Табиғаттағы элементтерді пайдалану.

Табиғатты пайдалануға да болады.Табиғи элементтер де фотосуретке шынайы сезім береді. Дегенмен, олар көбінесе көзді тақырыптан алшақтатады, сондықтан табиғи элементтерді қолданғанда абай болу керек.Ең жақсы табиғи жақтау элементтерінің кейбірі:

- Сарқырамала
  р Гүлдер
- Ағаштар
- Бақшалар
- Үңгірлер
- Жартастар
- Бұлттар

Фотосуреттерді жиектеуге арналған қосымша идеялар.

Жақтауға арналған кейбір идеялар сізге кадрлық фотосурет композициясын түсіруге көмектеседі. Міне, сізге бастау үшін бірнеше идея. Әр тақырып бойынша бірнеше әрекет жасап көріңіз. Содан кейін, ең жақсы жалпы композицияны шығаратын біреуін таңдаңыз.

- Нысаныңызды екі ағаштың арасына түсіріңіз
- Арқалар мен бағандарды көп жақтаулар ретінде пайдаланыңыз
- Фотосурет композициясын жақтау үшін табиғи және архитектуралық элементтерді араластырыңыз
- Гүлдер немесе өсімдіктер сияқты табиғи элементтерді пайдаланыңыз
- Жақтау тірегі ретінде бос жақтауды пайдаланыңыз
- Кескінді пішін немесе терезе арқылы алыңыз
- Жарық немесе көленке жақтауы бар фотосуреттер түсіру үшін жарықтандыруды өзгертіңіз.

Қолданылған әдебиеттер:

1. Рахимжанова Г.Б. Электронный учебник по предмету Фотографика 2021
2. М.Е. Карагодина Фотографика. Методическое указание.- Краснодар 2017. 37 стр.
3. Иоханнес Иттен. Искусство цвета. - Дмитрий Аронов 2007 г. - 96 с.
4. Компьютерная графика. Учебник. – СПб.: Питер, 2007 г
5. https://prophotos.ru › lessons › 208..
6. https://myslide.ru/presentation/skachat-osnovy-kompozicii-pravila-i-oshibki-kompozicii
7. https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.colescla8.ssroom.com/framing-photography-composition-tips-from-the pros/&ved=2ahUKEwiQ65Lzgr32AhVaRPEDHQWFAIAQFnoECEUQBQ&usg=AOvVaw3uAuFl9dZ7KXkdDFuTx3a4
9. https://www.movavi.io/ru/framing-ru/

УДК 72.012

WEB-ДИЗАЙНДАҒЫ ТЕХНОЛОГИЯЛАРДЫ ТАЛДАУ

Сағындық Әнел Сәкенқызы
lenfar_a@mail.ru
Студентка 5-курса кафедры «Дизайн и инженерная графика»
ЕНУ им. Л.Н.Гумилева, Нур-Султан, Казахстан
Научный руководитель – Тусупбекова Ш.М.

Преобразование веб-макета в полноценный функциональный сайт – достаточно долгий и трудоемкий процесс. Важнейшую роль играет код, с помощью которого готовый дизайн переводится в рабочий проект. Но должен ли веб-дизайнер знать, как воплотить свою творческую работу в жизнь? Это довольно спорная тема в ИТ-индустрии. Одни специалисты

считают, что это вовсе не нужно, другие – что просто необходимо. Мы пытаемся выяснить, какое из этих утверждений верно и как связаны веб-дизайн и программирование.

Рисунок 1. Завершающий этап разработки веб проекта

Любой, кто занимается дизайном, знает, что разработка веб-сайта не заканчивается созданием веб-макета. Процесс состоит из множества этапов, на которых разные специалисты отвечают за конкретные задачи и иногда взаимодействуют друг с другом. В результате такого взаимодействия создаются такие веб-страницы, какими их видит рядовой пользователь: с определенной структурой, текстом, изображениями и другими графическими объектами. Чтобы лучше представить себе всю картину и понять, что такое кодирование, давайте вкратце пройдемся по этапам от дизайна до верстки.

Первым шагом является написание основного HTML-кода. Любой сайт делается на основе HTML. Впоследствии браузер считывает код и переводит его в то, что мы видим на экране.

При написании кода верстальщик выделяет общую логическую структуру, составляет план компонентов и создает страницы с использованием соответствующих обозначений. Все элементы преобразуются в html и располагаются в иерархической последовательности, определяющей порядок их загрузки и отображения.

Рисунок 2. Код HTML

Специальный метатег в разметке указывает тип кодировки текста. Чтобы понять, что это такое и зачем оно нужно, сделаем небольшое отступление. Кодировки – определенный набор символов, благодаря которым текстовая информация преобразуется в биты данных и передается в Интернет. Это своего рода таблица, устанавливающая соответствия между буквами алфавита, цифрами, специальными символами и машинными кодами.

Каждому символу присваивается уникальный числовой код. Для корректного отображения текстовой информации на веб-странице сервер и браузер должны использовать одинаковую кодировку. Для этого на стороне сервера устанавливается система, которая заранее отправляет браузеру сообщение о шифровании отправляемой страницы. Иначе непонятные иероглифы могут заменить текст.

Основным параметром, отличающим разные типы кодирования, является количество битов, передающих значение. Например, один бит может кодировать два символа, два – четыре, три – восемь. Каждый добавленный бит удваивает количество символов, доступных для кодирования. Существуют 8-, 16- и 32-битные кодировки. В настоящее время наиболее используемой системой кодирования является UTF-8 [1].

HTML-код отвечает за основное содержание и семантику, формирует структуру, «каркас» веб-страницы. Визуальные компоненты, то есть большая часть информации о дизайне, хранятся отдельно от контента в CSS. Это еще один язык программирования, который используется разработчиками для стилизации HTML-элементов, определяющих внешний вид страницы.

С помощью CSS специалист создает стиль: описывает фон, шрифт, цвета, отступы. Стили записываются непосредственно в разметке с помощью тега style или в отдельном файле, который интегрируется поверх HTML с помощью тега link с атрибутом rel= «stylesheet». Это позволяет избежать многократного описания внешнего вида отдельных элементов, сокращает код и экономит время.

JavaScript – это код, отвечающий за динамику, реализацию интерактивных элементов. Этот язык расширяет функционал сайта, буквально оживляя его. В ее основе креативные анимированные меню, выпадающие формы поиска, контактные формы с автоматическим подбором слов, счетчики посещений, различные эффекты. Для решения большинства веб-задач доступны обширные библиотеки функций, API и плагинов для JavaScript. Разработчики пишут или находят нужные им скрипты и размечают их с помощью тега .

```
<html>
  <head>
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
      window.onload = function(){
        google.load("visualization", "1", {packages:["corechart"]});
        google.setOnLoadCallback(drawChart);
        function drawChart() {
          var data = new google.visualization.DataTable();
          var categories = ['Western Restaurant', 'Khmer Restaurant', 'Hotels', 'Resort'];
          var values = [100, 49, 120, 30];

          for (var i = 0, len = categories.length; i < len; i ++){
            data.addColumn('number', categories[i]);
          }

          data.addRows(1);

          for (var j = 0, jlen = values.length; j < jlen; j ++){
            data.setValue(0, j, values[j]);
          }

          var chart = new google.visualization.ColumnChart(document.getElementById('chart_div'));
          chart.draw(data, {width: 600, height: 180, chartArea: {left:50,top:30,width:"50%",height:"75%"},
backgroundColor:{strokeWidth:0}});
        }
      }
    </script>
  </head>

  <body>
    <div id="chart_div"></div>
  </body>
</html>
```

Рисунок 3. Код JavaScript

Таким образом, в процессе многих этапов кодирования реализуется нарисованный дизайнером макет: ресурс становится динамичным, получает окончательную структуру [2].

Каким бы качественным и подробным ни был отрендеренный макет, он остается графическим изображением. По сути, это снимок сайта, так называемый «мертвый» макет. Многие люди говорят, что пока дизайн не отображается в браузере, он не существует.

На самом деле, даже если макет не использует динамические элементы, он может сильно отличаться от окончательного макета страницы. Часто это происходит из-за непонимания между дизайнером и кодером, абсолютного отсутствия общего представления о кодировании первого.

Не зная HTML/CSS, веб-дизайнер создает макеты практически вслепую, не до конца понимая, как будет выглядеть тот или иной объект и возможно ли его реализовать с технической точки зрения. Это часто приводит к созданию проектов, мало применимых в действии[3].

Рисунок 4. Код HTML/CSS

Чтобы получить полноценный веб-сайт с большим функционалом, графический дизайнер и верстальщик должны работать бок о бок. И хотя они разные специалисты, один мыслит более графически, другой мыслит тегами и классами, они должны гармонично взаимодействовать [4].

Кроссбраузерная поддержка на основе стандартов кода, оптимизированный, не избыточный код, гибкая разметка, CSS-спрайты – все не так просто. Желательно изучить хотя бы основы HTML/CSS. Не обязательно уметь писать код самостоятельно, но вы должны хотя бы понимать принципы его работы и знать об ограничениях стандартов HTML. Очень желательно ориентироваться на прямоугольность деталей, разбираться в механизмах отображения различных элементов в браузере, учитывать специфику перестроения блоков на разных устройствах и другие нюансы.

Эти знания позволяют:

- избежать ненужных проблем и доработок;
- повысить эффективность работы;
- браться за более сложные и разнообразные задачи;
- понимать свои возможности и ограничения развития.

Подводя итог, выделим основные преимущества для веб-дизайнера, умеющего писать код:

- Высокая ценность как специалиста, соответствие требованиям рынка. Дизайнеры, знакомые с программированием, встречаются редко. Они относятся к числу особо ценных кадров, им часто предлагают хорошую, интересную работу в профессиональных веб-студиях. Кроме

того, в списке требований к высокооплачиваемым вакансиям всегда присутствует пункт «знание HTML/CSS». И это отличное конкурентное качество на рынке.

- Эффективная коммуникация с разработчиками и заказчиками. Зная нюансы базового программирования, веб-дизайнер умеет правильно расставить приоритеты, рационально оценить задачу и возможность ее реализации. Он понимает, какие элементы можно кодировать, а какие нельзя, учитывает все ограничения и тонкости, логично задает стили. Это облегчает процесс взаимодействия с веб-разработчиками, позволяет сделать проект, точно соответствующий макету.
- Сэкономить время разработки. Если дизайнер ориентируется на кодирование, этапы проектирования и макета будут выполняться один за другим довольно быстро. Работа выполняется более эффективно и выполняется за меньшее время.
- Создание логического дизайна. Специалист понимает логику макета и соответственно составляет дизайн-проект. При таком подходе сразу производится оптимальный выбор стиля, расположение элементов друг к другу.
- Преподносите информацию в доступной и понятной форме. Надлежащее кодирование помогает улучшить доступность: упростить навигацию, представить информацию в визуально привлекательном, ясном и лаконичном виде. Знание кода дает понимание всех аспектов, которые необходимо учитывать для создания легкодоступного ресурса.
- свобода анимации. Понимание технологий HTML, CSS и особенно JavaScript облегчает дизайнерам разработку элементов анимации для веб-сайта.
- Разработка адаптивного дизайна. Зная устройство адаптивной верстки, веб-дизайнер может легко установить логику расположения элементов для экранов разного разрешения.

Навыки верстки способствуют развитию динамического мышления, помогают избежать неправильных решений, противоречащих логике веб-дизайна. С их помощью можно создавать более интересные, динамичные и элегантные сайты, в полной мере используя возможности современных технологий [5].

Макет или кодирование в контексте веб-дизайна – это преобразование веб-сайта, представленного в графической форме, в HTML-форму с включенным CSS. Подготовленный дизайн-проект разбирается на составные части, а затем собирается заново с использованием HTML-кода и каскадных таблиц стилей.

Другими словами, дизайнер создает некое руководство по стилю, которое затем превращается в код, и из нескольких строк кода делается целый сайт. А чтобы реально оценить сложность и выполнимость задачи, он должен знать, как это все работает. Знать – да, уметь программировать – не обязательно. Но это, безусловно, будет огромным плюсом в работе.

Подводя итоги, можно сказать, что перед разработкой сайта необходимо исследовать, какие платформы, браузеры, технические новинки предпочтительны, какие скорости соединения используют потенциальные пользователи, какие характеристики сайта следует учитывать: увеличение посещаемости, совместимость версий, дизайн, функциональность, скорость, а также отображение графических объектов на сайтах.

Список использованной литературы

1. http://webmaster.info.aol.com.
2. Основы Web-технологий. Храмцов П.Б., Брик С.А., Русак А.М.Спб. 2005.
3. Крамер. HTML: наглядный курс Web-дизайна. Спб. 2005.
4. Нидерст. Web-мастеринг для профессионалов, настольный справочник. М.: Высшая школа. 2004.
5. Кон Артур. Секреты Интернета. Серия «Учебный курс», Ростов Н/Д.: