Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСТЕТ»

ДИЗАЙН ЭЛЕКТРОННЫХ И ВЕБ-ИЗДАНИЙ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Учебно-методическое пособие
для студентов специальности 1-47 01 02
«Дизайн электронных и веб-изданий»

Минск 2020

УДК 004.92:004.451.46(076.5)
ББК 32.973.4я73
Д12

Составители :
Н.И. Потапенко, М.Ф. Кудлацкая

Рецензенты :
заведующая кафедрой информационных технологий и моделирования экономических процессов учреждения образования «Белорусский государственный аграрный технический университет» кандидат педагогических наук, доцент *О. Л. Сапун*;
заведующий кафедрой коммуникативного дизайна Белорусского государственного университета кандидат филологических наук, доцент *О. А. Воробьева*

Д12 **Дизайн электронных и веб-изданий. Лабораторный практикум :** учеб.-метод. пособие для студентов специальности 1-47 01 02 «Дизайн электронных и веб-изданий» / сост. : Н. И. Потапенко, М. Ф. Кудлацкая. – Минск : БГТУ, 2020. – 125 с.

В учебно-методическом пособии представлены лабораторные работы в объеме, соответствующем учебному плану специальности, рассчитанные на 5-й и 6-й семестры 3-го курса дневной формы получения образования. Они содержат необходимый теоретический материал, варианты индивидуальных заданий, вопросы для закрепления материала.

Пособие предназначено для студентов специальности 1-47 01 02 «Дизайн электронных и веб-изданий», а также может использоваться студентами заочной формы обучения.

УДК 004.92:004.451.46(076.5)
ББК 32.973.4я73

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Учебная дисциплина «Дизайн электронных и веб-изданий» предназначена для реализации на первой ступени высшего образования. Учебная программа дисциплины разработана в соответствии с образовательным стандартом и учебным планом по специальности 1-47 01 02 «Дизайн электронных и веб-изданий». Данная дисциплина – значимая составляющая профессиональной подготовки дизайнеров-программистов. Она дает будущему специалисту широкий набор практических навыков по определению цели, задач и этапов проектирования веб-изданий, разработке дизайна и реализации веб-проектов, что позволит в дальнейшем эффективно использовать полученные знания на практике.

Цель курса – подготовка специалиста, владеющего фундаментальными знаниями и практическими навыками в области проектирования, разработки дизайна веб-изданий, а также разработки веб-проектов на основе современных веб-технологий с учетом требования дизайна и юзабилити.

Основные задачи курса: овладеть навыками графической деятельности; использовать ведущие термины и понятия современного веб-дизайна и интернет-программирования; изучить веб-технологии для проектирования и разработки дизайн-макетов веб-изданий; освоить веб-технологии для разработки веб-изданий. В результате изучения дисциплины студент должен знать методы проектирования электронных и веб-изданий как информационной системы; принципы применения стилей CSS для форматирования содержания и организации дизайна электронных и веб-изданий; язык JavaScript для организации динамического дизайна и управления содержанием веб-сайта на стороне клиента; программные средства для размещения и сопровождения электронных и веб-изданий; методы оптимизации электронных и веб-изданий для продвижения в сети интернет.

В учебно-методическом пособии представлены лабораторные работы, необходимые для формирования практических навыков в сфере веб-дизайна.

Лабораторная работа № 1

СОСТАВЛЕНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ НА РАЗРАБОТКУ САЙТА

Цель работы: ознакомиться с тематикой сайта по варианту и составить техническое задание на разработку сайта.

ТЕОРИЯ

Техническое задание полностью определяет, каким будет ваш будущий сайт. Оно представляет собой набор четких детальных инструкций, удобных в применении и полностью описывающих необходимые требования.

Грамотно составленное техническое задание должно раскрывать следующие аспекты разработки сайта:

– название сайта, компании, слоган, основание для разработки;

– назначение сайта, цель создания, потенциальная целевая аудитория, сущности и их атрибуты, бизнес-задачи сайта, критерии успешности сайта (количественные);

– технические особенности проекта: адаптивность, кросс-браузерность, система управления и т. д.;

– тип сайта (сайт-визитка, промосайт, интернет-магазин, портал, каталог, блог, форум, инфосайт, корпоративный сайт или landing page и др.);

– структура сайта (разделы, вложенность страниц), сквозные элементы, уникальные страницы, связи;

– терминология, используемая в процессе разработки;

– прототипы экранов;

– дизайн, логотип, шрифты, модульная сетка, цветовая гамма – их смысловая нагрузка и целесообразность выбора;

– верстка (описание поведения элементов форм и других динамических объектов на сайте);

– функционал сайта, технологии, используемые для разработки;

– описание внутренних страниц сайта;

– графический и текстовый контент, форматы, объемы текстовой и графической информации, качество графики;

– тестирование сайта (наборы поведенческих действий пользователя);

– размещение сайта на хостинге;

– форма передачи, права;
– сроки и бюджет разработки.

ЗАДАНИЕ

Составить техническое задание по варианту. Варианты задания представлены в таблице.

Варианты задания

№ варианта	Тема
	Банки
	Строительные компании
	Университеты
	Военные организации
	Магазины продуктовые
	Магазины универсальные
	Производство молочной продукции
	Производство мясной продукции
	Школы
	Детские сады
	Ветеринарные клиники
	Рестораны
	Доставка пиццы
	Магазины бытовой техники
	Клубы по интересам

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– титульный лист;
– название и цель лабораторной работы;
– техническое задание.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое техническое задание? Для каких целей оно используется?
2. Кратко перечислите, какую информацию содержит в себе техническое задание.
3. Для решения каких вопросов составляется техническое задание?
4. Надо ли включать терминологический словарь в техническое задание?

Аргументируйте свой ответ.

Лабораторная работа № 2

СРАВНИТЕЛЬНЫЙ АНАЛИЗ САЙТОВ

Цель работы: научиться анализировать сайты определенной тематики, выделять целевые группы посетителей, определять яркие элементы дизайна и функционал.

ТЕОРИЯ

Анализ сайта – это процедура оценки состояния различных аспектов сайта (технических, семантических) и удобства использования (юзабилити). Это серьезная исследовательская работа, которая проводится с целью увеличить посещаемость, продвижение, попасть в топ-выдачу и т. д.

На этапе составления технического задания многие параметры можно не учитывать, а обратить внимания на те, которые станут основой для создания собственного проекта. При проведении анализа необходимо сформулировать критерии, по которым будут оцениваться сайты. Для этого можно рассматривать следующие компоненты и аспекты деятельности сайта.

Сервисы – поиск, форма обратной связи, интерактивный помощник, форум, рассылки, ссылки-баннеры, переходы на другие сервисы или сайты, регистрация и т. д.

Навигация – расположение меню (повторы, доступность, понятность меню).

Логотип – рисунок, текст, комбинированный, узнаваемость и понимание.

Дизайн – стиль дизайна, цветовое решение, типографика.

Графика и анимация – качество графики, синемаграфы, видеофон, абстракция, панорамы, коллажи, интересные баннеры, нестандартные кнопки, слайдеры, параллаксы, микроанимация и пр.

Реализация – построена на CMS, использованы фреймворки, чистый HTML, CSS, JS и пр.

Коэффициент привлекательности – сервис WebScore AI по загруженной странице определяет балл привлекательности сайта по шкале от 0 до 10. Система анализирует страницу по параметрам, собранным в 15 базовых категорий, в том числе: микроразметка и базовое SEO, адаптивность под разные экраны, цвета, шрифты, изображения, структурированность информации.

Корректность отображения в браузерах – можно воспользоваться сервисом http://browsershots.org/ или проверить вручную в основных браузерах (не менее трех).

Корректность отображения при разных разрешениях – в Google Chrome (Настройка и управление Google Chrome/Дополнительные инструменты/ Инструменты разработчика/Из меню Device выбрать нужный элемент).

GTmetrix (https://gtmetrix.com) – инструмент для тестирования производительности и скорости загрузки веб-страниц. Можно остановиться на показателе YSLOW Score, отображающем производительность сайта в сравнении с просмотренными GTmetrix за месяц: зеленая стрелка – выше среднего, красная – ниже среднего.

Скорость загрузки страниц (F) – оранжевый значок с ромбом, означающий, что результат находится в пределах +/– 5% от среднего значения.

Ключевые слова. Известно, что поисковые роботы индексируют сайты по ключевым словам, поэтому важно обратить внимание на формулировки поисковых запросов. Для этого можно воспользоваться сервисом SerpStat (https://serpstat.com/ru) или https://spywords.ru (бесплатная регистрация).

Как правило, результаты анализа удобно представлять в табличной форме, группируя критерии по выделенным группам.

При анализе сайтов не рекомендуется давать личностные оценки типа «сайт имеет устаревший дизайн», «сайт производит гнетущее впечатление», «сайт имеет современный дизайн» и т. п.

ЗАДАНИЕ

Выбрать себе тему для будущей разработки. Тема может быть любой, но этот сайт в дальнейшем будет разрабатываться в различных стилях веб-дизайна.

Например, медицина – сайт медицинского центра; строительство – сайт фирмы, разрабатывающей дизайн и выполняющей строительство дач; образование – сайт музыкальной школы и т. д.

Необходимо сформулировать цель разработки сайта. Для уточнения и раскрытия функционала найти и проанализировать 5 сайтов аналогичной тематики. **Рекомендация**: выбрать веб-ресурсы Беларуси, России, Европы, США для того, чтобы потом можно было сделать выводы о предпочтениях в дизайне.

Составить сводную таблицу описания 5 сайтов со следующими критериями и показателями (показатели могут быть расширены, изменены) (таблица).

Сводная таблица

Характеристика	Сайт				
	1-й	2-й	3-й	4-й	5-й
Визуальный анализ					
Адрес (скриншот главной страницы в приложении)					
Название					
Целевая аудитория					
Сервисы (функционал)					
Навигация					
Логотип и его тип (скриншот)					
Дизайн и его яркие элементы					
Коэффициент привлекательности					
Технологический анализ					
Кросс-браузерность					
Адаптивность					
Производительность (YSlow)					
Скорость загрузки (*F*)					
Ключевые слова					

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:

– титульный лист, название и цель работы;

– сводную таблицу описания дизайна сайтов;

– общие выводы, сделанные в процессе выполнения лабораторной работы.

Отчет сохраняется и демонстрируется в электронном виде.

Лабораторная работа № 3

СОЗДАНИЕ ПРОТОТИПА САЙТА В ПАКЕТЕ AXURE

Цель работы: ознакомиться с пакетом Axure; разработать интерактивные прототипы страниц в программе Axure.

Работа рассчитана на 2 пары.

ТЕОРИЯ

Прототип в веб-дизайне – это схема страницы сайта в виде наброска, эскиза или html-документа, в котором отображены структурные элементы будущего сайта: текстовые и графические блоки, меню, кнопки, формы и др. Он может быть статичным изображением или динамичным.

Прототип сайта используется при разработке дизайна сайтов, чтобы показать, как в дальнейшем будет выглядеть структура сайта, а также в каком месте будут располагаться функциональные блоки.

Основные требования к прототипу:

– эскиз должен быть простым и понятным;

– размеры блоков реальными и соответствовать пропорциям;

– следует создавать несколько макетов с учетом адаптивности для возможности выбрать наиболее функциональный.

Прототип разрабатываем, принимая во внимание размеры устройств и используя модульную сетку.

Плюсы прототипирования:

1) прототип можно быстро прочитать;

2) на эскизе есть возможность наглядно показать, где, что и как будет расположено на будущей веб-странице;

3) экономия времени, так как согласование требований к проекту проще уточнить в ходе проектирования, а не переделки дизайна-макета или сверстанного сайта;

4) прототипирование помогает усилить действие контента. Можно визуально оценить расположение блоков, их видимость и значимость.

Прототип строится на основе модульной сетки. Модульную сетку можно разработать самостоятельно или воспользоваться пакетом Bootstrap. Наиболее популярный вариант создания блоков для 960 или 1200 пикселей тот, в котором ширина будущего контента распределяется по 9 или 12 колонкам.

Как правило, при разработке адаптивных интерфейсов на основе Bootstrap применяют следующие размеры экранов по ширине в пикселях:

– 480 и ниже для смартфонов;
– 768 и выше для планшетов;
– 980 и выше для стандартного экрана;
– 1200 и выше для широкоформатного экрана.

Основной функционал программы Axure RP Pro. Функционал приведен для версии Axure RP Pro 8, в последней доступной версии 9 появился новый интерфейс и возможности.

1. Основные инструменты для создания карты сайта (Sitemap)

Изображение	Описание
	Создает вложенную страницу (Child page)
	Перемещают выбранную страницу вверх или вниз. Работают только со страницами одного уровня. Если надо выделить и переместить вверх или вниз сразу несколько страниц, использовать Shift
	Изменяют уровень вложенности страниц. Стрелка влево выносит выбранные страницы на уровень выше, стрелка вправо подчиняет страницу родительскому элементу, расположенному над ней
	Удаляет страницу вместе с вложенными элементами
	Редактирование страницы, также можно сделать двойной клик по странице

Перечисленные инструменты используются для создания каркаса сайта. Эти же действия можно выполнять с использованием контекстного меню, которое выпадает по клику правой клавишей мыши на элементе (если кликнуть правой кнопкой мыши в свободной области, будет доступна только функция добавления новой страницы). По умолчанию открывается проект с тремя пустыми страницами. При создании каркаса рекомендуется сразу же выстраивать структурную схему сайта, используя многоуровневую иерархию, и переименовывать страницы в соответствии с их назначением (рис. 1).

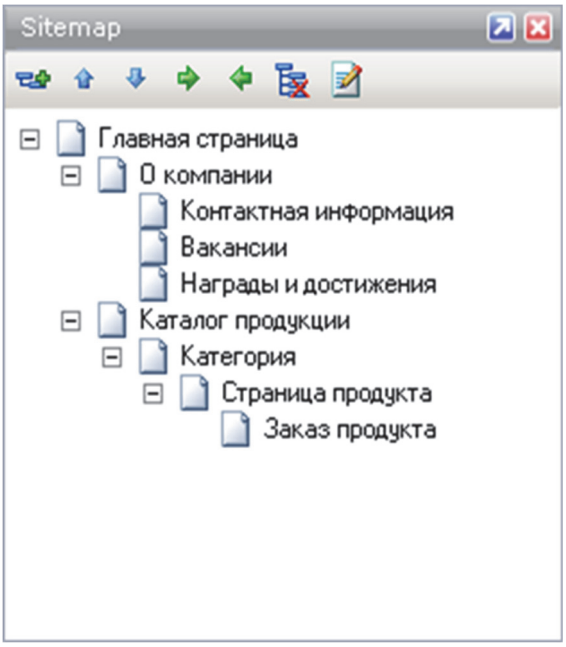


Рис. 1. Карта сайта

2. Панель виджетов. Панель (рис. 2) содержит набор интерфейсных элементов, которые постоянно используются при работе над проектом.

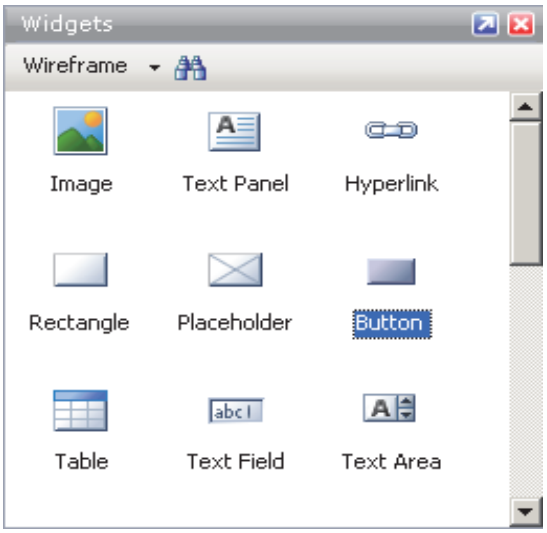


Рис. 2. Панель виджетов

Стандартные библиотеки содержат только самое необходимое: прямоугольники, текстовые панели, кнопки, элементы форм и т. д. В область можно подгружать либо все элементы из всех библиотек одновременно (для этого нужно выбрать All libraries), либо только ту библиотеку элементов, которая нужна в настоящий момент. Библиотеки элементов можно создавать самостоятельно. Для размещения элемента

на странице достаточно его туда перетащить. Основная библиотека имеет название Wireframe. В таблице приведены описания основных виджетов.

Виджеты

Виджет	Описание
 Image	Заглушка для изображения. Стандартный размер – 50×50 px
 Text Panel	Текстовое поле (100×16 px). По умолчанию используется Arial, 10, черный цвет
 Hyperlink	Гиперссылка. По умолчанию используется Arial, 10, синий цвет + подчеркивание
 Rectangle	Прямоугольник 180×80 px с белой заливкой и черной рамкой в 1 px
 Placeholder	Плейсхолдер 180×80 px, используется для указания текстового или графического блока, рамка и диагонали – черные, 1 px
 Button	Кнопка 100×25 px
 Table	Таблица. По умолчанию 3×3
 Text Field	Поле для ввода текста (одна строка)
 Text Area	Область для ввода текста (любое количество строк и столбцов)

Продолжение таблицы

 Droplist	Выпадающий список
 List Box	Многострочный список
 Checkbox	Чекбокс
 Radio Button	Радиокнопка
Horizontal Line	Горизонтальная линия
Vertical Line	Вертикальная линия
 Button Shape	Кнопка со скругленными углами, может превратиться в прямоугольник или квадрат. Радиус скругления можно задавать
 Image Map Region	Область наложения для изображений
 Inline Frame	Фрейм
 Dynamic Panel	Динамическая панель. Может использоваться для проставления активности пунктов меню на страницах
 Menu (Vertical)	Вертикальное многоуровневое выпадающее меню

Окончание таблицы

 Menu (Horizontal)	Горизонтальное многоуровневое выпадающее меню
 Tree	Раскрывающийся список

3. Панель мастеров (Masters). Предназначена для сохранения многократно повторяющихся элементов сайта (хедер, футер, типовая страница).

По умолчанию мастер-панель не создается. Для создания мастер-панели необходимо:

– создать панель – кнопка AddMaster (). Двойной клик по созданной мастер-панели откроет в рабочей области вкладку, на которой можно редактировать содержимое (рис. 3);

– редактирование мастер-панели включает добавление на нее необходимых элементов;

– правой кнопкой мыши по названию мастер-панели вызывается контекстное меню, кнопка Add To Pages добавляет страницы;

– при необходимости выводить мастер-панель на всех страницах можно воспользоваться кнопкой Check All;

– позиционирование панели определяется как Place in background – мастер-панель сохранит то же самое расположение, в котором она выполнена. Specify Location и задать левый и верхний отступы, мастер-панель займет указанное положение на страницах;

– удаление мастер-панели (сначала отменить ее размещение на страницах прототипа, а только потом удалять).

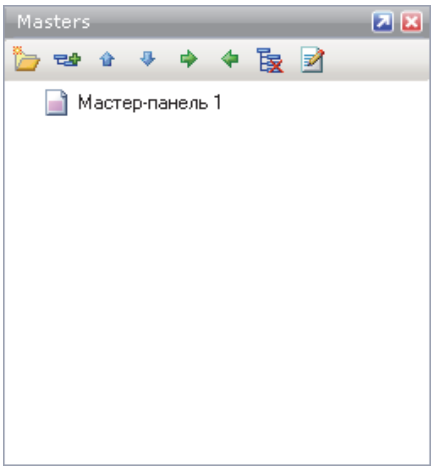


Рис. 3. Мастер-панель

Ознакомиться с процессом создания прототипа можно по ссылке: https://skill-box.ru/media/design/prototip_sayta_v_axure/;

https://www.uplab.ru/blog/kak-my-delaem-prototipy-saytov/

ЗАДАНИЕ

Разработать в AXURE прототипы не менее трех страниц одного из сайтов, участвующих в аналитическом обзоре для разрешений экрана: 1200 и 480 px.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение понятию «прототип».
2. Перечислите основные требования к прототипу.
3. Чем отличается статический прототип от интерактивного?
4. Перечислите инструменты для создания прототипов.
5. Назовите основные преимущества и недостатки пакета Axure.

Лабораторная работа № 4

СОЗДАНИЕ ПРОТОТИПА САЙТА В FIGMA

Цель работы: ознакомиться с основным функционалом пакета Figma; разработать прототип сайта.

ТЕОРИЯ

Figma – кросс-платформенный онлайн-сервис для создания прототипов и дизайн-макетов. Figma работает как онлайн-приложение, так и десктопное. Пакет условно бесплатный. Бесплатная версия предоставляет командную работу не более чем для двух дизайнеров, не более трех проектов, история версий хранится не дольше месяца (30 дней). Рекомендуется сохранять прототипы и макеты в форматах, совместимых с просмотром в других пакетах.

Особенности пакета:

– создание макета онлайн в браузере;

– возможность совместной работы над проектами;

– документирование;

– элемент *Компоненты* позволяет задавать общие стили, при изменении одного компонента происходит автоматическое изменение во всем проекте;

– история версий позволяет просматривать версии файла и восстанавливать или дублировать любую из них. Версии автоматически сохраняются, если в течение 30 мин в файле не было никаких изменений. В истории версий сохраняется автор изменений, время и т. д.;

– элемент *Фреймы* задает предустановленные размеры устройств: Phones, Tablet, Desktop, Watch, Paper, Social Media;

– панель *Grid Layout* создает сетку. Количество сеток на одном проекте не имеет ограничений, сетки могут быть фиксированной ширины, резиновыми, полноэкранными.

Более подробно о Figma по ссылке: https://skillbox.ru/media/design/razbiraemsya_v_zakonakh_kompozitsii_za_10_minut/

ЗАДАНИЕ

Разработать прототипы не менее трех страниц сайта из аналитического обзора в пакете Figma. Предусмотреть создание интерактивного прототипа.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите преимущества и недостатки пакета.
2. Как построить интерактивный прототип?

Лабораторная работа № 5

СТРУКТУРНАЯ СХЕМА САЙТА

Цель работы: ознакомиться с программами для визуализации схем; построить структурную схему (ментальную карту) сайта.

ТЕОРИЯ

Структура (схема) сайта – это связь страниц между собой. Правильная, продуманная структура позволяет поисковым роботам быстро совершать обход ресурса, а посетителям легко перемещаться между его страницами. Информационная архитектура сайта раскрывает и описывает контент-стратегию и способы поиска информации на веб-сайте. Контент-стратегия определяет структуру сайта и способы ориентации на ней, планирование продвижения веб-сайта. На уровне веб-сайта или приложения информационная архитектура определяет, какие данные должны быть размещены на каждой странице и как связать страницы друг с другом.

На данном этапе необходимо составить описание для каждой страницы. Для этого определяется:

– название;
– цели, которые она выполняет;
– информация, которая будет на ней присутствовать;
– функциональность;
– точки входа на страницу и выхода из нее.

Далее определяется модель организации контента или, другими словами, структура сайта (ментальная карта).

Ментальные карты – это техника визуализации информационной архитектуры, мышления. Применение ментальных карт очень разнообразно, например, их можно использовать для фиксации, понимания и запоминания содержания книги или текста, генерации и записи идеи, для того, чтобы разобраться в новой для себя теме, подготовиться к принятию решения, а также показать взаимное расположение страниц будущего проекта.

Модели структур сайта

Одна страница. Подходит для сайтов с ограниченным содержимым, узкой целью (промо-сайты, суб-сайты больших компаний или персональные страницы).

Плоская структура или линейная. Имеет сквозную навигацию, количество страниц на сайте не более 10 (портфолио агентства, обучающий ресурс – электронный учебник, небольшой интернет-магазин с ограниченным видом товаров или услуг).

Строгая иерархия. Переход на следующую страницу можно сделать только со страницы-родителя (электронные книги, учебники с заданной последовательностью прохождения тем, презентации, портфолио).

Индекс. Структура, похожая на плоскую, однако на главной странице есть список всех страниц (примеры те же, что и для плоской структуры).

Ромашка. Такая структура предполагает возврат на главную страницу с любой целевой страницы (веб-приложения, образовательные сайты).

Многомерная иерархия. Одна из наиболее распространенных структур. Предусматривается наибольшее количество элементов навигации, при которой каждая страница доступна отовсюду.

Примеры схем моделей структур сайта представлены на рис. 1, 2.

Рис. 1. Пример схемы сайта архитектурного агентства

Для построения схемы можно использовать различные **онлайн-инструменты для визуализации схем**:

1. ***Mockplus*** (http://mockplus.com). Располагает большим набором инструментов, содержит шаблоны схем и заготовки под разные устройства, есть библиотека иконок и пр. Можно использовать для проектирования интерфейсов, карт и т. д. Требуется регистрация. Действует триал-версия.

2. ***Cacoo*** (https://cacoo.com/) – универсальный онлайн-инструмент для рисования диаграмм, карт сайтов, создания прототипов, сетевых диаграмм. Есть ограничение на бесплатное использование – 14 дней.

3. ***Coggle*** (https://coggle.it/) – универсальный онлайн-инструмент для построения ментальных кар, схем. Позволяет создавать до 3 диаграмм, загружать изображения, сохраняет историю изменений, поддерживает совместную работу, скачивание в формате PDF или изображений.

4. ***Lovely Charts*** (https://lovelycharts.com/) – приложение для создания диаграмм, таких как блок-схемы, карты сайтов, бизнес-процессы, организационные структуры, прототипы и пр.

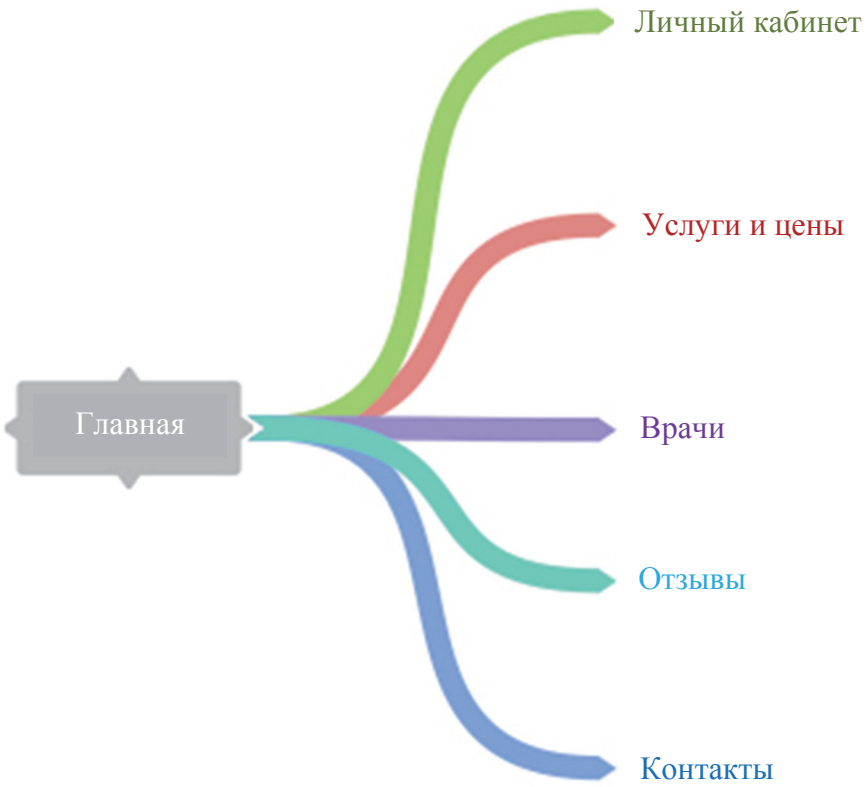


Рис. 2. Пример схемы сайта медицинского центра

5. ***LucidChart*** (https://www.lucidchart.com/) – онлайн-инструмент для создания диаграмм, чертежей, блок-схем, диаграмм идей, сетевых диаграмм, диаграмм UML, прототипов, проектов интерфейса пользователя и др. Имеет много шаблонов.

6. ***Mirosoft Visio*** входит в профессиональную версию Microsoft Office. Предлагает наборы вкладок, ориентированные на определенные группы схемы.

Для создания структурных схем также можно воспользоваться графическими пакетами, такими как Adobe InDesign, OpenOffice Draw, PhotoShop.

ЗАДАНИЕ

– дать краткое описание контента страниц;
– разработать ментальную карту (схему) сайта;
– использовать не менее двух программ визуализации;
– в отчете представить схемы, выполненные в двух программах;
– сделать выводы об удобстве использования программ для построения схем.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите виды структур сайта.
2. Дайте определение понятию «ментальная карта».
3. Перечислите способы создания ментальных карт.
4. Расскажите о возможностях выбранного вами инструмента. Каковы его достоинства и недостатки?

Лабораторная работа № 6

РАЗРАБОТКА ЛОГОТИПА, ПРОТОТИПА И ДИЗАЙНА-МАКЕТА В КЛАССИЧЕСКОМ СТИЛЕ

Цель работы: получить практические навыки по созданию логотипа, написанию гайдлайна, разработке прототипа и дизайна-макета сайта в классическом стиле.

ТЕОРИЯ

Логотип – оригинальное начертание или сокращенное наименование организации, товара, идеи. При создании логотипа (а это одна из сложнейших задач дизайнера) необходимо продумать следующие вопросы:

1. Концепция – это образ, который должен раскрывать суть организации или передавать направление ее деятельности.

2. Ключевая фраза (слоган). Рекомендуется проверить, как эта фраза может быть понята на других языках, так как оригинальное написание на русском, белорусском языках может на иностранном иметь совсем другой смысл.

3. Форма, размер, тип. Выделяют следующие типы логотипов:

Текстовые логотипы:

– аббревиатуры;

– слова (торговые марки).

Для текстовых логотипов необходим качественный, запоминающийся шрифт, цветовая гамма текста. Шрифт должен отражать суть компании, легко читаться, иметь определенное начертание.

Графические логотипы:

– знаки и символы;

– абстрактные;

– персонаж.

Комбинированные логотипы:

– текстово-графические;

– эмблема.

Классический стиль в веб-дизайне предполагает наличие модульной сетки, деление на колонки, использование горизонтальной и/или вертикальной навигации, в хедере (шапке страницы), как правило слева, располагается логотип, посередине – название, слоган (если он

есть), возможно телефоны для связи. Футер содержит данные о копирайте, контактную информацию. Контент распределяется с помощью боковых колонок и блоков. Шрифты используются классические, стандартные, наиболее удобочитаемые, например Tahoma или Arial, темные, как правило, на светлом фоне. Цветовая гамма без ярких или слишком мрачных цветов, с минимальным количеством графики и практически полным отсутствием анимации. Если анимация присутствует, то она должна быть сдержанной и не отвлекать от контента. Контент – грамотный, актуальный.

Гайдлайн – это документ, содержащий инструкции, которые регламентируют правила использования и размещения логотипа, шрифтов, цветовой гаммы на сайте, на элементах айдентики компании, рекламной продукции. Гайдлайн регламентирует цветовую палитру, возможное использование шрифтов, расположение логотипа и его размеров (коробка логотипа).

Прототип – графический эскиз будущего веб-проекта, на котором размечаются блоки расположения контента и компонентов сайта в соответствии с выбранным или заданным стилем.

Дизайн-макет представляет собой скомпонованный макет, в котором уже определены цвета, шрифты, подобран графический контент.

О фирменном стиле. *Фирменный стиль* – это совокупность и смысловое единство постоянных графических и текстовых деталей, посредством которых возможно установить принадлежность того или иного элемента к конкретной компании, т. е. просматривается единство в оформлении сайта, продукции, рекламы, помещений, полиграфии, одежды сотрудников и т. д.

Элементы фирменного стиля:

– цветовая гамма (фирменные цвета);

– товарный знак;

– логотип;

– фирменный блок – традиционное, часто употребляемое сочетание нескольких элементов фирменного стиля;

– слоган – постоянно используемый фирменный оригинальный девиз;

– фирменный комплект шрифтов;

– постоянный коммуникант (лицо, образ компании).

Вышеперечисленные компоненты относят к айдентике компании, организации. Фирменный блок может включать товарный знак, название предприятия, почтовые, банковские реквизиты, перечень товаров и услуг, рекламный символ фирмы, слоган. В него могут входить все перечисленные элементы или только некоторые из них. Классический

стиль – это не фирменный стиль. Фирменный стиль может быть реализован в различных стилях дизайна. Но так как корпоративный стиль разрабатывается в основном для крупных компаний, которые предпочитают классику, то многие иногда (ошибочно) отождествляют эти понятия.

ЗАДАНИЕ

В лабораторной работе № 5 Вы определились с темой будущей разработки, сформулировали цели и составили примерную структуру будущего сайта. В данной работе:

1. Определиться с названием компании, организации и пр. Уточнить цели деятельности организации и цели разработки сайта.
2. Разработать логотип, дать ему обоснование и описание. Рассмотреть варианты создания различных логотипов (предложить как минимум два варианта). Разработать минимальный вариант логотипа, сохранив его узнаваемость.
3. Разработать прототипы не менее двух (главной и типовой, например, страниц) с использованием одного из пакетов прототипирования.
4. Выбрать цвета фирменного стиля и обосновать их.
5. Подобрать шрифты.
6. Разработать слоган компании.
7. Разработать гайдлайн, в котором указать, какие цвета использованы, что означает тот или иной элемент, как можно использовать логотип на различных носителях, описать идею логотипа.
8. Подобрать графические элементы для сайта.
9. Разработать дизайн-макет сайта в классическом стиле.

Полезные ссылки:

Обзоры о дизайне и веб-разработке: logologika.ru, workdone.ru, www.designonstop.com, www.designonstop.com, holmax.ru, www.lookatme.ru, koloro.ua.

Сервисы-построители логотипов: Flickrlogomakr (изготавливает логотипы в стиле Flickr – синий шрифт с красной буквой на конце), Free Logo Generator (позволяет делать логотипы с зеркальным отображением), FREE Logo Creator, CoolText, LogoMaker, LogoSnap, LogoEasy, Logaster.

Типографика: gfto.ru, canva.com, fonts-online.ru, pinterest.ru

Фотостоки: skillbox.ru/media/design/

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:

– прототипы трех страниц;

– логотип в разном исполнении;

– гайдлайн;
– лозунг (слоган);
– цветовую схему;
– шрифты;
– дизайн-макет не менее трех страниц.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие типы логотипов вы можете назвать?
2. Есть ли зависимость между стилем веб-дизайна и выбором типа логотипа?
3. Что включает понятие «айдентика» компании?
4. Можно ли утверждать, что фирменный и классический стили это одно и то же?
5. Что включает в себя гайдлайн и для чего он разрабатывается?
6. Перечислите основные характеристики классического стиля в веб-дизайне.

Лабораторная работа № 7

РАЗРАБОТКА ПРОТОТИПА И ДИЗАЙНА-МАКЕТА В СТИЛЕ МИНИМАЛИЗМ

Цель работы: изучить принципы стиля минимализм и разработать прототипы и дизайн-макет сайта в данном стиле.

ТЕОРИЯ

Минималистический веб-дизайн (minimalist web-design) – это дизайн, для которого характерно упрощение интерфейса путем исключения элементов или контента, не требующихся для решения задач пользователя.

Основные характеристики стиля минимализм:

– паттерны и текстуры. Используются для создания визуального интереса или привлечения внимания пользователя без добавления дополнительных элементов дизайна либо графики. Чем меньше визуальной информации применяется, тем большее воздействие имеет цвет. При использовании ограниченной цветовой палитры важно учитывать два основных момента: цветовая схема должна обеспечивать достаточный контраст и быть различима для людей со слабым зрением или дальтоников. Акцентные цвета используются для выделения особо важной информации или первичных действий;

– использование минимального набора функций и графических элементов для функционала сайта. Элементом считается любая отдельно взятая единица интерфейса, включая такие пункты: меню, ссылки, изображения, графика, линии, заголовки, текстуры, цвета, шрифты, иконки;

– отрицательное пространство (белое, негативное) – это пустые области интерфейса. Грамотное использование отрицательного пространства помогает привлечь внимание пользователя к важному контенту. Необходимо учитывать восприятие иерархии страниц, отображение значимой информации в верхней части страницы и то, как отрицательное пространство будет изменяться при различных разрешениях экрана;

– крупные фоновые изображения или фоновое видео допустимо, главное – основной элемент страницы не должен конфликтовать

с фоном. При этом необходимо учесть следующее: изображения или видео должны служить конкретной цели или помогать пользователям понять сайт, не отвлекать от важного контента, текст должен быть читабелен и понятен; необходимо учесть скорость загрузки «тяжелого» изображения на различных устройствах; включить режим добровольного просмотра видео и звука;

– скрытая глобальная навигация в виде гамбургер-меню. Для сайта с большим количеством контента необходимо продумать данный пункт, и, возможно, в «гамбургер» завернуть «мега-меню», например, как на портале tut.by.

Пример стиля минимализм представлен на рис. 1.

Минимализм

ЗАДАНИЕ

1. Изучить принципы минимализма в веб-дизайне.
2. Разработать не менее двух прототипов страниц сайта в стиле минимализм.
3. Разработать не менее двух дизайнов-макетов сайта в стиле минимализм.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:

– прототипы страниц сайта;

– дизайны-макеты страниц сайта.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите характерные особенности стиля минимализм.
2. Для каких сайтов лучше всего применять данный стиль?
3. Какие типографические решения наиболее предпочтительны для данного стиля? Ответ обоснуйте.
4. Обоснуйте расположение элементов на вашем прототипе сайта и дизайне-макете с точки зрения данного стиля.
5. Допустимо ли размещать текстовые блоки большого размера на макете в стиле «минимализм»?
6. Возможно ли в данном стиле использовать текстовое меню «гамбургер»?
7. Допустимо ли использовать выразительную типографику в негативном пространстве?
8. Возможно ли использовать насыщенные видеофоны?

Лабораторная работа № 8

РАЗРАБОТКА ДИЗАЙНА-МАКЕТА В СТИЛЕ РЕТРО (ВИНТАЖ)

Цель работы: изучить принципы стиля ретро (винтаж) и разработать дизайн-макет сайта в данном стиле.

ТЕОРИЯ

Ретро – термин, объединяющий в себе моду второй половины XIX и первой половины XX в. Элементы дизайна, повторяющие модные направления после 1950-х гг., получили название «винтаж».

Основные элементы для создания ретро и винтажного дизайна:

– иллюстрации старых постеров, киноафиш, фотографии и др.;

– разнообразные старинные и рукописные шрифты;

– использование высококонтрастных неоновых цветов или темных грязных цветов и фактуры (например, старая бумага);

– элементы поп-арта;

– минимум анимации или вообще без нее (хотя, заметим, что элементы анимации и синемаграфы могут присутствовать).

Пример использования ретро-стиля на главной странице сайта представлен на рисунке.

Стиль ретро

ЗАДАНИЕ

1. Изучить принципы разработки сайта в стиле ретро в веб-дизайне.
2. Разработать дизайн-макет не менее двух страниц сайта в стиле ретро.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать дизайн-макет сайта.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите характерные особенности ретро-стиля (винтаж).
2. Для каких сайтов лучше всего применять данный стиль?
3. Какие типографические решения наиболее предпочтительны для данного стиля? Ответ обоснуйте.
4. Обоснуйте расположения элементов на вашем дизайне-макете сайта с точки зрения представленного стиля.

Лабораторная работа № 9

РАЗРАБОТКА ДИЗАЙНА-МАКЕТА В СТИЛЕ МЕТРО (КАРТОЧНЫЙ СТИЛЬ)

Цель работы: изучить принципы разработки сайта в стиле метро и получить практические навыки по созданию дизайна-макета сайта в данном стиле.

ТЕОРИЯ

Стиль метро в веб-дизайне – это дизайн стиля на основе интерактивных карточек, которые содержат изображения, контент, гиперссылки.

Карточный стиль имеет характерные особенности:

– продуманная модульная блочная сетка. Блоки могут быть как одинакового размера, так и различаться по ширине и высоте;

– адаптивность. Дизайн корректно отображается на любых устройствах;

– минималистичность, отсутствие отвлекающих элементов. Все внимание – функциональным карточкам;

– современный и эстетичный внешний вид, широкая палитра используемых цветов;

– применение легкочитаемых шрифтов (рекомендуемый – семейство Segoe);

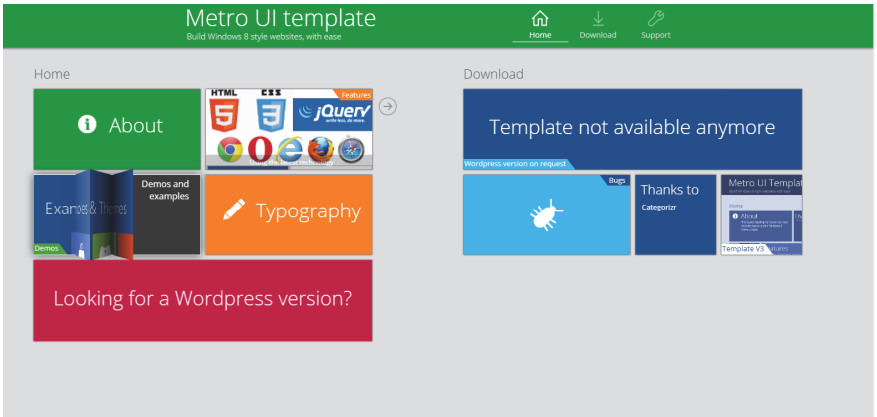
– графика и анимация. Присутствие слайдеров, графических элементов, рисованных элементов (иконок, объемных кнопок, инфографики, анимационных эффектов), которые облегчают восприятие и стимулируют к целевым действиям;

– навигация представлена в нескольких вариантах – традиционное горизонтальное меню и блоки-карты с понятными надписями или изображениями, кликнув на которые осуществляется переход на внутренние страницы сайта. Это увеличивает доступность и понимание функционала для пользователя;

Пример использования стиля метро (карточного стиля) представлен на рисунке.

ЗАДАНИЕ

1. Изучить принципы стиля метро в веб-дизайне.
2. Разработать дизайн-макет сайта в стиле метро.
3. Разработать прототип сайта.



Стиль метро

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– прототип главной страницы сайта;
– дизайн-макет сайта (не менее двух страниц).

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите характерные особенности стиля метро.
2. Для каких сайтов лучше всего применять данный стиль?
3. Какие типографические решения наиболее предпочтительны для данного стиля? Ответ обоснуйте.
4. Обоснуйте расположение элементов на макете сайта с точки зрения представленного стиля.

Лабораторная работа № 10

РАЗРАБОТКА ДИЗАЙНА-МАКЕТА В СТИЛЕ ГРАНЖ

Цель работы: изучить принципы стиля гранж в веб-дизайне и получить практические навыки по созданию дизайна-макета сайта в данном стиле.

ТЕОРИЯ

Стиль гранж в веб-дизайне характеризуется тем, что он не следует строгим формам и правилам, является нарочито небрежным с неряшливыми текстурами фона, хаотическим расположением элементов, «грязными» цветами.

Выделяют следующие основные составляющие данного стиля:

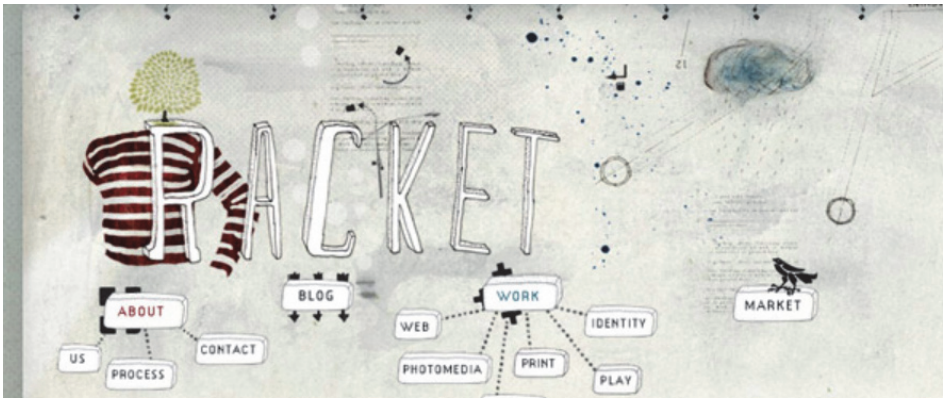
– типичные цвета – бежевый, коричневый, черный и серый. Данный стиль избегает пастельных тонов;

– текстуры и узоры. Для фонов используются различные «неопрятные» визуальные элементы, такие как изображения стальной арматуры, разорванные джинсы, деревянные поверхности, а также поверхности с различными шумами и потертостями;

– реалистичные детали. Это могут быть стикеры, поляроидные снимки, обрывки бумаги, кусочки канцелярского скотча, скрепки, кляксы, пятна, брызги и разводы на поверхности, потертые иконки;

– шрифты. Гранжевый шрифт может иметь потертости, шероховатости, заломы. Он может состоять из неровных и неправильных линий, рукописный.

Пример использования стиля гранж на сайте представлен на рисунке.



Стиль гранж

ЗАДАНИЕ

1. Изучить принципы стиля гранж в дизайне сайтов.
2. Разработать дизайн-макет сайта в стиле гранж.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– прототип сайта;
– дизайн-макет (не менее двух страниц) сайта.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите характерные особенности стиля гранж.
2. Для каких сайтов лучше всего применять данный стиль?
3. Какие типографические решения наиболее предпочтительны для данного стиля? Ответ обоснуйте.
4. Обоснуйте использование элементов на вашем макете сайта с точки зрения представленного стиля.

Лабораторная работа № 11

РАЗРАБОТКА ДИЗАЙНА-МАКЕТА В ПРОМОСТИЛЕ

Цель работы: изучить особенности промостиля и разработать дизайн-макет сайта по выбранной тематике.

ТЕОРИЯ

Промостиль – это рекламный стиль в веб-дизайне, его основная цель – презентация или реклама товара, услуги либо компании. Отличительные особенности сайтов в таком стиле – красочность, обилие графики, большие рекламные слоганы. В целом акцент делается на интересный дизайн. Сайт, выполненный в промостиле, всегда имеет свою изюминку, креатив и яркую индивидуальность. Его легко запомнить и им легко поразить. Такой стиль больше всего подходит для сайтов-визиток, одностраничников и лендинга.

Характерные черты промостиля:

– обилие графики, фото, иллюстрации, эффекты для фона и т. п.;

– использование ярких образов и метафор;

– сочетание рисованных элементов, качественной графики, видеофонов, анимаций. Визуальная графическая составляющая в данном стиле является первичной. Задача промостиля – это воздействие на эмоции, а не на ум пользователей;

– преимущественно рекламное содержание сайта, отсутствие посторонней рекламы.

Пример сайта в стиле промо представлен на рисунке.

Промостиль

ЗАДАНИЕ

1. Изучить принципы промостиля.
2. Разработать дизайн-макет в промостиле (одна страница).

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– титульный лист;
– название и цель работы;
– дизайн-макет сайта;
– описание / перечень элементов, которые относятся к стилю.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите характерные особенности промостиля.
2. Для каких сайтов лучше всего применять данный стиль?
3. Какие типографические решения наиболее предпочтительны для данного стиля? Ответ обоснуйте.
4. Обоснуйте выбор графики и остальных элементов в дизайне-макете.

Лабораторная работа № 12

РАЗРАБОТКА ДИЗАЙНА-МАКЕТА СТРАНИЦЫ 404

Цель работы: получить практические навыки по созданию дизайна-макета страницы 404.

ТЕОРИЯ

Код 404 – один из кодов состояния HTTP, принятых Консорциумом Всемирной паутины (W3C) в 1992 г.

Код состояния HTTP – часть первой строки ответа сервера при запросах по протоколу HTTP. Он представляет собой целое число из трех арабских цифр. Первая цифра указывает на класс состояния. За кодом ответа обычно следует отделенная пробелом поясняющая фраза на английском языке, которая разъясняет причину такого ответа.

404 Page Not Found – Страница 404 «Не найдено»

Клиент узнает по коду ответа о результатах его запроса и определяет, какие действия ему предпринимать дальше. Набор кодов состояния является стандартом. Они описаны в соответствующих документах.

В настоящее время выделено ***пять классов кодов состояния***:

– 1xx: Informational (информационные) – коды, информирующие о процессе передачи;

– 2xx: Success (успешно) – сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента;

– 3xx: Redirection (перенаправление) – коды этого класса сообщают клиенту, что для успешного выполнения операции необходимо сделать другой запрос, как правило, по другому URI;

– 4xx: Client Error (ошибка клиента) – класс кодов 4xx предназначен для указания ошибок со стороны клиента;

– 5xx: Server Error (ошибка сервера) – коды 5xx сообщают об ошибках сервера.

Некоторые ошибки из класса 4xx:

1) 400 Bad Request («плохой, неверный запрос»);
2) 401 Unauthorized («не авторизован»);
3) 402 Payment Required («необходима оплата»);
4) 403 Forbidden («запрещено»);
5) 404 Not Found («не найдено»);
6) 405 Method Not Allowed («метод не поддерживается»);

Причины возникновения ошибки 404:

– был изменен URL;

– страница была удалена;

– ошибка в написании ссылки;

– сайт на данный момент находится офф-лайн;

– сайт больше не существует;

– пользователь стер часть URL для того, чтобы перейти на уровень выше;

– битая ссылка;

– время загрузки страницы превышает 10 с (считается битой) и др.

Задача дизайнера – видоизменить стандартную страницу 404, чтобы она соответствовала дизайну сайта. Дизайн страницы 404 должен быть удобным для пользователя, не отпугивать, а удержать его на сайте. Страница должна содержать полезный контент, который бы предоставил пользователю ответ на вопрос «что мне теперь делать?» и дал простые и ясные инструкции. Не рекомендуется размещать на данной странице рекламу.

При разработке страницы 404 следует учесть такие моменты:

1) не все посетители знают, что такое 404. Предусмотрите фразу, поясняющую, почему посетитель сюда попал и как ему действовать;

2) предложите пользователям проверить написание ссылки. Если они вводили адрес ссылки самостоятельно, то вполне вероятно могли допустить ошибку;

3) придерживайтесь единого стиля сайта;

4) страница ошибки 404 не должна выглядеть как обычная страница. Посетитель должен видеть и понимать, что страница не была найдена и не содержит какого-либо контента, поэтому лишний текст не нужен;

5) юмор и забавная анимация приветствуются, можно использовать некоторые элементы игры;

6) наилучший вариант – размещение одной ссылки, возвращающей на сайт, можно добавить строку поиска, плашку с основным меню.

Пример оформления страницы 404 представлен на рисунке.

Страница 404

ЗАДАНИЕ

1. Ознакомиться с классами кодов состояния HTTP.
2. Изучить рекомендации по разработке дизайна страницы 404.
3. Разработать дизайн-макет страницы 404 для вашего сайта

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– титульный лист;
– название и цель работы;
– дизайн-макет страницы 404 (можно использовать любой стиль из ранее разработанных).

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое код состояния HTTP? Какие классы он в себя включает? Какие ошибки из класса 4xx вы знаете?
2. Какие причины появления ошибки 404 вы знаете?
3. Перечислите основные рекомендации по разработке дизайна для страницы 404.
4. Зависит ли вид страницы 404 от выбранного стиля? Ответ обоснуйте.
5. Какое влияние оказывает ошибка 404 на пользователей?

Лабораторная работа № 13

ВЕРСТКА ДИЗАЙНА-МАКЕТА. КЛАССИЧЕСКИЙ СТИЛЬ. ГОРИЗОНТАЛЬНОЕ И ВЕРТИКАЛЬНОЕ МЕНЮ

Цель работы: научиться использовать блочную, семантическую верстку; получить практические навыки верстки горизонтального и вертикального меню сайта.

ТЕОРИЯ

В HTML5 для навигации используется семантический тег <nav>. Меню, как правило, создается с использованием тега списка <ul>, в котором в каждом пункте <li> содержится одна ссылка <a>.

Пример 1. Вертикальное меню (рис. 1)

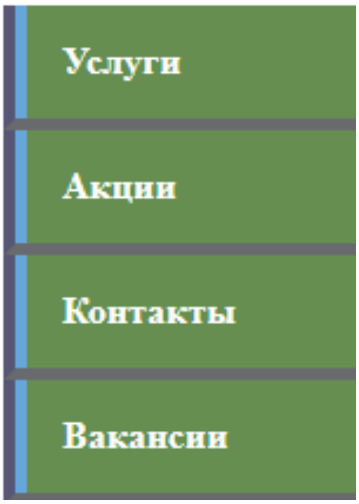


Рис. 1. Вертикальное меню

Алгоритм создания вертикального меню:

Шаг 1. Создать маркированный список, добавив к нему атрибут id с идентификатором "navbar":

```
<ul id="navbar">
    <li><a href="#">Услуги</a></li>
    <li><a href="#">Акции</a></li>
    <li><a href="#">Контакты</a></li>
    <li><a href="#">Вакансии</a></li>
</ul>
```

Шаг 2. Сбросить установленные по умолчанию стили списка и задать требуемую ширину меню:

```
#navbar {
    margin: 0;
    padding: 0;
    list-style-type: none;
    width: 150px;}
```

Шаг 3. Добавить к ссылкам фоновый цвет, а также цвет, размер и насыщенность шрифта. Убрать подчеркивание, добавить небольшие отступы и переопределить отображение элемента <a> со строчного на блочный. Добавить левую и нижнюю рамки к пунктам списка, задать им блочное отображение.

Таким образом, ссылка будет занимать все доступное пространство пунктов списка, т. е. для перехода по ссылке не надо наводить курсор точно на текст.

```
#navbar a {
    background-color: #559444;
    color: #fff;
    padding: 15px;
    text-decoration: none;
    font-weight: bold;
    border-left: 5px solid #55AfFF;
    display: block;
}
```

Шаг 4. Необходимо изменить внешний вид ссылки при наведении на нее курсора. Для этого определим псевдо-класс :hover:

```
#navbar a:hover {
    background-color: #666;
    border-left: 5px solid #3333FF;
}
```

Итоговый вариант

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

```
            #navbar {
                margin: 0;
                padding: 0;
                list-style-type: none;
                width: 150px;
            }
            #navbar li {
                border-left: 5px solid #557;
                border-bottom: 5px solid #666;
            }
            #navbar a {
                background-color: #559444;
                color: #fff;
                padding: 15px;
                text-decoration: none;
                font-weight: bold;
                border-left: 5px solid #55AfFF;
                display: block;
            }
        </style>
    </head>
    <body>
        <ul id="navbar">
            <li><a href="#">Услуги</a></li>
            <li><a href="#">Акции</a></li>
            <li><a href="#">Контакты</a></li>
            <li><a href="#">Вакансии</a></li>
        </ul>
    </body>
</html>
```

Алгоритм создания горизонтального меню

Алгоритм создания горизонтального меню такой же, отличие лишь в третьем шаге: свойству display для элементов <li> надо присвоить значение inline, т. е. сделать его строчным, чтобы пункты списка располагались друг за другом.

```
#navbar li {
    display: inline;
}
```

Пример 2. Выпадающее меню (рис. 2)

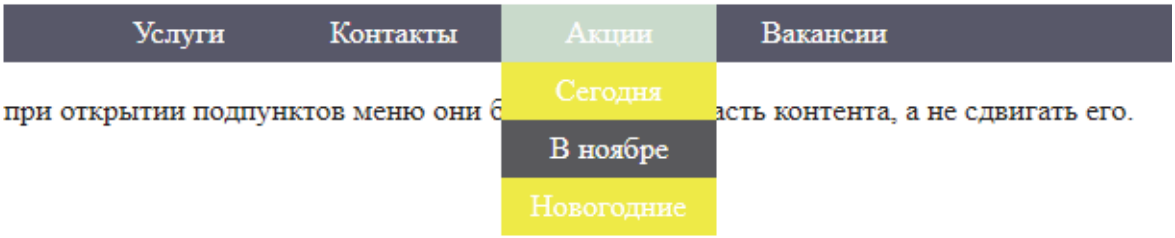


Рис. 2. Выпадающее меню

Алгоритм создания выпадающего меню

Алгоритм такой же, как при создании вертикального меню. Особенность заключается в том, что для создания выпадающего меню используется список в списке: комбинация горизонтального и вертикального меню.

```
<ul id="navbar">
    <li><a href="#">Услуги</a></li>
    <li><a href="#">Контакты</a></li>
    <li><a href="#">Акции</a>
        <ul>
            <li><a href="#">Сегодня</a></li>
            <li><a href="#">В ноябре</a></li>
            <li><a href="#">Новогодние</a></li>
        </ul>
    </li>
    <li><a href="#">Вакансии</a></li></ul>
```

Шаг 1. Скрыть вложенный список с подпунктами, используя display: none;, чтобы они не отображались на веб-странице все время.

Шаг 2. Преобразовать вложенный список в блочный при наведении на пункт меню (элемент <li>):

```
#navbar ul {
    display: none;
}
#navbar li:hover ul {
    display: block;
}
```

Шаг 3. Убирать у обоих списков отступы и маркеры, установленные по умолчанию. Элементы списка с навигационными ссылками сделать плавающими, чтобы сформировалось горизонтальное меню. При этом для элементов списка, содержащих подпункты, задаем float: none;, чтобы они отображались друг под другом.

```
#navbar, #navbar ul {
    margin: 0;
    padding: 0;
    list-style-type: none;}
#navbar li {
    float: left;}
#navbar ul li {
    float: none;}
```

Шаг 4. Необходимо, чтобы выпадающее подменю не смещало контент, расположенный под панелью навигации вниз. Для этого следует задать пунктам списка position: relative; а списку, содержащему подпункты, – position: absolute; добавить свойство top со значением 100%, чтобы абсолютно позиционированное подменю отображалось точно под ссылкой.

```
#navbar ul {
    display: none;
    position: absolute;
    top: 100%;
}
#navbar li {
    float: left;
    position: relative;
}
#navbar {
    height: 30px;
}
```

Высота для родительского списка введена для того, чтобы список не был проигнорирован браузером и контент, следующий за списком, обтекал меню. Браузеры не учитывают в качестве содержимого элемента плавающий контент.

Итоговый вариант:

```
<!DOCTYPE html>
```

```
<html>
    <head>
        <meta charset="utf-8">
        <title>Документ</title>
        <style>
            #navbar ul{
                display: none;
                background-color: #ff0;
                position: absolute;
                top: 100%;
            }
            #navbar li:hover ul {
                display: block;
            }
            #navbar, #navbar ul{
                margin: 0;
                padding: 0;
                list-style-type: none;
            }
            #navbar {
                height: 30px;
                background-color: #556;
                padding-left: 35px;
                min-width: 480px;
            }
            #navbar li {
                float: left;
                position: relative;
                height: 100%;
            }
            #navbar li a {
                display: block;
                padding: 6px;
                width: 100px;
                color: #fff;
                text-decoration: none;
                text-align: center;
            }
            #navbar ul li {
               float: none;
            }
```

```
            #navbar li:hover {
               background-color: #cdc;
            }
            #navbar ul li:hover {
                 background-color: #555;
            }
        </style>
    </head>
    <body>
        <ul id="navbar">
            <li><a href="#">Услуги</a></li>
            <li><a href="#">Контакты</a></li>
            <li><a href="#">Акции</a>
                <ul>
                    <li><a href="#">Сегодня</a></li>
                    <li><a href="#">В ноябре</a></li>
                    <li><a href="#">Новогодние</a></li>
                </ul>
            </li>
            <li><a href="#">Вакансии</a></li>
        </ul>
                    <p>при открытии подпунктов меню они будут скрывать
      часть контента,  а не сдвигать его.</p>
    </body>
</html>
```

Пример 3. Семантическая верстка меню. Семантические теги используются для четкого определения содержимого контента.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="unicod">
        <title>Документ</title>
    </head>
    <body>
        <header>
            <h1>пример</h1>
        </header>
        <nav>
            <a href="page/1.html">Услуги</a>
            <a href="page/2.html">Акции</a>
```

```
            <a href="page/3.html">Контакты</a>
            <a href="page/4.html">Вакансии</a>
        </nav>
        <article>
            <h2>Добро пожаловать!</h2>
        </article>
    </body>
</html>
```

ЗАДАНИЕ

Сверстать сайт в классическом стиле, содержащий минимум три страницы. На сайте должно быть горизонтальное и вертикальное меню. Два пункта меню должны содержать не менее трех пунктов второго уровня. Ссылки (пункты меню) должны быть работающими.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– титульный лист;
– название и цель работы;
– листинги верстки вашего сайта;
– ответы на следующие вопросы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите методы верстки сайта.
2. Какая разница между применением табличной и блочной верстки при отображении контента на странице?
3. Можно ли использовать фреймы при верстке сайта?
4. Дайте определение понятию «навигация на сайте».
5. Приведите алгоритм создания меню.
6. С помощью какого свойства изменить вид меню при наведении на него курсором мыши?
7. Для чего применяют блочное оформление для ссылок?
8. Что надо задать, чтобы список не был проигнорирован браузером и контент, следующий за списком, обтекал меню?
9. Почему мы сбрасываем свойства списка по умолчанию при использовании его в меню?
10. Для чего задаем пунктам списка position: relative?

Лабораторная работа № 14

ВЕРСТКА САЙТА В СТИЛЕ МИНИМАЛИЗМ. АДАПТИВНОЕ МЕНЮ

Цель работы: сверстать сайт по разработанному дизайну-макету в стиле минимализм. Реализовать адаптивное меню «гамбургер».

ТЕОРИЯ

«Гамбургер»-меню. Иконку в виде гамбургера создал дизайнер компании Xerox Норм Кокс в 1981 г., однако развитие и использование этого элемента началось гораздо позже, в 2000-х гг. с переходом веб-дизайна на адаптивность. Идея такого меню заключалась в скрытии сложной навигации за значком трех горизонтальных полосок, напоминающих три составляющие гамбургера.

Особенности меню «гамбургер»:

– позволяет не акцентировать внимание пользователя на отвлекающих факторах на главном экране, скрывая все опции под узнаваемым значком с тремя полосками;

– не перегружает пользователя большим выбором вариантов и не запутывает.

Применение меню «гамбургер» представлено на рис. 1 и 2.

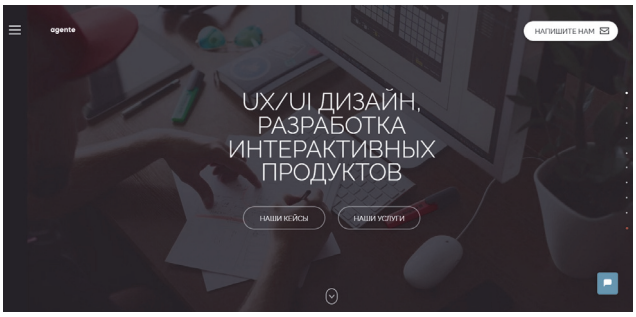


Рис. 1. Меню в левом верхнем углу

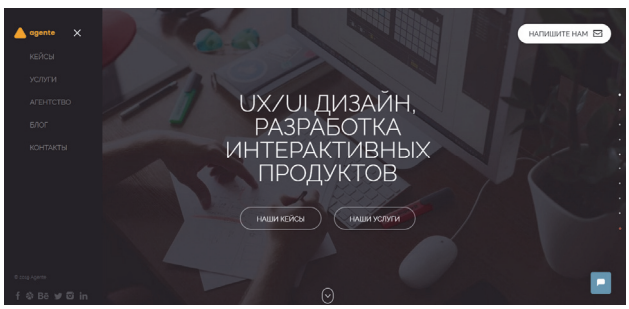


Рис. 2. Развернутый вариант меню

ЗАДАНИЕ

Согласно вашему дизайну-макету сверстать страницу в стиле минимализма, трансформировав ее таким образом, чтобы применить «гамбургер»-меню (минимум три пункта), декоративные блоки с типографикой. Меню «гамбургер» должно быть с работающими ссылками.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– титульный лист;
– название и цель работы;
– сверстанный сайт;
– листинги верстки сайта, включающего меню «гамбургер».

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Кто разработал «гамбургер-меню»?
2. В каких случаях лучше всего использовать меню данного типа?
3. Какие альтернативы существуют для меню «гамбургер»?
4. Какие плюсы и минусы у названного меню?

Лабораторная работа № 15

ВЕРСТКА САЙТА В АМЕРИКАНСКОМ БИЗНЕС-СТИЛЕ

Цель работы: получить практические навыки верстки сайта, дизайн которого выполнен в американском бизнес-стиле.

ТЕОРИЯ

***Характерные особенности для данного стиля*:**

– фиксированная ширина, стандартное расположение элементов, привычные образы, баннерная реклама;

– совмещение вертикального и горизонтального меню, «хлебные крошки», визуальное выделение активного пункта меню;

– качественная графика с использованием узнаваемых образов, вызывающих быстрые ассоциации у среднестатистического человека.

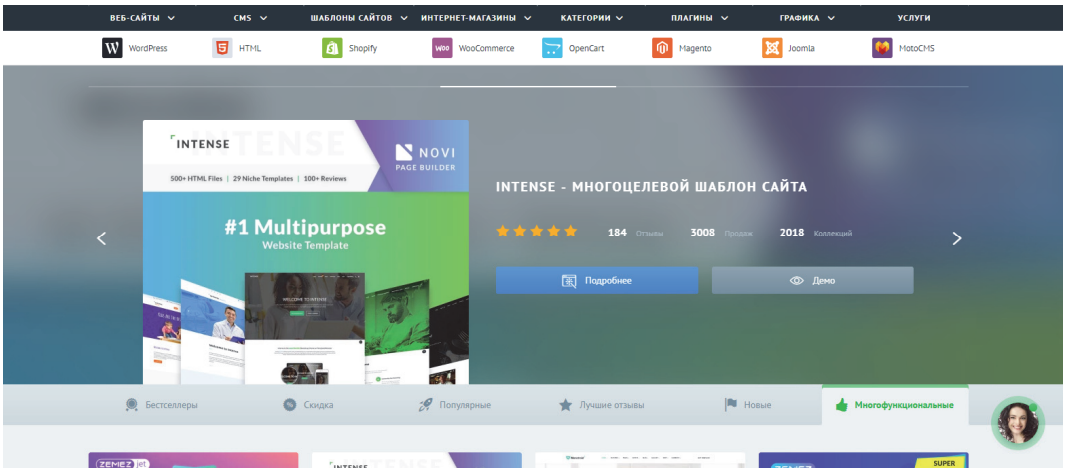
Современные тенденции предусматривают несколько вариантов оформления фона (сплошная заливка, градиентная заливка, текстурное заполнение, эффект «фальшивой резины» – сайт верстается жестко, но при этом шапка и фон под ней сливаются, образуя композиционное целое). В результате кажется, что шапка тянущаяся, хотя на самом деле тянется фон под ней.

***Рекомендации по созданию и верстке сайта в американском бизнес-стиле*:**

1. Модульная сетка должна быть простой и узнаваемой. Достаточно на главной странице использовать несколько ярких блоков, ведущих в самые важные разделы, а внутренние страницы можно оставить более простыми. Как правило, сайт в АМБ повторяет макет классического стиля, но в АМБ есть много элементов декора, рекламы и пр.

2. Применение качественного графического материала. Он должен быть ярким, простым, наглядным. Необходимо подбирать избыточный набор фотографий, чтобы можно было их разместить в шапке сайта или использовать для основы коллажа или баннеров, слайдеров.

Пример сайта в американском бизнес-стиле представлен на рисунке.



Американский бизнес-стиль

ЗАДАНИЕ

1. Видоизменить дизайн-макет вашего сайта в классическом стиле, добавив элементы американского бизнес-стиля.
2. Выполнить верстку сайта.
3. Использовать на сайте слайдер. Вид слайдера выбрать самостоятельно.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– титульный лист;
– название и цель работы;
– сверстанный сайт (не менее 3 страниц);
– листинги верстки с комментариями как выполнен слайдер (заимствован, разработан самостоятельно на чистом CSS, JS).

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для сайтов какой тематики используется бизнес-стиль?
2. Перечислите ключевые особенности стиля и общие закономерности.
3. Какое стандартное расположение элементов в американском бизнес-стиле?
4. Какая модульная сетка подойдет для сайта данного стиля? Почему?
5. Обоснуйте расположение элементов на вашем прототипе сайта с точки зрения представленного стиля.

Лабораторная работа № 16

РАЗРАБОТКА АНИМАЦИОННОГО БАННЕРА

Цель работы: получить практические навыки по разработке анимационного баннера.

ТЕОРИЯ

Можно выделить 2 вида баннеров для интернет-сайтов: статические и динамические (анимированные). Статические баннеры не отвлекают внимания от чтения основного текста и доступны к просмотру в браузерах, не поддерживающих анимацию. Анимированные баннеры за счет анимационных элементов привлекают больше внимания, однако и являются раздражающим элементом. Динамические баннеры можно разработать в графических редакторах (GIF-баннеры) и средствами HTML5.

GIF-баннер представляет собой последовательность растровых кадров, которые сменяют друг друга. Смена кадров происходит последовательно с учетом запрограммированной задержки для каждого кадра. Данный вид анимации обычно используется при несложном сценарии. Плавность движений в таком баннере может достигаться за счет множества промежуточных кадров, что существенно сказывается на объеме баннера.

Для создания анимированного баннера можно применять различные программные средства – Adobe Photoshop, Adobe Animate, Adobe Premiere и др.

HTML5-баннер – это комбинация HTML-элементов с применением анимаций и визуального оформления, адаптированных под любые устройства и браузеры. HTML-баннер может быть разработан с помощью CSS3 и JavaScript, элемента «canvas» и JavaScript, в котором реализуются анимация, рисунки, графики, игры, а также с привлечением сторонних библиотек, таких как CreateJS, или пакетов Adobe Animate CC, Google Web Designer и др.

Баннер встраивается на страницу через элемент «iframe».

ЗАДАНИЕ

Разработать сценарий и реализовать анимационный баннер для вашего сайта.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:

– титульный лист;

– название и цель работы;

– определение баннера, его сценарий;

– способы размещения баннеров;

– краткое описание алгоритма изготовления и разработки баннера для вашего сайта;

– описание инструментов разработки вашего баннера.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие виды баннеров вы знаете?
2. В чем отличие статических баннеров от динамических?
3. Какие динамические баннеры вы знаете?
4. Опишите плюсы и минусы GIF-баннеров.
5. Опишите плюсы и минусы HTML5-баннеров.
6. На каких устройствах отображаются HTML5-баннеры?

Лабораторная работа № 17

CSS-АНИМАЦИЯ И ТРАНСФОРМАЦИЯ

Цель работы: получить практические навыки по использованию CSS для анимации и трансформации объектов.

ТЕОРИЯ

Преобразования в CSS представляют собой набор функций, которые позволяют определенным образом придавать элементу форму:

– translate: перемещает элемент вдоль трех осей (*x*, *y* и *z*);

– rotate: вращает элемент вокруг центральной или опорной точки;

– scale: изменяет размер элемента;

– skew: искажает элемент.

Свойства трансформации

Есть три доступных свойства для трансформации:

1) transform определяет, какая функция будет применяться (translate, rotate, scale и др.);

2) transform-origin позволяет изменять точку начала преобразования (работает как background-position);

3) transform-style используется для настройки 3D.

Свойство transform не является сокращенным. Независимо от вращения, масштабирования или перемещения, они не будут влиять на другие элементы.

Функция translate() позволяет перемещать элемент в плоскости (по осям *x* и *y*).

Она принимает:

– один параметр: перемещает элемент вдоль оси *x*;

– два параметра: первое значение для оси *x*, второе – для оси *y*.

Transform – это стилевое свойство, translate() – значение (и одновременно функция), применяемое к этому свойству.

Можно использовать translateX() и translateY() для перемещения элемента вдоль только оси *x* и *y*, соответственно.

Функция rotate() позволяет вращать элемент вокруг неподвижной точки. По умолчанию вращение происходит вокруг центра элемента, rotate() принимает только один параметр, который является значением угла и определяется в градусах (deg), градах (grad), радианах (rad) или в оборотах (turn), где один оборот эквивалентен полному кругу.

```
@keyframes rotating {
0% { transform: rotate(0deg); }
100% { transform: rotate(360deg); }
}
p { animation: rotating 4s linear infinite; }
```

Функция scale() позволяет изменять размер элемента. Она может увеличить или уменьшить элемент. Функция принимает:

– один параметр: изменение размеров элемента одинаково по высоте и ширине;

– два параметра: первое значение изменяет размер элемента по горизонтали, второе – по вертикали.

Диапазон возможных значений:

1: элемент сохраняет свой первоначальный размер;

2: элемент удваивается в размере;

0.5: элемент уменьшается наполовину;

0: элемент в основном исчезает (так как его высота и ширина равны нулю);

–1: элемент отражается.

```
@keyframes scaling {
0% { transform: scale(1); }
20% { transform: scale(2); }
40% { transform: scale(0.5); }
60% { transform: scale(0); }
80% { transform: scale(-1); }
100% { transform: scale(1); }
}
p { animation: scaling 10s steps(1) 0s infinite; }
```

Есть версии для *x* и *y*: scaleX() и scaleY(), для изменения размера по горизонтали и вертикали, соответственно.

Функция skew() позволяет искажать элемент, сдвигая его стороны вдоль линий. Эта функция преобразования используется редко, поскольку ее последствия весьма непредсказуемы, а результат не обязательно привлекателен.

Функция skew() принимает :

– один параметр: элемент искажается по горизонтали;

– два параметра: первое значение искажает элемент по горизонтали, второе – по вертикали.

Функция skew() принимает только значения угла в градусах (deg). Пример представлен ниже.

```
@keyframes skewing {
0% { transform: skew(0deg); }
20% { transform: skew(10deg); }
40% { transform: skew(45deg); }
60% { transform: skew(90deg); }
80% { transform: skew(120deg); }
100% { transform: skew(0deg); }
}
p { animation: skewing 10s steps(1) 0s infinite; }
```

Для translate() есть версия *translate3d()*, которая выполняет преобразование в трех измерениях, а это значит, что она также включает в себя ось *z* (кроме того, существует отдельная функция translateZ).

Параметр *z* в основном заставляет элемент двигаться ближе и дальше: в зависимости от уменьшения или увеличения значения. Это как увеличение и уменьшение масштаба.

```
@keyframes zooming {
0% { transform: translate3d(0, 0, 0); }
100% { transform: translate3d(0, 0, 200px); }
}
p { animation: zooming 5s alternate; }
```

К родительскому элементу требуется применить свойство perspective: 500 px, чтобы трехмерное пространство стало заметным. В качестве альтернативы также может быть использовано свойство transform: perspective(500px).

ЗАДАНИЕ

1. Разработать и применить на вашем сайте CSS-анимацию.
2. Выполнить трансформацию отдельных объектов сайта.
3. Анимировать страницу 404.
4. Проверить HTML-разметку валидатором (validator.w3.org).

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:

– титульный лист;

– название и цель работы;

– макет и листинги разработки страницы 404 с анимационными эффектами;

– краткое описание алгоритма разработки CSS-анимации и трансформации;

– страницу со скриншотом отчета валидатора HTML.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите основные свойства трансформации.
2. Какие параметры они в себя включают?
3. Кратко опишите принцип их работы.
4. Сколько параметров может принимать функция scale()?
5. Что выполняет функция skew?
6. В каких единицах измерения угла принимает значения функция skew?
7. По каким осям функция scale() может изменять размеры?

Лабораторная работа № 18

ИСПОЛЬЗОВАНИЕ ФОРМАТА SVG

Цель работы: ознакомиться с возможностями формата SVG; получить практические навыки по использованию SVG для рисования и анимации.

ТЕОРИЯ

Формат SVG (от англ. Scalable Vetor Graphics – масштабируемая векторная графика) – язык разметки масштабируемой векторной графики. Предназначен для описания двумерной векторной или векторно-растровой графики в формате XML (т. е. на языках, родственных языкам HTML и CSS). Поддерживает как неподвижную, так и анимированную интерактивную графику, т. е. элементами SVG-графики можно управлять средствами CSS или JavaScript.

Пример. SVG в коде страницы – inline-SVG (рисунок)

```
<div style='background-color: #FFFF80'>
<svgwidth="256"height="256""xmlns="http://www.w3.org/2000/svg">
<path d="m 133.08501, 28.75909 a 7.4746049, 7.4746049 0 0 0 -11.54314, 1.75487 l -54.12787, 93.74883 -54.12786, 93.74883 a 7.4746049,
7.4746049 0 0 0 6.45401, 1 1.21164 l 108.25569,0 108.25576, 0 a 7.4746049, 7.4746049 0 0 0 6.47348, -11.19217 l -54.10833, -93.7683 -54.12785, -93.74883 a 7.4746049, 7.4746049 0 0 0 -1.40389, -1.75487 z m -5.08917, 20.45394 47.65436, 82.51769 47.65444, 82.55672 -95.3088, -3e-5 -95.30869, 0 47.65437, -82.55669 47.65432, -82.51769 z"
style="color: #000000; fill: #ff0000; stroke-width:17" />
<path id=IMalanka d="m 140.6108, 93.17314 -36.08065, 55.83917 18.04032, 1.2886 -18.04032, 48.53712 36.93969, -53.69151 -16.32215, -3.86579 c 0, 0 16.32215, -48.96665 15.46311, -48.10759 z"
style="fill: #000000; stroke: #000000; stroke-width:0.5px;
stroke-linecap:butt; stroke-linejoin:miter" />
</svg>
</div>
```

SVG используется для наложения масок. Более подробно об SVG масках и способах их создания по ссылке: http://css.yoksel.ru/svg-masks/

Изображение в формате SVG

Основные теги и атрибуты для работы с SVG

Теги	Атрибуты
<svg> парный тег, включает в себя все описания и построения рисунка	width, height – размеры рисунка на HTML-странице, в пикселях; viewBox – четыре координаты (через пробел) в системе координат рисунка, задающие прямоугольник, который отображается на HTML-странице (он может охватывать не всю площадь построений); пропорции viewBox могут не совпадать с пропорциями атрибутов width и height. Если не задан, предполагается что координатная система рисунка совпадает с сеткой пикселей HTML-страницы и рисунком является прямоугольник от точки 0, 0 до точки width, heigh; preserveAspectRatio позволяет задать, как выравнивается и как масштабируется рисунок внутри прямоугольника width, height, по умолчанию рисунок растягивается, игнорируя пропорции
<defs> парный тег	служит для различного рода описаний, которые затем могут быть использованы в рисунке; применяется внутри тега <svg>
<desc> парный тег	служит для текстового комментирования графики, которое может быть использовано альтернативными агентами (речевыми браузерами, поисковыми роботами и т. д.); содержимым тега должен быть текст; используется внутри любых SVG-тегов
Геометрические фигуры	
<line> рисует отрезок прямой линии <line x1=0 y1=0 x2=100 y2=100 stroke=red />	*x*1, *y*1 – координаты начала линии; *x*2, *y*2 – координаты конца линии; stroke – цвет линии (как в CSS);

Продолжение таблицы

	stroke-width – толщина линии, по умолчанию 1; здесь и далее – в значении SVG-атрибутов размера единицы измерения не указываются, они всегда в пикселях; в значениях стилевых свойств размера единицы измерения обязательны!
<polyline> рисует ломаную линию, незамкнутую последовательность отрезков <polyline points='5.85 80.100 15.115' stroke=red fill=none />	points – строка с парами координат узлов ломаной через пробел, каждая в виде *x*, *y*; stroke – цвет линии; stroke-width – толщина линии
<polygon> рисует многоугольник, замкнутую (и возможно залитую) последовательность отрезков <polygon points='5.125 80.140 15.155' stroke=red fill=yellow />	points – строка с парами координат узлов ломаной; stroke – цвет линии; stroke-width – толщина линии; fill – цвет заливки (как в CSS)
<rect> рисует прямоугольник (возможно залитый)	*x*, *y* – координаты левого верхнего угла прямоугольника; width, height – ширина и высота прямоугольника; rx, ry – радиус скругления углов прямоугольника; stroke – цвет линии; stroke-width – толщина линии; fill –цвет заливки
<circle> рисует окружность (возможно залитую)	*cx*, *cy* – координаты центра окружности; *r* – радиус окружности; stroke – цвет линии; stroke-width – толщина линии; fill – цвет заливки
<ellipse> рисует эллипс (возможно залитый) <ellipse cx=120 cy=40 rx=45 ry=30 stroke=green fill='#FFD0D0' />	*cx*, *cy* – координаты центра окружности; *rx*, *ry* – радиусы эллипса; stroke – цвет линии; stroke-width – толщина линии; fill – цвет заливки
<path> рисует линию, состоящую из прямых и/или криволинейных участков (возможно замкнутую и залитую) В строке, описывающей участки линии, могут быть использованы следующие команды M, H, L, V, Z, задающие координаты прорисовки	*d* – строка, содержащая команды построения участков линии, разделенные пробелами; stroke – цвет линии; stroke-width – толщина линии; fill – цвет заливки

Продолжение таблицы

Заливки	
для любых тегов, рисующих линии	stroke-opacity – непрозрачность линии, дробное число от 0 до 1 (по умолчанию 1); stroke-linecap – форма концов линии, одна из констант; butt (торец, по умолчанию), round (круглое перо), square (квадратное перо); stroke-linejoin – форма углов линии, одна из констант; miter (косые, по умолчанию), round (скругленные), bevel (конические); также атрибут и стилевое свойство stroke-miterlimit ограничивает выступ угла, образующегося при stroke-linejoin=miter; stroke-dasharray – длины штрихов и интервалов между штрихами через запятую; stroke-dashoffset – отступ первого штриха (т. е. на сколько короче должен быть первый штрих, чем указано в stroke-dasharray)
Текст	
<text> парный тег, выводит строку текста; должен содержать текст либо тег **<textPath>**, может содержать теги <tspan>	*x*, *y* – координаты текста font, font-family, font-size, font-style, font-weight, letterspacing, word-spacing, text-decoration и другие стилевые свойства и одноименные атрибуты задают различные свойства шрифта (аналогично CSS); textLength – ширина текста (если не задана, будет определена автоматически; если задана, будут увеличены интервалы между буквами и, возможно, ширина букв, чтобы довести ширину текста до указанной); lengthAdjust – задает, что следует увеличивать для получения установленной через textLength ширины текста; spacing (по умолчанию) – только интервалы между буквами; spacingAndGlyphs – интервалы между буквами и ширину букв; rotate – задает углы поворота для каждого символа строки через запятую; если символов больше чем задано углов, последний угол действует на все символы до конца строки; stroke – цвет линии (обводки букв);

	fill – цвет заливки букв; text-anchor – выравнивание текста относительно координат, заданных атрибутами *x*, *y*; start (по умолчанию) – *x*, *y* задают координаты левого конца базовой линии текста; middle (по умолчанию) – *x*, *y* задают координаты середины базовой линии текста; end (по умолчанию) – *x*, *y* задают координаты правого края базовой линии текста;
<tspan> парный тег, задает подстроку текста, которую можно позиционировать и / или стилизовать отдельно от остального текста; должен содержать текст	*dx*, *dy* – относительные координаты подстроки текста font, font-family, font-size, font-style, fontweight, letter-spacing, word-spacing, textdecoration и другие стилевые свойства и одноименные атрибуты задают различные свойства шрифта (аналогично CSS); stroke – цвет линии (обводки букв); fill – цвет заливки букв
<textPath> парный тег, задает кривую, вдоль которой нужно расположить текст; должен содержать текст	xlink:href – URI описанной ранее кривой; обычно кривая описана в этом же документе, тогда URI можно записать в виде "#XXX", где XXX – идентификатор кривой; startOffset – начальное смещение текста вдоль кривой
Растровые изображения	
<image> вставляет растровое изображение (аналогично тегу <img> в HTML)	*x*, *y* – координаты изображения; width, height – размеры прямоугольника, в который должно быть вписано изображение (пропорции изображения не меняются); эти атрибуты указываются в единицах SVG (как и все координаты и размеры в SVG), а не в пикселях; xlink:href – URI изображения (так как атрибут принадлежит пространству имен xlink, следует в теге <SVG> указывать URI этого пространства имен атрибутом xmlns:xlink='http://www.w3.org/1999/xlink'
Группировка	
<g> парный тег; позволяет сгруппировать некоторые построения, для применения к ним общих атрибутов, стилевых свойств, трансформаций и т. д.	<svg width=170 height=100 xmlns='http://www.w3.org/2000/svg'> <g stroke=green stroke-width=3> <circle cx=35 cy=40 r=30 fill=yellow /> <ellipse cx=120 cy=40 rx=45 ry=30 stroke=red fill=orange /> </g> </svg>

Преобразования и эффекты	
для любых тегов, рисующих что-либо	transform – преобразование (аналогично стилевому свойству transform в CSS3), значение представляет собой список операций, разделенных запятыми или пробелами, в списке могут быть указаны следующие операции; translate(*x y*) – перенос построений на *x* вправо и на *y* вниз; scale(*s*) или scale(*sx sy*) – увеличение построений в s раз (во втором варианте указывается отдельные коэффициенты для *x* и для *y*); rotate(α) или rotate(α *x y*) – поворот построений на угол α (в градусах, по часовой стрелке) относительно начала координат (во втором варианте координаты x и y указывают, относительно какой точки должен быть осуществлен поворот); skewX(α) и skewY(α) – наклон построений на угол α по оси *x* и *y* соответственно; здесь и далее в значении SVG-атрибутов угла единицы измерения не указываются, они всегда в градусах: transform="rotate(45)" в значениях стилевых свойств угла единицы измерения обязательны; style="transform: rotate(45deg)"
Библиотека символов	
<symbol> парный тег, описывает (неотображаемые) построения, которые затем можно отобразить многократно в рисунке (тегом <use>) с возможностью стилизации «по месту»; внутри тега <symbol> – своя система координат	id – идентификатор, по которому можно затем ссылаться на символ. <symbol id=BRICK1> <circle cx=33 cy=33 r=30 fill=green /> <rect x=13 y=18 width=40 height=30 stroke=red fill=yellow /> </symbol> <use xlink:href='#BRICK1' x=30 y=10 /> <use xlink:href='#BRICK1' opacity=0.5 transform='scale(0.5)' x=90 y=150 />
<use> отображает описанные ранее построения; построения могут быть описаны тегом <symbol> или любым тегом, обычно внутри тега <defs>	*x*, *y* – координаты, в которых отображаются построения; xlink:href – URI построений, обычно построения описаны в этом же документе, тогда URI можно записать в виде "#XXX", где XXX – идентификатор построений (например, тега <symbol>); если построения описаны во внешнем SVG-файле, URI может быть в виде "путь/ имя.svg#XXX", причем этот внешний SVG-файл в целях безопасности должен иметь одно происхождение с документом (т. е. один домен, протокол и порт)

ЗАДАНИЕ

1. Добавить на сверстанные страницы анимационные эффекты, выполненные с использованием SVG. На странице должно быть не менее трех анимационных эффектов. Применить SVG-маску.
2. Проверить HTML-разметку валидатором (validator.w3.org).

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– титульный лист;
– название и цель работы;
– листинги верстки веб-страниц с анимационными эффектами;
– краткое описание алгоритма разработки;
– страницу со скриншотом отчета валидатора HTML.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое SVG?
2. Какие преимущества SVG над другими форматами?
3. Какие недостатки SVG?
4. Перечислите основные теги и атрибуты для работы с SVG.
5. Каким образом можно получить SVG-код изображения?

Лабораторная работа № 19

ФОРМА И ВАЛИДАЦИЯ ЗНАЧЕНИЙ ПОЛЕЙ

Цель работы: научиться создавать на веб-странице формы; изучить возможные элементы формы и их свойства, а также способы доступа к элементам формы и проверки вводимых значений.

ТЕОРИЯ

HTML формы могут содержать такие элементы ввода, как:

– текстовые поля;

– флажки;

– радио-кнопки;

– кнопки отправления и др.

Формы также могут содержать списки выбора, многострочные текстовые поля, метки и др.

Для создания формы в HTML используется тег **<form>**.

Элементы ввода применяются для приема пользовательских данных. Они отличаются друг от друга в зависимости от значения атрибута type:

– текстовое поле <input type="text"/> определяет однострочное текстовое поле, в которое пользователь может вводить различную информацию;

– поле пароля <input type="password"/> определяет поле для ввода пароля;

– флажок ☐ позволяет пользователям выбирать несколько пунктов с предварительно заполненной информацией из группы;

– радио-кнопка ☐ позволяет выбрать только один пункт с предварительно заполненной информацией из группы;

– кнопка отправления <input type="submit"/> предназначена для отправления на сервер введенных данных. Адрес, на который будут пересылаться данные формы, указывается в атрибуте тега form – action. Если приведенный атрибут отсутствует, данные будут отправлены на текущую страницу;

– текстовая область <textarea> позволяет ввести многострочный текст.

– список <select> создает выпадающий список, элементы которого определяются с помощью тега <option>. С помощью атрибута multiple можно указать, что в выпадающем списке могут быть выбраны одновременно несколько элементов;

– заголовки в формах <fieldset> позволяют сгруппировать желаемую часть формы и затем с помощью тега <legend> установить желаемое заглавие.

Доступ к элементам формы осуществляется с помощью JavaScript. Можно отправить данные формы на сервер, очистить ее, а также получить доступ к любому элементу формы для изменения его значений.

Существует несколько способов обращения к формам с использованием объектной модели. При создании формы автоматически создается массив forms. Для доступа к форме требуется указать номер элемента или имя формы, заданное параметром name, как показано в примере.

```
<html>
    <body>
        <form name=data>
            …
        </form>
        <script>
            alert(document.forms.length) // количество форм на стр.
            alert(document.forms[0].name) // имя первой формы
            alert(document.forms.data.length) // количество элем.
alert(document.forms["data"].length) //
        </script>
    </body>
</html>
```

Нумерация элементов массива всегда начинается с нуля, поэтому обращение к первой форме будет document.forms[0], ко второй – document.forms[1].

Второй способ – через семейство all. Обращение к форме происходит как к элементу массива с именем, совпадающим с именем формы – document.all["data"], или напрямую – document.all.data, как показано в примере.

```
<html>
    <body>
        <form name=data>
            ...
        </form>
        <script>
            alert(document.all["data"].length)
            alert(document.all.data.name)
        </script>
    </body>
</html>
```

Обращение к элементам формы осуществляется посредством семейства elements или напрямую по имени элемента (смотрите пример).

```
<html>
    <body>
        <form name=data>
 <input type=text name=userName value="Введите ваше имя">
        </form>
        <script>
            // Общее число элементов в форме
            alert(document.forms.data.elements.length)
            // Значение первого элемента
            alert(document.forms[0].elements[0].value)
            // Значение элемента с именем userName
            alert(document.forms["data"].userName.value)
        </script>
    </body>
</html>
```

Нумерация массива elements, как и в случае с семейством forms, ведется с нуля, поэтому обращение к первому элементу формы будет elements[0]. Для большого количества данных в форме значения элементов лучше получать по их имени.

Доступ к элементам формы можно также осуществить по их ID с помощью метода document.getElementById().

ЗАДАНИЕ

Реализовать в своем проекте форму (форму заявки, заказа, обратной связи, согласно вашему макету) с проверкой валидации не менее трех полей.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– титульный лист;
– название и цель работы;
– листинги верстки форм;
– JS-листинги проверки валидации форм.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие элементы ввода вы знаете?
2. Кратко опишите синтаксис каждого элемента.
3. Что такое валидация?
4. Как организуется нумерация массива в зависимости от выбранного способа обращения к форме?

Лабораторная работа № 20

СИНЕМАГРАФЫ И ПАРАЛЛАКС-ЭФФЕКТ НА САЙТЕ

Цель работы: ознакомиться с техникой синемаграфии и параллакс-эффектами; получить практические навыки по разработке синемаграфов и параллаксов.

ТЕОРИЯ

Термин *синемаграфия* был введен в 2011 г. фотографами Кевином Бургом (Kevin Burg) и Джеймсом Беком (James Beck), которые впервые применили данную технику. Она основывается на объединении фотографии и видео, где за основу берется видео, в котором некоторые элементы оставляют статичными, а другие – подвижными, анимированными.

Синемаграф – определенным образом созданное видео в бесконечном цикле на статическом фоне. Цикл видео не превышает 1–5 с.

Синемаграфия является сильным визуальным инструментом для электронного маркетинга и брендинга.

Для создания синемаграфа необходимо:

1. Разработать сценарий. В сценарии указать, что хотим отобразить, какие элементы будут статичными, а какие динамичными. Определить ключевые кадры и их описание. Для этого выбрать или снять исходное видео с постановкой кадра. Определить движущийся объект и замороженный фон либо наоборот. Важно, чтобы движущийся объект был циклическим. Лучше всего для этого подходят природные явления: снегопад, вода, волны, огонь, дым и др.

2. Фокусировка. Определить объект, на котором будет сконцентрировано внимание. Как правило, это один объект, поэтому важно сохранить фокусировку на этом объекте при съемке. Движение лучше организовать в центральной зоне кадра. Чаще движение придают одному объекту, но в зависимости от цели их может быть несколько.

3. Съемка. При съемке камера должна быть неподвижна, лучше использовать штатив. Очень важно обратить внимание на постановку кадра, фокус, количество света и т. п.

4. Обработка. Необходимо правильно наложить слои, выделить динамический объект и сохранить качество видео.

5. Размещение. Синемаграфы имеют формат видео или анимации. Тип контента зависит от цели и места его размещения. Например, синемаграфию в инстаграм лучше выкладывать в формате видео.

Для загрузки на сайт рекомендуется использовать GIF-формат.

Для окончательной обработки можно использовать Photoshop. Вначале следует импортировать видео в слои, после чего можно работать с отдельными кадрами. Они отображаются на шкале времени, что упрощает поиск цикличного момента видео. Если такого кадра нет, можно отзеркалить видео.

Анимировать объекты помогает кейфрейм – ключевой кадр.

При помощи программы объекты на видео можно замораживать, перемещать, изменять размер и т. п.

Сохранить синемаграф можно в форматах: gif, mp4, mwv.

Ознакомиться с примерами синемаграфов и приемами работы по ссылке https://sunny-alison.livejournal.com/76598.html.

Параллакс – специальная техника, при которой фоновое изображение в перспективе двигается медленнее, чем элементы переднего плана.

Достигается такой эффект трехмерного пространства с помощью нескольких слоев, которые накладываются друг на друга и при прокручивании движутся с различной скоростью. С помощью данной технологии можно создать не только искусственный трехмерный эффект, но и применять ее к иконкам, изображениям и другим элементам страницы.

С помощью CSS3 есть возможность создать похожий эффект: контент сайта размещается на одной странице, а перемещение по подстраницам происходит методом CSS3-перехода.

Примеры по ссылкам параллакс-скроллинг – https://ruseller.com/lessons.php?rub=28&id=1889; параллакс-эффекты – https://www.kasper.by/blog/parallaks-effekt-na-saite/

ЗАДАНИЕ

1. Разработать синемаграф. Разместить на одной из страниц сверстанного сайта. Если это не вписывается в концепцию, представить как отдельное произведение.

2. На главной странице сайта (стиль по выбору) применить параллакс-эффект.

3. Проверить HTML-разметку валидатором (validator.w3.org).

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– титульный лист;
– название и цель работы;
– сверстанные страницы;
– листинги верстки сайта;
– страницу со скриншотом отчета валидатора HTML.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое синемаграфия?
2. По какому принципу она работает?
3. Что такое параллакс-эффект?
4. Какими способами он достигается?

Лабораторная работа № 21

БИБЛИОТЕКА JQUERY. СОЗДАНИЕ ИНТЕРАКТИВНОЙ КАРТЫ

Цель работы: освоить создание анимационных эффектов с помощью библиотеки jQuery.

ТЕОРИЯ

JQuery – библиотека JavaScript, предназначенаая для облегчения работы с анимационными и прочими эффектами на веб-странице, работает одинаково во всех основных браузерах.

Подключение библиотеки JQuery:

– скачать библиотеку JQuery, источник – jQuery.com, подключить файл:

– <head><script src="jquery-1.11.3.min.js"></script></head>

– включить JQuery из CDN, например Google:

<script type="text/javascript" <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>

Код jQuery состоит из последовательно идущих команд. Стандартный синтаксис jQuery команд:

$(селектор).метод();

Знак $ сообщает, что символы, идущие после него, являются jQuery кодом; селектор позволяет выбрать элемент на странице; метод задает действие, которое необходимо совершить над выбранным элементом.

Методы в jQuery разделяются на следующие группы:

– для манипулирования DOM;

– оформления элементов;

– создания AJAX-запросов;

– создания эффектов;

– привязки обработчиков событий.

jQuery можно комбинировать с обычным JavaScript. Если строка начинается с $ – это jQuery, если $ в начале строки отсутствует – это строка JavaScript кода.

Селекторы jQuery

С помощью селекторов можно выбирать элементы на странице для применения к ним определенных действий.

В табл. 1 приведены примеры использования селекторов для выбора желаемых элементов.

Таблица 1

Примеры селекторов

Пример	Результат
$("p")	Будут выбраны все элементы p, которые находятся на странице
$(".par")	Будут выбраны все элементы на странице с class="par"
$("#par")	Будет выбран первый элемент на странице с id="par"
$(this)	Позволяет выбрать текущий HTML-элемент
$("p.par")	Будут выбраны все элементы p на странице с class="par"
$("p#par")	Будут выбраны все элементы p на странице с id="par"
$(".par,.header,#heat")	Будут выбраны все элементы на странице со значениями атрибутов class="par", class="header" и id='heat'
$("[src]")	Будут выбраны все элементы на странице, имеющие атрибут src
$("[src='значение']")	Будут выбраны все элементы со значениям атрибута src="значение"
$("[src$='.gif']")	Будут выбраны все элементы со значениями атрибута src, заканчивающимися на .gif
$("div#wrap .par1")	Будут выбраны все элементы с class=par1, которые находятся внутри элементов div с id=wrap
$(":input")	Будут выбраны все input элементы на странице
$(":тип")	Будут выбраны все input элементы с <input type='тип' />. Например :button выберет все <input type='button' /> элементы, :text выберет все <input type='text' /> элементы и т. д.

Методы обработчиков событий jQuery

События (Events)

Все действия различных пользователей, на которые веб-страница может реагировать, называются событиями:

- перемещение мыши над элементами;
- выбор радиокнопок (переключателей);
- нажатие мыши.

В табл. 2 представлены наиболее распространенные события мыши (MouseEvents), клавиатуры (KeyboardEvent), обработки форм (FormEvents), окна приложения(WindowEvents).

Таблица 2

Методы событий

MouseEvents	KeyboardEvents	FormEvents	Document/ WindowEvents
Click	Keypress	Submit	Load
Dblclick	Keydown	Change	Resize
Mouseenter	Keyup	Focus	Scroll
Mouseleave	Blur		Unload

Синтаксис методов обработчиков событий JQuery

```
$("p").click(function(){ // что-то делаем});
```

Часто используемые методы обработчиков событий JQuery приведены в табл. 3.

Таблица 3

Примеры методов событий

Метод	Пример	Описание
$(document).ready()	$("p").click(function(){ $(this).hide(); })	Обеспечивает выполнение функции после полной загрузки документа
dblclick()	$("p").dblclick(function(){ $(this).hide(); });	Скрывает абзацы по двойному клику на них мыши
mouseenter()	$("#p1").mouseenter(function(){ alert("You entered p1!"); });	Функция выполняется при наведении указателя мыши наведен на HTML-объект
mouseleave()	$("#p1").mouseleave(function(){ alert("Bye! You now leave p1!"); });	Функция выполняется, как только указатель мыши покидает HTML-объект
mousedown()	$("#p1").mousedown(function(){ alert("Mouse down over p1!"); });	Присоединяет функцию обработчика события к элементу HTML. Функция выполняется при нажатых левой, средней и правой кнопки мыши, и нахождении мыши над элементом HTML
mouseup()	$("#p1").mouseup(function(){ alert("Мышь сверху p1!"); });	При освобожденной мыши

hover()	$("#p1").hover(function(){ alert ("вы выбрали p1!"); }, function(){ alert("вы покинули p1!"); })	Имеет две функции, которые выполняются как комбинации mouseenter() andmouse-leave()
focus()	$("input").focus(function() { $(this).css("background-color", "#cccccc"); });	Добавляет обработчик события к полям формы, выполняется, как только поле формы получает фокус
blur()	$("input").blur(function(){ $(this). css("background-color", "#ffffff"); });	Выполняется при потери полем формы фокуса

Эффекты JQuery

Показ и скрытие

Синтаксис:

$(selector).hide(speed,callback);

$(selector).show(speed,callback);

Параметры (необязательные):

1) speed – определяет скорость скрытия / показа и может принимать следующие значения: "slow", "fast" или milliseconds;

2) callback – функция, которая будет выполнена после завершения методов hide() или show().

Пример 1.

```
html>
<head>
 <script src="https://ajax.gooeapis.com/ajax/libs/jquery/2.2.0/jquery.min.js">
</script><script>
$(document).ready(function(){
$("#hide").click(function(){
$("p").hide();
});
$("#show").click(function(){
$("p").show();
});
});
</script>
</head>
<body>
<p>При нажатии на кнопку, все скроется.</p>
```

```
<button id="hide">Скрыть</button>
<button id="show">Показать</button>
</body>
</html>
```

Пример 2. Параметр скорости

```
<script>
$(document).ready(function(){
$("button").click(function(){
$("p").hide(1000);
});
});
</script>
</head>
<body>
<button>Скрыть</button>
<p>Это параграф.</p>
<p>Это другой небольшой параграф.</p>
</body>
</html>
```

Переключение

Синтаксис*:*

$(selector).toggle(speed,callback);

Переключение между показом и скрытием.

Пример

$("button").click(function(){ $("p").toggle(); });

Видимость элементов

Методы, управляющие видимостью элементов, представлены в табл. 4.

Таблица 4

Методы зависимости

fadeIn()	$(selector).fadeIn(speed,callback);	Используется для плавного отображения скрытого элемента
fadeOut().	$(selector).fadeOut(speed,callback);	Используется для удаления из видимости элемента

fadeToggle().	$(selector).fadeToggle(speed,callback);	Переключение между вид / невид
fadeTo()	$(selector).fadeTo(speed,opacity, callback);	Позволяет изменить цвет, задавая значение непрозрачности (value between 0 and 1)

Пример. Скрытие абзаца.

```
$(document).ready(function(){
  $("#but1").click(function(){$("#par1").fadeOut(3000)});
  $("#but2").click(function(){$("#par1").fadeIn(3000)});
$("#but3").click(function(){$("#par1").fadeTo(3000,0.3)});
$("#but4").click(function(){$("#par1").fadeTo(3000,0.8)});
$("#but5").click(function(){$("#par1").fadeOut(3000,function(){ alert("Абзац был полностью скрыт.");});
});
});
```

Методы скольжения:

1) slideDown() – метод выполняет плавный разворот элемента до исходного размера вниз;

2) slideUp() – метод выполняет плавное изменение высоты элемента до 0;

3) slideToggle() – переключает методы slideDown() и slideUp().

Синтаксис:

$(selector).slideUp(speed,callback).

ЗАДАНИЕ 1

Разработать веб-страницу с тематической интерактивной картой. Страна и тип карты представлены в табл. 5. Ваш вариант – номер по списку в журнале подгруппы.

На странице отображается карта страны. При наведении курсора мыши на область карты изменяется ее цвет, должны появляться всплывающие блоки с информацией. Если это политическая карта – название провинции, области, главный город, население, еще три параметра на выбор, если это климатическая карта – название местности, средние температуры сезонов и т. д. НЕ МЕНЕЕ 4 позиций информации. При создании карты и оформлении страницы учитывать политические, национальные мотивы.

Для создания карты использовать SVG.

Таблица 5

Варианты

№	Страна	Тип карты	Примечание
1	Беларусь	Природная	Заповедники
2	Польша	Политическая	Административные округа, воеводства, герб воеводства, население
3	Австрия	Географическая	Курорты, горы, долины
4	Аргентина	Туристическая	Национальные достопримечательности
5	Вьетнам	Туристическая	Курорты
6	Греция	Историческая	Памятники
7	Грузия	Историческая	Памятники и исторические области
8	Нидерланды	Географическая	Каналы
9	Испания	Гастрономическая	Области и национальные танцы / блюда
10	Кения	Природная	Заповедники
11	Куба	Природная	Курорты
12	Мексика	Историческая	Национальные достопримечательности
13	Франция	Историческая	Замки
14	Турция	Природная	Климатические зоны
15	Германия	Политическая	Земли, названия, краткая история

ЗАДАНИЕ 2

Разработать веб-страницу, которая должна содержать фон, текстовые блоки (минимум 2), графический блок (рисунок). В качестве темы текста выбрать рассказ про котов и мышей. В тексте должно присутствовать слово КОТ не менее 5 раз. Используя методы JQuery, реализовать следующее: при наведении мыши на слово КОТ, слово исчезает и появляется изображение кота с мышью (задача заключается в смене текстовых и графических блоков).

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– веб-страница;
– описание технологий, функций по созданию карты и интерактива.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего используются селекторы? Привести примеры и их значения.
2. Какие существуют методы обработчиков событий, их значения?
3. Какие методы используются для видимости элементов, для скольжения?

Лабораторная работа № 22

АНИМАЦИЯ НА JS

Цель работы: изучить возможности управления процессами во времени; научиться строить анимационные графики и траектории движения.

ТЕОРИЯ

Можно выделить следующие методы управления процессами во времени:

1. ***Метод setInterval().*** Для организации постоянного периодического (через заданный интервал времени) выполнения некоторого выражения или функции служит метод setInterval() объекта window.

Синтаксис:

setInterval (выражение, период, [язык]).

Первый параметр – строка, содержащая выражение (обычно вызов функции). *Второй параметр* – целое число, указывающее временную задержку в миллисекундах. *Третий*, необязательный параметр, указывает язык, на котором написано выражение; по умолчанию – JavaScript. Метод setInterval() возвращает некоторое целое число – идентификатор временного интервала, который может быть использован в дальнейшем для прекращения процесса.

Пример: myfunc() выполняется периодически через 0,5 с.

```
setInterval("myfunc()", 500)
```

2. ***Метод clearInterval().*** Используется для остановки запущенного временного процесса, он принимает в качестве параметра целочисленный идентификатор, возвращаемый соответствующим методом setInterval(), например:

```
var myproc = setInterval("myfunc () , 100")
if (confirm("Прервать процесс ?"))
clearInterval(myproc)
```

3. Метод setTimeOut(). Для выполнения выражения с некоторой временной задержкой используется метод setTimeout() объекта window, setTimeout (выражение, задержка [, язык]).

Выражение – строка, содержащая выражение (обычно вызов функции). *Второй параметр* – целое число, указывающее временную задержку в миллисекундах. Третий, необязательный параметр, указывает язык, на котором написано выражение; по умолчанию – JavaScript.

Метод setTimeout() возвращает целое число – идентификатор временного интервала, который может быть использован в дальнейшем для отмены задержки выполнения процесса.

Для отмены задержки процесса, запущенного с помощью метода setTimeout(), используется метод clearInterval(идентификатор), который принимает в качестве параметра целочисленный идентификатор, возвращаемый соответствующим методом setTimeout.

Движение по произвольной кривой. В примере рассматривается задача движения изображения по синусоиде с 50 пикселов и горизонтальной скоростью 10 пикселов в секунду. Начальные ординаты графического объекта равны 100 и 50 пикселов по вертикали и горизонтали соответственно. Функция движения в качестве параметров будет иметь два выражения, которые описывают изменения вертикальной и горизонтальной координат элемента. Эти выражения будут содержать одну переменную *x*, которую можно рассматривать как независимый параметр движения (например, время). С помощью встроенной функции eval() вычисляются значения этих выражений при конкретном значении *x* и присваиваются параметрам left и top таблицы стилей перемещаемого элемента. Функция move(), которая все это выполняет, передается в качестве первого параметра методу setInterval(). Он периодически вызывает ее через заданный интервал времени.

Функция инициализации движения curvemove() принимает три строковых параметра (ID перемещаемого элемента, выражение для вертикальной координаты и выражение для горизонтальной координаты) и один числовой параметр (период времени, через который координаты элемента пересчитываются). Ниже приводятся определения функций curvemove() и move():

```
function curvemove(xid, yexpr, xexpr, ztime) {
/* Движение по произвольной кривой.*/
if (!xid) return null
if (!yexpr) yexpr = "x"
if (!xexpr) xexpr = "x"
if (!ztime) ztime = 100  // интервал времени, мс
x = 0       /* глобальная переменная, входящая в выражения yexpr и xexpr */
setInterval("move('"+xid+" ', + yexpr + " "+ xexpr +"' ztime)
}
function move(xid, yexpr, xexpr) {
x++ ;
 document.all[xid].style.top = eval(yexpr);
  document.all[xid].style.left = eval(xexpr)
}
```

Параметры:

xid – id движущегося объекта, строка;

yexpr – выражение для вертикальной координаты;

xexpr – выражение для горизонтальной координаты;

ztime – интервал времени между вызовами функции move(), мс.

Пример:

```
<HTML>
<IMG ID = "myimg" SRC = "pict1.gif" STYLE = "posi-tion:absolute">
<SCRIPT>
function curvemove(xid, yexpr, xexpr, ztime) {
// код определения функции
}
function move(xid, yexpr, xexpr) {
// код определения функции
}
Curvemove()'myimg' , "100 + 50*Math.sin(0.03*x)" , "50 + x", 100)
</SCRIPT>
</HTML>
```

Рисование линий. В JavaScript нет специальных встроенных средств для рисования произвольных линий. Для решения этой задачи нужно вывести на экран изображение размером 1×1 пиксел, залитое цветом, отличающимся от цвета фона. Это изображение следует разместить несколько раз в соответствии с координатами, которые задаются параметрами позиционирования top и left атрибута STYLE тега <IMG>. С помощью сценария можно сформировать строку, содержащую теги <IMG> с необходимыми атрибутами, а затем записать ее в документ методом write().

Рисование прямой линии. Для отображения точки в HTML можно использовать следующий тег:

<IMG SRC = "point.bmp" STYLE = "position:absolute; top:y; left:x">

Здесь point.bmp – имя графического файла, содержащего один пиксел; *y*, *x* – числа, указывающие положение графического файла в пикселах. Изображение точки размером 1×1 пиксел можно создать в любом графическом редакторе в файле формата BMP. Задать размеры отображения точки на экране с помощью WIDTH и HEIGHT:

<IMG SRC = "point.bmp" STYLE = "position: absolute; top:y; left:x" WIDTH=n HEIGHT=n>

Пример функции, которая рисует прямую линию с заданными координатами *xl*, *yl*, *x2*, *y2* и толщиной линии n.

```
function line(xl, yl, x2, y2 , n){
/*xl, yl – начало линии, x2, y2 – конец, n – толщина */
var clinewidth = " WIDTH=" + n + "HEIGHT=" + n  /* строка для учета толщины
var xstr = "" // строка тегов для записи в HTML-документ
var xstr0 = ' <IMG SRC="point.bmp"' + clinewidth + ' STYLE = "position:absolute; '
var k = (y2 – yl)/(x2 – xl)          // коэффициент наклона линии
var x = xl                          // начальное значение координаты x
/* Формирование строки, содержащей теги <IMG. . . >: */
while (x <= x2) {
xstr += xstr9 + 'top:' + (yl + k* (x – xl)) + ': left:1 + x + '
x+ +
// запись в документ
```

Рисование кривой линии. Функция curve() для рисования кривой принимает следующие параметры: имя графического файла с изображением точки (может быть любое изображение), выражение, задающее кривую, координаты начала линии, количество точек линии, толщина линии и длина штриха (если потребуется штриховая линия). Далее следует код функции curve() для рисования кривых.

```
function curve(pict_file,yexpr, x0, y0, t, n. s){
/* pict_file – имя графического файла
yexpr – выражение с переменной x
x0, y0 – координаты начала кривой
t – количество точек кривой (значений переменной x)
n – толщина линии
s – длина штриха и паузы */
if (!yexpr) return null
if (!pict_file) pict_file = "point.bmp"
if (!s) s = 0
if (!t) t = 0
var clinewidth = "
if (!n)
clinewidth = 'WIDTH=' + n + 'HEIGHT=' + n
var x
xstrG = '<IMGSRO "' + pict_file + '"' + clinewidth + STYLE = "position:abso-
lute;top: xstr = ""
var i = 0, draw = true
for(x = 0; x < t; x++) {
if (draw)
xstr += xstr0 + (y0 + eval(yexpr)) + '; left:' + (x0 + x)
if (i > s&&s > 0) {
draw = !draw
i = 0
}
```

```
i ++
}
document .write (xstr)      // запись в документ
```

ЗАДАНИЕ 1

Разместить на странице две кнопки: ГРАФИК и ТАБЛИЦА, при нажатии на кнопку ГРАФИК строится анимационный график функции по варианту (таблица).

Задания

№ п/п	$y(x)$	№ п/п	$y(x)$
1	$y = (3^{\sin 2x} - \cos 2x)$	2	$y = \cos(e^{2x})$
3	$y = \dfrac{4\sin 2x}{\cos^2 3x}$	4	$y = \ln\sqrt{\lvert 3x^2 + 4x - 1\rvert}$
5	$y = \cos^2 3x$	6	$y = \cos(x + \dfrac{\pi}{2})$
7	$y = \sin\left(\ln x^2 - \dfrac{\pi}{4}\right)$	8	$y = \dfrac{1}{5}\operatorname{tg}2x$
9	$y = (2x - \ln x^2)$	10	$y = \cos^2 x^2$
11	$y = \operatorname{tg}x \cdot \cos x^2$	12	$y = x^4 \cdot \sin(2x - 1)$
13	$y = \ln\sqrt{1 + 2x^2}$	14	$y = (3x - 1) + \sin\dfrac{\pi}{12}$

При нажатии на кнопку ТАБЛИЦА на страницу выводится таблица со значениями *X* и *У*. *X* изменяется от 1 до 10 с шагом 0,5.

ЗАДАНИЕ 2

Используя созданную ранее карту из лабораторной работы № 21, реализовать передвижение некоторого объекта (самолет, всадник, точка, корабль, путник и т. д) по карте из лабораторной работы № 21. Движение может быть произвольным, или управляемым, или закреплено за кнопками. При попадании объекта на заданную область открывается модальное окно с описанием местности либо объекта.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите методы управления процессами во времени, их параметры.
2. Какие существуют функции для рисования?

Лабораторная работа № 23

БИБЛИОТЕКА CHART.JS

Цель работы: ознакомиться с возможностями библиотеки **chart.js**; научиться строить диаграммы различных видов.

ТЕОРИЯ

Библиотека содержит 8 типов графиков. Графики создаются на основе создания стандартной модели, при этом:

– графики адаптивны;

– можно совмещать разные типы графиков;

– простое API.

Ознакомиться с видами и примерами диаграмм можно по ссылке:

– https://gostash.it/ru/stashes/1655-chartjs-20-krasivye-grafiki-s-ispol-zo-vaniem-html5

– https://code.tutsplus.com/ru/tutorials/getting-started-with-chartjs-line-and-bar-charts--cms-28384

ЗАДАНИЕ

Необходимо построить диаграмму используя библиотеку chart.js. Выбрать тип диаграммы и данные для нее исходя из номера варианта (таблица). При построении диаграмм использовать анимацию, всплывающие подсказки, при оформлении показателей стран – государственную символику.

Задания

Вариант	Данные	Тип диаграммы
1	Изменение цен (https://www.belnovosti.by/belarus-v-cifrah/belstat-rasskazal-chto-bolshe-vsego-podeshevelo-i-podorozhalo-v-yanvare)	Line
2	Сведения о средствах массовой информации на 1 февраля 2019 года (http://mininform.gov.by/activities/statisticheskiy/_)	Bar
3	Государственные расходы на образование (в процентах к ВВП) (https://edu.gov.by/statistics	Radar
4	Общий коэффициент охвата населения начальным образованием (МСКО 1) (процентов) стр. 9 – 10 стран https://edu.gov.by/statistics	Polar Area

Окончание таблицы

5	Общий коэффициент охвата населения средним образованием (МСКО 2 и 3) (процентов) с. 10 https://edu.gov.by/statistics – 10 стран	Pie Doughnut
6	Учреждения дошкольного образования (на конец года) с. 11 https://edu.gov.by/statistics	Bubble
7	Число дневных учреждений общего среднего образования и *численность учащихся в них (на начало учебного года; тысяч человек) – всплывающая подсказка https://edu.gov.by/statistics*	Line
8	Число учреждений профессионально-технического образования *и численность учащихся в них (на начало учебного года, тыс. человек)* https://edu.gov.by/statistics	Bar
9	Число учреждений высшего образования и *численность студентов в них (на начало учебного года)* https://edu.gov.by/statistics	Radar
10	Количество студентов учреждений высшего образования на 10 000 человек населения https://edu.gov.by/statistics	Polar Area
11	Количество учащихся учреждений среднего специального образования на 10 000 человек населения https://edu.gov.by/statistics	Bubble
12	Динамика изменения количества учреждений профессионально-технического образования Республики Беларусь в период с 2005/2006 по 2017/2018 учебные годы, с. 39 https://edu.gov.by/statistics	Line
13	Распределение количества учащихся из стран СНГ (по дневной и заочной формам обучения) приведено в таблице, с. 45 https://edu.gov.by/statistics	Bar
14	Учреждения высшего образования Республики Беларусь https://edu.gov.by/statistics	Radar
15	Распределение количества студентов и магистрантов, с. 52 https://edu.gov.by/statistics	Polar Area

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать диаграмму на оформленной странице.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите виды диаграмму, которые доступны в данной библиотеке.
2. Какие ограничения есть в библиотеке charts?

Лабораторная работа № 24

РАБОТА С COOKIE-ФАЙЛАМИ

Цель работы: освоить приемы работы с cookie-файлами и управления ими.

ТЕОРИЯ

Файл cookie – это ограниченный фрагмент текста, передаваемый в браузер с сайта, который вы посетили. Он помогает сайту запомнить информацию о вас (язык прочтения, рекламные предпочтения, безопасность-логины и т. д.).

Для работы и обработки cookie-файла можно воспользоваться библиотекой JavaScript, которая предоставляет методы для их обработки.

Подключение js-cookie к странице

<script src="/path/to/js.cookie.js"></script>

Запись cookie осуществляется с помощью функции set:

// name – ключ (имя) cookie

// value – значение, связанное с ключом name

// attributes (необязательный параметр) – атрибуты cookie в формате объекта

Cookies.set('name', 'value'[,attributes]);

Установить cookie для всех страниц сайта:

Cookies.set('nameCookie', 'valueCookie'); // => "nameCookie=valueCookie;

path=/" Создать cookie, действительную 30 дней (от текущего момента времени) и видимую любыми страницами сайта:

Cookies.set('nameCookie', 'valueCookie', { expires: 30 }); // => "nameCookie=valueCookie; path=/; expires=Thu, 14 Mar 2020 13:00:15 GMT"

Выполнить запись cookie, доступ к которой будет иметь только текущая страница (срок хранения 365 дней):

Cookies.set('nameCookie', 'valueCookie', { expires: 365, path: '' }) => "nameCookie=valueCookie; expires=Wed, 14 Mar 2020 13:00:36 GMT"

Получение (get) cookie

Чтение значения cookie осуществляется с помощью метода get:

Cookies.get('nameCookie'); // => "valueCookie"

Cookies.get('otherCookie'); // => undefined Получить все доступные cookies:

Cookies.get(); // => {nameCookie: "valueCookie"}

Для удаления cookie предназначена функция remove:

Cookies.remove('nameCookie');

Удаление cookie с указанием атрибута path:

При удалении cookie необходимо указать тот же самый путь и домен, который использовался для его установки. Если атрибуты cookie в Cookies.remove не указать, то будут браться те, которые заданы по умолчанию в Cookies.defaults.

Установка атрибутов cookie

Передача атрибутов при установке cookie осуществляется посредством указания их в качестве значения последнего аргумента функции Cookies.set.

```
Cookies.set('name', 'value', {
  expires: 365,
  path: '/news/',
  domain: 'm.mydomain.ru',
  secure: true
}); // => "name=value; path=/news/; expires=Wed, 14 Mar 2020 12:54:28 GMT; domain=m.mydomain.ru; secure"
```

expires – указывает длительность хранения cookie в браузере. Значение можно задавать как в формате числа (количество дней от текущего момента времени), так и в виде даты. Если данный параметр не указать, то cookie будет сессионным, т. е. он удалится после закрытия браузера.

Например, установим cookie на 30 дней (в качестве формата expires будем использовать дату):

```
// функция, возвращающая дату, которая будет через указанное количество дней от текущей
function getDateExpires(days) {
  var date = new Date;
  date.setDate(date.getDate() + days);
  return date;
}
 запись cookie nameCookie=valueCookie
Cookies.set('nameCookie', 'valueCookie', { expires: getDateExpires(30) });
```

path – строка, указывающая путь (по умолчанию: /), посредством которого будет определяться видимость (доступность) cookie. Cookie для некоторой страницы доступна только тогда, когда ее path входит в адрес. По умолчанию cookie видимы для всех страниц сайта.

Cookie, которая будет иметь в качестве пути текущее местоположение документа:

Cookies.set('nameCookie', 'valueCookie', { path: '' }); // => "nameCookie=valueCookie"

domain – строка, указывающая домен (по умолчанию: текущий), в котором cookie должна быть видна. При этом cookie также будут доступны во всех поддоменах этого домена.

secure – устанавливает, необходимо ли при передаче cookies использовать безопасный протокол (https). По умолчанию: false (нет не требуется).

Кроме индивидуальной установки атрибутов cookie при каждом вызове функции Cookies.set, их также можно назначить глобально посредством свойства defaults объекта Cookie.

```
Cookies.defaults = {
  path: '/',
  expires: 365
};
```

Cookies.set('nameCookie', 'valueCookie'); // => "nameCookie=valueCookie; path=/; expires=Wed, 14 Mar 2020 12:53:23 GMT"

Данные атрибуты будут использоваться функцией Cookies.set по умолчанию.

Cookie и формат JSON

Библиотека js-cookie позволяет установить в качестве значения cookie не только текстовое значение, но и данные в формате массива или объекта (их строковое представление посредством JSON.stringify):

Cookies.set('name', { name1: 'value1', name2: 'value2' });

На выходе получаем строку (а не массив или объект).

Cookies.get('name'); // => "{"name1":"value1","name2":"value2"}"

Cookies.get(); // => {name:"{"name1":"value1","name2":"value2"}"}

Для этого можно воспользоваться функцией Cookies.getJSON. Данная функция не только получает cookie, но и разбирает ее с помощью JSON.parse:

Cookies.getJSON('name'); // => {name1: "value1", name2: "value2"}

Cookies.getJSON('name')['name1']; // => "value1"

Cookies.getJSON('name')['name2']; // => "value2"

Cookies.getJSON(); // => {name: {name1:"value1", name2:"value2"}}

Cookie – эффективное сильное средство для сбора статистики в рекламных целях.

ЗАДАНИЕ 1

Разработать страницу, при загрузке которой запрашиваются логин, пароль, секретное слово. Сохранить это в cookie. Сделать запрос: *показать секретные данные?* Секретные данные отображаются

при корректном вводе секретного слова. Вывести на страницу данные из файла cookie текущего сеанса. Получить текущую дату, установить срок хранения cookie = сегодня+2 дня. Проверить срок годности, изменив дату.

Удалить ваше имя из cookie. Закрыть браузер.

ЗАДАНИЕ 2

Разработать страницу, на которой разместить ваше фото, имя и перечень хобби в зависимости от сезона (4 изображения): например, хоккей – зимой, футбол – летом или вязание – зимой, плавание – летом, осень – походы. При посещении страницы в зависимости от сезона показывать картинку с тем видом хобби, которое соответствует сезону. Добавить приветствие, запросив имя при открытии страницы. Сопровождая примерно таким текстом при последующих открытиях: «О, привет, Максим! Вы посещали эту страницу (время, дата или сезон). Сегодня плаваем, так как наступило лето.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– страницы с авторским дизайном;
– описание функций, использованных при обработке cookie;
– скриншоты экранов с расположением файлов cookie.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего используются файлы cookie?
2. Как отключить файлы cookie в браузере?

Лабораторная работа № 25

REDIS – ХРАНИЛИЩЕ ДАННЫХ ДЛЯ БЫСТРОЙ ОБРАБОТКИ. ТЕХНОЛОГИЯ «ПУБЛИКАЦИЯ – ПОДПИСКА»

Цель работы: изучить работу с хранилищем REDIS и технологию «публикация – подписка».

ТЕОРИЯ

Redis (REmote DIctionary Server) – нереляционная высокопроизводительная СУБД. Redis *хранит все данные в памяти*, доступ к данным осуществляется по ключу. Опционально копия данных может храниться на диске. Этот подход обеспечивает высокую производительность, превосходящую производительность реляционных СУБД. Redis благодаря своей структуре можно использовать как базу данных, систему кеширования или посредник сообщений. Redis поддерживает следующие типы данных: *строки, списки, хеши, множества, сортированные множества*. Особенность в том, что, в отличие от реляционной базы данных, он не имеет четкой структуры таблиц, жестко заданных типов полей. Значения хранятся в виде пары *ключ – значение*.

Типы структур в REDIS

Redis использует концепцию базы данных. В Redis база данных идентифицируется целым числом, которое по умолчанию равняется 0. Сменить (перейти) базу данных – команда select. Например, select 1 выдаст

Redis:
OK
redis 127.0.0.1:6379[1]>.

Обратный переход:
select 0.

По умолчанию можно использовать до 16 БД.

Команды, Ключи и Значения

Ключ – это метка части информации.

Пример ключа: season: summer.

Ключ-*season* – значение – *summer*.

Значение – это данные, ассоциированые с ключом, т. е. summer может быть строкой, числом, сериализованным объектом в виде JSON или др. Redis рассматривает значение как массив байт без учета структуры.

Длина ключа в Redis до 231 байт, длина строки – до 512 Мб, списки и множества могут содержать до 232 элементов, один экземпляр Redis может хранить до 232 ключей.

Особенность Redis заключается в том, что это – однопоточный сервер. Такое решение упрощает поддержку кода, обеспечивает атомарность (нецелостность) операций и позволяет запустить по одному процессу Redis на каждое ядро процессора. При этом каждый процесс будет прослушивать свой порт. Решение нетипичное, но оправданное, так как на выполнение одной операции Redis тратит небольшое количество времени (порядка одной стотысячной секунды).

Типы данных и команды

Строки – основная структура, одна их 5 базовых, а также основа для более сложных структур.

Строки используются:

– в сценариях использования HTML-страниц;

– во избежание уже закодированных данных;

– для возможности хранить объекты как строки.

Команды:

SET: устанавливает пару ключ – значение. Префикс EX устанавливает время недоступности для значения

SET one ' hello Student !!!'

SET two 'student Roman good bye ' EX 1000

GET: возвращает значение для ключа.

Get one

hello Student !!!

DEL – удаляет ключ и значение. *DEL one*;

INCR – увеличивает значение;

INCRBY – увеличивает значение на указанную величину;

EXPIRE – устанавливает время жизни ключа (в секундах).

Списки используются, если нужна ограниченная коллекция из *N* элементов, доступ в которой осуществляется к верхнему или нижнему элементам или когда *N* небольшое.

Основные команды:

LPUSH – добавить значение в начало списка;

RPUSH – добавить в конец списка;

LPOP – возвращает и удаляет первое значение в списке;

RPOP – возвращает и удаляет последнее значение в списке;

LREM – удаляет значения из списка;

LRANGE – возвращает диапазон значений из списка;

LTRIM – изменяет список, оставляя только указанный диапазон.

Множества – неупорядоченный набор данных. Множество хорошо представляет отношения, например, «Каковы друзья пользователя *X*?»

Множества поддерживают сложные операции, такие как пересечение, объединение, вхождение.

Основные команды:

SADD – добавить одно или несколько значений;

SMEMBERS – возвращает все элементы набора;

SINTER – пересечение двух наборов;

SISMEMBER – проверяет вхождение значения в наборе;

SRANDMEMBER – возвращает случайный элемент набора.

Операции в Redis:

– *репликация* с несколькими главными серверами не поддерживается. Каждый подчиненный сервер может выступать в роли главного для других. Репликация в Redis не приводит к блокировкам ни на главном сервере, ни на подчиненных. На репликах разрешена операция записи. Когда главный и подчиненный сервер восстанавливают соединение после разрыва, происходит полная синхронизация (resync);

– *транзакцию* (будут последовательно выполнены либо все операции, либо ни одной) и пакетную обработку команд (выполняем пачку команд, затем получаем пачку результатов) ничто не мешает использовать совместно.

Общие сведения о модели публикация – подписка

Механизм *публикация – подписка* (*Publish/Subscribe*) позволяет поставлять информацию от поставщика к потребителю. Одним из типичных примеров такой информации являются данные на рынке акций и валют, еще пример – подписки на сообщения, твиттер и пр.

В такой модели издателю необязательно знать о местонахождении получателя и наоборот. В модели Request / Reply движение информации начинается по запросу потребителя (клиента). В модели *Publish / Subscribe* движение информации начинается по мере ее появления и поступления от поставщика.

Поставщика информации принято называть *издателем* (publisher). *Издатель* предлагает информацию на определенные темы. Потребитель информации называется *подписчиком* (subscriber), он подписывается на получение информации на определенные темы. Информация, которую получает один *подписчик*, может передаваться другим *подписчикам*.

Информация, посылаемая как сообщение, характеризуется темой. *Издатель* посылает информацию только на те темы, на которые сделана подписка. Взаимодействие между *издателями* и *подписчиками* контролируется посредником, или *брокером* (*broker*). Взаимосвязанные

темы группируются совместно в потоки (stream). *Издатель* может применять потоки для ограничения диапазонов тем издателей и *подписчиков*. Через *брокера* идет поток, который использует все темы. На рисунке представлена схема, иллюстрирующая взаимодействие *брокеров*, издателей и *подписчиков*.

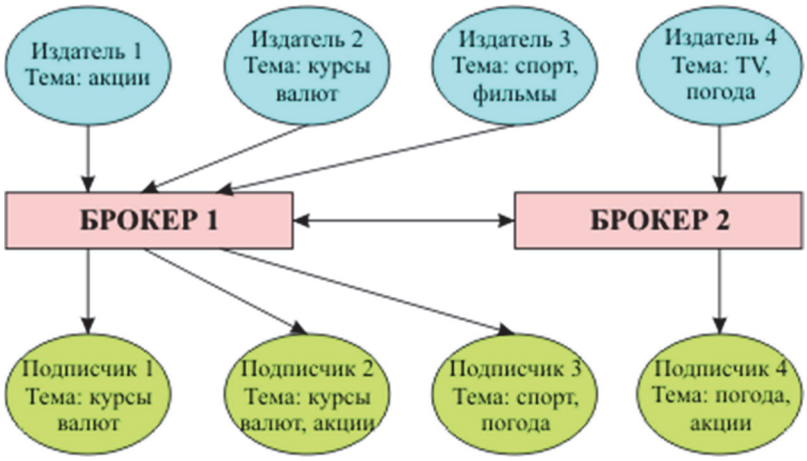


Схема взаимодействия брокеров

ЗАДАНИЕ

Общие указания. Работа выполняется с оформлением дизайна веб-страницы / страниц. Стиль – на ваш выбор. Задания представлены в таблице.

Варианты заданий

1	Используя REDIS (хеши), разработать модуль, считающий количество посещений страницы. Пара *ключ – значение* в этом случае число – ip-адрес. Придумать дизайн страниц с отображением посещенных страниц или количеством их посещений
2	Имеется букмекерская конторка, которая отслеживает результаты достижений по легкой атлетике. Каждый день публикуются результаты трех наилучших и трех наихудших результатов
3	На сайте некий издатель публикует новости книжного магазина. Реализовать механизм подписки и публикации на сайте подписчика новых публикаций
4	Вы – президент некой странной страны, в которой происходит голосование в парламенте. В парламенте 3 партии (название придумаете сами). Итоги голосования обновляются с быстрой частотой, но результаты голосования одной партии блокируются. Все отображать. Дизайн должен учитывать национальные особенности

Окончание таблицы

5	Разработать форму регистрации данных пользователя. При повторном входе предложить новые данные (в случае их изменения), старые удалить, установить срок хранения данных по запросу пользователя
6	Ваш магазин получает данные *ключ-инфо* о клиенте (скорее всего объект, содержащий несколько полей и функций управления ими). Реализовать операции добавления новой пары, операции поиска и операцию удаления пары по ключу
7	Вы – владелец музыкального магазина (на выбор, можно и другой). Идет регистрация и хранение данных посетителей и их предпочтений. Поиск, публикация наиболее предпочтительных товаров, рейтинги
8	Реализовать подписку и публикацию новостей университета, сгруппированную по факультетам
9	Вы – издатель сайта, на котором есть несколько групп подписчиков. Используя **схему**, разработать сайт с подпиской на несколько новостей от нескольких брокеров
10	Сайт с объявлением о конференции. Регистрация на конференцию с публикацией списка зарегистрированных и их докладов
11	На некотором сайте публикуется информация о компьютерах, ноутбуках, планшетах. Организовать подписку на новости о каждой группе товаров
12	Питомник разводит ценных рыбок. Есть несколько типов рыб и условия их содержания разные (тип воды, температура, насыщенность воздухом). Клиент покупает некоторое количество рыб, информация изменяется о текущем состоянии питомника
13	На сайте биржи публикуются котировки трех групп интересов. Реализовать механизм подписки с тремя брокерами

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать:
– страница проекта;
– код с комментариями.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Особенности технологии публикация – подписка.
2. Особенности Redis.
3. Перечислите основные структуры Redis и их особенности.

Лабораторная работа № 26

ТЕХНОЛОГИЯ AJAX

Цель работы: изучить и реализовать основы технологии AJAX; написать клиентское и серверное приложения, обменивающиеся асинхронными данными.

ТЕОРИЯ

AJAX – набор техник разработки веб-интерфейсов, позволяющих делать динамические запросы к серверу без перезагрузки веб-страницы. AJAX обеспечивает динамичность и асинхронность веб-разработок при отсутствии необходимости обновления страниц. AJAX позволяет писать быстрореагирующие веб-приложения, в которых не нужно постоянно обновлять страницы, и AJAX поддерживается всеми браузерами.

AJAX – это технология, которая обрабатывает операции в JavaScript и асинхронно запускает на стороне сервера операции, предоставляющие желаемый результат. В основе технологии AJAX лежит объект XMLHttpRequest.

Пример: создание объекта XMLHttpRequest

```
<script language="javascript" type="text/javascript">
var request = new XMLHttpRequest();
</script>
```

Ajax помещает технологию JavaScript и объект XMLHttpRequest между веб-формой и сервером. При заполнении пользователями формы данные передаются в JavaScript-код, а не прямо на сервер. Вместо этого JavaScript-код собирает данные формы и передает запрос на сервер. Пока это происходит, форма на экране пользователя не исчезает и не блокируется. Другими словами, код JavaScript передает запрос в фоновом режиме и пользователь не замечает, что происходит запрос на сервер. Так как запрос передается асинхронно, то пользователи могут продолжать вводить данные и совершать действия на странице. После получения запроса и его обработки сервер передает данные обратно в JavaScript-код (все еще находящийся в веб-форме), и JS должен иметь алгоритм, который реализует действия, что сделать с полученными данными, который решает, что делать с данными. JavaScript-код используется для управления DOM и для работы со структурой HTML-формы и всеми XML-данными, возвращаемыми сервером.

Объект XMLHttpRequest (кратко «XHR») дает возможность из JavaScript делать HTTP-запросы к серверу без перезагрузки страницы. XMLHttpRequest работает с любыми данными (таблица).

Объект XMLHttpRequest

Методы объекта XMLHttpRequest	
open(method, url, async, user, password)	Конфигурация запроса, например: open('GET', 'phones.json', false);
send(body)	Отослать запрос
abort()	Отмена запроса
setRequestHeader(name, value)	Указать заголовок
getResponseHeader(name)	Получить заголовок
getAllResponseHeaders()	Получить все заголовки
Свойства XMLHttpRequest	
timeout	Максимальная продолжительность асинхронного запроса
responseText	Текст ответа сервера.
status	HTTP-код ответа: 200,404,403, т. д. Если = 0, то сервер не ответил или другой домен.
statusText	Текстовое описание статуса от сервера: OK, Not Found,Forbidden
События	
onreadystatechange	Происходит несколько раз в процессе отсылки и получения ответа. «Текущее состояние запроса» доступно через свойство .readyState
loadstart	Запрос начат
progress.	Можно прочитать текущие полученные данные в responseText
abort	Запрос был отменен вызовом abort()
Error	Ошибка
load	Запрос был успешно (без ошибок) завершён
timeout	Запрос был прекращен по таймауту
loadend	Запрос был завершен (успешно или неуспешно)

Метод OPEN()

Синтаксис:

open(method, URL, async, user, password)

Этот метод вызывается первым после создания объекта XMLHttpRequest.

Параметры запроса:

– Method – HTTP-метод. Как правило, используется GET либо POST, хотя доступны TRACE/DELETE/PUT и т. п;

– URL – адрес запроса. Можно использовать не только http/https, но и другие протоколы, например ftp://, file://.

Внимание! Запрос со страницы можно отправлять только на тот же протокол://домен:порт, с которого она пришла.

Async – false – запрос производится синхронно, если true – асинхронно; – user, password – логин и пароль для HTTP-авторизации, если нужны.

Вызов open не открывает соединение, а лишь настраивает запрос, а коммуникация инициируется методом send.

Синтаксис:

send([body])

Этот метод открывает соединение и отправляет запрос на сервер.

В body находится *тело* запроса. Не у всякого запроса есть тело, например у GET-запросов тела нет, а у POST – основные данные как раз передаются через body.

Пример:

```
Создаем новый объект XMLHttpRequest
var xhr = new XMLHttpRequest();
// 2. Конфигурируем его: GET-запрос на URL 'phones.json'
xhr.open('GET', 'phones.json', false);
// 3. Отсылаем запрос
xhr.send();
// 4. Если код ответа сервера не 200, то это ошибка
if (xhr.status != 200) {
  // обработать ошибку
  alert( xhr.status + ': ' + xhr.statusText ); // пример вывода: 404: Not Found
} else {
  // вывести результат
  alert( xhr.responseText ); // responseText – текст ответа.
}
```

Для более легкой работы с HTML при помощи JavaScript были разработаны несколько фреймворков, которые упрощают повседневные рутинные функции и помогают программисту не задумываться о реализации примитивов, например доступа к полям DOM-а, их изменение

стандартными средствами JavaScript, а предоставляют удобные инструменты для данных функций.

Библиотека jQuery предоставляет удобный API по работе с Ajax, так как jQuery является более простой для изучения, то остановимся на ней подробнее.

Возможности:

– переход по дереву DOM включая поддержку Xpath как плагина;

– события;

– визуальные эффекты;

– AJAX-дополнения.

Добавим в форму дополнительное поле, в котором будем изменять значение параметра учета количества сделанных изменений.

Добавление дополнительного поля в форму:

```
<form>
 <p>City: <input type="text" name="city" id="city" size="25"
onChange="getResult();" /></p>
 <p>ZipCode: <input type="text" name="zipCode" id="zipCode"
size="5"
readonly /> получен с помощью AJAX запроса</p>
 <p>Значение было изменено <input type="text" name="jq" id="jq"
size="3" value="0" readonly /> раз с помощью jQuery</p>
</form>
```

Дополним функцию updatePage() вызовом дополнительной функции jq(), которая, используя механизмы jQuery, будет изменять данные в поле с id=”jq”, в котором указывается количество обновлений страницы:

Исходный код функции jq()

```
function jq() {
 //Записываем в переменную X значение поля с id='jq'
   $x = $("#jq").attr("value");
 //Увеличиваем переменную на 1
   $x++;
 //Обновляем значение атрибута value поля с id='jq'
   $("#jq").attr("value", $x);
}
```

ЗАДАНИЕ

Внимание!!! Страница не перегружается при переходах

1. Проверка логина при регистрации – проверка на существование вводимого логина в базе данных пользователей при наборе логина в соответствующем поле.

2. Корзина товаров в онлайн-магазине. Добавление / удаление.
3. Города и области – выборка области по выбранному городу.
4. Подсчет хеша при заполнении строки.
5. Транслитерация.
6. Простейший почтовый клиент – выборка темы письма по запросу и по клику на теме письма, доставка его содержимого.
7. Мониторинг сервисов на ПК. Опрос списка запущенных процессов и асинхронное обновление.
8. Гостевая книга, Комментарии – добавление / удаление.
9. Чат.
10. На странице есть форма обратной связи. После заполнения данные появляются на странице без перегрузки страницы.
11. Показать: кто, куда и когда кликал за указанный период времени по вашим страничкам.
12. Создание функции рейтинга (проставить количество лайков на несколько изображений).

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем заключается суть синхронных и асинхронных запросов к серверу?
2. Можно ли установить продолжительность ожидания ответа?
3. Может ли объект работать с данными JSON?

Лабораторная работа № 27

ПРЕПРОЦЕССОР HTML И ШАБЛОНИЗАТОР PUG (JADE)

Цель работы: получить навыки работы с шаблонизаторами.

ТЕОРИЯ

Шаблонизатор в веб – это программное обеспечение, позволяющее использовать html-шаблоны для генерации конечных html-страниц.

Основная цель использования шаблонизаторов – это отделение представления данных от исполняемого кода. Часто это необходимо для обеспечения возможности параллельной работы программиста и дизайнера-верстальщика. Данный подход значительно ускоряет время разработки и прототипирования приложения, дизайнеру не нужно вникать в программирование, а программисту – беспокоиться об интерфейсе.

Использование шаблонизаторов улучшает читаемость кода и упрощает внесение изменений во внешний вид, когда проект целиком выполняет один человек.

Шаблон – это строка в специальном формате, которая путем подстановки значений (текст сообщения, цена и т. п.) и выполнения встроенных фрагментов кода превращается в DOM/HTML (таблица).

Шаблонизаторы

Название	Документация	Характеристика
doT.js	http://olado.github.io/doT/	Простой и легкий. Используется с Node.js и браузера
HyperScript	https://github.com/hyperhype/hyperscript	Ориентирован на клиент-сервер
Marko	https://github.com/marko-js/marko	Ориентирован для создания динамических пользовательских интерфейсов, поддерживает как однофайловые, так и многофайловые компоненты
Mustache	http://mustache.github.io/mustache.5.html	Совместим с Node, поддерживается многими языками

ЗАДАНИЕ

Pug – это краткий и простой язык шаблонов с сильным акцентом на производительность и мощные функции.

Установить Jade(Pug) с помощью Node: $ npm install pug

Установка последней версии $ npm install pug-cli -g

Синтаксис здесь: http://jade-lang.com/

Создать такой jade-шаблон, чтобы в результате получилась html-страничка на произвольную тему вида (добавьте div):

```
<!DOCTYPE html>
<html>
<head>
   <title>Ваш заголовок</title>
<script  type="text/javascript">
       if  (foo)  {
          bar(1  +  5)
       }
        </script>

</head>
<body>
<p><a href="ind.htm">Главная</a></p>
            <p align="center">Что такое Интернет? </p>
<p id="bold">Небольшой по объему абзац</p>
<p class="blue"> Длинный текст </p>

<p>Рано или поздно каждый человек, бороздящий просторы сети интернет, задается вопросом: Что же это такое интернет? </p>
<p>Ответ как ни странно прост. Интернет – это обычная сеть, такая же, как сеть электропроводки в здании, городе, стране и т. д. Это совокупность проводов от дешевых до очень дорогих, а иногда и радиосигналов и сигналов спутника. Самым главным элементом этой сети является то, что на концах каждого провода находится компьютер или телефон, или еще какое-либо устройство цифрового мира.

Информация в интернете – фильмы, музыка, стихи, книги, открытки и т. д. появляется благодаря пользователям, т. е. тем, кто на концах этой сети. Чем больше пользователей, тем больше в интернете информации. На сегодняшний день интернет является местом циркуляции и всемирным хранилищем информации. </p>
<ul>
   <li>xxx</li>
```

```
        <li>xxx</li>
      </ul>
<p>Интернет начал зарождаться в 1957 г. в США как ответ на запуск спутника Советским Союзом. Планировалось создать надежную сеть для передачи электронных сообщений на случай войны. Первая сеть носила имя ARPANET. Она объединила четыре научных центра США. Это были университеты в Лос-Анжелесе, Стендфорде, Санта-Барбаре и штате Юта. Позже к этой сети присоединились и другие научные центры. </p>
<p>У первой сети ARPANET вскоре появился серьезный конкурент. </p>
<a href="xxx.xx"> XXX </a>
</body>
</html>
```

1. Вывести HTML-страницу несколькими способами.
2. Создать композитный шаблон из разных самостоятельных частей при помощи инструкции include. Предварительно свой шаблон разбить минимум на 3 части.

Для шаблонизатора можно использовать любой свой сайт.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите виды шаблонизаторов.
2. Всегда ли оправдано использование шаблонизаторов?
3. Назовите основные характеристики PUG.

Лабораторная работа № 28

БИБЛИОТЕКА BACKBONE.JS

Цель работы: освоить приемы работы с библиотекой backbone.js.

ТЕОРИЯ

Ряд современных JS фреймворков позволяет организовывать свой код с помощью паттерна MVC (Model-View-Collection). Он состоит из следующих модулей:

– Views (представления);
– Models (модели);
– Collections (коллекции);
– Events (события);
– Routers (маршрутизаторы, роутеры).

Backbone – это javascript-библиотека для приложений, в которых логика интерфейса ложится на браузер.

В качестве ядра библиотеки используются наследуемые «классы».

Router и History принимают URL и говорят, какой View надо запустить.

View привязывается к dom-элементам и в зависимости от их вложенности отвечает за их данные. Именно View вызываются из Route, чтобы открыть прошлое состояние, и View реагирует на пользователя через подписку на события (поле events).

Collection – массив моделей привязывается к View и может оповещать его об изменениях.

Model – основные классы сущностей, имеют URL для получения и изменения данных.

Подключение:

```
<script src = ”js/jquery.js”></script>
<script src = ”js/underscore.js”></script>
<script src = ”js/backbone.js”></script>
```

Представления:

Для создания представления нужно расширить класс Backbone.view.

Представления имеют четыре основных свойства:

1. el состоит из 4 атрибутов:

а) className;

б) eltagName;

в) id;

г) attribute.

Если один из них не указан, то this.el является пустым div.

2. initialize – возможность передать параметры, которые будут прикреплены к модели, коллекции или View.el.

3. render – вставляет отметки в элементы.

4. events – события записываются в следующем формате:

{“<EVENT_TYPE><ELEMENT_ID>”:”<CALLBACK_FUNCTION>”}

Модель содержит некоторое состояние приложения (JSON), называемое Backbone-атрибутами, а также события, позволяющие управлять изменением состояния. Упорядоченная группа моделей – это коллекция. Модели содержат интерактивные данные и логику, связанную с ними, такую как получение, установка и проверка данных, значения по умолчанию, инициализация данных, преобразования и другие, т. е. модели хранят данные вашего приложения.

Backbone-модели можно создать путем вызова функции:

```
var Book = Backbone.Model.Extend({});
//Каждый экземпляр может содержать некоторые начальные поля.:
var Book = Backbone.Model.extend({
Defaults: {
Title: ‘no titile’,
Author: ‘unknown’,
releaseDate: 2018,
description: ‘’
}
});
```

Коллекции – это упорядоченные множества моделей, в которых можно получить и установить модели в коллекции, прослушивать события, когда любой элемент в коллекции изменяется, и изменять данные модели с сервера. Это аналог результата запроса в базах данных.

Определить коллекцию можно так:

```
var BooksCollection = Backbone.Collection.extend({ model: Book });
```

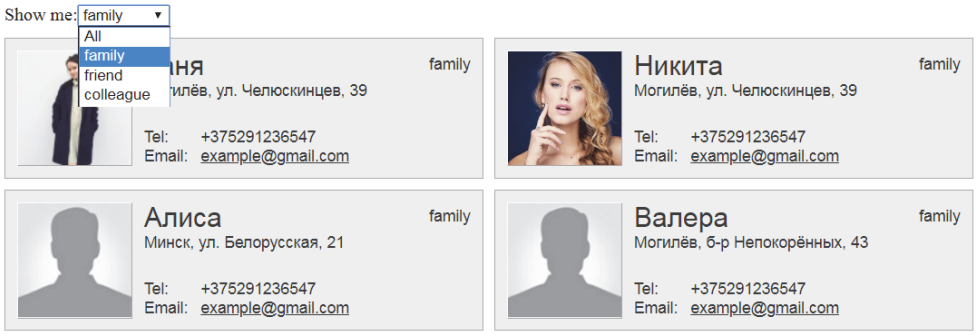
С моделями коллекций можно совершать следующие действия – сортировать, фильтровать и т. д.

```
var BooksCollection = Backbone.Collection.extend({
model: Book,
old: function() {
return this.filter(function(book) {
return book.get(‘releaseDate’) < 1900;
}); } });
```

Пример создания и получения элемента коллекции с id = 0.

```
var oBooks = new BoksCollection();
oBooks.get(0);
```

Пример. Создание менеджера контактов с использованием Backbone



Вид страницы контактов

При выборе категории отображаются отобранные контакты

1. Добавить модели в коллекцию.

Создайте форму с id = «addContact», которая должна содержать следующие области: фото, тип, имя, адрес, телефон, email, а также кнопку добавить. Необходимо следить за тем, чтобы значения атрибутов элементов <input> совпадали с теми полями, что вы используете в модели.

Затем нужно добавить обработчик события в главное представление для того, чтобы можно было собрать данные при отправке формы; добавить следующий код после всех существующих элементов типа ключ – начение в объекте событий:

```
"click #add": "addContact"
```

Будем отслеживать событие клика по элементу с id, равным add (кнопка в форме). Обрабатывать клик мы будем в методе addContact. Добавьте следующий код после метода thefilterByType:

```
addContact: function (e) {
    e.preventDefault();

    var newModel = {};
    $("#addContact").children("input").each(function (i, el) {
        if ($(el).val() !== "") {
            newModel[el.id] = $(el).val();
      }
    });

    contacts.push(formData);

    if (_.indexOf(this.getTypes(), formData.type) === -1) {
        this.collection.add(new Contact(formData));
        this.$el.find("#filter").find("select").remove().end().append(this.createSelect());
    } else {
        this.collection.add(new Contact(formData));
    }
}
```

2. Отобразить новый контакт на странице.

Теперь необходимо отобразить новую модель на странице. Для этого потребуется еще один обработчик события. Прикрепим его к событию add. Добавьте следующий фрагмент в метод initialize() коллекции:

```
this.collection.on("add", this.renderContact, this);
```

Метод on() используется для отслеживания добавления нового контакта и вызова уже существующего метода для формирования HTML-кода, который будет отображен на странице.

Если все поля формы addContact будут пустыми, то в дальнейшем могут возникнуть проблемы с моделью. Одно из решений – это выставление значений по умолчанию, в случае если данные не введены. Если это невозможно сделать, то в качестве значений можно передать пустые строки. Обновите объект класса Contact для назначения значений полей по умолчанию:

```
name: "",
address: "",
tel: "",
email: "",
type: ""
```

3. Удалить модель из коллекции.

Создайте для каждого контакта свою кнопку удаления.

Процесс удаления модели может быть добавлен в класс представления, который является по сути каждым отдельным контактом. Итак, нам нужно добавить событие и обработчик события «клика по кнопке» удаления. Добавьте следующий код в конец класса ContactView:

```
events: {
    "click button.delete": "deleteContact"
},

deleteContact: function () {
    var removedType = this.model.get("type").toLowerCase();

    this.model.destroy();

    this.remove();

    if (_.indexOf(directory.getTypes(), removedType) === -1) {
        directory.$el.find("#filter select").children("[value='" + removedType + "']").remove();
    }
}
```

Мы используем объект событий для того, чтобы назначить перечисленное событие. На этот раз мы слушаем событие клика по элементу <button> для удаления контакта. Обработчик данного события – метод deleteContact. Сохраняем тип контакта, который хотим удалить. Преобразуем данное значение к нижнему регистру, чтобы избежать непредвиденных ситуаций. Затем вызываем метод destroy() от объекта модели. Мы также удаляем HTML-код, отображающий контакт на странице с помощью jQuery-метода remove() и очищаем все события и обработчики событий, ассоциированные с данным элементом.

Потом проверяем тип всех моделей в коллекции для того, чтобы знать, оставить ли тип удаленного контакта в элементе <select>. Если никакая модель не относится к данному типу, то преобразуем элемент <select> и отображаем его новое содержимое на странице.

Извлекаем элемент <option> со значением в атрибуте value, совпадающим со значением переменной removedType, которое мы сформировали раньше.

4. Удалить данные из модели.

Помимо удаления контакта из коллекции и представления, необходимо удалить его из массива. Если этого не сделать, то модель может возникнуть позже, при фильтрации.

Функционал удаления данных из оригинального массива можно расположить в главном представлении. К коллекции мы прикрепим событие remove и назначим обработчик для того, чтобы мы могли быстро определить, когда нужно удалить данные из массива. Добавим следующий код сразу после существующих привязок событий к модели:

```
this.collection.on("remove", this.removeContact, this);
```

Далее создаем метод removeContact(). Добавляем его после метода addContact():

```
removeContact: function (removedModel) {
    var removed = removedModel.attributes;

    if (removed.photo === "/img/placeholder.png") {
        delete removed.photo;
    }

    _.each(contacts, function (contact) {
        if (_.isEqual(contact, removed)) {
            contacts.splice(_.indexOf(contacts, contact), 1);
        }
    });
}
```

Спрятать форму и отображать ее при клике на специальную кнопку. Добавьте следующий код к элементу <header>:

```
<a id="showForm" href="#">Add new contact</a>
```

Для начала нужно скрыть форму и отображать при клике на специальный элемент. Добавим обработчик события к классу DirectoryView:

```
"click #showForm": "showForm"
```

Метод showForm():

```
showForm: function () {
    this.$el.find("#addContact").slideToggle();
}
```

ЗАДАНИЕ

Разработать менеджер задач с функцией добавления, удаления, сортировки по важности задачи (высокая, средняя, низкая).

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите свойства представлений.
2. Что такое коллекции?

Лабораторная работа № 29

ПРЕПРОЦЕССОР SASS

Цель работы: освоить работу с препроцессором SASS.

ТЕОРИЯ

Препроцессор SASS используется для расширения функционала CSS и помогает упорядочить и облегчить работу с большими таблицами стилей. SASS позволяет использовать такие функции, как:

– переменные;

– вложенности;

– миксины;

– наследование и др.

Переменные. Используются для хранения цветов, шрифтов, размеров и т. д. Переменная начинается с символа **$**

```
$prim-color: #3ff;
$font-stack: Helvetica, sans-serif;
body {
  font: 100% $font-stack;
  color: $prim-color;
}
```

Фрагментирование. Можно создавать фрагменты SASS-файла, содержащие в себе небольшие отрывки CSS, которые можно будет использовать в других SASS-файлах. Это делает CSS модульным и облегчает его обслуживание. *Фрагмент* – это простой SASS-файл, имя которого начинается с нижнего подчеркивания, например **_part.scss.** Нижнее подчеркивание в имени SASS-файла говорит компилятору о том, что это только фрагмент и он не должен компилироваться в CSS. Фрагменты SASS подключаются при помощи директивы @import.

Импорт. CSS имеет возможность импорта, которая позволяет разделить ваш CSS-файл на более мелкие и облегчить их в обслуживании. Но у этого способа есть весомый недостаток: каждый раз, когда в CSS используете @import, то в CSS создается еще один HTTP-запрос. SASS берет идею импорта файлов через директиву @import, но вместо создания отдельного HTTP-запроса SASS импортирует указанный в директиве файл в тот, в котором он вызывается, т. е. на выходе получается один CSS-файл, скомпилированный из нескольких фрагментов.

Например, есть несколько фрагментов SASS-файлов – _reset.scss и base.scss. И мы хотим импортировать _reset.scss в base.scss.

```
// _reset.sass
html,
body,
ul,
ol
  margin:  0
  padding: 0

// base.sass
@import reset
body
  font: 100% Helvetica, sans-serif
  background-color: #efefef
```

Миксины (примеси) позволяют создавать группы деклараций CSS, которые могут использовать по нескольку раз на сайте. Для создания миксина используется директива **@mixin + название** миксина, в миксине используется переменная $radius внутри скобок, тем самым позволяя себе передавать в переменной все, что надо. После того, как создали миксин, можно его использовать в качестве параметра CSS, начиная вызов с **@include** имя миксина.

При определении миксина аргументы пишутся как имена переменных, разделенные запятыми внутри круглых скобок сразу после имени. Затем при подключении миксина значения могут быть переданы аналогичным образом. Например:

```
@mixin sexy-border($color, $width) {
  border: {
    color: $color;
    width: $width;
    style: dashed;
  } }

p { @include sexy-border(blue, 1in); }
компилируется в
p {
  border-color: blue;
  border-width: 1in;
  border-style: dashed; }
```

В миксинах также можно определять значения аргументов по умолчанию, используя обычный синтаксис установки значений переменных.

Затем при подключении миксина, если ему не будет передан аргумент, будет использовано значение по умолчанию. Например:

```
@mixin sexy-border($color, $width: 1in) {
  border: {
    color: $color;
    width: $width;
    style: dashed;
  }
}
p { @include sexy-border(blue); }
h1 { @include sexy-border(blue, 2in); }
```

компилируется в

```
p {
  border-color: blue;
  border-width: 1in;
  border-style: dashed; }

h1 {
  border-color: blue;
  border-width: 2in;
  border-style: dashed; }
```

Расширение / наследование. Это одна из самых полезных функций SASS. Используя директиву @extend, можно наследовать наборы свойств CSS от одного селектора другому. Это позволяет держать SASS-файл в «чистоте». В нашем примере мы покажем вам, как сделать стили оповещений об ошибках, предупреждениях и удачных исходах, используя другие возможности SASS, которые идут рука-об-руку с расширением, классами-шаблонами.

Класс-шаблон – особый тип классов, который выводится только при использовании расширения, что позволит сохранить скомпилированный CSS чистым.

Синтаксис SASS:

```
// Данный отрывок кода не попадет в CSS, так как %equal-heights никогда не расширялся.
%equal-heights
  display: flex
  flex-wrap: wrap
// Данный отрывок кода попадет в CSS потому, что %message-shared расширен.
```

```
%message-shared
  border: 1px solid #ccc
  padding: 10px
  color: #333
.message
  @extend %message-shared
.success
  @extend %message-shared
  border-color: green
.error
  @extend %message-shared
  border-color: red
.warning
  @extend %message-shared
  border-color: yellow
```

Данный код сообщает классам .message, .success, .error и .warning вести себя как %message-shared. Это означает, что где бы ни вызывался %message-shared, то и .message, .success, .error и .warning тоже будут вызваны. Магия происходит в сгенерированном CSS, в котором каждый из этих классов получает CSS-свойства, как и %message-shared. Это позволит избежать написания множества классов в HTML-элементах.

Можно расширить большинство простых CSS-селекторов прибавлением к классам-шаблонам в SASS, однако использование шаблонов – простейший способ быть уверенным, что вы не расширяете класс везде, где он используется в стилях, что могло бы привести к непреднамеренным наборам стилей в заданном CSS.

При генерировании CSS он будет выглядеть так, как пример, представленный ниже. Обратите внимание, %equal-heights не попадает в CSS, так как ни разу не был использован.

```
.message, .success, .error, .warning {
  border: 1px solid #cccccc;
  padding: 10px;
  color: #333;
}
.success {
  border-color: green;
}
.error {
  border-color: red;
}
```

```
.warning {
  border-color: yellow;}
```

Математические операторы. Использовать математику в CSS очень полезно. SASS имеет несколько стандартных математических операторов, таких как +, –, *, / , %. В рассматриваемом примере совершаются простые математические вычисления для расчета ширины aside и article.

Синтаксис SASS:

```
.container
  width: 100%
article[role="main"]
  float: left
  width: 600px / 960px * 100%
aside[role="complementary"]
  float: right
  width: 300px / 960px * 100%
```

В примере создается простая адаптивная модульная сетка с шириной в 960 пикселей. Используя математические операторы, полученные данные с пиксельными значениями конвертируются в процентные. Скомпилированный CSS выглядит так:

```
.container {
  width: 100%;
}
article[role="main"] {
  float: left;
  width: 62.5%;
}
aside[role="complementary"] {
  float: right;
  width: 31.25%;
}
```

Установка SASS:

https://sass-scss.ru/install/

Подробно о SASS по ссылке: https://sass-scss.ru//

ЗАДАНИЕ

Взять за основу сверстанный макет в классическом стиле.

1. Используя SASS, определить стили для элементов страницы. Скомпилировать SASS-файл, отобразить страницу.

2. В SASS переобозначить цвет. Скомпилировать SASS-файл, отобразить страницу. Сделать скриншоты (файлы sass, css, html), результат.

3. Переменные для цвета, размера, фона вынести в отдельный файл. Подключить в основном директивой @import “ _file”. Для изменения цветов использовать арифметические операции.

4. Ввести оператор @if, проверить цвет любого элемента и изменить на другой, типа так:

```
$color: black;
.link {
@if $color == black {
color: white;
} @else {
color: black;
} }
```

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен содержать скриншоты этапов выполнения, скриншоты SASS, CSS, HTML, результата.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите задачи препроцессора SASS.
2. Какие возможности имеет препроцессор SASS?

Лабораторная работа № 30

ОСНОВЫ РАБОТЫ С ANGULARJS

Цель работы: освоить приемы работы с AngularJS

ТЕОРИЯ

AngularJS – открытый фреймворк, работающий на модели MVC.

Бизнеслогика отделена от кода интерфейса, что позволяет улучшить тестируемость и расширяемость приложений. Также в нем реализовано двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом. Таким образом, AngularJS синхронизирует модель и представление.

AngularJS поддерживает такие функциональности, как Ajax, управление структурой DOM, анимация, шаблоны и т. д.

Официальный сайт фреймворка: http://angularjs.org/.

Начало работы с AngularJS:

Загрузить или скачать:

https://ajax.googleapis.com/ajax/libs/angularjs/1.7.8/angular.min.js.

При загрузке zip-пакета, кроме самой библиотеки (angular.js), имеется еще ряд файлов и их минимизированные версии:

– angular-touch.js – поддержка событий сенсорного экрана;

– angular-animate.js – анимации;

– angular-aria.js – поддержка aria-атрибутов (accesible rich internet application);

– angular-mocks.js – mock-объекты для юнит-тестирования;

– angular-route.js – обеспечивает механизм маршрутизации;

– angular-sanitize.js – предоставляет функционал для управления потенциально опасным контентом (javascript, html);

– angular-cookies.js – управление cockies;

– angular-loader.js – загрузка angularjs-скриптов;

– angular-messages.js – вывод сообщений;

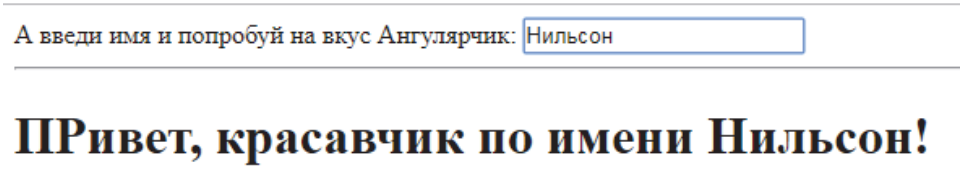
– angular-resource.js – работ с ресурсами.

Пример:

```
<!doctype html>
<html ng-app> // объявление, что это корневое приложение для ANG
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.8/angular.min.js"></script>
  </head>
```

```
<body>
  <div>
    <label>  А введи имя и попробуй на вкус Ангулярчик:</label>
    <input type="text" ng-model="yourName" placeholder="Введи имя здесь"> //
указывает модель "name", к которой будет привязано значение элемента input.
    <hr>
    <h1> ПРивет, красавчик по имени {{yourName}}!</h1> // Двойные скобки,
собственно передают значение из input, на страницу
  </div>
 </body>
</html>
```

Результат представлен на рисунке.



ПРивет, красавчик по имени Нильсон!

Результат работы

Для работы с AngularJS подключаем директивы, начинающиеся с префикса **ng-**.

Для выполнения задания используйте директивы Angular (ng-click, ng-hide, ng-show) для задания и считывания активной переменной. Ее изменение инициирует изменения HTML-кода элемента. В терминологии Angular такая переменная называется моделью. Она доступна для всех директив текущей области. Кроме того, примените шаблоны JavaScript с синтаксисом команды {{VAR}}. Когда фреймворк видит такую строку, он заменяет содержимое переменной. Эта операция повторяется каждый раз при изменении переменной.

При клике на пункт меню на странице отображается ВЫ ВЫБРАЛИ название пункта меню.

ЗАДАНИЕ

Необходимо реализовать сайт на тему, представленную в таблице вариантов.

Варианты

1	Сайт доставки еды. Сперва выбор категории еды (пицца, лапша, суши). Затем выбор конкретного блюда (в каждом блюде дополнительная информация, изображение). На сайте должна быть форма заказа (имя, адрес, время доставки). После выбора заказа отображается заказ и его стоимость

Окончание табл.

2	Сайт портфолио. В меню – о себе, оставить заказ, просмотреть работы
3	Магазин товаров для животных. О нас: товары для кошек, товары для собак. Заказать. Контакты
4	Сайт любителей путешествовать в Китай
5	Незнакомая Беларусь
6	Про суши и не только
7	Разговор о моде
8	Брокерская контора
9	Караоке
10	Рецепты блюд
11	Сайт техники. Категории – смартфоны, планшеты и т. д.
12	Сайт для покупки билетов на квест
13	Сайт, который предоставляет варианты подарка друзьям, семье т. д.
14	Достопримечательности города
15	Сайт доставки цветов

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. С какого префикса начинаются директивы Angular?
2. Особенности Angular.

Лабораторная работа № 31

EMBER.JS

Цель работы: получить навык работы по созданию веб-приложений с использованием Ember.js.

ТЕОРИЯ

Ember.js – свободный JavaScript каркас веб-приложений, реализующий MVC-шаблон, предназначенный для упрощения создания масштабируемых одностраничных веб-приложений. Фреймворк используется такими компаниями, как TED, Yahoo!, Twitch.tv и Groupon.

***Основные принципы*:**

1. Маршруты являются одним из основополагающих принципов Ember.js и подчеркивают важность URL в управлении состоянием приложения. Маршруту объекта соответствует URL-адрес, который определяет текущее состояние приложения. Маршруты определены в единственном объекте маршрутизатора.

2. Модели. Каждому маршруту соответствует модель, в которой содержатся данные, соответствующие текущему состоянию приложения. И несмотря на то, что есть возможность использовать jQuery, чтобы загружать с сервера JSON-объекты, большинство приложений использует для этих целей библиотеку с моделью данных.

3. Контроллеры. Используются для того, чтобы добавить модели некую логику отображения. Ранее стандартной практикой было наследовать контроллер от ObjectController, если модель представляла собой один объект, и от ArrayController – если модель была массивом записей. Сейчас эти базовые классы устаревшие и нормальной практикой считается обращение к свойствам модели из Ember.Controller.

4. Шаблоны написаны на языке HTMLBars (HTML + handlebars = HTMLbars) и описывают пользовательский интерфейс. Шаблоны используются для построения HTML-кода приложения и позволяют встраивать в него динамически обновляемые выражения.

Его отличительные черты:

– рассчитан на визуализацию, на фронтенд;

– поддержка компилятора JS Babel для поддержкки TypeScript;

– brocolli.js для сборки ассетов (неделимых сущностей);

– шаблонизация с использованием Handlebars http://handlebarsjs.com/);

– навигация на основе роутинга;

– поддержка JSON API;

– необходима последняя версия Node.js и npm. Скачать по ссылке: Node.js.

Бекенд фреймворком не управляется.

Использование Ember может сильно повысить эффективность разработки фронтенда. Ember дает весь необходимый функционал без дополнительных усилий. Использование ember-cli и настроенных процессов сборки приложения упрощает процесс разработки. Ember сложно встраивать в существующий проект. Он хорошо работает для старта нового проекта.

Документация по Ember по ссылке: https://flexberry.github.io/ru/gbt_emberjs.html, https://guides.emberjs.com/release/getting-started/quick-start/.

ЗАДАНИЕ

Реализовать одностраничный сайт с функционалом на тему исходя из варианта.

Варианты

1	Сайт салона автомобилей
2	Сайт аренды велосипедов в парке
3	Сайт проката фортепиано
4	Игра в кости с ограниченным количеством бросков
5	Сайт проката платьев
6	Результаты сессии (студенты, оценки и т. д.)
7	Телефонный справочник
8	Сайт компании (количество работников, рейтинг)
9	Сайт доставки еды
10	Сайт проката детских товаров
11	Сайт проката скутеров
12	Список участников конкурса
13	Участники проекта, их должности
14	Список преподавателей, предметы
15	Сайт проката фототехники

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Особенности Ember.
2. Перечислите примеры проектов, в каких может быть задействован Ember.

Лабораторная работа № 32

БИБЛИОТЕКА KNOCKOUT.JS

Цель работы: ознакомиться с принципами разработки веб с использованием библиотеки.

Ссылки:
http://flash2048.com/post/Knockout-simple-app
http://learn.knockoutjs.com/#/?tutorial=intro
https://habr.com/ru/post/121926/

ЗАДАНИЕ

Разработать страницу с соответствующей варианту темой. При вводе данных в поля формы на странице обновляется список зарегистрированных (таблица).

Варианты

1	Регистрация на самолет
2	Регистрация на конференцию
3	Список покупок
4	Бронирование билетов на концерт
5	Регистрация на ЦТ
6	Регистрация на курсы
7	Регистрация на закрытую вечеринку
8	Регистрация на мастер-класс
9	Регистрация на прием к врачу
10	Регистрация на конкурс
11	Список возможных и утвержденных гостей на день рождения
12	Регистрация на фотосессию
13	Регистрация на открытие выставки
14	Регистрация на экскурсию
15	Бронирование мест в маршрутку (максимум 15 мест)

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите особенности библиотеки.
2. В чем отличие библиотек от фреймворков?

Лабораторная работа № 33

CSS-ПРЕПРОЦЕССОР LESS

Цель работы: изучить особенности использования препроцессора LESS.

ТЕОРИЯ

Отработайте полученные на лекции знания, выполнив следующий цикл заданий: https://htmlacademy.ru/courses/85/run

Скачайте Less с официального сайта по этой ссылке: http://lesscss.org/#download-options

или с сайта GitHub: https://github.com/less/less.js/archive/master.zip

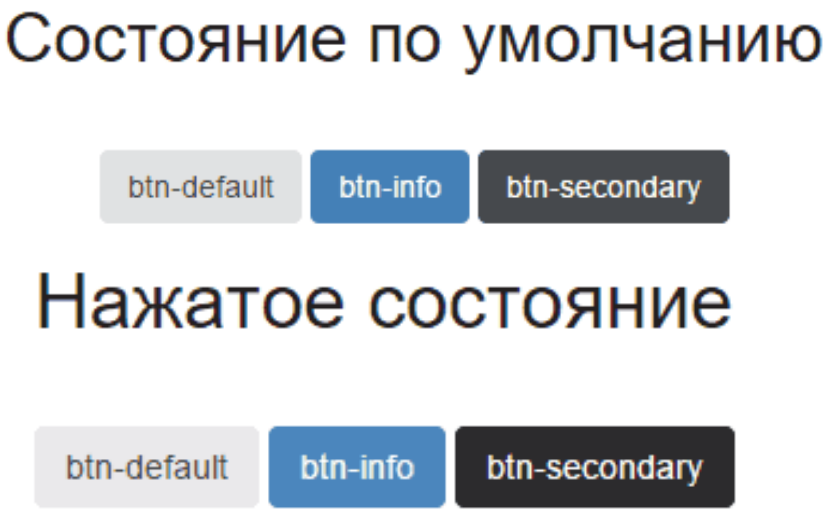
или установите с помощью npm. Не забудьте подключить его к своему документу.

ЗАДАНИЕ

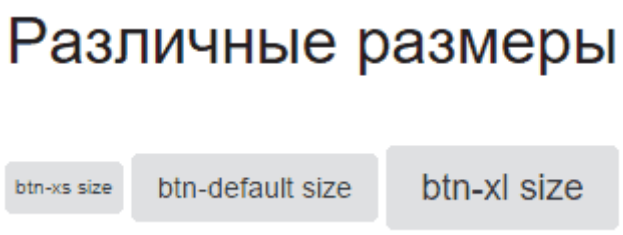
Каждая кнопка имеет общий стиль, но при этом различается размерами и цветами. Обязательным условием выполнения будет использование переменных для хранения цветов, размеров и прочих величин, которые могут поменяться на любой стадии развития проекта, а также примесей, которые будут менять размер и цвет кнопки в зависимости от передаваемых параметров.

Ниже приведены изображения кнопок, которые необходимо разработать.

Состояние до нажатия и после



Размеры кнопок



Для выполнения задания для вас подготовлена таблица со всеми значениями переменных, которые могут понадобиться в работе.

Цвет фона и шрифта

Свойство	По умолчанию	Второй тип	Третий тип
background	#e0e0e0	#3b83c0	#404245
color	#333	#fff	#fff
hover background	#e8e8e8	#458ac6	#1b1c1d
hover color	#333	#fff	#fff

Размеры

Свойство	Маленький	Стандартный	Большой
font-size	10px	14px	18px
line-heigth	20px	20px	20px
padding	2px 4px	6px 12px	10px 16px

Пример использования примесей:

```
.mix{
        width: 100px;
        height: 100px;
}
p{
        .mix;
        color: red;
}
```

Результат выполнения кода:

```
.mix {
        width: 100px;
```

```
        height: 100px;
}
p {
        width: 100px;
        height: 100px;
        color: red;
}
```

С помощью LESS-функции spin можно повернуть цветовое колесо на определенный угол относительно заданного цвета и получить новый цвет. Функция принимает два параметра.

Синтаксис:

spin(цвет, угол_поворота)

Цвет можно задавать в любом цветовом формате. Значение угла может быть как положительным, так и отрицательным. При положительном угле функция повернет колесо по часовой стрелке, при отрицательном – против.

Примеры:

color: spin(red, 90); // цвет повернется от красного на 90° по часовой

border-color: spin(#f0f, -45); // цвет на 45° от #f0f против часовой

Противоположный цвет на колесе называется комплементарным. Он находится под углом 180° к заданному цвету. Комплементарные цвета используют для создания контраста.

ЗАДАНИЕ 1

Разработать анимированную кнопку (круг). При повороте /захвате мышью будет менять цвет

Используя вложенные правила, измените внешний вид объекта прямоугольник на треугольник или круг, добавив цветовую схему. (https://htmlacademy.ru/courses/85/run/5).

Функция spin и переменная @distance.

Над любыми численными значениями в LESS-коде можно произвести математические операции сложения, вычитания, умножения или деления:

padding-top: 10px + 20; // = 30px

padding-bottom: 100px – 50; // = 50px

font-size: 2em * 2; // = 4em

left: 50% / 2; // 25%

LESS выполняет математическую операцию и возвращает в CSS вычисленное значение. Единицы измерения всегда берутся от первого параметра в выражении.

Для вычисления отрицательного значения @distance надо умножить переменную на –1.

ЗАДАНИЕ 2

Разработать три кнопки с тремя состояниями, например: *верно*, *не верно*, *ой, ошибочка вышл*а, применить эффекты.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что определяет свойство background?
2. Какой цвет считается по умолчанию для фона?
3. Какой размер шрифта является стандартным?
4. Поясните смысл работы примесей.
5. Что выполняет Less-функция spin()?
6. Какие операции допустимо выполнять над переменными в Less-коде?

Учебное издание

Потапенко Наталья Ивановна
Кудлацкая Марина Федоровна

ДИЗАЙН ЭЛЕКТРОННЫХ И ВЕБ-ИЗДАНИЙ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Учебно-методическое пособие

Редактор *Е. И. Гоман*
Компьютерная верстка *Д. С. Жих*
Корректор *Е. И. Гоман*

Издатель:
УО «Белорусский государственный технологический университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/227 от 20.03.2014.
Ул. Свердлова, 13а, 220006, г. Минск.