ALIGN=bleedleft прижимает содержимое ячейки вплотную к левому краю.
  - ALIGN=left выравнивает содержимое ячейки по левому краю с учетом отступа, заданного атрибутом CELLPADDING.
  - ALIGN=center располагает содержимое ячейки по центру.
  - ALIGN=right выравнивает содержимое ячейки по правому краю с учетом отступа, заданного атрибутом CELLPADDING.
- Атрибут VALIGN осуществляет выравнивание текста и графики внутри ячейки по вертикали. Вертикальное выравнивание может быть задано несколькими способами:
  - VALIGN=top выравнивает содержимое ячейки по ее верхней границе.
  - VALIGN=middle центрирует содержимое ячейки по вертикали.
  - VALIGN=bottom выравнивает содержимое ячейки по ее нижней границе.

### **Атрибут BORDER**

В теге <TABLE> часто определяют, как будут выглядеть рамки, то есть линии, окружающие ячейки таблицы и саму таблицу. Если вы не зададите рамку, то получите

таблицу без линий, но пространство под них будет отведено. Того же результата можно добиться, задав `<TABLE BORDER=0>`. Иногда хочется сделать границу потолще, чтобы она лучше выделялась. Можно для привлечения внимания к рисунку или тексту задать исключительно жирные границы. При создании вложенных таблиц приходится делать для разных таблиц границы различной толщины, чтобы их легче было различать.

### Атрибут **BGCOLOR**

Данный атрибут позволяет установить цвет фона. В зависимости от того, с каким тегом (TABLE, TR, TD) он применяется, цвет фона может быть установлен для всей таблицы, для строки или для отдельной ячейки. Значением данного атрибута является RGB-код или стандартное название цвета.

### Атрибут **BACKGROUND**

Данный атрибут задает фоновое изображение для таблиц. Применим к тегам TABLE и TD. Его значением является URL файла с фоновым изображением.

### Атрибут **BORDER**

В теге `<TABLE>` часто определяют, как будут выглядеть рамки, то есть линии, окружающие ячейки таблицы и саму таблицу. Если вы не зададите рамку, то получите таблицу без линий, но пространство под них будет отведено. Того же результата можно добиться, задав `<TABLE BORDER=0>`. Иногда хочется сделать границу потолще, чтобы она лучше выделялась. Можно для привлечения внимания к рисунку или тексту задать исключительно жирные границы. При создании вложенных таблиц приходится делать для разных таблиц границы различной толщины, чтобы их легче было различать.

### Порядок выполнения работы:

1. Постройте таблицу следующего вида:


Таблица №1.

2. Постройте таблицу следующего вида:


Таблица №2.

3. Постройте таблицу следующего вида:


Таблица №3.

4. Постройте таблицу следующего вида:


Таблица №4.

5. Использование стандартного параметра `BORDER` приводит к неоднозначности отображения границы таблицы. Точнее каждый браузер интерпретирует вид границы по своему. Для устранения подобного разночтения создают таблицу с простой однопиксельной рамкой. Последовательность действий такова:

1. Создать таблицу "подложку", состоящую из одной строки и одного столбца. Залить ее требуемым цветом.
2. Далее в эту таблицу помещаем вложенную, заливая каждую *ячейку* нужным цветом.
3. Устанавливаем свойство `CELLSPACING` вложенной таблицы равным 1, тогда нижняя таблица будет просвечивать и образует рамку толщиной 1 пиксель.

Создайте таблицу 6x6 с простой однопиксельной рамкой.

6. Создайте для таблицы № 4 простую однопиксельную рамку.

**Форма представления результата:**

Отчет по выполненной лабораторной работе.

**Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

## Тема 08.01.01 Основы web-технологий

### Лабораторное занятие № 2 Создание формы на html-странице

**Цель:** получение практических навыков работы с формами на html-странице

**Выполнив работу, Вы будете:**

**уметь:**

- У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов
- У.02.2. Определять необходимые источники информации
- У.09.2. Использовать современное программное обеспечение

**Материальное обеспечение:**

Методические указания для выполнения практических работ, текстовый редактор: Atom, Sublime, Visual Studio Code, PHPStorm.

**Задание:**

Создайте формы на html-страницах.

**Краткие теоретические сведения:**

С помощью средств HTML можно создавать формы для ввода информации посетителем Web-страницы.

Описание формы помещается между тэгами <FORM>и </FORM>. На странице одновременно может располагаться несколько форм, однако они не могут быть вложены одна в другую.

**Атрибуты**

- **METHOD.** Задаёт метод – GET или POST. По умолчанию предполагается GET. Хотя большинство из существующих форм использует метод POST. Различия между GET и POST, а также ситуации, в которых следует использовать тот или иной метод.
- **ACTION.** Задаёт имя программы, которая будет обрабатывать форму.
- **ENCTYPE.** Задаёт метод кодирования (обычно не задаётся). По умолчанию равен "application/x-www-form-urlencoded".
- **NAME.** Задаётся для JavaScript, чтобы иметь возможность обращаться к форме по имени, а не по номеру.
- **TARGET.** Задаётся во фреймосодержащих документах. Определяет, в какой фрейм отправить полученную информацию.

Например:

```
<form name="MyForm" action="request.php" method="pos">  
...  
</form>
```

**Элементы форм**

**<INPUT>**

Чаще всего в формах используется тэг <INPUT>. Он не имеет закрывающего тэга. Вся информация, необходимая браузеру для обработки, содержится непосредственно в тэге <INPUT> и задаётся с помощью различных атрибутов.

Атрибуты

- **TYPE**. Задаёт тип поля ввода. Может принимать значения: кнопка (BUTTON, SUBMIT, RESET), поле ввода (TEXT), поле ввода пароля (PASSWORD), скрытое поле (HIDDEN), флажок (CHECKBOX), переключатель (RADIO), файл (FILE).
- **LABEL**. Метка.
- **NAME**. Задаёт имя поля. Каждое создаваемое поле ввода должно иметь собственное уникальное имя, иначе сценарий не определит, к каким полям относятся полученные значения. Конечно, имя поля ввода должно соответствовать имени, которое описано для него в программе обработки.
- **VALUE**. Задаёт значение поля по умолчанию или надпись на кнопке.
- **ONCLICK**. Задаёт обработчик щелчка на кнопке.
- **SIZE**. Задаёт размер поля типа TEXT.
- **MAXLENGTH**. Задаёт ограничение в поле типа TEXT вводимого количества символов.

### Поле ввода - TEXT

Текстовое поле ввода используется в формах наиболее часто. Более того, его можно по праву считать основным и главнейшим элементом форм. Этот тип используется тэгом <INPUT> по умолчанию, его не нужно каждый раз указывать, чтобы вывести текстовое поле. Таким образом, на каждом текстовом поле вы экономите 12 нажатий на клавиши, а главное, размер HTML-кода будет меньше. Имя поля, задаваемое атрибутом NAME, обязательно, так как базируясь именно на этом параметре, браузер передаёт сценарию пару имя=значение. Например:

Вид в браузере	Код HTML
Имя: <input type="text"/>	<pre>&lt;form name="MyForm" action="request.php" method="post"&gt; &lt;label&gt;Имя:&lt;/label&gt; &lt;input type="text" name="name" /&gt; &lt;/form&gt;</pre>

### Поле ввода пароля - PASSWORD

Поле ввода пароля очень похоже на простое текстовое поле. Отличается оно тем, что вместо вводимых символов в нём отображаются звездочки. Такая возможность очень важна, когда нужно выспросить у пользователя секретную информацию типа пароля, которую не должны видеть другие.

Например:

Вид в браузере	Код HTML
Пароль: <input type="password"/>	<pre>&lt;form name="MyForm" action="request.php" method="post"&gt; &lt;label&gt;Пароль:&lt;/label&gt; &lt;input type="password" name="password" /&gt; &lt;/form&gt;</pre>

### Переключатель - RADIO

Переключатель напоминает флажок, поскольку он тоже может находиться во включённом или выключённом состоянии. По смыслу всегда предполагается, что в форме имеется несколько переключателей с одинаковым атрибутом NAME. У каждого из них своё значение атрибута VALUE. Группа переключателей с одним и тем же именем в форме ведёт себя таким образом, что только один из них может быть включённым. При передаче данных

передается значение только выбранного переключателя. Один переключатель из группы может быть изначально выбран по умолчанию с помощью атрибута CHECKED.

Например:

Вид в браузере	Код HTML
Пол: <input checked="" type="radio"/> Муж. <input type="radio"/> Жен.	<pre>&lt;form name="MyForm" action="request.php" method="post"&gt; &lt;label&gt;Пол:&lt;/label&gt; &lt;input type="radio" name="pol" value="0" /&gt; Муж. &lt;input type="radio" name="pol" value="1" /&gt; Жен. &lt;/form&gt;</pre>

### Флажок - CHECKBOX

Браузер отображает поле этого типа в виде небольшого квадрата. По смыслу флажок служит для того, чтобы быть установленным (квадрат перечеркнут), либо нет (квадрат пуст). Когда он установлен, его значение, заданное атрибутом VALUE, передается программе сценария. Если он не установлен, то его значение не передается совсем. Флажок может быть сразу установлен по умолчанию, если указан атрибут CHECKED. По умолчанию атрибут VALUE имеет значение ON (установлен). Так как атрибут VALUE здесь задает не надпись на флажке, а его внутреннее значение, передаваемое программе сценария, то если надо что-то подписать, пишете рядом с флажком.

Например:

Вид в браузере	Код HTML
Есть у Вас дома ПК: <input checked="" type="checkbox"/>	<pre>&lt;form name="MyForm" action="request.php" method="post"&gt; &lt;label&gt;Есть у Вас дома ПК:&lt;/label&gt; &lt;input type="checkbox" name="pc" value="0" /&gt; &lt;/form&gt;</pre>

### Файл - FILE

Позволяет передать сценарию любой файл. Максимальный размер файла задается скрытым полем MAX\_FILE\_SIZE.

Например:

Вид в браузере	Код HTML
Загрузить файл: <input type="text" value="Выберите файл"/> Файл не выбран	<pre>&lt;form name="MyForm" action="request.php" method="post"&gt; &lt;label&gt;Загрузить файл:&lt;/label&gt; &lt;input type="file" name="file" value="0" /&gt; &lt;/form&gt;</pre>

### Список - <SELECT>

Предназначен для создания меню, из которого пользователь может выбрать один или несколько из предложенных вариантов.

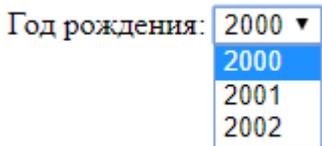
#### Атрибуты

- **NAME**. Задаёт имя поля. Обязательный атрибут.
- **SIZE**. Задаёт вертикальный размер окна для вариантов выбора. Если атрибут опущен или его значение равно 1, выводится всплывающий список вариантов. Если указано число больше 1, то варианты выводятся в окне с полосой прокрутки. Если значение атрибута больше, чем фактическое количество элементов списка, добавляются пустые строки. При их выборе пользователем возвращаются пустые поля.
- **MULTIPLE**. Этот атрибут позволяет выбрать несколько вариантов одновременно. Следует обратить внимание, что при множественном выборе в потоке данных от одной формы может присутствовать несколько переменных с одним и тем же именем. Ваша программа обработки должна предусматривать подобные ситуации и корректно их обрабатывать.

Список вариантов включается в контейнер `<SELECT>...</SELECT>` при помощи тэгов `<OPTION>`. Тэг имеет два необязательных атрибута:

- **VALUE**. Задаёт значение, передаваемое сценарию в случае выбора варианта пользователем. Если этот атрибут отсутствует, то в сценарий будет отослан текст, расположенный сразу после тэга `<OPTION>`.
- **DISABLED**. Блокирует вариант для его выбора.
- **SELECTED**. Указывает вариант, выделенный по умолчанию.

Например:

Вид в браузере	Код HTML
	<pre>&lt;form name="MyForm" action="request.php" method="post"&gt; &lt;label&gt;Год рождения:&lt;/label&gt; &lt;select name="year"&gt; &lt;option value="2000"&gt;2000&lt;/option&gt; &lt;option value="2001"&gt;2001&lt;/option&gt; &lt;option value="2002"&gt;2002&lt;/option&gt; &lt;/select&gt; &lt;/form&gt;</pre>

#### Текстовая область - `<TEXTAREA>`

Это область для ввода многострочной текстовой информации. В контейнере `<TEXTAREA>...</TEXTAREA>` допускается размещать любой текст, который будет выведен в поле ввода по умолчанию.

#### Атрибуты

- **NAME**. Задаёт название информации. Обязательный атрибут.
- **ROWS**. Задаёт высоту поля, т.е. число строк в нём.
- **COLS**. Задаёт ширину поля, т.е. длину строки.
- **disabled**. Запрещает доступ пользователя к элементу формы. Поле не может быть выбрано. Значение не отправляется на сервер.
- **readonly**. Объявляет поле только для чтения. Поле доступно для выбора, но не может быть изменено.

Хотя атрибуты **ROWS** и **COLS** не являются обязательными, они не имеют определенных значений по умолчанию (для каждого браузера эти значения различны), поэтому лучше их всегда указывать.

Например:

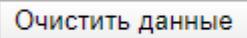
Вид в браузере	Код HTML
	<pre>&lt;form name="MyForm" action="request.php" method="post"&gt; &lt;label&gt;Введите текст:&lt;/label&gt; &lt;br/&gt; &lt;textarea rows="3" cols="25"&gt; &lt;/textarea&gt; &lt;/form&gt;</pre>

## Кнопки

### Кнопка -RESET

Это кнопка очистки формы. При ее нажатии всем измененным элементам возвращается значение по умолчанию. Применяется она достаточно редко. Однако в некоторых случаях может быть весьма полезно. Совет: осторожно относитесь к выбору надписи на кнопке RESET. Вполне наглядным (и, главное, интуитивно понятным даже чайнику из чайников) будет что-нибудь вроде “Очистить”, “Начать сначала”, “Удалить ввод” и т.п. В общем, надо, чтобы у пользователя не закралось и тени сомнения относительно предназначения этой клавиши.

Например:

Вид в браузере	Код HTML
	<pre>&lt;form name="MyForm" action="request.php" method="post"&gt; &lt;input type="reset" name="clear" value="Очистить данные"/&gt; &lt;/form&gt;</pre>

### Кнопка - SUBMIT

Эта кнопка предназначена для передачи формы. В большинстве браузеров внешне почти неотличима от кнопки BUTTON. Опять же сама она не передается, а служит только для управления. Атрибут ONCLICK практически не используется, так как лучше использовать обработчик событий ONSUBMIT, заданный в тэге <FORM>. Ведь чтобы передать введенные в форму данные, в общем случае совсем не обязательно нажимать на кнопку SUBMIT. Можно просто нажать на клавиатуре клавишу ENTER, находясь в текстовом поле ввода. При этом произойдет передача данных.

Например:

```
<form name="MyForm" action="request.php" method="post">
  <input type="submit" name="send" value="Отправить"/>
</form>
```

### Кнопка - BUTTON

В форме изображается кнопка с надписью, заданной атрибутом VALUE, при нажатии на которую вызывается JavaScript-обработчик, заданный атрибутом ONCLICK. Атрибут NAME служит для JavaScript-именования кнопки, а не для передачи на сервер. Атрибут TYPE, который может принимать значения submit, reset и button. Первые два значения и так ясно для его, а вот третье предназначено для тех случаев, когда надо исполнять какой-нибудь скрипт. То есть на кнопку вешается событие OnClick и вызывается нужная функция.

Например:

```
<form name="MyForm" action="request.php" method="post">
  <input type="button" name="but" value="Кнопка"/>
</form>
```

## Кнопка - IMAGE

Вместо кнопки вставляется изображение.

Например:

```
<form name="MyForm" action="request.php" method="post">
  <input type="image" name="button_img" src="images/butt1.png"/>
</form>
```

*Примечание:*

Атрибут `for` тега `label` — задает уникальный идентификатор, определяемый с помощью атрибута `id` элемента `<input>`, с которым следует установить связь.

Например:

```
<form name="MyForm" action="#" method="post">
  <label for="name">Имя:</label>
  <input id="name" type="text" name="name" />
</form>
```

## Порядок выполнения работы:

**Задание 1:** Создайте страницу по образцу (оформление анкеты через таблицы).

**Анкета**

Имя:	<input type="text"/>
Фамилия:	<input type="text"/>
Пароль:	<input type="text"/>
Ваш пол:	<input type="checkbox"/> муж. <input type="checkbox"/> жен.
Любимые предметы:	
<input type="checkbox"/> русский язык	<input type="checkbox"/> литература
<input type="checkbox"/> математика	<input type="checkbox"/> физика
<input type="checkbox"/> физика	<input type="checkbox"/> история
<input type="checkbox"/> информатика	<input type="checkbox"/> английский язык
<input type="checkbox"/> язык	<input type="checkbox"/> физкультура
Сколько времени тратите на выполнение домашнего задания:	
<input type="text" value="меньше 20 минут"/>	
Предложение:	<input type="text"/>
<input type="button" value="Очистить анкету"/> <input type="button" value="Отправить анкету"/>	

**Задание 2:** Создайте формы следующего вида.

# ЗАО Птицефабрика №1

Фамилия

Инициалы

Пароль

Адрес получателя

Количество

Тип

Доставка  Почтой  Электронно  Курьером

Требуется накладная

Дополнительная информация

### Проверка формы

Логин:\*

Пароль:\*

Подтвердите пароль:\*

Email адрес:\*

Показывать Email

Телефон:

О себе:\*

Получать рассылку:

**Задание 3:** Создайте документ с несколькими формами.

Данные компании	
Название компании:	<input type="text" value="Работа в России"/>
Наименование юридического лица:	<input type="text" value="ООО 'Работа в России'"/>
Юридический адрес:	<input type="text"/>
Почтовый адрес:	<input type="text"/>
Телефон 1:	<input type="text"/> <a href="#">+</a> <a href="#">Добавить еще телефон</a>
Телефон 2:	<input type="text"/> <a href="#">✗</a> <a href="#">Удалить</a>
Факс:	<input type="text"/>
E-mail:	<input type="text"/> <a href="#">+</a> <a href="#">Добавить еще e-mail</a>
Сайт	http:\\ <input type="text"/> <a href="#">+</a> <a href="#">Добавить еще сайт</a>
Платежная информация	
Название банка:	<input type="text"/>
Город банка:	<input type="text"/>
Расчетный счет:	<input type="text"/>
Корреспондентский счет:	<input type="text"/>
БИК:	<input type="text"/>
ИНН:	<input type="text"/>
КПП:	<input type="text"/>
Руководство компании	
ФИО генерального директора:	<input type="text"/>
ФИО лица, подписывающего документы:	<input type="text"/>
На основании чего действует организация:	<input type="text"/> (устав, закон, постановление и т.д.)
ФИО главного бухгалтера:	<input type="text"/>

**Форма представления результата:**

Отчет по выполненной лабораторной работе.

**Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

## Тема 08.01.01 Основы web-технологий

### Лабораторное занятие № 3

#### Форматирование web-страниц с использованием каскадных таблиц стилей

**Цель:** получение практических навыков форматирования web-страниц с использованием каскадных таблиц стилей

#### Выполнив работу, Вы будете:

##### *уметь:*

У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов

У.5. Учитывать существующие правила корпоративного стиля

У.09.2. Использовать современное программное обеспечение

#### Материальное обеспечение:

Методические указания для выполнения практических работ, текстовый редактор Atom, Sublime, Visual Studio Code, PHPStorm.

#### Задание:

Создайте стилевой файл содержащий оформление web-сайта по варианту, предложенному преподавателем.

#### Краткие теоретические сведения:

##### Технология CSS

*Каскадные таблицы стилей* или CSS (от английского Cascading Style Sheets) являются следствием дальнейшего развития HTML и дают нам возможность перейти на следующий уровень представления информации. Таблицы стилей позволяют разделить смысловое содержимое странички и его оформление.

В первых версиях стандарта HTML не было предусмотрено никаких средств для управления внешним видом информации. Общая концепция гипертекста была направлена на доступность информации для любых устройств, способных воспроизводить текст. Для разметки рекомендовалось использовать только логические теги, определяющие заголовки, подзаголовки, списки, абзацы, цитаты и т.д. - то есть, те элементы, которые и составляют структуру документа. Интерпретация же внешнего вида оставалась полностью на совести оконечного терминала.

Однако с тех пор много что изменилось, и стандарт HTML потерял первоначальную стройность. Вначале Netscape добавил "улучшенные теги", которые позволили более широко управлять внешним видом представляемой информации. Нововведение прижилось, и все расширения Netscape стали стандартом de facto. Потом точно также поступила Microsoft. Когда спохватились, то HTML представлял собой ужасную смесь логических и оформительских тегов, несовместимых расширений и полностью перестал отвечать первоначальной концепции - представлять информацию на любом устройстве независимо от его характеристик по выводу информации.

Тогда была предпринята широкомасштабная стандартизация. В результате чего на свет явился стандарт HTML 3.2. Он не был революционным, а лишь расставил по местам все нововведения и выработал общие рекомендации для производителей браузеров. Революционные изменения были введены в новом стандарте - HTML 5.0 или, как его стали называть, Dynamic HTML.

Каким же образом была решена проблема с представлением внешнего вида информации? В этом и заключается революционность подхода. Все оформление рекомендуется вынести во внешний стилевой файл. Основная же страничка будет содержать только информацию и ссылки на необходимые стили.

При показе странички конкретному устройству должна быть задействована соответствующая случаю таблица стилей. Для сотового телефона и монитора компьютера они, разумеется, должны быть разными. В первом случае мы используем минимальное оформление, которое позволит представить информацию наиболее оптимально и компактно. Во втором же случае в нашем распоряжении имеется все богатство шрифтового и цветового оформления.

Таблицу стилей нужно написать всего один раз при создании сайта для каждого из устройств, на котором планируется вывод информации. К тому же таблица стилей может быть единой для целого сайта. И, следовательно, не нужно будет повторять одни и те же описания стилей на каждой из страниц.

Размещение всей стилевой информации в одном внешнем файле открывает нам и другие полезные возможности - ведь изменив содержимое только одного (!) стилевого файла, мы можем в считанные секунды сменить весь дизайн сайта. Причем никаких других переделок не понадобится. Разумеется, это верно лишь в том случае, если первоначально сайт был спроектирован верно.

### **Подключение таблиц стилей**

Для осуществления этой задачи мы можем воспользоваться одним из 3-х предлагаемых методов:

- внешний файл,
- inline-описание,
- описание в секции заголовка.

#### **inline-описание**

или описания, встроенное в тег:

```
<p style="color:red; text-align:center;">
```

Этот текст переопределен стилем

```
</p>
```

При помощи дополнительного атрибута `style` мы можем определить нужные нам стилевые параметры в любом теге. Это самый легкий способ, и действует он в пределах лишь одного тега. Но представьте, насколько вырастет размер файла, и насколько неудобно будет его исправлять, если мы будем указывать стиль у каждого тега. Этот способ не слишком отличается, к примеру, от прямого описания внешнего вида при помощи тега `<font>`.

#### **Описание в секции заголовка**

Его действие распространяется на всю страничку. Определение стилей происходит при помощи классов, которые представляют собой списки с определением всех необходимых параметров оформления.

При использовании этого метода описание стилей необходимо разместить в секции заголовка:

```
<head> ....
<style type="text/css">
<!--
.header {
text-align :center;
font-size : 27pt;}
.red {color : red; }
-->
```

```
</style>
</head>
```

Теперь эти стили можно применять в любом месте html-кода. Для этого используется следующая конструкция:

```
<p class=header>Этот текст написан стилем header</p>
```

```
<p class=red>Этот текст написан красным цветом</p>
```

Как видите, все не так уж сложно. Главное понять основные принципы. Кроме определения новых классов мы также имеем возможность переопределять стандартные теги. Например, тег `<p>`:

```
<style type="text/css">
<!-- p { text-align : center; font-size :12pt;}
--> </style>
```

Теперь весь текст, заключенный в теги `<p></p>`, будет выглядеть так, как определено данным стилем. Это очень удобно и позволяет легко адаптировать уже существующие странички к использованию стилей. Кроме того, это несколько уменьшает объем файла за счет отсутствия лишних атрибутов `class`.

### Вынесение описания стилей во внешний файл

Диапазон его воздействия простирается на все файлы, в которые включено описание. Это способ наиболее соответствует концепции HTML 4.0. В случае, если нам потребуется изменить внешний вид сайта, то будет достаточно скорректировать лишь один этот файл. Сравните его с предыдущими способами. В одном из них придется менять описание на каждой страничке, а в другом даже более того - около каждого тега, что, разумеется, совершенно не вдохновляет.

Каким же образом производится внедрение внешнего файла? Для начала создается стилевой файл с описанием всех нужных нам классов (`mystyle.css`):

```
.header { text-align : center; font-size : 27pt;}
.red { color :red; }
p { text-align : center; font-size : 12pt;}
```

А потом ссылка на него внедряется в документ при помощи тега `<link>`:

```
<head> .... <link rel="stylesheet" type="text/css"
href="css/mystyle.css" title="MyStyleSheet"> .... </head>
```

Это самый удобный способ, и для основной таблицы стилей рекомендуется пользоваться именно им.

### Каскадность стилей

Каскадность заключается в том, что стили могут переопределяться. Приведенный выше список способов внедрения стилей соответствует порядку переопределения. Нижерасположенный способ может переопределять вышерасположенный.

Например, мы определили во внешнем стилевом файле, что текст в теге `<p>` должен быть написан при помощи шрифта высотой 10 пунктов. Но если в заголовке странички мы дополнительно укажем, что тот же текст в теге `<p>` должен быть написан шрифтом в 12 пунктов, то текст будет выведен именно таким кеглем - т.е. стиль в заголовке странички переопределил стиль во внешнем файле.

### Синтаксис CSS

Описание каждого класса делается при помощи конструкции, подобной этой:

```
.small { font-size: 9pt; }
```

Сначала указывается имя класса - оно может быть произвольным, но желательно все-таки давать осмысленное название. Далее, в фигурных скобках перечисляются все

необходимые параметры для данного класса. Параметры отделяются друг от друга точкой с запятой. Вот еще один пример, в котором используется более длинное описание:

Заметьте, что имя класса начинается с точки и таким образом определяет универсальный класс, т.е. такой, который может быть применен к любому тегу. И делается это при помощи следующей конструкции:

```
<p class=small>Накладываем стиль на этот текст</p>
```

Существуют универсальные классы и, так называемые, теговые классы:

```
p.small { font-size: 9pt; }
```

Класс, определенный таким образом, сработает только в том теге, для которого он предназначен, а для всех остальных будет проигнорирован.

Мы можем определять параметры не только для одного тега, но и сразу для нескольких. Для этого в определении стиля достаточно перечислить их через запятую:

```
p, td { font-size: 9pt; color:green; }
```

Такой прием называется группировкой, и в данном случае мы определили и для <p>, и для <td> одинаковый размер и цвет текста.

В случае переопределения существующих тегов, в описании стиля можно указывать не все параметры, а лишь те из них, которые мы хотим изменить. Все остальные параметры примут значения по умолчанию, которые для разных тегов различны.

### Псевдоклассы

В CSS есть такое понятие как псевдокласс. В отличие от обычного класса, действие псевдокласса распространяется не на весь текст, к которому применен данный стиль, а лишь на его часть и возможно лишь в определенном состоянии. Чтобы было понятнее, давайте разберем эффект, при котором ссылки подчеркиваются лишь при наведении на них курсора. Эффект достаточно распространенный, и его можно наблюдать в том числе и на этом сайте. Вот фрагмент таблицы стилей, который позволяет достигать вышеописанного эффекта:

```
a { text-decoration: none; }  
a:hover { text-decoration: underline; }
```

Верхняя строчка - это переопределение стандартного тега <a>, которое запрещает подчеркивать ссылки, а вот нижняя - это определение стиля для псевдокласса hover, который описывает стиль ссылки в момент, когда курсор находится над ней.

А вот и другой пример псевдокласса - определение буквы в начале абзаца:

```
p:first-letter { font-size: 200%; font-weight: bold; }
```

Заметьте, что и в том, и в другом случае действие стиля распространяется либо на определенное состояние (пользователь собирается щелкнуть по ссылке), либо на фрагмент текста (изменяется только первая буква абзаца). В этом и заключается смысл псевдоклассов.

### Примечания

Как и в любом достаточно сложном языке, при создании таблицы стилей можно пользоваться комментариями. Их формат аналогичен классическому C:

```
/* Этот текст является комментарием */
```

Для небольших сайтов эта возможность Вам вряд ли пригодится, а вот при создании сложных, многоуровневых таблиц стилей комментарии могут пригодиться. Кстати, здесь будет уместно привести золотое правило - чем понятнее названа переменная (в данном случае имя класса), тем меньше комментариев необходимо.

### Основные параметры CSS

Все параметры, используемые для определения стиля, условно можно разделить на несколько больших групп:

управляющие видом шрифта (гарнитура, кегль, цвет, наклон, жирность,..)  
управляющие форматированием абзаца (выравнивание, интерлиньяж, расстояние между словами, отступ красной строки,..) управляющие свойствами блока (отступы слева-сверху-

справа-снизу, местоположение блока на страничке, видимость блока,..) другие, которые не вписываются ни в одну из перечисленных выше групп (цвет ссылок странички, изменение внешнего вида курсора,..) Рассмотрим подробнее параметры, используемые для управления внешним видом текста и форматирования абзацев - как наиболее часто употребляемые.

### **Основные параметры шрифта**

- font-weight: [bold|normal|number] - жирность шрифта
- font-style: [normal|italic|oblique] - наклон шрифта
- font-size: number - размер шрифта
- font-family: name - гарнитура шрифта
- color:number - цвет шрифта
- background-color: number - цвет подложки
- background: url - текстурная подложка

### **псевдоклассы Ссылок**

- A:active{ } Таблица стилей для активных ссылок (при нажатии)
- A:link{ } Таблица стилей для собственно ссылок
- A:visited{ } Таблица стилей для посещённых ссылок
- A:hover { } Таблица стилей для ссылок при наведении указателя мыши

### **Основные параметры абзаца (и Элементов типа "Вох")**

- text-align: [left|right|center|justify] - выравнивание
- text-indent: number - отступ красной строки
- line-height: number -интерлиньяж
- letter-spacing: number - трекинг
- padding-left: number - отступ от текста слева
- padding-right: number - отступ от текста справа
- padding-top: number - отступ от текста сверху
- padding-bottom: number - отступ от текста снизу
- margin-left: number - отступ от границы слева
- margin-right: number- отступ от границы справа
- margin-top: number - отступ от границы сверху
- margin-bottom: number - отступ от границы снизу

### **Единицы измерения в CSS**

В свойствах, которым требуется указание размеров, можно использовать несколько способов для их задания:

- относительный размер в процентах (%)
- относительный размер при помощи словесного описания (larger, smaller, xx-small, x-small, small, medium, large, x-large, xx-large)
- абсолютный размер в типографских единицах - размер может задаваться в пунктах (pt), пиках (pc), пикселях (px), средней шириной буквы "m" (em), средней шириной буквы "x" (ex)
- абсолютный размер в стандартных единицах длины - размер может задаваться в сантиметрах (cm), миллиметрах (mm), дюймах (in) абсолютный в пикселях (px)

### **Задание цвета в CSS**

Цвет для тех свойств, где это нужно, может быть определен одним из трех способов:

- при помощи названия цвета: yellow, red, green, grey,..
- шестнадцатеричным заданием цвета в формате #RRGGBB: #ff0000, #883490, #ffffff,..
- десятичным заданием составляющих цвета в формате rgb(red, green, blue): rgb(255,0,0), rgb(100,23,78),..

### **Примеры описания таблицы стилей:**

```

.epigraph {
    font-size: 12pt;
    font-style: italic;
    text-align: right;
    color: rgb(127,127,0);
}

p.big {
    font-size: 16px;
    font-weight: bold;
    color: #ff0000;
}

.menu {
    font-weight: bold;
    font-size: 9pt;
    font-family: arial, helvetica, sans-serif;
}

a:hover {
    color: #b63a3a;
    text-decoration: none;
}

```

## Несколько простых правил

### Фон

Текст должен читаться. Дикий, аляповатый фон сильно мешает восприятию содержания. При выборе фона учитывайте также, что распечатывать удастся лишь темный текст на светлом фоне.

### Картинки

Текст с иллюстрациями смотрится лучше, чем просто текст. Нужно учитывать, что в России пользователи Интернета страдают от медленной связи, графика загружается долго, поэтому лучше, если вы поместите небольшие по размеру и количеству килобайт картинки. **Скачивая понравившуюся картинку с чужого сайта, вы нарушаете закон об авторских правах.** Хорошо, когда вы сами рисуете иллюстрации к своей страничке.

### Шрифты

Слишком много разных шрифтов на странице - дурной тон. Лучше выделять слова размером, курсивом или жирным стилем. Подчеркивания воспринимаются как ссылки, советую этого избегать. Помните, ничто не раздражает читателя так, как мигающий текст.

### Стиль

Не важно, в каком стиле будет оформлен ваш сайт, поместите ли вы сердечки, цветочки и вензелочки, или используете в оформлении строгую линейную графику, главное, чтобы все страницы сайта были оформлены единообразно, подчинены одной идее.

### Порядок выполнения работы:

Создайте стилевой файл, содержащий оформление web-сайта. Обязательно определение оформления следующих элементов:

- Заголовки 1-4 уровней.
- Пункты меню 1-4 уровней.

- Гиперссылки.
  - Таблицы и ячейки таблиц.
  - Основной текст.
2. Для оформления страницы используются стандартные графические элементы. Удобно использовать специализированные редакторы, например PhotoShop. Разработайте:
- Баннер.
  - Разделители.
  - Списковые элементы (буллеты).
  - Пиктограммы и иконки (домой, назад, и т.д.).
3. Тематика сайтов уточняется с преподавателем.

**Форма представления результата:**

Отчет по выполненной лабораторной работе.

**Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

## Тема 08.01.01 Основы web-технологий

### Лабораторное занятие № 4 Вёрстка

**Цель:** получения практических навыков верстки сайтов

**Выполнив работу, Вы будете:**

**уметь:**

У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов

У.5. Учитывать существующие правила корпоративного стиля

У.02.2. Определять необходимые источники информации

У.02.5. Выделять наиболее значимое в перечне информации

У.09.2. Использовать современное программное обеспечение

**Материальное обеспечение:**

Методические указания для выполнения практических работ, текстовый редактор Atom, Sublime, Visual Studio Code, PHPStorm.

**Задание:**

Сверстать сайт по PSD-макету.

**Краткие теоретические сведения:**

Если табличная вёрстка подразумевает, что содержимое страницы находится внутри тега `<table>`, то концепция блочной вёрстки основана на активном использовании универсальных тегов `<div>`, внутрь которых помещается содержимое, включая другие теги.

Блочная вёрстка лишена недостатков табличной – поисковыми системами она индексируется лучше, её код не такой развесистый, да и блоки `<div>`, которые так любят называть «слоями», изначально задумывались универсальными, то есть «для всего», тогда как `<table>` — это таблица, которую нужно использовать для отображения табличных данных и не более того.

Единственный ощутимый минус блочной вёрстки – сделанные на ней сайты могут по-разному отображаться в обозревателях. Чтобы этого избежать, нужно делать вёрстку «кроссбраузерной», то есть одинаково отображаемой любым обозревателем.

**Суть блочной вёрстки**

В графическом редакторе создаётся макет сайта: размечается, где какая область страницы (шапка, низ, боковая панель, основной контент) будет находиться и сколько места занимать, готовятся картинки, фоны.

Каждая часть страницы помещается в свой блок `<div>`: верх сайта – в первый, меню – во второй, контент – в третий и т. д. Каждый блок наполняется содержимым средствами HTML, а также позиционируется и оформляется с помощью CSS-разметки.

Конечный HTML-документ представляет собой набор блоков `<div>` с контентом внутри. Оформление зачастую находится в отдельном CSS-файле, подключенном к странице тегом `<link>`, или как минимум в контейнере `<style>` секции `<head>`.

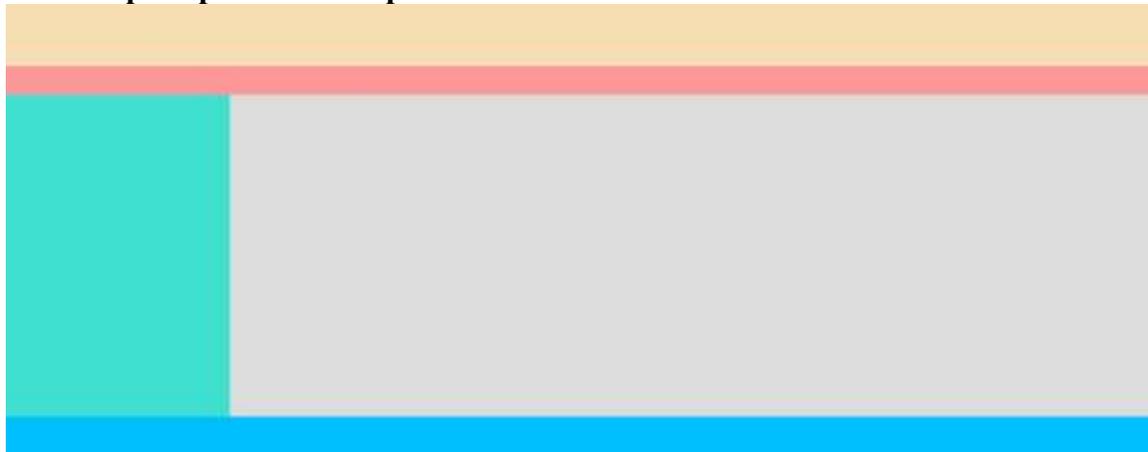
**Принципы блочной вёрстки**

Первый – конечно же, повсеместное использование тега `<div>`, который можно назвать базовым элементом блочной структуры, но об этом было сказано уже достаточно много.

Второе правило – принцип разделения кода, согласно которому содержимое от оформления нужно отделять. Говоря проще и ближе к нашему случаю: HTML — отдельно, CSS — отдельно (желательно в разные файлы). Такой подход делает структуру понятней. Программисту не нужно открывать CSS, дизайнеру – HTML.

Третий принцип – таблицы нужно использовать не везде, а по необходимости, при этом полностью отказываться от них так же странно, как и применять не к месту.

### Пример блочной вёрстки



Согласно макету, страница сайта будет содержать пять блоков: «шапку», навигационное меню, боковую панель, основной блок с контентом и «ноги».

Сначала создадим HTML-страницу: обозначим структуру, разметим её. HTML-код будет таким:

```
<!DOCTYPE html>
<html>
<head>
  <title>Блочная вёрстка</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<div id="container">
  div id="header">
    <h2>header (шапка сайта)</h2>
  </div>

  <div id="navigation">
    <h2>Блок навигации</h2>
  </div>

  <div id="sidebar">
    <h2>Левая панель</h2>
  </div>

  <div id="content">
    <h2>Основной контент страницы</h2>
  </div>

  <div id="clear">

</div>
```

```

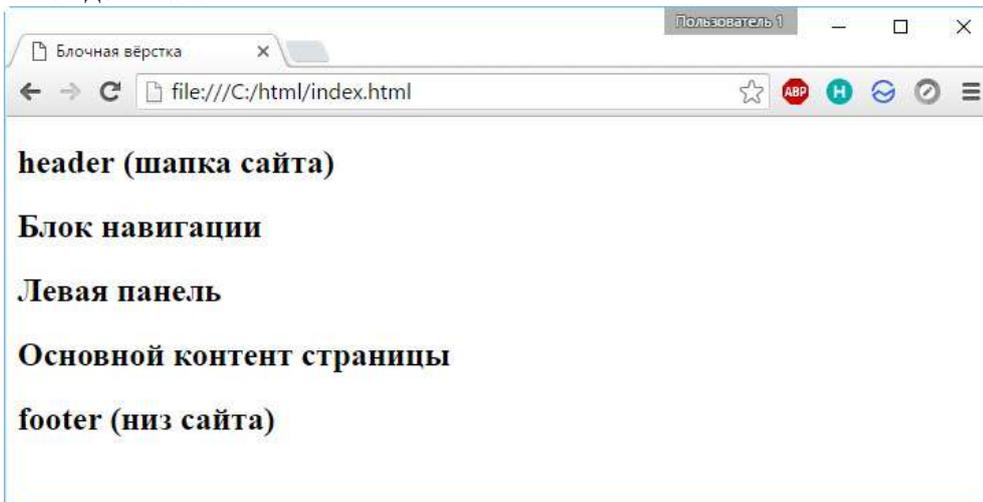
    <div id="footer">
      <h2>footer (низ сайта)</h2>
    </div>
  </div>
</body>
</html>

```

Разберём некоторые моменты.

**<div id="container">** - это блок-родитель, внутри которого расположились все остальные блоки. Как ячейки таблицы внутри **<table>**. Назначение дочерних контейнеров должно быть понятно, за исключением разве что блока **<div id="clear">**. Это вспомогательный слой, его смысл будет понятен, когда вы увидите код CSS.

Если открыть HTML-файл в браузере, не подключая таблицу стилей, страница будет выглядеть так.



Теперь добавим файл CSS, код которого приведён ниже.

```

body {
  background: #FFF;
  color: #000;
  font-family: Arial, sans-serif;
  font-size: 14px;
}

```

```

#header {
  background: #F5DEB3;
  width: 100%;
  height: 55px;
}

```

```

#container {
  background: #FFD700;
  margin: auto auto;
  text-align: center;
  width: 80%;
  height: 400px;
}

```

```

#navigation {

```

```

background: #FE9798;
width: 100%;
height: 25px;
}

#sidebar {
background: #40E0D0;
float: left;
width: 20%;
height: 280px;
}

#content {
background: #DCDCDC;
float: right;
width: 80%;
height: 280px;
}

#clear {
clear: both;
}

#footer {
background: #00BFFF;
width: 100%;
height: 40px;
}

```

С помощью стилей мы последовательно оформляем содержимое тега **<body>** и всех находящихся внутри контейнеров с помощью ранее изученных правил.

**#clear { clear:both; }** запрещает обтекание элемента слева и справа. Если убрать это правило, вёрстка «поедет» и низ сайта перестанет корректно отображаться.



Вот суть блочной структуры. Дальше можно только наполнять сайт содержимым и усложнять оформление, но делаться это будет всё равно по изложенному выше принципу.

### Порядок выполнения работы:

Сверстайте сайт по PSD-макету.



### Подготовительный этап

Создайте в отдельной папке файл **index.html**. В этот же каталог добавьте директорию **images**. Она будет содержать все картинки, используемые в шаблоне и на странице. Так как графические элементы мы вырезали заблаговременно, сразу скопируем их в папку images и дадим такие названия:

- **back\_all** — подложка сайта.
- **header\_top** — фон шапки.
- **big\_pic** — логотип.
- **title** — фон заголовков левой панели.
- **footer** — заливка низа сайта.
- **1mini** — первое фото для основной части страницы.
- **2mini** — второе фото.

В папке со страницей index.html создайте файл **styles.css** — в нём будут размещены таблицы стилей шаблона.

Делим документ на блоки

Откройте документ **index.html** и впишите в него следующий код:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Шаблон сайта</title>
```

```

<meta name="keywords" content="">
<meta name="description" content="">
<link href="styles.css" rel="stylesheet" type="text/css" media="screen">
</head>
<body>
</body>
</html>

```

Сохраните файл. Кодом выше мы создали основу HTML-страницы, теперь нужно разделить её на блоки — указать структуру документа, что за чем в нём будет идти.

Блоков у нас 7, перечислим их по идентификатору (значению атрибута **id**):

1. **content** — блок, внутри которого будут храниться остальные блоки.
2. **header** — блок шапки, внутри которого будут:
  - 2.1. **menu** — верхняя навигация.
  - 2.2. **logo** — картинка с текстом.
3. **right** — основная часть страницы.
4. **left** — панель слева.
5. **footer** — низ сайта.

Так и запишем (в контейнер **<body>** вставьте следующий код):

```

<div id="content">
  <!-- Шапка -->
  <div id="header">
    <div id="menu">
    </div>
    <div id="logo">
    </div>
  </div>
  <!-- Конец шапки -->
  <!-- Основной блок -->
  <div id="right">
  </div>
  <!-- Конец основного блока -->
  <!-- Левая панель -->
  <div id="left">
  </div>
  <!-- Конец левой панели -->
  <!-- Ноги сайта -->
  <div id="footer">
  </div>
  <!-- Конец -->

```

В браузере страница будет по-прежнему пустой, но структуру документа уже можно понять, она готова.

### Устанавливаем базовое форматирование

Теперь перейдём к CSS-оформлению, чтобы задать документу начальное оформление. Откройте **style.css** и добавьте туда строки кода, которые встретите ниже.

Убираем отступы и поля на странице по умолчанию:

```

*
{
margin: 0px;
padding: 0px;
}

```

Устанавливаем цвета ссылок в зависимости от поведения пользователя (навёл курсор, не навёл, посетил) и убираем подчёркивание у ссылок, над которыми находится указатель мыши:

```
a:link {
    color: #D72020;
}

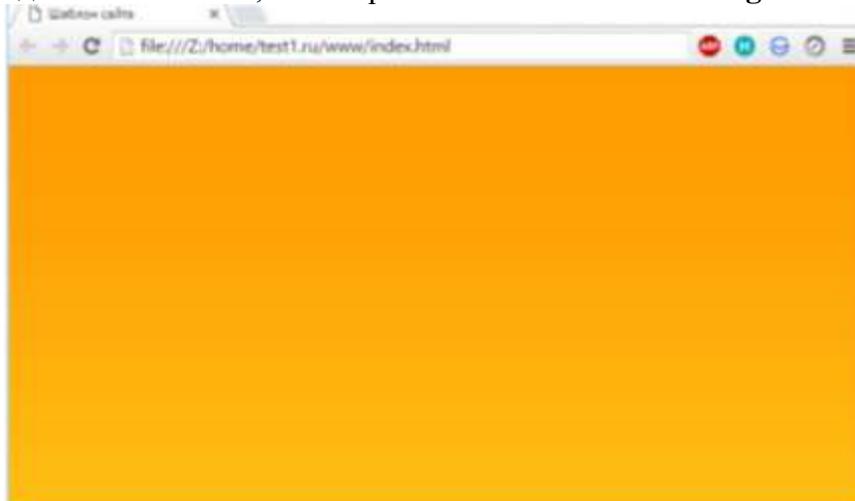
a:hover {
    text-decoration: none;
    color: #FF0000;
}

a:visited {
    color: #D72020;
}
```

Настраиваем основное оформление страницы: прописываем фоновый цвет и изображение-подложку (картинке задаём горизонтальное заполнение), устанавливаем цвет, стиль и размер шрифта:

```
body {
    background: #FFD723 url(images/back_all.jpg) repeat-x;
    font: 13px Tahoma, Verdana, Arial, Helvetica, sans-serif;
    color: #333333;
}
```

Вот теперь можно обновить страницу. Она заполнена рисунком-подложкой. Пока единственное видимое изменение, за которое отвечает свойство **background** класса **body**.



### Оформляем горизонтальное меню

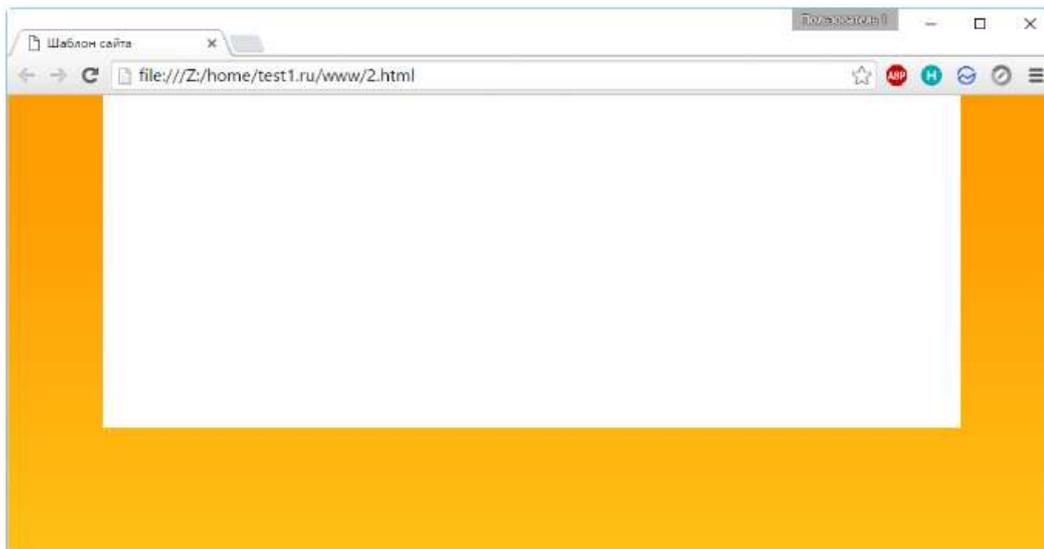
Начало есть, и теперь можно приступить к вёрстке уже непосредственно основных блоков.

Начнём с шапки. Которая, в свою очередь, состоит из блоков горизонтального меню и логотипа.

Сначала зададим общее оформление обоих элементов шапки: выравнивание текста по левому краю, белый фон и высоту 30брх:

```
#header {
    background: #ffffff;
    height: 30брх;
    text-align: left;
}
```

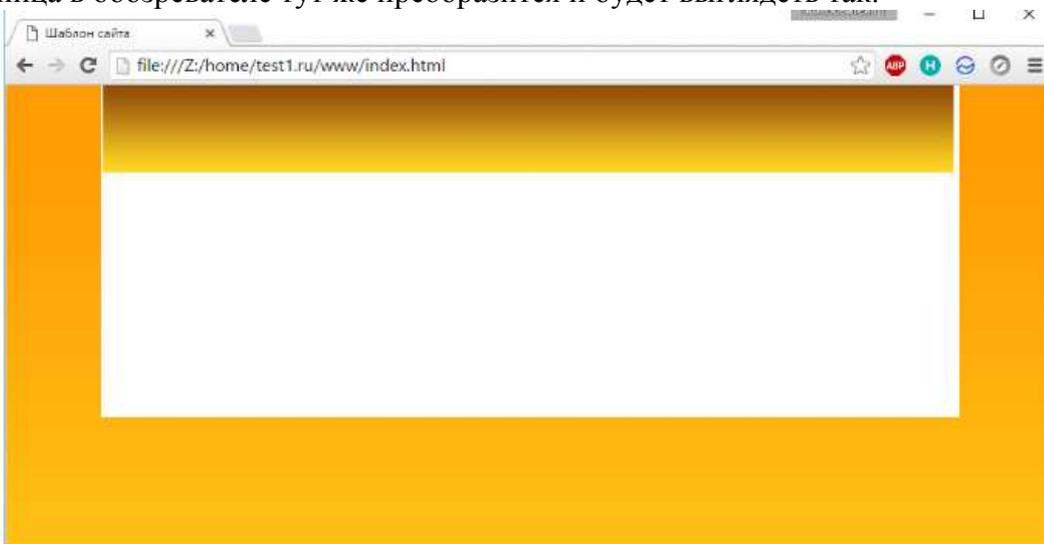
Пространство шапки: белый прямоугольник, на котором будут располагаться её элементы.



Горизонтальное меню. Внесём первые коррективы: зададим левую границу в 2 пикселя толщиной, ширину и высоту нашего меню, а также повторяющийся по оси X фоновый рисунок:

```
#menu
{
border-left: 2px solid #ffffff;
width: 779px;
height: 80px;
background: url(images/header_top.gif) repeat-x;
}
```

Страница в обозревателе тут же преобразится и будет выглядеть так.



Теперь можно добавить и само меню в файл **index.html**:

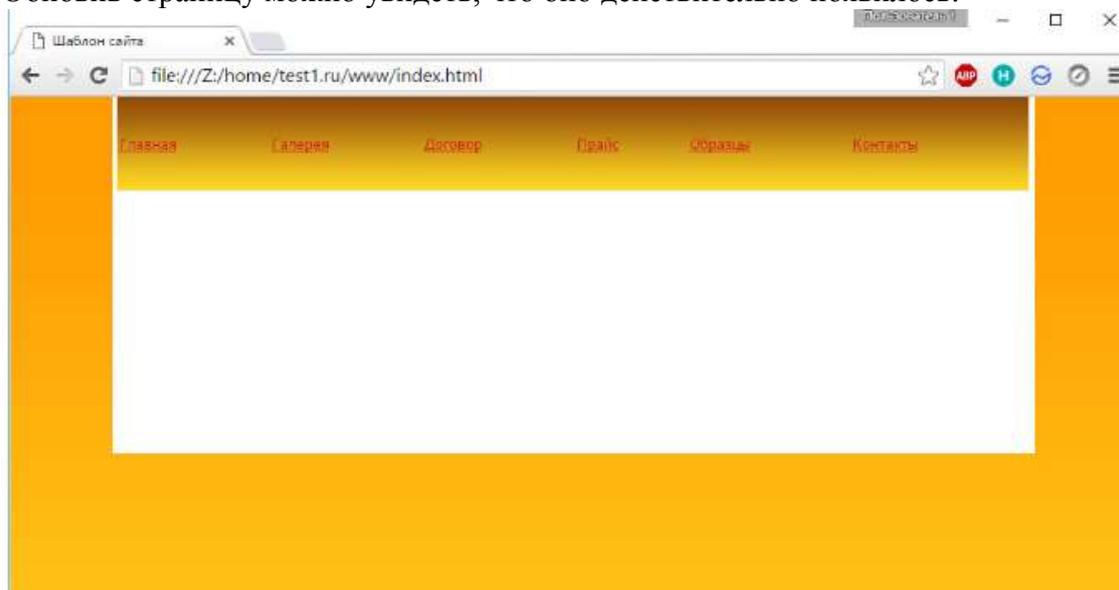
```
<table id="menu">
<tr>
<td><a href="#" title="">Главная</a></td>
<td><a href="#" title="">Галерея</a></td>
<td><a href="#" title="">Договор</a></td>
<td><a href="#" title="">Прайс</a></td>
<td><a href="#" title="">Образцы</a></td>
```

```

        <td><a href="#" title="">Контакты</a></td>
    </tr>
</table>

```

Обновив страницу можно увидеть, что оно действительно появилось.



Только вот вид ссылок оставляет желать лучшего. Установим для них свои правила (выравнивание, ширину, цвет, жирность и т. д.), а ссылкам при наведении зададим смену цвета и вернём убранное по всему шаблону подчёркивание:

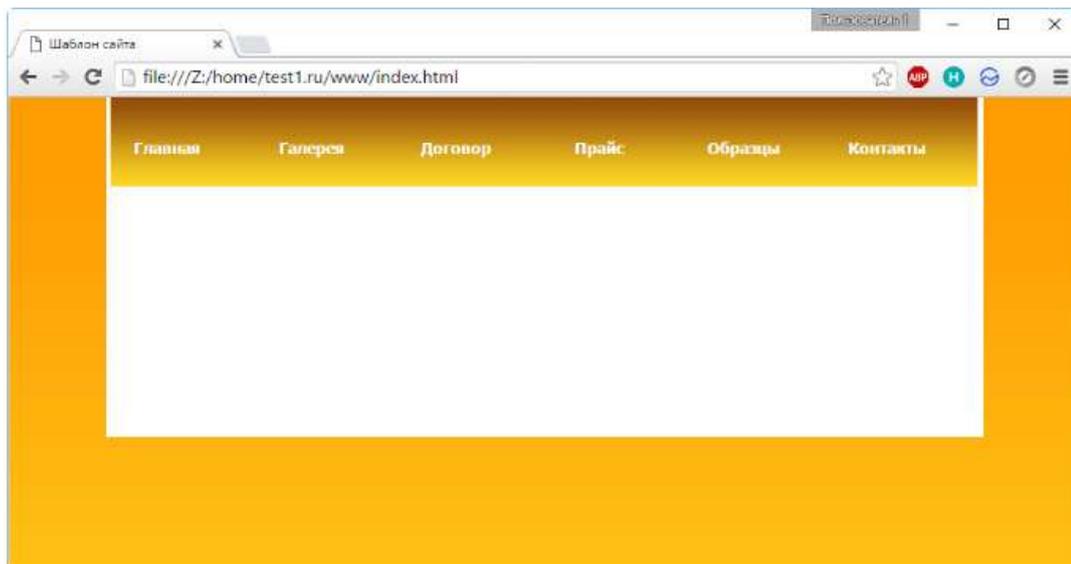
```

#menu a {
    float: left;
    width: 99px;
    height: 46px;
    display: block;
    text-align: center;
    text-decoration: none;
    color: #ffffff;
    font-weight: bold;
    font-size: 14px;
    padding-top: 35px;
}

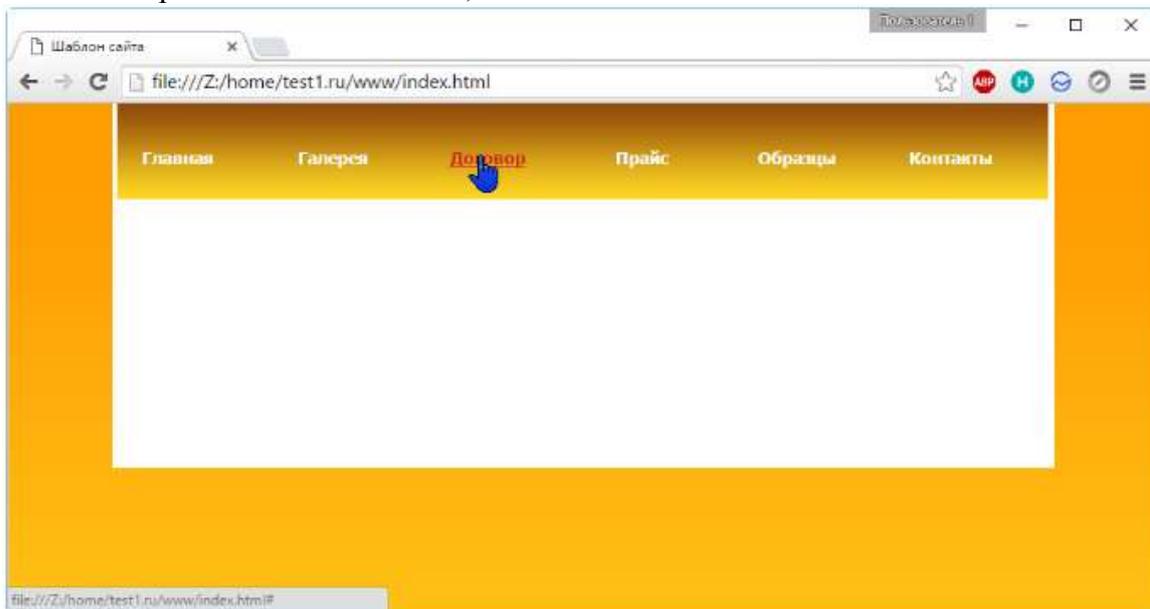
#menu a:hover {
    color: #D72020;
    text-decoration: underline;
}

```

Теперь форматирование меню можно сопоставить с PSD-шаблоном.



Обратите внимание, как меняется оформление пункта, если подвести к нему указатель (за это отвечают правила **#menu a: hover**).

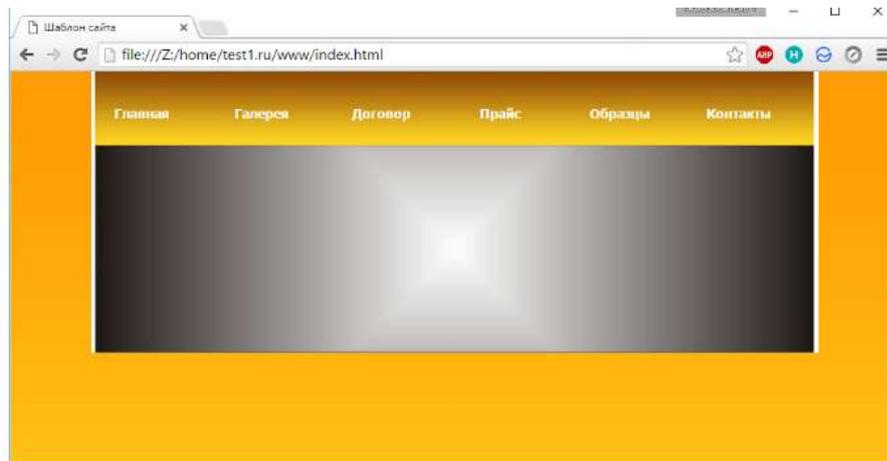


### Настраиваем логотип

Логотип у нас уже есть и лежит в папке, остаётся добавить его на страницу и отформатировать правилами. И то, и другое можно сделать средствами CSS, чем мы и не преминем заняться.

```
#logo {  
  background: #ffffff url(images/big_pic.jpg) no-repeat;  
  width: 738px;  
  height: 146px;  
  text-align: left;  
  padding-top: 80px;  
  padding-left: 40px;  
  border-left: 4px solid #ffffff;  
}
```

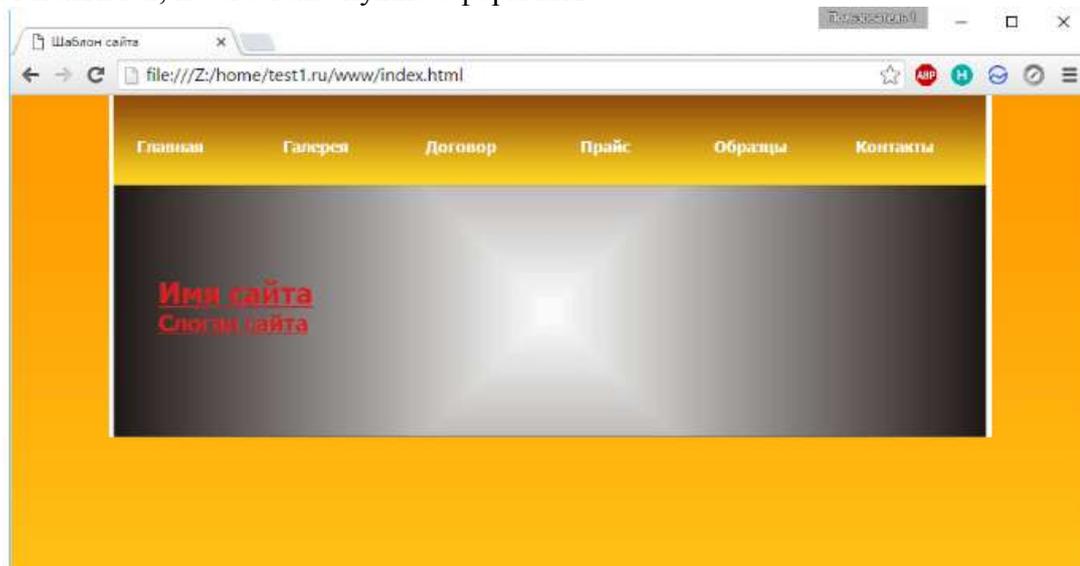
Логотип вставлен ровно по размеру.



Единственное, чего ему не хватает, так это текста. Вставим недостающее в блок **logo** файла **index.html**, чтобы получилось:

```
<div id="logo">  
  <h1><a href="#">Имя сайта</a></h1>  
  <h2><a href="#" id="metamorph">Слоган сайта</a></h2>  
</div>
```

Текст появился, но его тоже нужно оформлять.



Исходя из того, что логотип почти всегда бывает ссылкой, зададим оформление соответствующим классам.

```
#logo a {  
  text-decoration: none;  
  text-transform: lowercase;  
  font-style: italic;  
  font-size: 36px;  
  color: #FFFFFF;  
}  
#logo h2 a  
{  
  font-size: 24px;  
}
```

Внешний вид текста изменился, и в целом шапка теперь выглядит даже лучше, чем на PSD-макете.



### Верстаем основную часть страницы

Далее настраиваем самый большой блок, на котором будет размещён весь уникальный контент. Он будет занимать 500px и располагаться в правой части сайта. Установим правила позиционирования, оформления заголовков, абзацев и ссылок (обо всех свойствах мы уже говорили в статьях по CSS).

```
#right
{
float: right;
width: 500px;
padding-right: 10px;
}
```

```
#right h4
{
margin: 0;
padding: 0px;
font-size: 12px;
color: #D72020;
}
```

```
#right a
{
color: #D72020;
text-decoration: none;
}
```

```
#right p {
margin: 0;
padding: 0;
padding-bottom: 10px;
}
```

```
#right h2 {
margin: 0;
padding: 0;
```

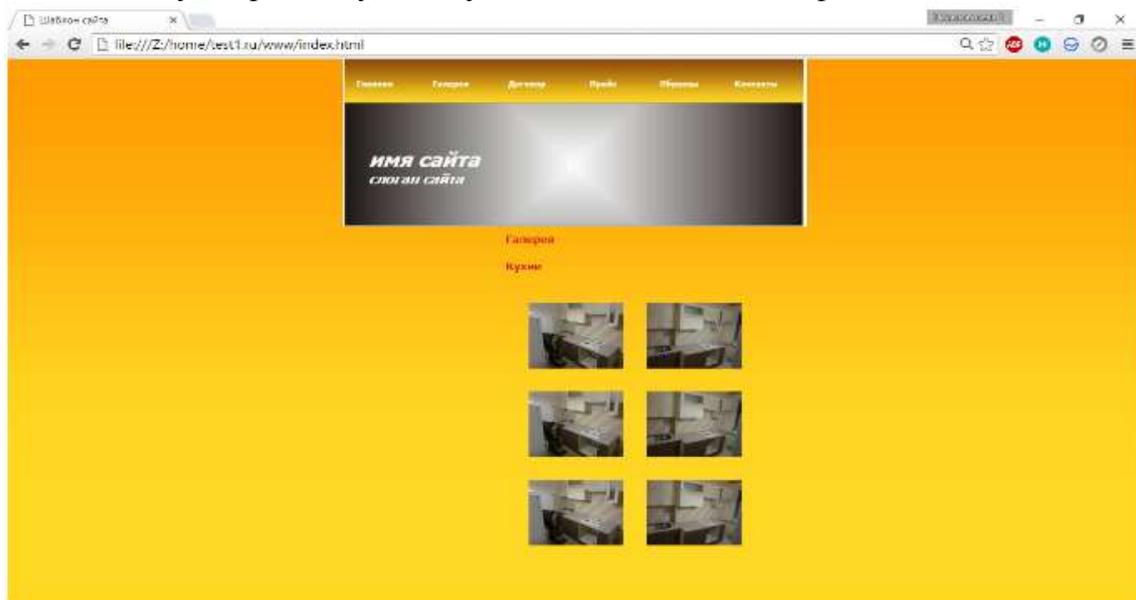
```
padding-top: 10px;
color: #D72020;
}
```

Так как мы зафиксировали только ширину блока, видимых изменений наблюдаться не будет до тех пор, пока мы не наполним его контентом — высота страницы будет меняться в зависимости от содержимого.

Заполним контейнер **right**. Изображения поместим в простую таблицу.

```
<h2>Галерея</h2><br />
<h2>Кухни</h2><br />
<table cellspacing = 40>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</table>
```

Контент получил разметку, но ему явно не хватает белого фона.



### Создание левой панели

Во время вёрстки, но уже после отрисовки макета, выяснилось, что боковое меню ещё будет иметь подпункты, причём они должны появляться при наведении на главный пункт и исчезать, когда указателя на нём нет.

Ситуации, когда приходится дорабатывать оформление «на ходу», встречаются довольно часто. Расстояние между меню и основным содержимым позволяет нам вставить подпункты, однако интересен фокус с исчезновением и появлением подменю.

В файл CSS впишите следующий код.

```
#left
{
padding: 10px;
```

```
width: 237px;
padding-right: 1em;
}
```

```
#left h3
{
width: 225px;
height: 25px;
font-size: 14px;
font-weight: bold;
padding-left: 15px;
padding-top: 15px;
text-transform: uppercase;
color: #ffffff;
background: url(images/title.gif) no-repeat
}
```

```
#left ul {
margin: 0;
padding: 10px;
list-style: none;
width: 100px;
font-size: 18px;
}
```

```
#left li ul {
position: absolute;
left: 90px;
top: 0;
display: none;
}
```

```
#left ul li {
position: relative;
margin-bottom: 20px;
}
```

```
#left ul li a {
display: block;
text-decoration: none;
color: #fffcc;
background: #ff9900;
padding: 5px;
border: 1px solid gold;
border-bottom: 0;
}
```

```
#left li: hover ul {
display: block;
}
```

```
#left li li {
    margin-bottom: 0px;
    width: 150px;
}
```

```
#left p
{
padding: 10px;
border-bottom: 1px solid #D72020;
border-left: 1px solid #D72020;
border-right: 1px solid #D72020;
}
```

Обратите внимание на правила классов **ul** и **li** — секрет исчезновения меню находится там, раскройте его самостоятельно.

В контейнер left HTML-документа добавим сначала информационный блок без меню.

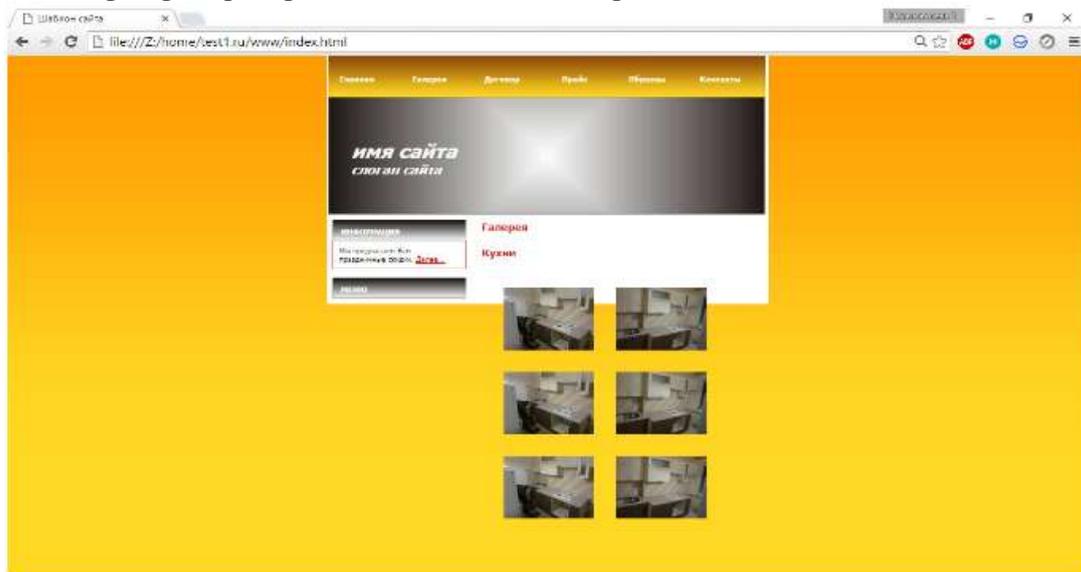
```
<h3>Информация</h3>
```

```
<p>Мы предлагаем Вам праздничные скидки. <a
```

```
href="http://test1.ru/news.php">Далее...</a></p><br />
```

```
<h3>Меню</h3>
```

Белый фон распространился ещё ниже по странице.



Теперь самое время вставить в HTML-файл код меню левой панели. Оно, в отличие от верхней навигации, реализовано списками, что можно было заметить ещё из CSS-правил.

```
<ul>
```

```
<li><a href="#">Галерея</a>
```

```
<ul>
```

```
<li><a href="#">Кухни</a></li>
```

```
<li><a href="#">Кровати</a></li>
```

```
<li><a href="#">Стенки</a></li>
```

```
<li><a href="#">Прихожие</a></li>
```

```
<li><a href="#">Шкафы-купе</a></li>
```

```
<li><a href="#">Компьютерные
```

```
столы</a></li>
```

```
</ul>
```

```
</li>
```

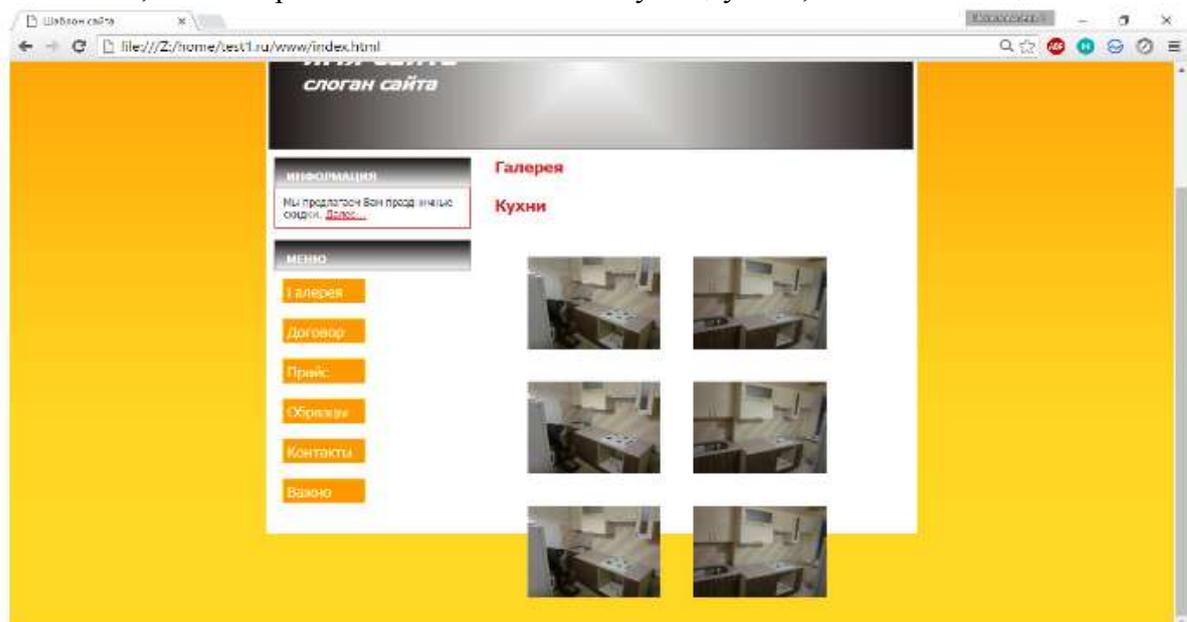
```
<li><a href="#">Договор</a></li>
```

```

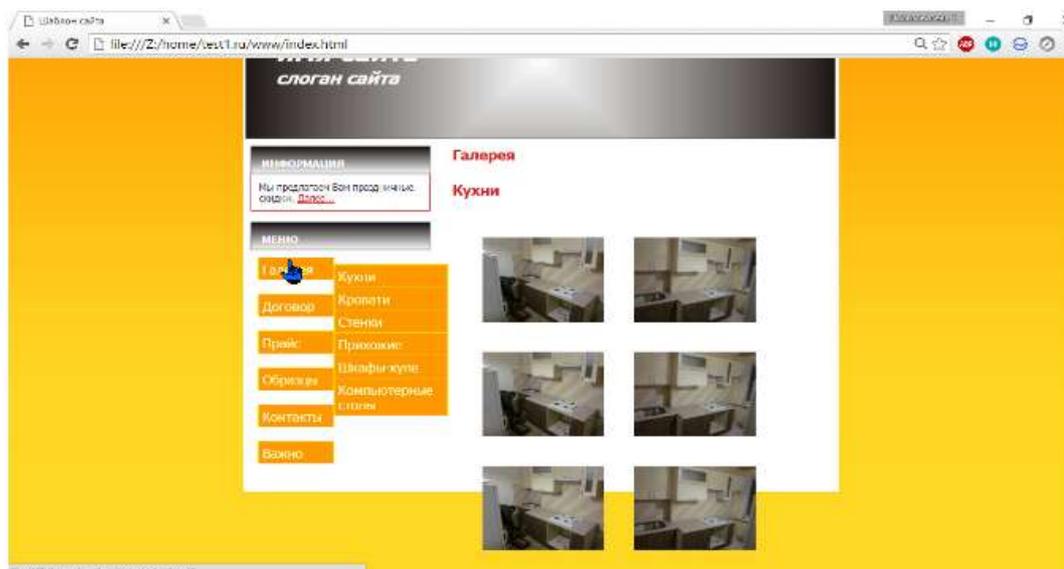
<li><a href="#">Прайс</a>
  <ul>
    <li><a href="#">Кухни</a></li>
    <li><a href="#">Кровати</a></li>
    <li><a href="#">Стенки</a></li>
    <li><a href="#">Прихожие</a></li>
    <li><a href="#">Шкафы-купе</a></li>
    <li><a href="#">Компьютерные
столы</a></li>
  </ul>
</li>
<li><a href="#">Образцы</a>
  <ul>
    <li><a href="#">Стекло</a></li>
    <li><a href="#">ДСП</a></li>
    <li><a href="#">Фурнитура</a></li>
    <li><a href="#">И т.д.</a></li>
  </ul>
</li>
<li><a href="#">Контакты</a>
<li><a href="#">Важно</a>
</li>
</ul>

```

Взгляните, как смотрится список. Многие могут подумать, что это таблица.



Подменю работает, надо только подвести мышку к пункту, его содержащему (у нас это Галерея, Прайс и Образцы).



### Делаем ноги

Завершающая часть шаблона — футер. Простое оформление, из элементов только абзацы и ссылки.

```
#footer {
    height: 44px;
    clear: both;
    padding-top: 20px;
    background: url(images/footer.gif) repeat-x;
    border-top: 5px solid #A6640E;
}
```

```
#footer p {
    margin: 0;
    font-size: 10px;
    text-align: center;
    color: #ffffff;
}
```

```
#footer a {
    color: #ffffff;
}
```

Обычно в футере размещается вспомогательная информация, добавим её в HTML-код (контейнер **footer**).

```
<p>Copyright © 2017. <a href="http://test1.ru/" title="Адрес сайта">Адрес сайта</a>
| <a href="#">Слоган сайта</a></p>
<p>+7-(4832)-37-77-77 | <a href="#">Тула</a></p>
```

На этом оформление сайта завершено. В результате мы получаем законченный, готовый к использованию шаблон, части которого можно вынести в отдельные PHP-файлы, сделав из статической страницы динамическую.

### Форма представления результата:

Отчет по выполненной лабораторной работе.

### Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

## Тема 08.01.01 Основы web-технологий

### Лабораторное занятие № 5

#### Использование языка сценариев JavaScript при создании web-сайта

**Цель:** получение практических навыков использования языка сценариев JavaScript при создании сайта

**Выполнив работу, Вы будете:**

**уметь:**

У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов

У.09.2. Использовать современное программное обеспечение

**Материальное обеспечение:**

Методические указания для выполнения практических работ, текстовый редактор Atom, Sublime, Visual Studio Code, PHPStorm.

**Задание:**

1. Сменить цвет фона.
2. Определить количество введенных знаков в поле формы
3. Динамическое добавление поля по желанию пользователя
4. Определение текущего времени и дата

**Краткие теоретические сведения:**

JavaScript — прототипно-ориентированный сценарный язык программирования. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

**Порядок выполнения работы:**

**Задание 1:** смена фона.

```
1 <html>
2 <head>
3 </head>
4 <body>
5 <FORM>
6 <SELECT onChange="document.bgColor=this.options[this.selectedIndex].value">
7 <OPTION VALUE="red">красный</OPTION>
8 <OPTION VALUE="2E8B57">морской волны</OPTION>
9 <OPTION VALUE="87CEEB">голубой</OPTION>
10 <OPTION VALUE="brown">коричневый</OPTION>
11 <OPTION VALUE="yellow">желтый</OPTION>
12 <OPTION VALUE="blue">синий</OPTION>
13 <OPTION VALUE="FFFFFF" SELECTED>белый</OPTION>
14 </SELECT>
15 </FORM>
16 </body>
17 </html>
```

**Результат:**



**Задание 2:** Определить количество введенных знаков в поле формы.

```
1 <html>
2 <head>
3 <title>Количество введенных знаков в поле формы.</title>
4 </head>
5 <body>
6 <form name=add>
7 <textarea class=forms name=descr rows=2 cols=25></textarea>
8 <br>
9 <SCRIPT language=javascript type="text/javascript">
10 document.write("введено знаков: <input type=text name=curtxt size=4 class=forms>");
11 </SCRIPT>
12 </form>
13 <SCRIPT language=javascript type="text/javascript">
14 functiongettxt()
15 {
16 document.add.curtxt.value=document.add.descr.value.length;
17 setTimeout("gettxt()",500);
18 }
19 gettxt();
20 </SCRIPT>
21 </body>
22 </html>
```

**Результат:**



**Задание 3:** динамическое добавление поля по желанию пользователя.

```

1 <html>
2 <head>
3 <title>Динамическое добавление поля по желанию пользователя</title>
4 <script language="javascript">
5 var items=1;
6 functionAddItem() {
7 div=document.getElementById("items");
8 button=document.getElementById("add");
9 items++;
10 newitem="<strong>Поле " + items + ": </strong>";
11 newitem+="<input type=\"text\" name=\"item\" + items;
12 newitem+="\" size=\"45\"><br>";
13 newnode=document.createElement("span");
14 newnode.innerHTML=newitem;
15 div.insertBefore(newnode,button);
16 }
17 </script>
18 </head>
19 <body>
20 <form name="form1">
21 <div ID="items">
22 <strong>Поле 1: </strong><input type="text" name="item1" size="45"><br>
23 <input type="button" value="Добавить поле" onClick="AddItem();" ID="add">
24 </div>
25 </form>
26 </body>
27 </html>

```

**Результат:**

The screenshot shows a web form with two input fields labeled "Поле 1:" and "Поле 2:". Below the second field is a button labeled "Добавить поле".

**Задание 4:** текущее время и дата.

```

1 <html>
2 <head>
3 <title>Дополнительное окно ТЕКУЩЕЕ ВРЕМЯ И ДАТА</title>
4 </head>
5 <body>
6 <a href="" onclick="alert(new Date());return false">Текущее время и дата</a>
7 </body>
8 </html>

```

**Результат:**

The screenshot shows a browser window with a link labeled "Текущее время и дата". Clicking the link has triggered a confirmation dialog box with the text "Подтвердите действие" and "Sun May 10 2020 13:02:49 GMT+0500 (Екатеринбург, стандартное время)". There is an "OK" button at the bottom right of the dialog.

**Форма представления результата:**

Отчет по выполненной лабораторной работе.

**Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

## Тема 08.01.01 Основы web-технологий

### Лабораторное занятие № 6

#### Подготовка и оптимизация графики на web-странице

**Цель:** изучение алгоритма подготовки графических элементов веб-страницы, изучение процесса обработки изображений для Веб, изготовление фавикон из картинки

#### Выполнив работу, Вы будете:

##### *уметь:*

- У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов
- У.5. Учитывать существующие правила корпоративного стиля
- У.02.2. Определять необходимые источники информации
- У.02.5. Выделять наиболее значимое в перечне информации
- У.09.2. Использовать современное программное обеспечение

#### Материальное обеспечение:

Методические указания для выполнения практических работ, графический редактор Adobe Photoshop.

#### Задание:

1. Сохранение документа для интернета и устройств
2. JPEG-сохранение
3. GIF-сохранение
4. Взвешенная оптимизация
5. Настройка параметров вывода

#### Краткие теоретические сведения:

Компьютерные изображения в Web — это файлы определенных форматов, которые распознаются браузерами и графическими приложениями. В Сети используются форматы JPEG, GIF, PNG. Но в компьютерном мире изображения могут существовать и во многих других форматах.

**GIF (Graphics Interchange Format)** — наиболее широко распространенный формат в Web. Файлы такого формата чаще всего являются созданными на компьютере: рисунок, текст. Максимальное число цветов в изображении в этом формате, не может превышать 256.

Схема сжатия, используемая в GIF, дает хорошие результаты для изображений с большими областями. Кроме этого, формат GIF позволяет создавать прозрачные изображения, так что, кажется, будто рисунок действительно нарисован на Web-странице и создавать анимацию.

**JPEG (Joint Photographic Experts Group).** Данный формат использует метод сжатия с потерями, при работе с ним всегда существует возможность выбора между меньшим размером файла и качеством изображения. Он обычно используется для передачи фотографий, поскольку он умеет неплохо справляться с изображениями, содержащими несколько миллионов цветов.

**PNG (Portable Network Graphics).** Этот формат также умеет делать изображения прозрачными и позволяет создавать анимацию.

Поскольку формат JPEG, хорошо подходит для фотографий, а формат GIF для рисованных иллюстраций, правильным выбором формата файла вполне можно достичь хорошего сочетания качества изображения и размера файла.

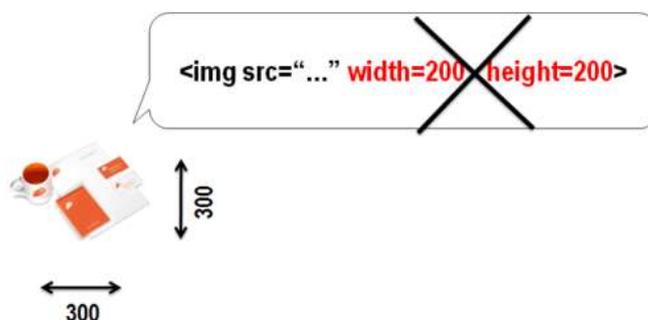
Вставить в Web-страницу изображение, довольно легко. Но гораздо сложнее — создание самого изображения.

Основные правила создания “хорошего” графического изображения:

1. Фотография и графика должны непосредственно относиться к информации на вашей странице.
2. Изображения должны загружаться быстро. А для этого файлы должны иметь разумный размер.

Общие пожелания для подготовки изображений:

1. Графические элементы должны быть представлены в формате GIF или PNG.
2. Фотографии должны быть представлены в формате JPEG.
3. Обеспечьте показ изображений с малым разрешением, пока идет загрузка больших изображений.
4. Используйте GIF-изображения с чересстрочным форматом.
5. Укажите атрибуты height и width элемента img.
6. Сократите количество анимации.
7. По возможности используйте пиктограммы изображений.
8. Обязательно создавайте «альтернативный» текст (атрибут alt).



При размещении градиента на фон, подбирается либо горизонтальный либо вертикальный градиент, и, в зависимости от этого, вырезается лишь тонкая линия данного градиента. При размещении в CSS устанавливаются соответствующие свойства фона (см. рис. 1.1)

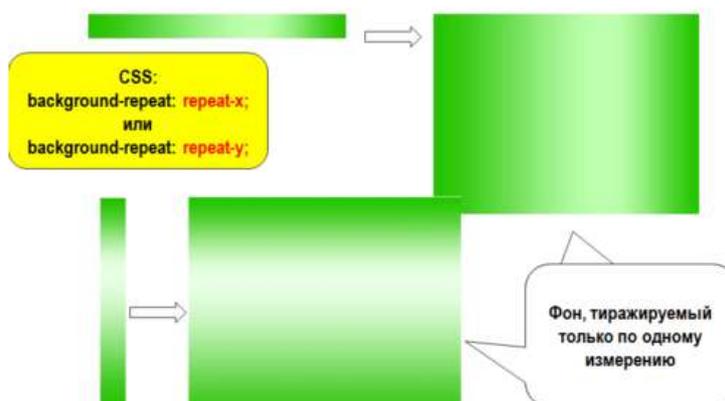


Рис. 1.1. Градиент на фон в Photoshop

Прежде чем использовать файл изображения в веб-странице, необходимо в графическом редакторе подготовить его для размещения. При этом важно не только определить в Photoshop размер изображения, но и правильно сохранить: используя **диалоговое окно Сохранить для Веб и устройств** (см. рис. 1.2):

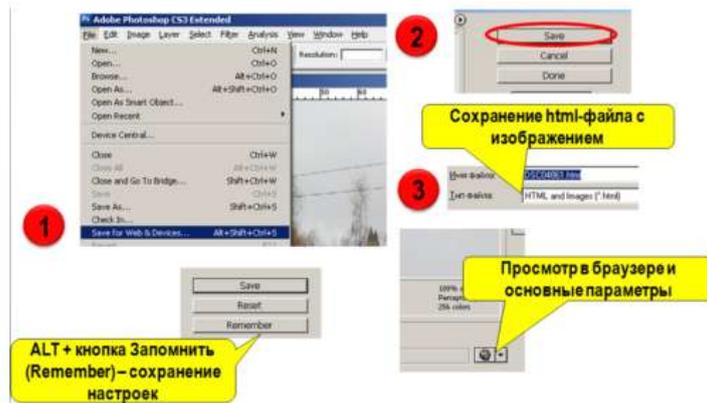


Рис. 1.2. Диалоговое окно Сохранить для Интернета и устройств

Устанавливаем настройки диалогового окна Сохранить для Интернета и устройств:

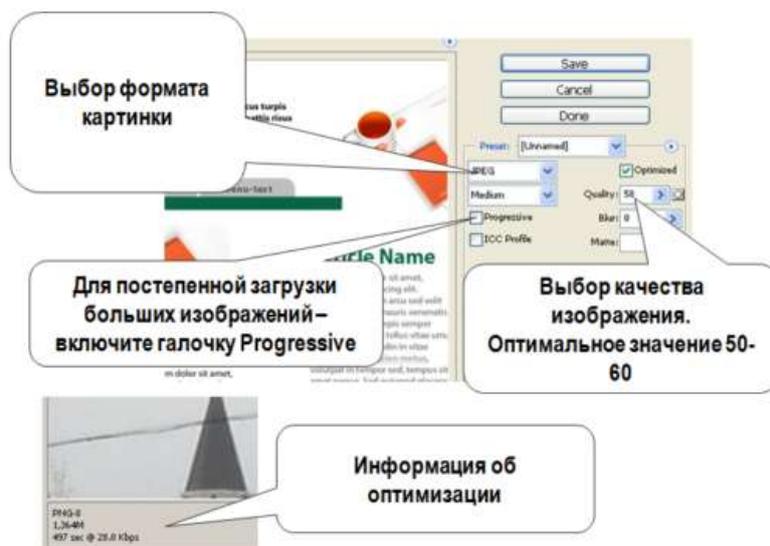


Рис. 1.3. Настройки диалогового окна Сохранить для Интернета и устройств  
Оптимизация и сжатие до заданного размера файла:

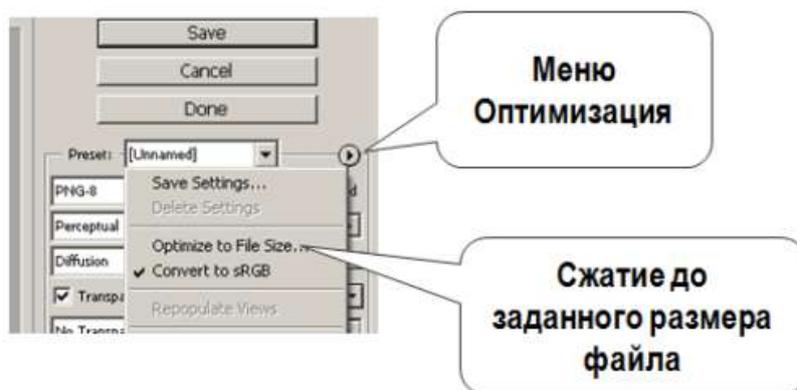


Рис. 1.4. Оптимизация и сжатие до заданного размера файла

Для размещения качественных фото, необходимо подготовить в Photoshop изображение в формате JPEG:



Рис. 1.5. Параметры оптимизации JPEG

Установка параметров оптимизации для форматов GIF и PNG

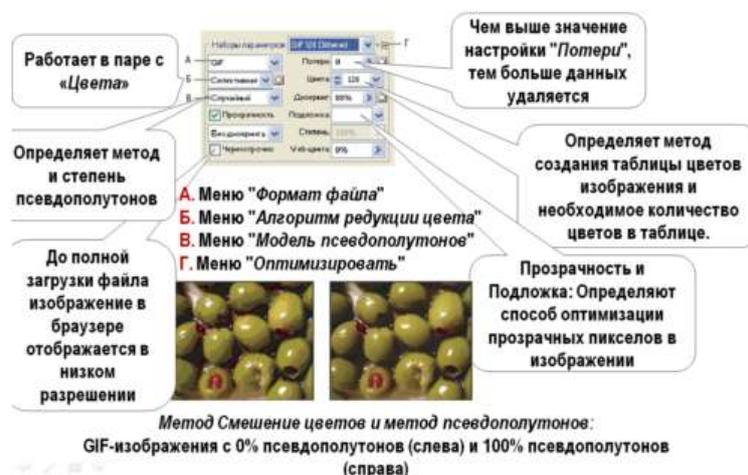


Рис. 1.6. Параметры оптимизации для форматов GIF и PNG

### Порядок выполнения работы:

#### Задание 1. Сохранение документа для интернета и устройств

1. Откройте изображение в Photoshop и выберите «Файл» -> «Сохранить для Интернета и устройств».
2. В диалоговом окне «Сохранить для Интернета и устройств» щелкните вкладку «Оптимизированный».
3. В меню «Стиль» выберите «Низкое качество JPEG».
4. Щелкните вкладку «Размер изображения».
5. Убедитесь, что установлен параметр «Сохранить пропорции» и введите ширину. Для электронной почты подходит размер **400** пикселей.
6. Нажмите кнопку «Сохранить». Введите имя файла и путь для сохранения файла. Убедитесь, что в меню «Формат» выбран вариант «Только изображения». Нажмите кнопку «Сохранить».

#### Задание 2. JPEG-сохранение

1. Откройте в редакторе изображение jpg формата (или скачайте изображение желтого цветка) и вызовите диалог «Сохранить для Веб и устройств» (Save for Web & Devices) в меню File.

2. Выберите вкладку «*Оптимизированный*» (*Optimized*).
3. Откройте список «*Наборы параметров*» (*Preset*) и выберите один из 12 предлагаемых вариантов – *JPEG Medium* (средний).
4. Оцените качество изображения и размер итогового файла.
5. Уровень качества, можно грубо задать в меню, а можно точно настроить ползунком «*Качество*» (*Quality*): передвиньте ползунок на 70.
6. Включите флажок опции «*Оптимизированный*» (*Optimized*): позволяет чуть уменьшить размер файла в обмен на ограничение совместимости со старинными браузерами.

### **Задание 3. GIF-сохранение**

1. Скачайте файл для работы.
2. Из «*списка форматов*» выберите пункт «*GIF*».
3. Выберите (или введите) количество оттенков в поле «*Цвета*» (*Colors*) и посмотрите, как это отразилось на изображении и размере итогового файла (установите 90).
4. Для уменьшения исходной гаммы картинки до заданного количества, можно воспользоваться одним из 9 доступных методов редукции цвета. Выберите одну из четырёх верхних опций: установите «*Ограниченная*» (*Restrictive*).
5. Смешение цветов (*Dithering*) отчасти компенсирует узость гаммы, «конструируя» отсутствующие цвета из набора имеющихся. Выберите «*Диффузия*» (*Diffusion*) и передвиньте ползунок *Dither* (степень количества полутонов) на 80.
6. В таблице цветов выберите один из цветов (желтый), дважды щелкнув по цвету, и замените его другим оттенком. Для выбора можно воспользоваться *Пипеткой* в левой части экрана.
7. Щелкните кнопку *Done*, чтобы сохранить параметры оптимизации в исходный документ.

### **Задание 4. Взвешенная оптимизация**

1. Скачайте файл для работы.
2. Откройте файл в редакторе. Выделите фрагмент картинки (цветок) *Прямолинейным Лассо* или любым другим инструментом.
3. Сохраните выделение в альфа-канал (*Select* → *Save Selection* (*Выделение* → *Сохранить выделение*)). Дайте ему имя (введите в поле *Имя*).
4. Откройте диалог «*Сохранить для Веб и устройств*», выберите «*JPEG*» в качестве формата на выходе и, если нужно, настройте уровень качества.
5. Щёлкните кнопку «*маска*» по соседству с полем «*Качество*» (*Quality*). Открывается диалоговое окно.
6. Выберите в меню «*Канал*» (*Channel*) только что сохранённый альфа-канал. Настройте «*Минимальное*» и / или «*Максимальное*» значения по вкусу и нажмите *Ok*.

### **Задание 5. Настройка параметров вывода**

1. В диалоговом окне «*Сохранить для Интернета и устройств*» в раскрывающемся меню «*Оптимизация*» выберите пункт «*Изменить настройки вывода*» (*Edit output settings*).
2. Отметьте флажок *Вывод XHTML*: При экспорте создаются web-страницы, соответствующие стандарту XHTML.

3. Щелкните по кнопке *Следующая* и настройте параметры в группе «Фрагменты».
4. Настройте вывод фона щелкнув по *Следующая*.
5. Щелкните *OK* и сохраните оптимизированный файл.
6. Просмотрите HTML-код результирующего файла.

**Форма представления результата:**

Отчет по выполненной лабораторной работе.

**Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

## Тема 08.01.01 Основы web-технологий

### Лабораторное занятие № 7 Создание баннера для web-страницы

**Цель:** получение практических навыков создания баннера для сайта

**Выполнив работу, Вы будете:**

**уметь:**

- У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов
- У.5. Учитывать существующие правила корпоративного стиля
- У.02.2. Определять необходимые источники информации
- У.02.5. Выделять наиболее значимое в перечне информации
- У.09.2. Использовать современное программное обеспечение

**Материальное обеспечение:**

Методические указания для выполнения практических работ, текстовый редактор Atom, Sublime, Visual Studio Code, PHPStorm.

**Задание:**

Создайте баннеры для своего сайта.

**Порядок выполнения работы:**

*Задание 1: создание баннера*

5. Зайдите на сайт <http://cooltext.com/>
6. Выберите шаблон «Legal».

Обратите внимание, что баннеры есть статичные, с графическими эффектами (что-то блестит, сверкает и анимированные (что-то движется).



Перед вами появится среда редактирования данного баннера.



7. В текстовое поле впишите текст баннера «Мой сайт». Через несколько секунд текст изменится в самом баннере.

### LEGAL Text Generator

Мой сайт

8. Измените шрифт баннера на любой вам понравившийся, нажав ссылку **Font**.

Font **CrutchShaded** AaBbCcDdEeFfGgHhIiJjKkLlMmNn

9. Установите размер шрифта «35».

10. Самостоятельно измените текстовые эффекты (цвет текста и подсветка).

Text Color

Highlight Color

11. Самостоятельно измените параметры тени (сдвиг и прозрачность).

Shadow Type  Medium Blur

Shadow Offset X  Y

Shadow Color  Opacity

12. Установите размер картинки.

Для этого сначала посмотрите размер баннера на вашем сайте в режиме конструктора и такой же размер установите на создаваемом баннере. Снимаем флажки Auto и устанавливаем ширину и высоту.

Alignment  Top Left

Width  Auto

Height  Auto

13. Выберите формат GIF.

14. Как только выбрали GIF-формат, появилась новая настройка фона картинки.

File Format

Background Color

Выберите фон картинки, соответствующий вашему сайту.

15. Создаем баннер. Нажмите кнопку

Create Logo

16. Появилась страничка, на которой вы видите готовый баннер и можете его скачать, нажав на Download.

17. Скачайте данный баннер в свою рабочую папку и запомните имя картинки.

18. Разместите данный баннер на вашем сайте.

*Задание 2: самостоятельно создайте два других баннера для вашего сайтаю*

*Задание 3: Самостоятельно освойте процесс создания баннера в среде*

*<http://www.artbanner.com.ua/generator-bannerov-onlayn> и создайте новый баннер.*

**Форма представления результата:**

Отчет по выполненной лабораторной работе.

**Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

## Тема 08.01.02 Web-дизайн

### Лабораторное занятие № 8 Разработка эскизов веб-приложения

**Цель:** получение практических навыков создания эскиза сайта

**Выполнив работу, Вы будете:**

**уметь:**

- У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов
- У.5. Учитывать существующие правила корпоративного стиля
- У.02.2. Определять необходимые источники информации
- У.02.5. Выделять наиболее значимое в перечне информации
- У.09.2. Использовать современное программное обеспечение

**Материальное обеспечение:**

Методические указания для выполнения практических работ, текстовый редактор Atom, Sublime, Visual Studio Code, PHPStorm.

**Задание:**

1. Ознакомиться с методическими указаниями.
2. Создать эскиз сайта.

**Краткие теоретические сведения:**

Любая web-страница содержит определенный набор стандартных элементов, являющихся обязательными компонентами каждого ресурса Интернета. Безусловно, ассортимент и количество подобных объектов могут варьироваться в зависимости от тематической направленности сайта, объема опубликованных на нем материалов, а также от целей и задач, которые ставит перед собой создатель данного ресурса. Компоновка таких элементов, проектирование их взаимного расположения и составляет одну из главных задач web-мастера.

Основные элементы страницы.

Зачастую основными элементами страницы являются:

- ✓ содержащий блок (wrapper, container)
- ✓ логотип
- ✓ навигация
- ✓ контент
- ✓ футер (нижний колонтитул)
- ✓ свободное пространство (по сути свободное пространство — это не элемент дизайна, но понятие, помня о котором при составлении макета страницы, проект не будет выглядеть как нагромождение блоков).

Содержащий блок (контейнер). Роль контейнера на странице может выполнять непосредственно элемент `body` или же `div`. Ширина содержащего блока может быть резиновой (`fluid`), а может быть фиксированной (`fixed`).

Изначально разработчику сайта ширина окна браузера пользователя неизвестна, поскольку она может меняться в самых широких пределах. Ширина зависит от разрешения монитора, длины его диагонали, размера окна и еще некоторых варьируемых данных. Иными словами предугадать ее заранее простыми средствами не представляется возможным. С учетом этой особенности утвердилось два способа верстки: фиксированный и «резиновый».

При работе с фиксированным макетом устанавливаем общую ширину макета жестко заданной и равной определенной величине. Если взять некоторую обобщенную статистику посетителей сайтов и посмотреть, какое разрешение монитора они преимущественно используют, то узнаем, что это 800 x 600 и 1024 x 768 пикселей. Получается, что ширина монитора пользователей в основном 800 и 1024 пикселя. Возьмем за ориентир 800 пикселей, тогда общая ширина макета за вычетом вертикальной полосы прокрутки и рамки браузера окажется 770–790 пикселей. На этот размер ориентируемся и устанавливаем ширину макета, например 770 пикселей.

«Резиновый» макет основывается на том, что в качестве одной из единиц измерения выступают проценты. Общая рабочая ширина окна браузера — 100%, и колонки макета в сумме не должны ее превышать, поэтому для удобства, как правило, везде применяют процентную запись. При изменении размеров окна происходит переформатирование данных страницы, чтобы они вписались в новую ширину.

Этот вид верстки набирает все большую популярность и практически все известные сайты выбрали именно его в силу того, что эффективно задействуется вся площадь веб-страницы.

Логотип - текстовая или графическая составляющая проекта и выделяющая его среди других. Логотип чаще всего располагается в верхнем левом углу страницы или же посередине (в зависимости от идеи, макета).

Навигация. Основная навигационная панель содержит ссылки на основные разделы сайта. Навигационная панель часто располагается в верхней части страницы (в независимости от того вертикально или горизонтально располагаются элементы навигации).

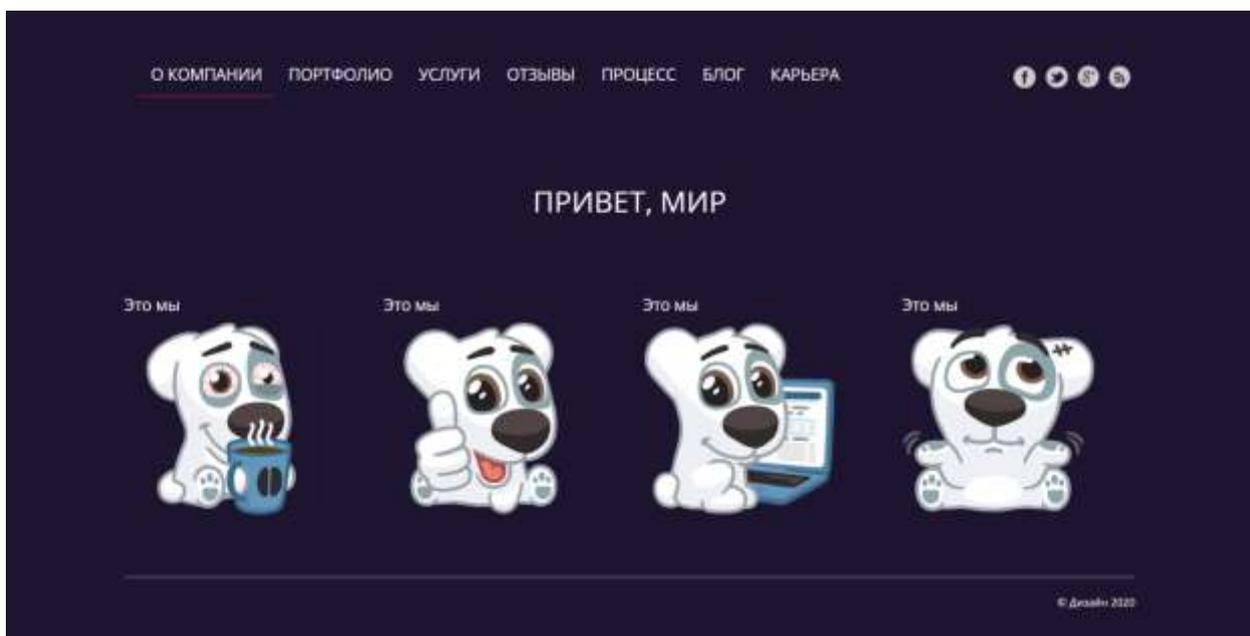
Контент – это основная составляющая веб-страницы. Он занимает главенствующую роль в дизайне страницы, поэтому занимает большее пространство, подкреплён, помимо текста, графикой.

Нижний колонтитул (footer). Данный элемент располагается внизу страницы и обычно содержит информацию о правообладателе, контактные и юридические данные, ссылки на основные разделы сайта (зачастую дублирует основную навигацию), ссылки на социальные сети, форму обратной связи и пр.

### Порядок выполнения работы:

1. Сформировать верхний колонтитул страницы.

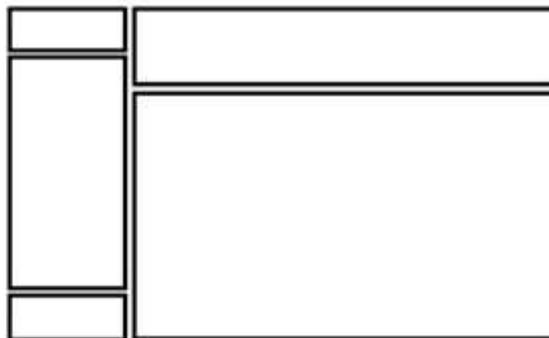
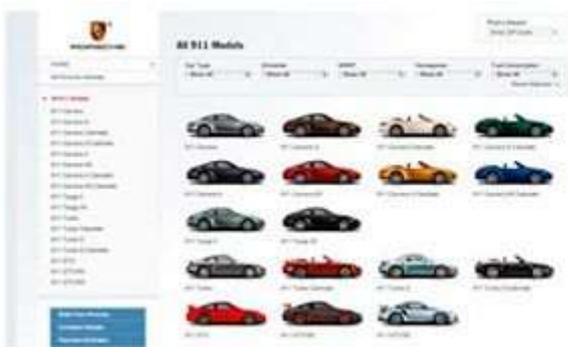
Типовые примеры верхнего колонтитула страницы представлены ниже:



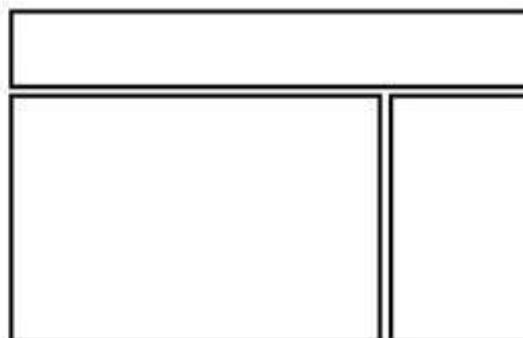
2. Сформировать навигацию для сайта.

Среди всего многообразия составления макета веб-страницы можно выделить четыре наиболее распространённых:

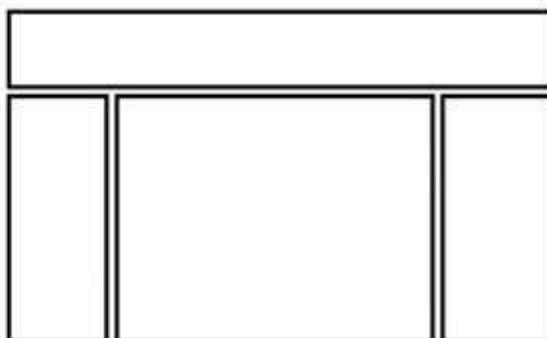
- Навигация в левом столбце



- Навигация в правом столбце



- Навигация в трёх столбцах



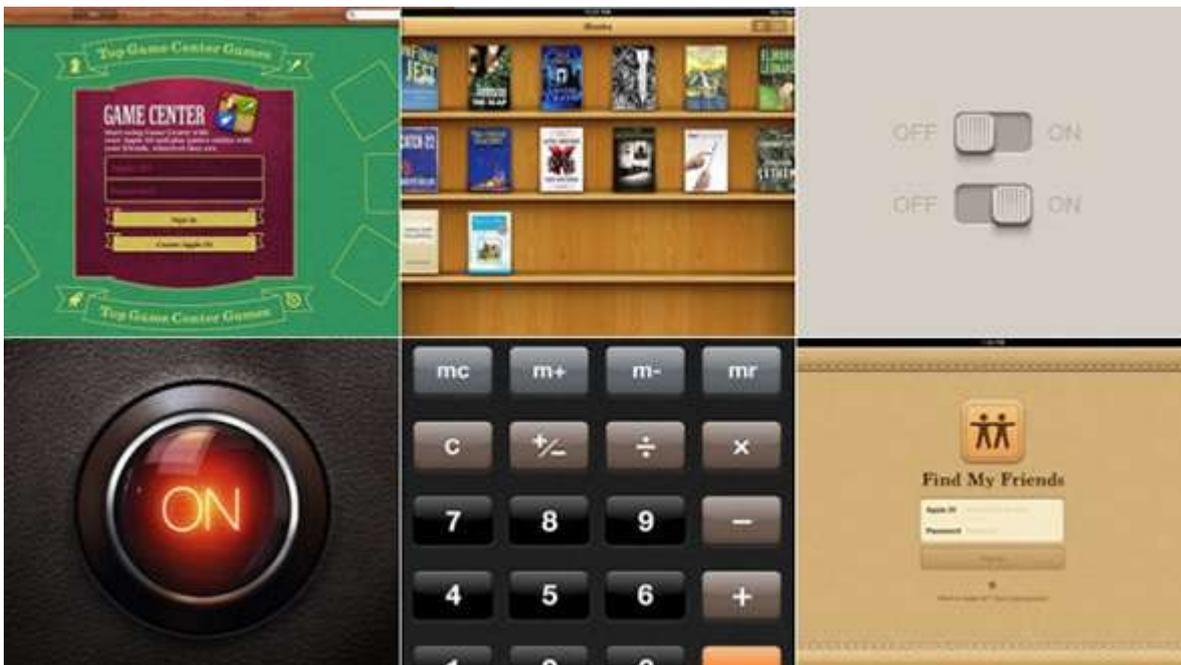
Горизонтальная навигация. На данном этапе сайты с таким типом навигации составляют большинство. Удобство такого подхода легко объяснить тем, что в данном случае у нас остаётся больше пространства для контента, составляющего наш сайт.

3. Сформировать нижний колонтитул для страницы.

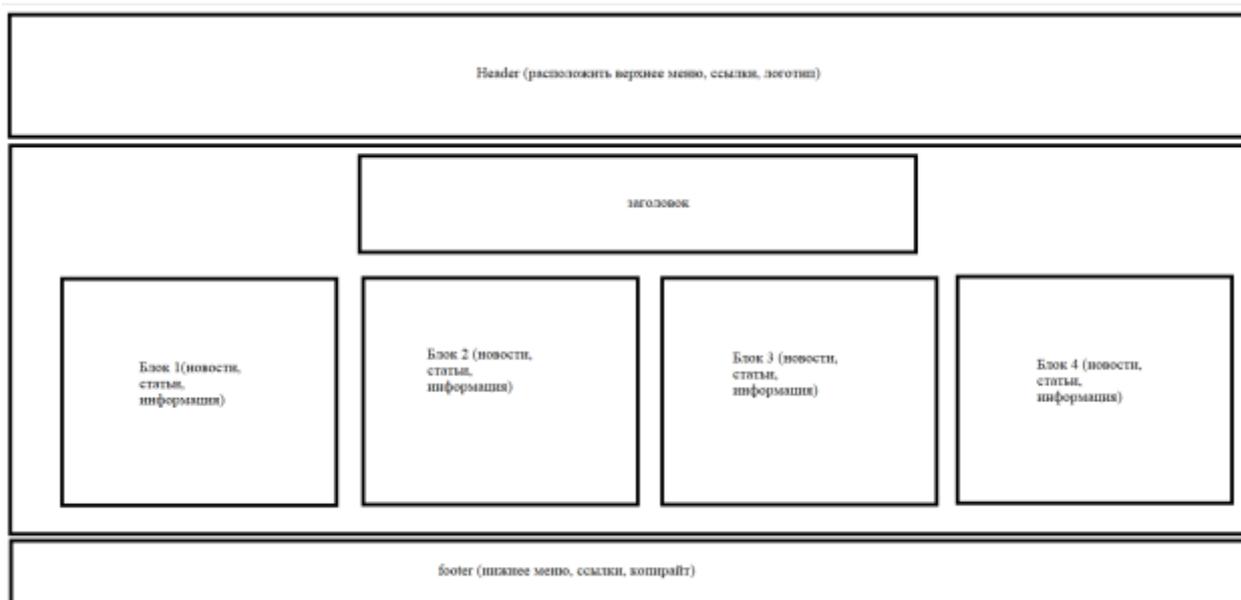
Нельзя не упомянуть о некоторых наметившихся трендах последнего времени в качестве компоновки и дизайна страниц. Во-первых, стоит упомянуть о так называемых лендинговых страницах, которые подразумевают под собой длинную страницу, разделённую на соответствующие секции и знакомящие пользователя с основным контентом сайта. Часто лендинг является единственной страницей, на которой сразу удаётся показать всю необходимую информацию, не заставляя пользователя переходить по страницам. Лендингам обычно сопутствует хороший дизайн, выверенная и продуманная подача информации, элементы call-to-action, интерактивность (счётчики, анимация и пр.).

Стоит так же сказать о внешнем виде страниц и вспомнить такие понятия как скевоморфизм и плоский дизайн.

Скевоморфизм уже продолжительное время уступает свои позиции плоскому дизайну. Данное понятие означает наделение интерактивных элементов качествами реальных. Например, оформление страницы с книгами в виде книжной полки, оформление кнопки на подобии настоящей с соответствующей имитацией нажатия, использование реальных текстур и пр. Данный принцип активно использовался при создании страниц буквально несколько лет назад, но затем тенденции сменились и на первый план вышел плоский дизайн (Flat, Material).



4. Финальный эскиз должен представлять из себя набросок будущего сайта с пояснением



**Форма представления результата:**

Отчет по выполненной лабораторной работе.

**Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

## Тема 08.01.02 Web-дизайн

### Лабораторное занятие № 9 Разработка прототипа дизайна веб-приложения

**Цель:** получение практических навыков создания прототипа дизайна сайта

**Выполнив работу, Вы будете:**

**уметь:**

- У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов
- У.5. Учитывать существующие правила корпоративного стиля
- У.02.2. Определять необходимые источники информации
- У.02.5. Выделять наиболее значимое в перечне информации
- У.09.2. Использовать современное программное обеспечение

**Материальное обеспечение:**

Методические указания для выполнения практических работ, текстовый редактор Atom, Sublime, Visual Studio Code, PHPStorm.

**Задание: Создать макета сайта в программе Photoshop**

**Краткие теоретические сведения**

**Adobe Photoshop** – многофункциональный графический редактор, разработанный и распространяемый фирмой Adobe Systems. В основном работает с растровыми изображениями, однако имеет некоторые векторные инструменты.

Программа Adobe Photoshop используется для создания дизайна сайта. Затем этот дизайн нужно будет разрезать на элементы, сверстать из него HTML страницу и превратить в итоге в готовый шаблон для сайта.

Adobe Photoshop – это многофункциональная программа, и в ней можно не только Изготовить план эвакуации из здания при пожаре, но создавать простые и сложные графические элементы для сайта.

Программу Adobe Photoshop можно использовать для того, чтобы нарисовать оригинальный логотип и любые эксклюзивные детали сайта, поэтому нарисованный в программе шаблон сайта будет уникальным и не иметь аналогов. Именно они задают настроение вашему сайту и привлекают к нему внимание посетителей. Создание привлекательных и функциональных веб-макетов – неотъемлемая часть жизни веб-дизайнера.

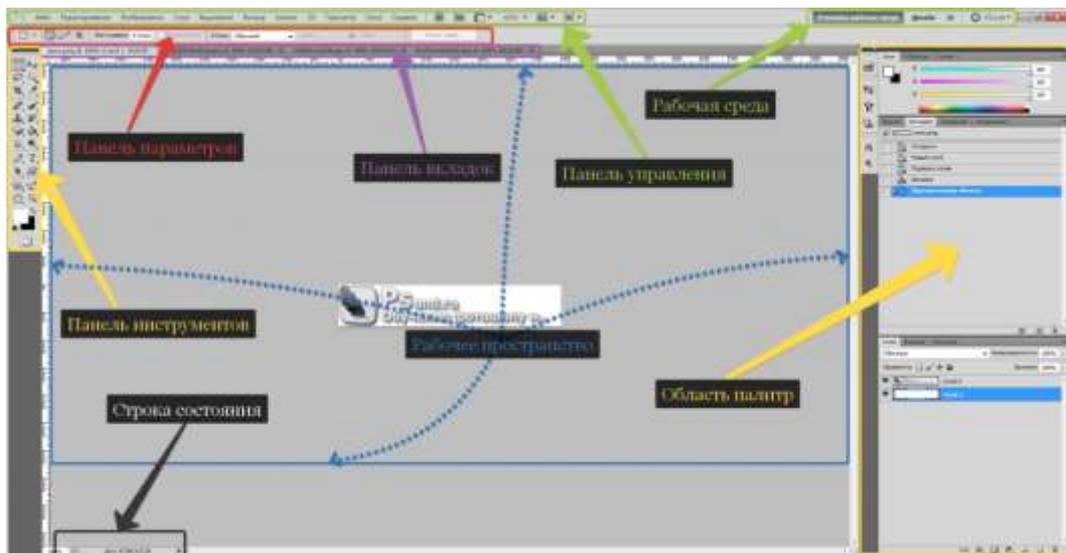


Рисунок 1 – Интерфейс Adobe Photoshop

## Порядок выполнения работы:



Рисунок 2

1. **Открываем Photoshop** и создаём в нём новый документ (**Файл -> Создать** или **Ctrl+N**).

2. **Устанавливаем начальные параметры.** На практике продумать дизайн с точностью до пикселя практически невозможно — в процессе вёрстки обязательно нужно будет что-то поменять, переместить, переделать. Поэтому размеры можно задавать приблизительные, причём ширину и высоту документа желательно указывать заведомо больше планируемых размеров сайта, чтобы в макете точно уместились все элементы. Наш документ сделаем 1000 пикселей шириной и 1500 пикселей высотой (планшет – 768 x 1024 пикселей, смартфон – 320 x 480 пикселей). Обратите внимание: пикселей, а не сантиметров. Другие параметры можно не трогать.

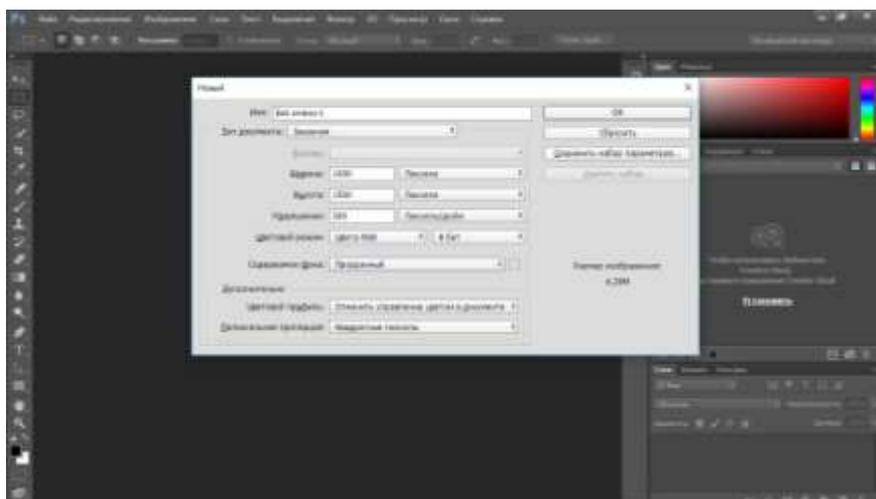


Рисунок 3

3. **Включаем линейки.** Во время работы они понадобятся, ведь линейки позволяют очень точно отмерять расстояния. Проверьте, включены ли линейки у вас. Если да, то вы увидите шкалы рядом с левой и под верхней панелью инструментов.



Если линеек нет — включите их (Просмотр -> Линейки или Ctrl+R).

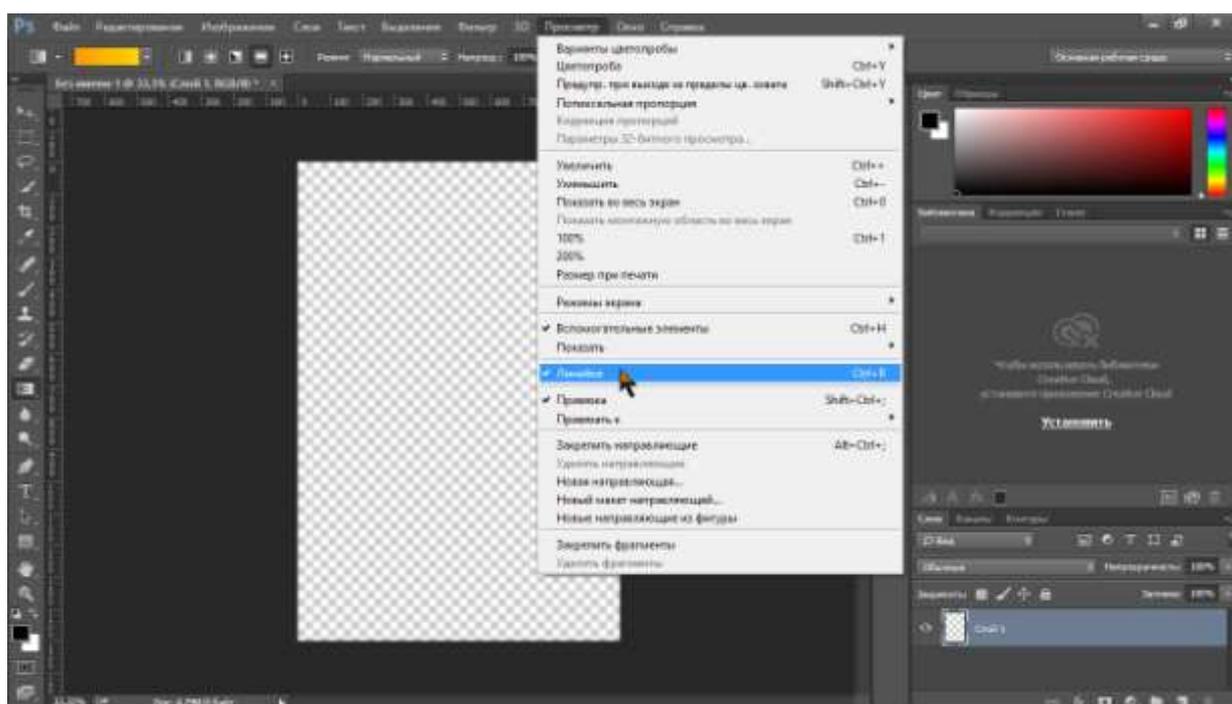


Рисунок 4

Линейки должны показывать величину в пикселях. Чтобы переключиться на них с другой меры длины, щёлкните на линейке правой кнопкой мыши и в открывшемся меню установите соответствующий флажок.

**Очень удобно использовать сетку Bootstrap, в которой заранее заданные размеры колонок, которые можно сразу же использовать, например ширина колонки 140 px**

(посмотреть видеоролик «Как в фотошопе сделать сетку бутстрап (bootstrap grid)» или Открыть Photoshop-Перейти на вкладку операции-Открыть папку 960-Grid-System-master > app\_plugins > photoshop и перенести файл 960\_GRIDS на вкладку Операции-после этого выбрать соответствующую сетку (12,16,24)).

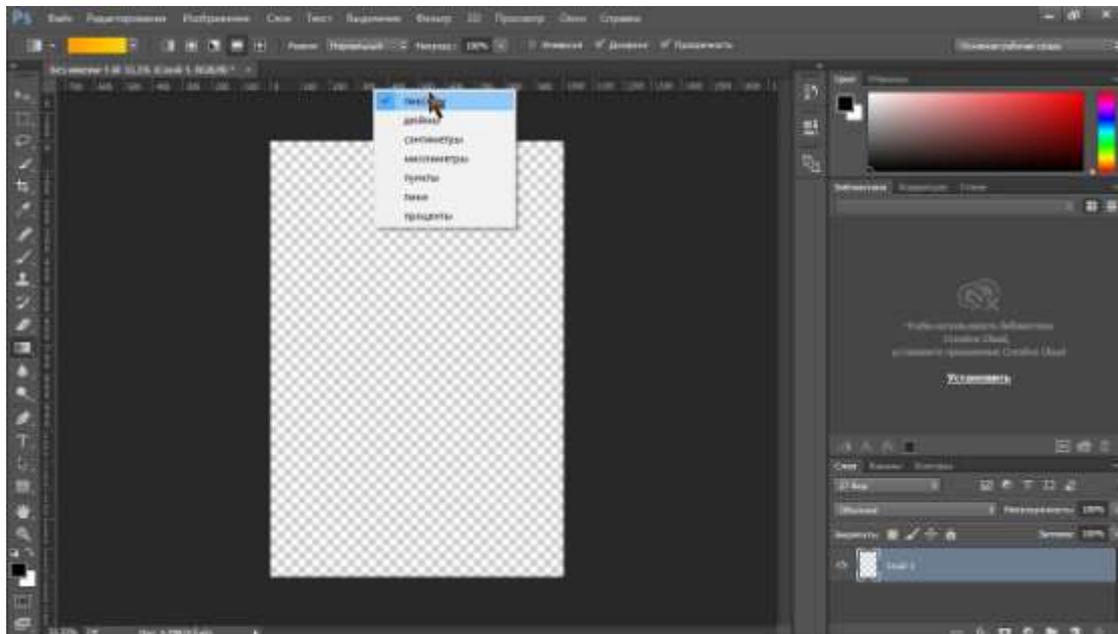


Рисунок 1

4. Проверяем, чтобы размер шрифта был указан в пикселях, а не в пунктах. Если настроено не так, идём в Редактирование -> Настройки -> Основные, в отобразившемся окне переходим на вкладку Единицы измерения и линейки, в выпадающем списке Текст выбираем Пиксели и нажимаем ОК.

5. Сразу делаем подложку сайта. У нас это градиентная заливка оранжевого, переходящего в жёлтый цвет. На панели слева выбираем инструмент Градиент.

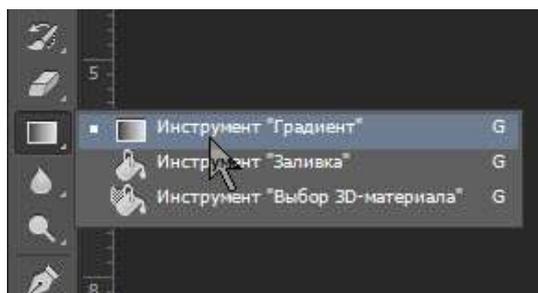


Рисунок 6

На появившейся сверху панели жмём кнопку Зеркальный градиент, выбираем цвет на палитре левее



Рисунок 7

Используя инструменты открывшегося окна **Редактор градиентов**, выбираем нужные цвета. Чтобы задать точный цвет контрольной точки, щёлкните на ней, нажмите кнопку **Цвет** и в окне палитры цветов укажите его в формате RGB, HSB, CSS или любом другом из доступных.

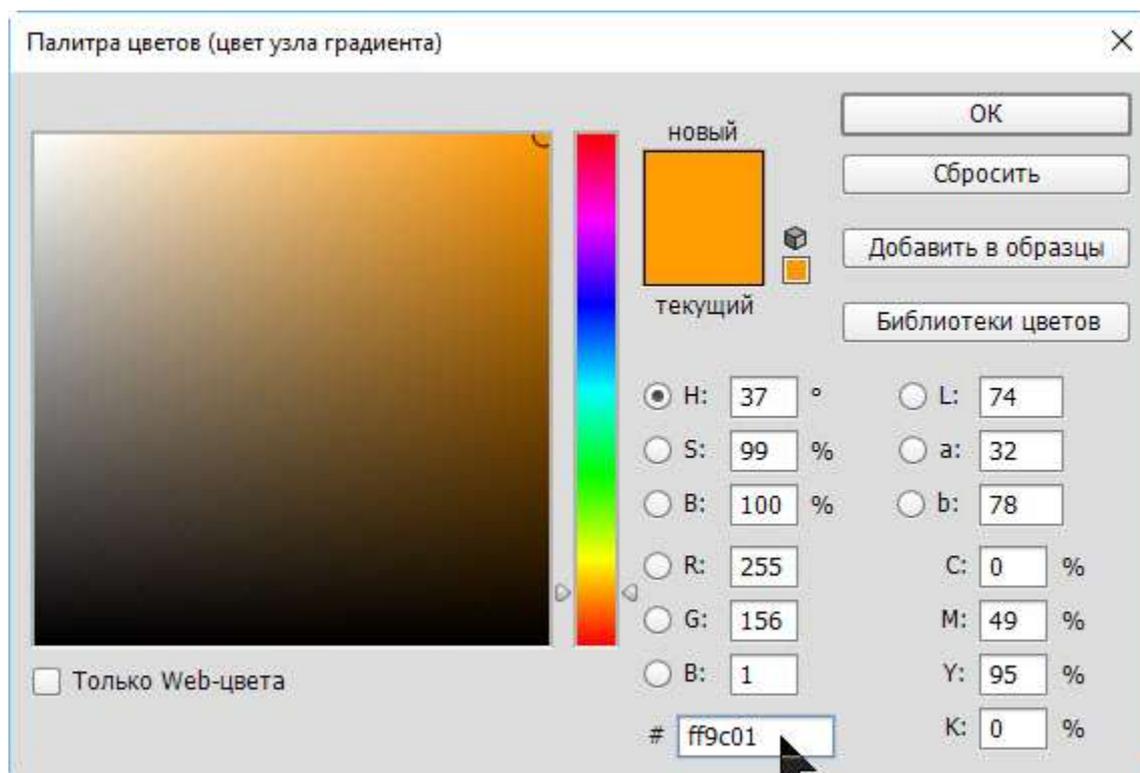


Рисунок 8

В результате манипуляций градиент получился следующим.

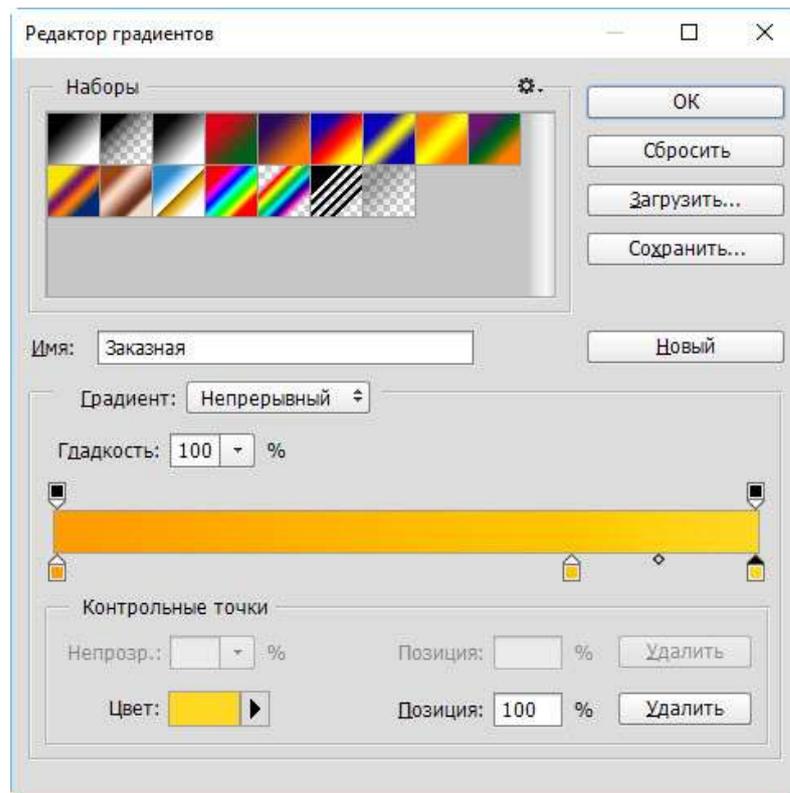


Рисунок 9

Чтобы применить градиент к текущему слою, проводим над ним указателем, удерживая при этом нажатой левую кнопку мыши.

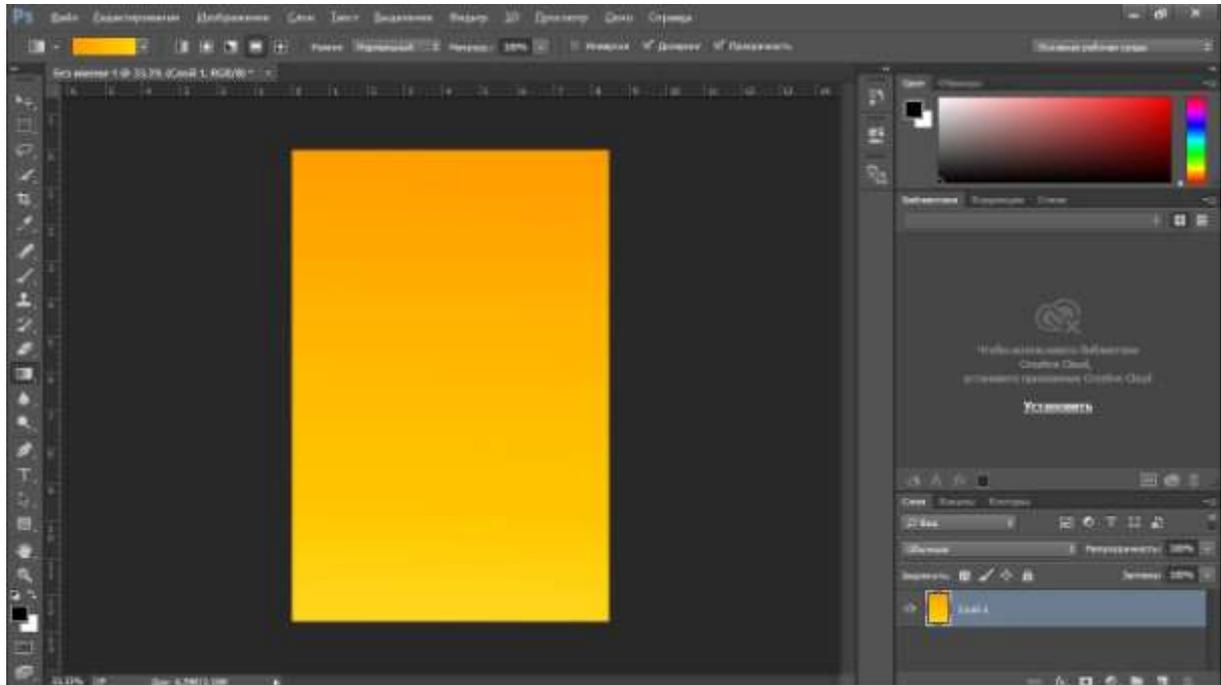


Рисунок 10

6. Сохраняем подложку в файл. То, что мы сделали, должно отображаться под основной страницей и занимать всё окно браузера целиком — своего рода подложка. Например, ширина сайта — 800 пикселей, а разрешение экрана у пользователя гораздо

больше. Оставшееся пространство (всё, кроме тех самых 800px, которые будут заняты блоком страницы) заполнится градиентным фоном.

Так как разрешение экрана нельзя предугадать, из созданного фона можно вырезать полосу толщиной в один пиксель и сохранить как картинку. Браузер будет заполнять задний фон ею по всей ширине.

Сохранить такую узкую полосу несложно.

6.1. Выбираем инструмент **Прямоугольная область**.

6.2. Выделяем полосу произвольной ширины, но по всей длине слоя.

6.3. Копируем выделенную область (**Ctrl+C**).

6.4. Создаём новый документ (**Ctrl+N**), устанавливаем для него ширину 10 пикселей и вставляем скопированное (**Ctrl+V**).

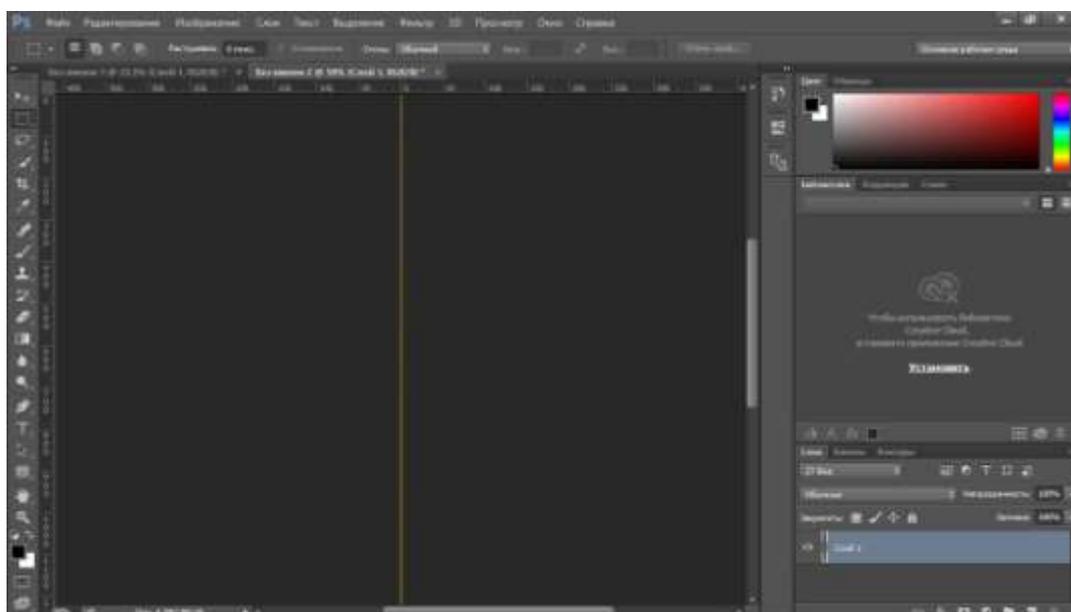


Рисунок 11

6.5. Сохраняем файл в **JPG**-формате.

7. Создаём фон страницы. Фоном будет простой белый цвет. Выбираем инструмент **Прямоугольник** и в окне свойств задаём нужные параметры. У нас получился прямоугольник 800x1100 пикселей, левый верхний угол которого лежит в точке 100,0.

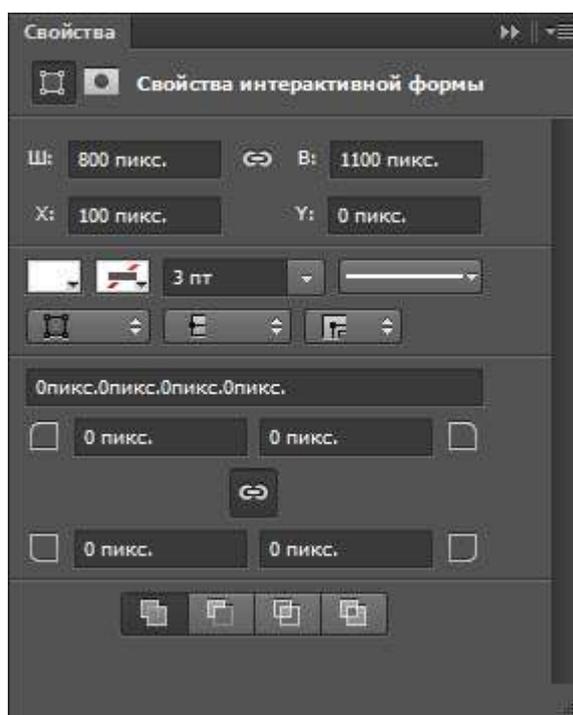


Рисунок 12

8. Делаем фон шапки. Градиентная заливка, похожая на подложку, размерами 780x80px.

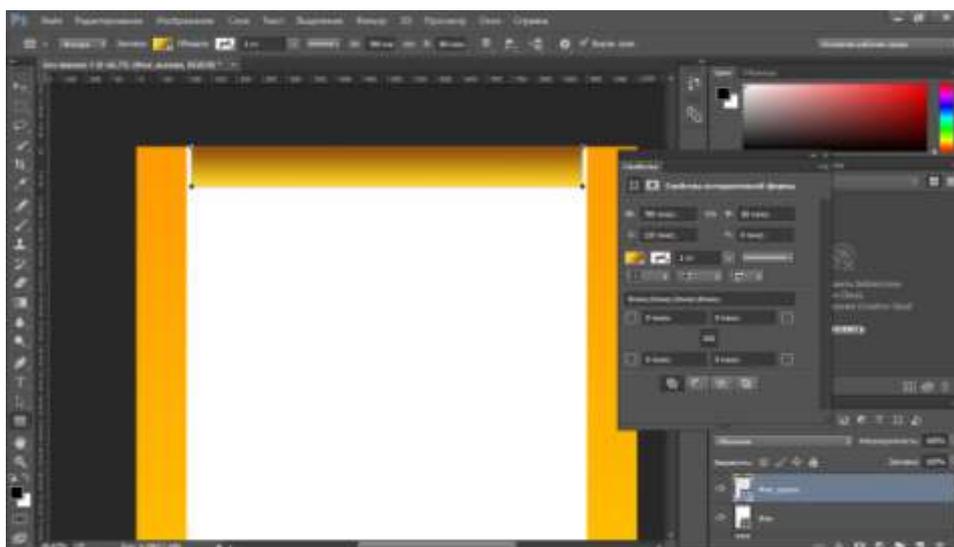


Рисунок 13

Сохраняем его отдельным графическим файлом шириной в 1 пиксель, как делали это с подложкой.

9. Создаём верхнее меню. С помощью инструмента Горизонтальный текст добавляем на макет первый пункт меню — **Главная**. Слой создастся и даже переименуется автоматически, так что с ним можно ничего не делать.

Здесь важно, чтобы все пункты распределились равномерно, поэтому крайне рекомендуется использовать линейки. Чтобы вытянуть вертикальную, проведите указателем,

удерживая при этом нажатой кнопку мыши, слева направо. С помощью линеек отмеряйте расстояние, учитывая, что каждый пункт меню должен занимать в нашем случае 120 пикселей.

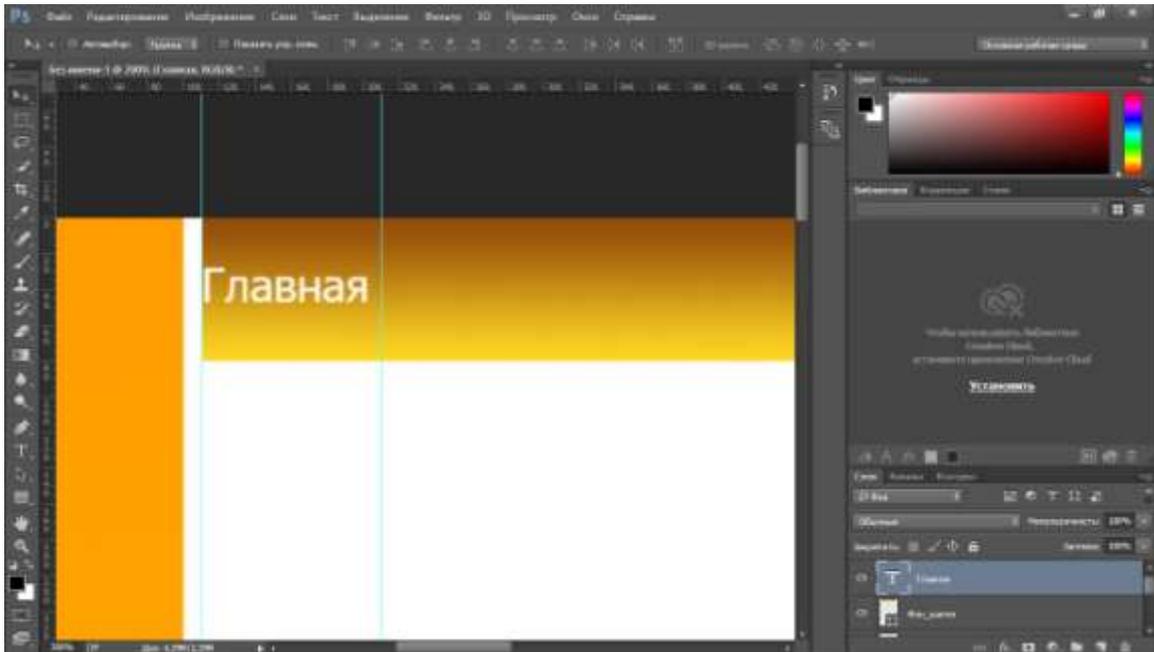


Рисунок 14

10. Аналогично вставляем остальные пункты меню.

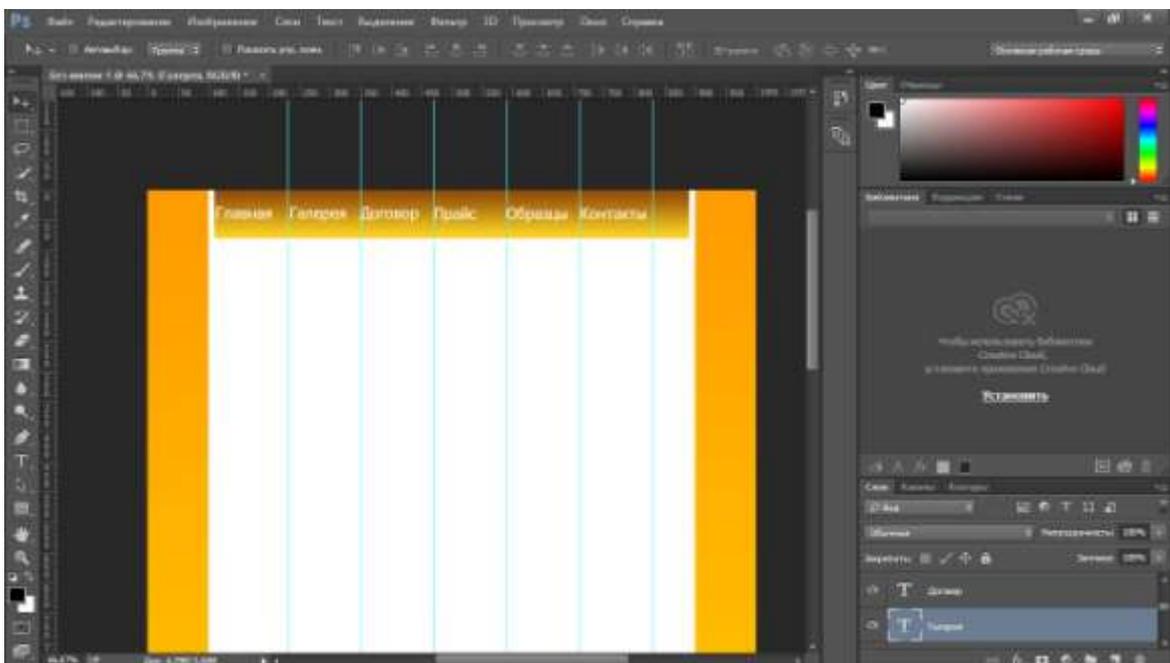


Рисунок 15

11. Добавляем логотип. У нас уже есть готовый, поэтому его остаётся только аккуратно вставить в макет. Для этого нажимаем **Файл -> Открыть**, затем щёлкаем на изображении и, удерживая нажатой кнопку мыши, перемещаем его на заголовок документа-

шаблона, когда он откроется, перетаскиваем картинку в нужное место макета и отпускаем кнопку мыши.

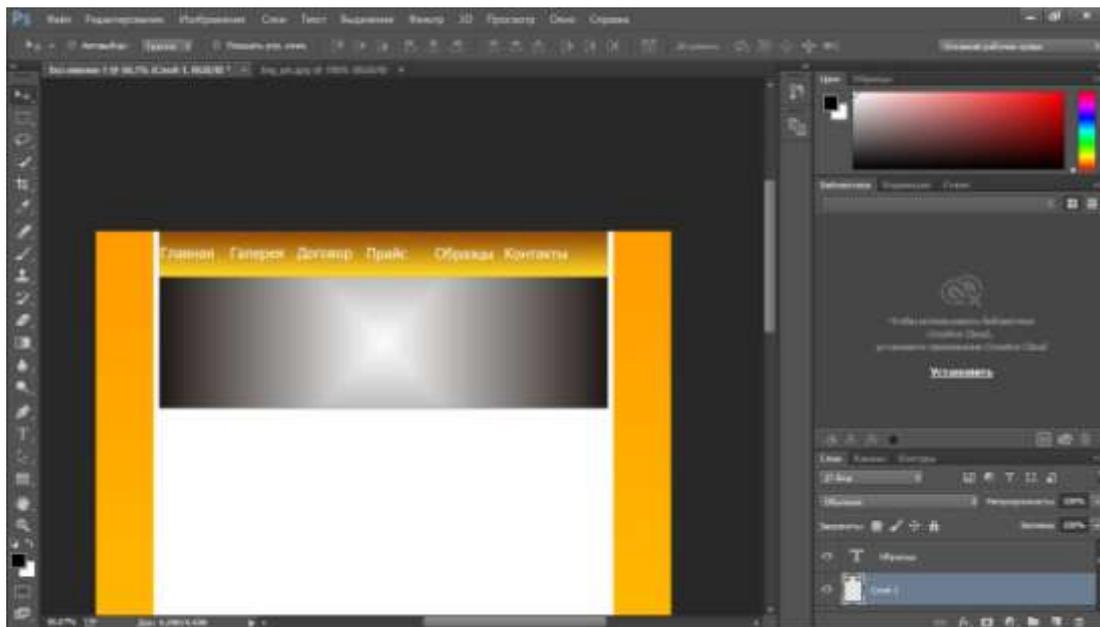


Рисунок 16

12. Пишем имя и слоган сайта. Уже известным нам инструментом **Горизонтальный текст** дополняем уже почти созданный логотип надписями.



Рисунок 17

13. Переходим к боковой панели. Для начала создадим и сохраним отдельным файлом градиентную заливку для её заголовков. С инструментами знакомы, пояснения не требуются.

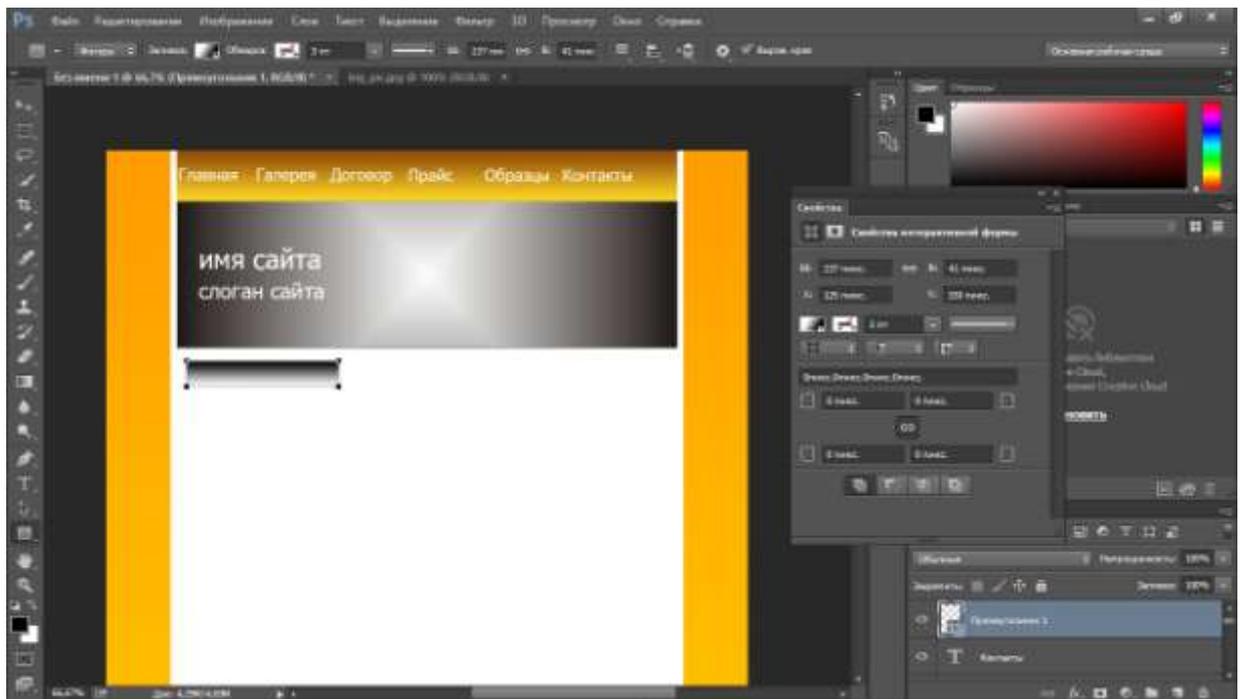


Рисунок 18

14. Добавляем на только что созданный градиент текст-заголовок информационного блока.

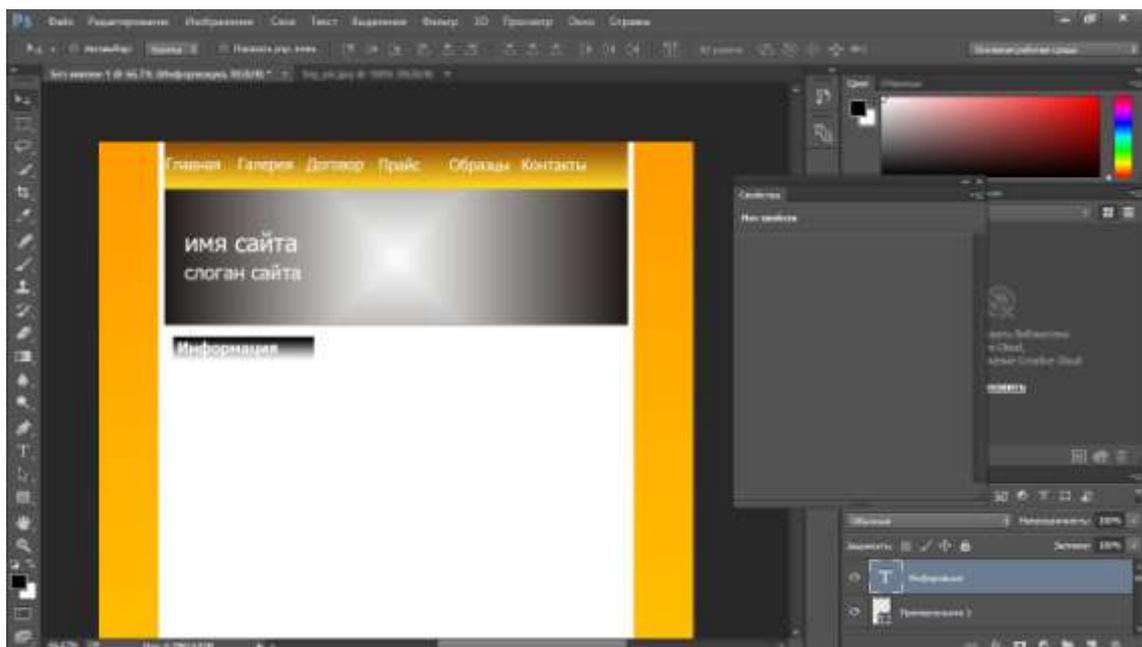


Рисунок 19

15. Рисуем обрамление области. Для этого достаточно использовать прозрачный прямоугольник с чёрными линиями контура. Выбираем инструмент Прямоугольник, задаём тип заливки фигуры Нет цвета, щёлкаем значок Задать тип штриха фигуры и выбираем чёрный цвет, иначе линий не будет. Если контур получился слишком толстым, устанавливаем ширине линий значение 0,5 пт.

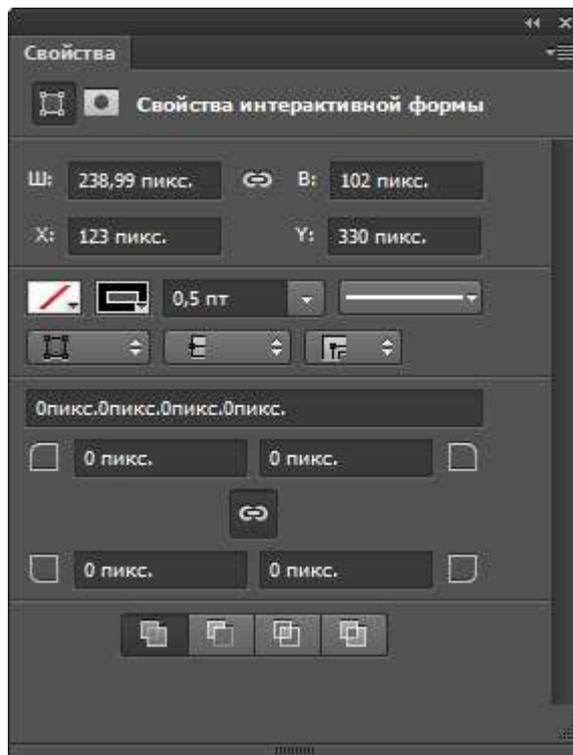


Рисунок 20

16. Ниже добавляем заголовок блока с градиентной заливкой, как в п. 12-13.

17. Создаём блок меню левой панели. Добавляем оранжевый прямоугольник шириной 100px с жёлтым контуром 0,2 пт.



Рисунок 21

18. Добавляем на него текст.



Рисунок 22

19. С помощью дублирования слоёв и линеек создаём ещё пять пунктов меню левой панели.

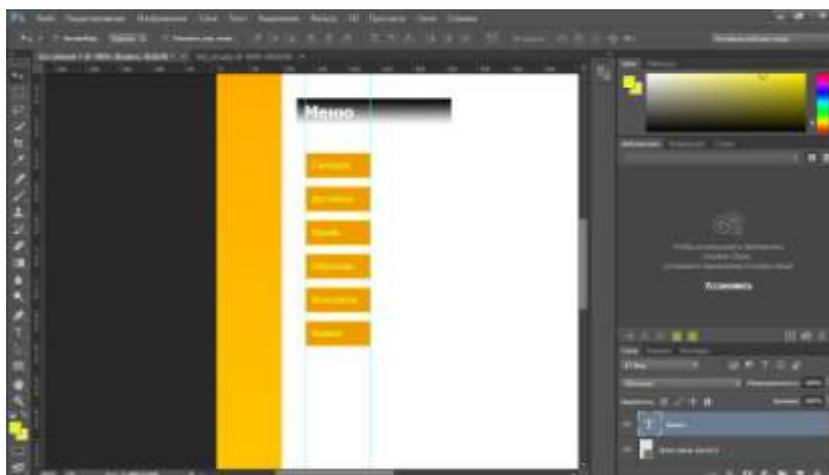


Рисунок 23

20. Прописываем текст в основной части страницы, используя всё тот же инструмент **Горизонтальный текст**.

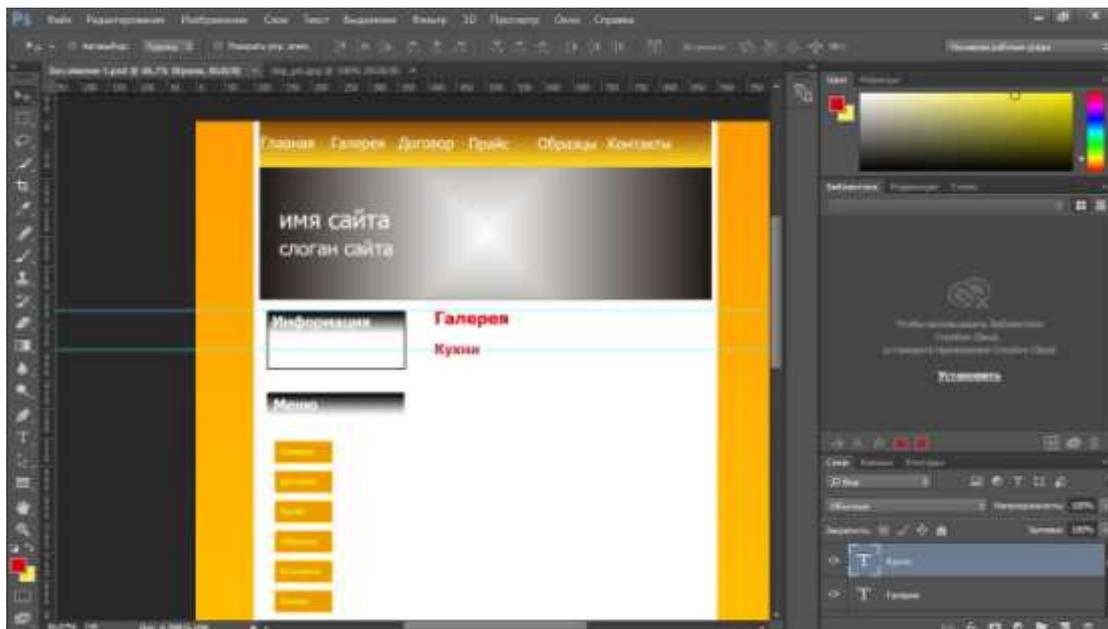


Рисунок 24

21. Добавляем фото в основную часть страницы, как мы делали это с логотипом. Для копирования изображения просто перемещайте его мышью, удерживая при этом нажатой клавишу **Alt**. Если вдруг картинка не подходит по размеру, используйте инструмент Трансформация (**Ctrl+T**).

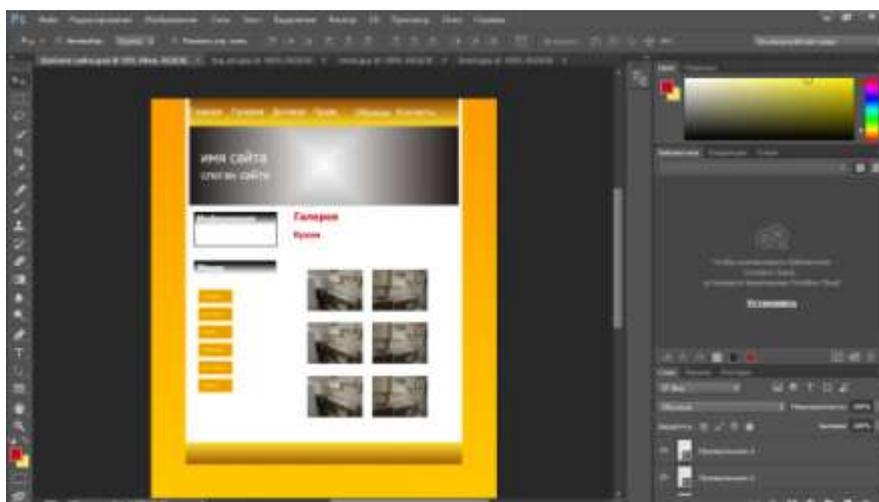


Рисунок 25

22. Рисуем фон нижней части сайта — оранжевый градиент длиной 64 пикселя.

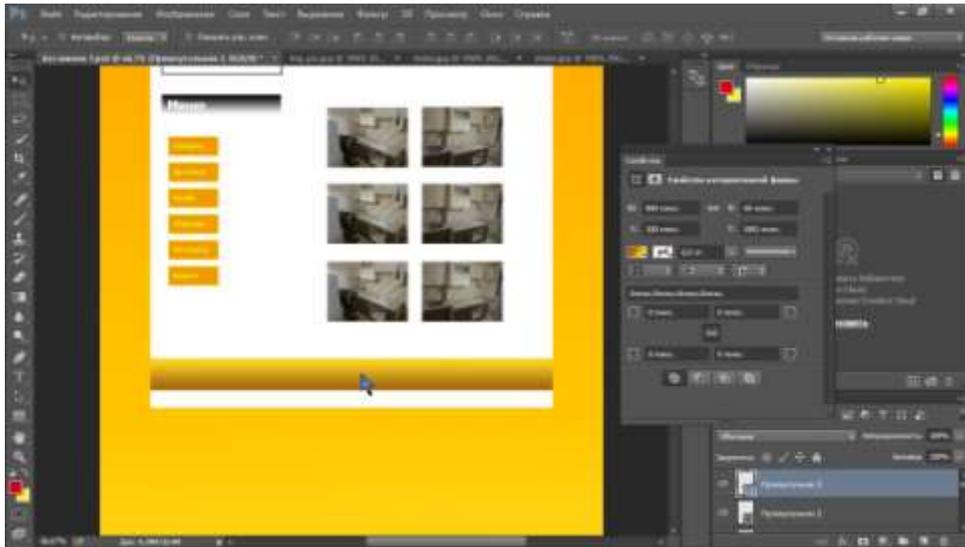


Рисунок 26

23. Сохраняем полосу нижнего фона шириной 1 пиксель в отдельный графический файл.

24. Уменьшаем высоту страницы. Получилось так, что все элементы уже прорисованы, а лишнее место ещё осталось. Вот тут-то нам и пригодились осмысленные имена слоёв. Среди прочих выбираем фоновый слой (у нас он так и называется — Фон) и с помощью инструмента Трансформация уменьшаем высоту нашего белого прямоугольника до нижнего края футера страницы.

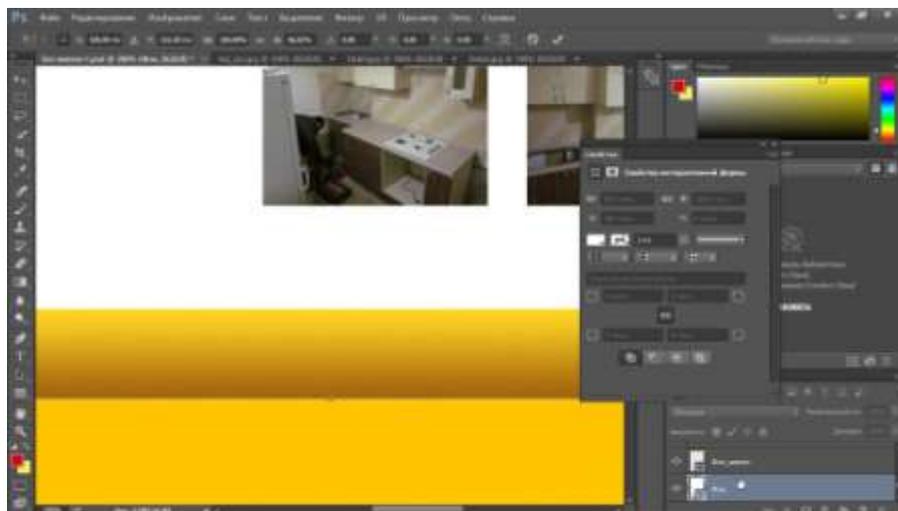


Рисунок 27

25. Сохраняем шаблон в файл формата **.PSD** (Файл -> Сохранить).

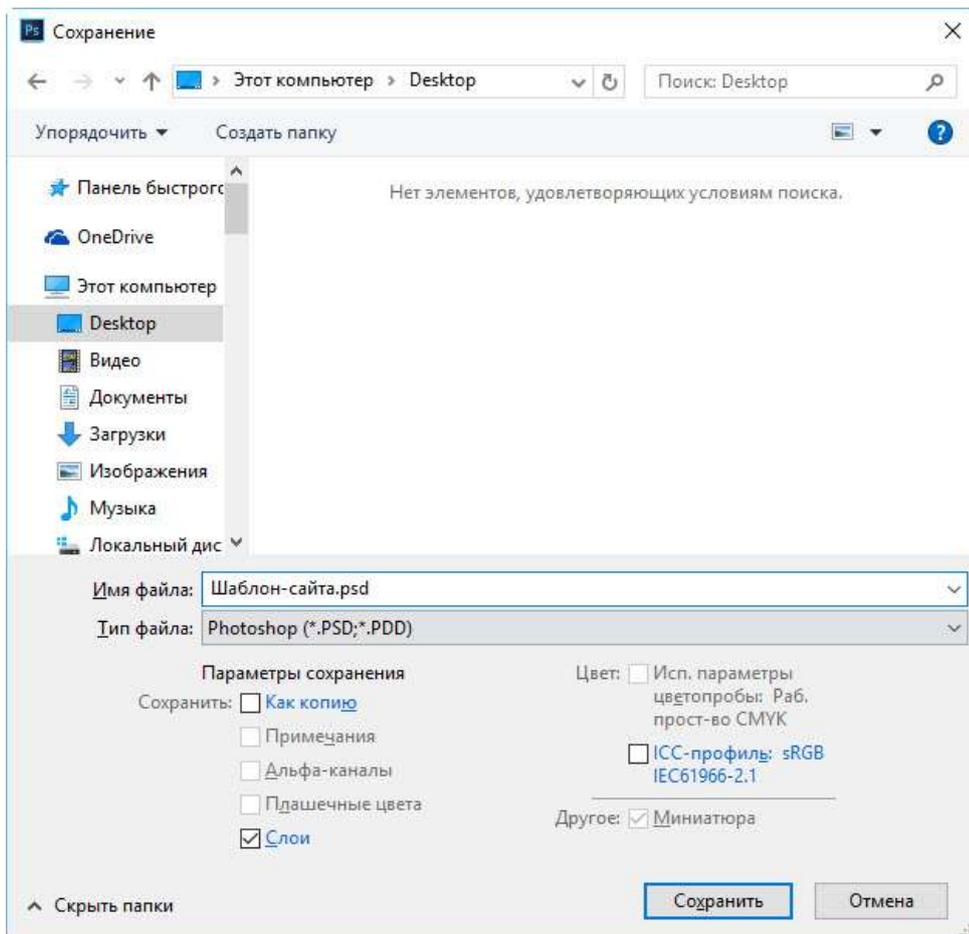


Рисунок 28

26. Результатом сего действия и стал ещё простой, но уже нормально выглядящий шаблон сайта.

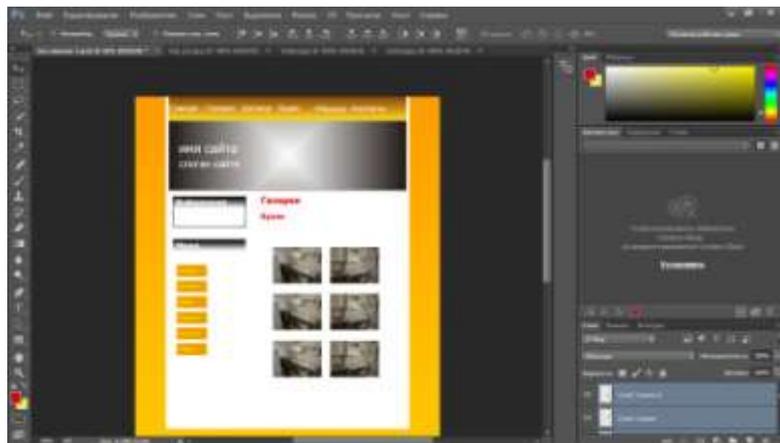


Рисунок 29

Теперь из PSD-макета остаётся средствами HTML/CSS сверстать шаблон сайта.

**Домашнее задание.** Разработать макет собственного блога, опираясь на макет приведенный ниже:

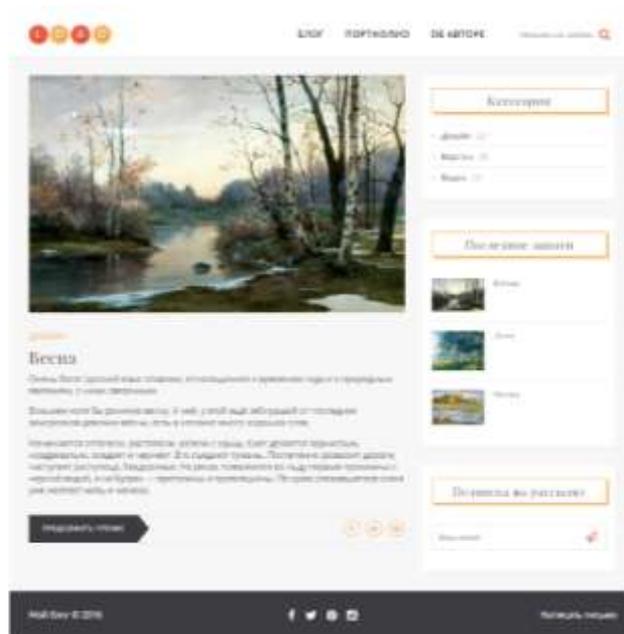


Рисунок 30 – Пример блога

### Контрольные вопросы

- 1 Что такое дизайн-макет сайта?
- 2 На каком из этапов создания сайта происходит разработка дизайн-макета?
- 3 Где создается дизайн-макет сайта?
- 4 Перечислите основные структурные элементы макета?
- 5 Почему так важно использовать сетку при создании макета сайта?

### Форма представления результата:

Отчет по выполненной лабораторной работе.

### Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

## Тема 08.01.02 Web-дизайн

### Лабораторное занятие № 10 Разработка схемы интерфейса веб-приложения

**Цель:** получение практических навыков создания схемы интерфейса сайта

**Выполнив работу, Вы будете:**

**уметь:**

У.4. Разрабатывать интерфейс пользователя для веб-приложений с использованием современных стандартов

У.5. Учитывать существующие правила корпоративного стиля

У.02.2. Определять необходимые источники информации

У.02.5. Выделять наиболее значимое в перечне информации

У.09.2. Использовать современное программное обеспечение

**Материальное обеспечение:**

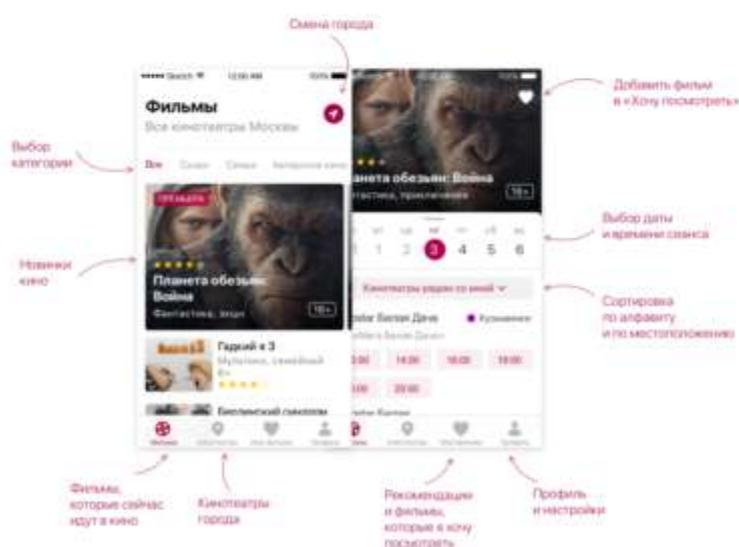
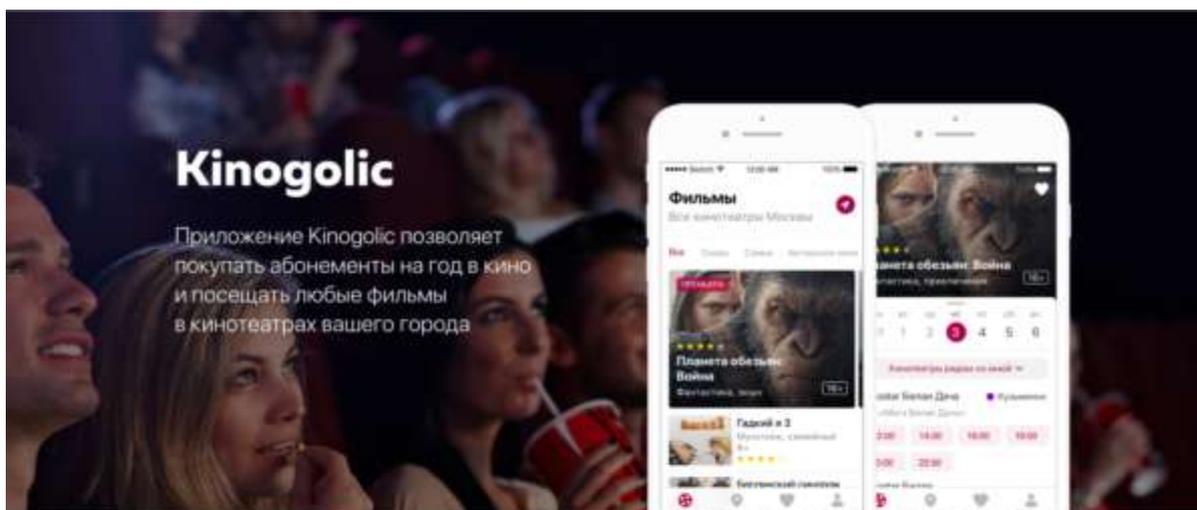
Методические указания для выполнения практических работ, текстовый редактор Atom, Sublime, Visual Studio Code, PHPStorm.

**Задание: Создать схемы интерфейса сайта**

**Порядок выполнения работ:**

1. Необходимо доработать макет сайта, созданного в лабораторной работе 9. Для созданного макета сформировать фирменный стиль, который складывается из цветовой палитры, шрифта, иконок и иллюстраций.

Пример выполненной работы представлен ниже:



**Форма представления результата:**

Отчет по выполненной лабораторной работе.

**Критерии оценки:**

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.