УДК 68.004.93

НЕКОТОРЫЕ СОВРЕМЕННЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ВЕБ-САЙТОВ С АНИМАЦИЕЙ

Э. И. Ясюкович
кандидат технических наук, доцент
Белорусско-Российский университет (г. Могилев)

В статье проведен обзор некоторых современных фронт-энд и бек-энд технологий разработки веб-сайтов. Рассмотрены технологии и средства построения анимированной графики, адаптивного веб-дизайна и адаптивной верстки сайтов. Приведены примеры построения анимированных веб-сайтов – на основе покадровой анимации и с использованием элемента "canvas".

Ключевые слова: веб-сайт, адаптивный веб-дизайн, фронт-энд технология, бек-энд технология, веб-анимация, элемент *canvas*, язык разметки гипертекста, каскадные таблицы стилей, языки веб-программирования, браузер, системы управления контентом, параллакс-эффект.

Введение

В настоящее время межсетевое объединение Internet стало основным источником информации [1], но эффективность и функциональность веб-ресурса зависит от выбора технологий его разработки. Чтобы веб-проект выглядел наиболее современно и привлекательно, при его разработке необходимо учитывать новейшие, ежегодно меняющиеся и поддерживаемые последними версиями браузеров технологии.

Современный веб-сайт должен иметь такую структуру, чтобы при просмотре даже его первой страницы в течение первых трех секунд у посетителя создалось четкое представление о содержании и структуре сайта. Поэтому веб-мастерам следует ответственно относиться к вопросам полезности, эффективности и информативности их сайтов, чтобы привлечь внимание как можно большего количества посетителей [2].

Язык гипертекстовой разметки *html5* предоставляет дополнительные функции для использования передовых технологий сайтостроения, таких, как видео, аудио, анимация и многие другие. Одна из них – это элемент *canvas* (холст), позволяющий строить анимированные растровые изображения. Для построения масштабируемой векторной графики в *html5* используется элемент *SVG (Scalable Vector Graphics)*, который, в отличие от *canvas*, может содержать интерактивные объекты.

Для эффективного создания *SVG* графики и анимации используются библиотеки *Java Script*, такие, как *Velocity*, *SVG.JS*, *Walkway*, *Raphael.JS*, *Snap.Svg*, *Bonsai* и другие, позволяющие создавать вращение и изменение форм объектов, а также анимационные сюжеты. Объектами векторной анимации могут быть изображения, текст, пользовательские интерфейсы, логотипы и другие.

Неотъемлемой частью сайта являются также каскадные таблицы стилей и процедуры на языках *Java Script*, *PHP* или других.

Для взаимодействия языка *JavaScript* и *HTML* используется библиотека *jQuery*, называемая также фреймворком, позволяющая существенно упростить использование языка *JavaScript* и сделать его более доступным за счет предоставления простых функций, существенно облегчающих решение различных задач сайтостроения. Для использования *jQuery* ее необходимо скачать с официального сайта *jQuery.com* и в *HTML* документе подключить как обычный *Java Script* файл:

<script type="text/javascript" src="js/jquery-1.9.1.min.js"></script>

Можно также, не скачивая, подключить ее последнюю версию с сайта *GoogleCode*:

<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>

Важное место в сайтостроении занимают веб-анимации, для построения которых используются различные инструменты, например, такие, как [3]:

— редактор *HTML5 Maker*, предоставляющий большую коллекцию изображений;

— Online*Sprite Box Tool*, позволяющий обрабатывать картинки с использованием *jQuery*, *CSS3* и *html5*;

— *Online 3D Sketch Tool*, использующийся для создания эскизов, 3D картинок, пунктирных линий и других элементов;

— *Tumult Hype* – специальная система покадровой анимации, позволяющая оживить неподвижный элемент;

— *Google Web Designer*, позволяющая создавать работающие на любом устройстве интерактивные проекты, основанные на *html5*;

— *GSAP*, содержащая набор инструментов скриптовой и высокопроизводительной полноцветной *html5* анимации;

— *Hippo Animator*, автоматически масштабирующая анимацию под любой размер веб-браузера;

— *Mugeda*, позволяющая создавать игры и веб-приложения, которые эффективно работают на мобильных устройствах;

— *Tween JS*, используемая для создания сложной анимации библиотека *JavaScript*;

— *Radi* – для создания видео, анимации и графики в реальном времени;

— *Animatron*, позволяющая создавать анимацию *html5* и интерактивный контент.

В современных технологиях сайтостроения активно используется также адаптивный веб-дизайн, представляющий собой новое направление, призванное обеспечивать качественное восприятие сайтов на различных устройствах, подключенных к сети Internet.

При создании адаптивного веб-дизайна для мобильных устройств используется метод *Mobile First* оптимизированной верстки, предполагающий сначала верстать сайт для мобильных устройств, а затем для больших экранов с учетом скорости подключения к сети. Данный метод демонстрирует в первую очередь самое важное содержание, обеспечивает легковесность и оптимизированность сайта, а также не допускает загрузки большего объема ресурсов, чем требуется пользователю [4].

Адаптивная верстка меняет дизайн веб-страниц в зависимости от поведения пользователя, платформы, размера экрана, ориентации девайса и является неотъемлемой частью современной веб-разработки. Она позволяет существенно экономить и не отрисовывать новый дизайн для каждого разрешения, а лишь менять размеры и расположение отдельных элементов.

Для автоматизации выполнения рутинных операций при разработке веб-сайтов широко используются *CMS* (Content Management Software – система управления контентом), позволяющие быстро строить сайты, наполнять их содержимым и выполнять редактирование. При этом использование *CMS* не требует глубоких знаний даже *html*.

1. Некоторые современные технологии веб-разработки

Несколько лет назад для разработки сайтов было достаточно знаний языка разметки гипертекста, каскадных таблиц стилей и основ компьютерной графики. В настоящее вре-

мя выполнение таких работ требует владения скриптовыми языками, одним из которых является *Java Script*, используемый для создания слайдеров – специальных графических элементов или текстов с эффектами анимации, модальными окнами и множеством других элементов, оживляющих сайт. То есть слайдеры на сайтах – это блоки на веб-страницах, в пределах которых с установленной периодичностью демонстрируются анонсы новостей, статей или изображений. Создаются эти блоки на *html*, *CSS*, *Java Script* и являются визуально привлекательными для посетителей веб-сайтов, так как стимулируют интерес к их материалам. Кроме этого, слайдеры позволяют экономить место, поскольку каждый их блок дает возможность демонстрировать несколько анонсов.

Основой в технологии создания сайтов является язык *html*, которым должны владеть и верстальщики, и веб-программисты. *CSS* используется для визуального оформления веб-страниц и адаптации их под все устройства, на которых будет отображаться сайт – от телевизоров до мобильных телефонов. Статистика показывает, что примерно 52% пользователей выходят в Интернет с помощью смартфонов [5], поэтому адаптивная версия сайта должна быть не хуже ее полной версии.

Современную веб-разработку разделяют на две части: *фронт-энд* и *бек-энд* [6]. *Фронт-энд* – это клиентская сторона пользовательского интерфейса, содержащая кнопки, панели, картинки, заставки и другие элементы, используемые для взаимодействия с сайтом. Разработка сайта во *фронт-энд* начинается с создания его эскиза и дизайна, которые затем реализуются верстальщиком. *Фронт-энд* разработчик не обязан быть профессиональным дизайнером, но в идеале должен уметь создавать макет и дизайн сайта, свободно владеть *html* и *CSS*, а также программировать на языке *JavaScript* с использованием *JQuery* и других *JavaScript* библиотек.

Бэк-энд – это написанная на языке *PHP* и выполняемая на стороне сервера скрытая от клиента программная часть сайта. Здесь может выполняться запрос к базе данных и поиск в ней необходимых вариантов, после чего генерируется ответ в виде веб-странички, которая отсылается клиенту как результат поиска, и клиент видит результат работы *бек-энд*а на *фронт-энд*е.

В последние два года наблюдается тенденция к росту *бек-энд* разработки на языке *Python*, который появился в 1991 г.

Быстро набирающим популярность является современный высокопроизводительный язык *Go*, который сочетает в себе лучшие черты статических и динамических языков: строгую типизацию и скорость исполнения, гибкость и скорость разработки, добавляя к этому производительную и надежную концепцию параллельного программирования.

Для мобильных устройств в 2017 г. появилась группа приложений *PWA* (ProgressiveWebApplications – прогрессивные веб-приложения), использующая стек веб-технологий (*JavaScript* + *html* + *CSS*), которые позволяют увеличить конверсию, количество пользователей и удобство использования веб-приложений на мобильных устройствах. Основными принципами *PWA* являются: полная адаптивность к любым “умным” устройствам; нативность, выдвигающая удобства пользователя на первое место, и автономность, обеспечивающая самостоятельное обновления приложений.

Важными параметрами качества сайта являются используемые шрифты, особенно на первом его экране, где можно применить каллиграфические или готические шрифты, которые обязательно должны быть свободно читаемыми.

Цвета, используемые на сайте, должны корректно отображаться в любом браузере и на любых типах устройств, а количество используемых цветов должено быть правильным и ограниченным [4].

Для оживления сайтов используется *параллакс-эффект* – специальная технология, когда фоновое изображение в перспективе двигается медленнее, чем элементы перед-

него плана. Такой эффект формируется с помощью нескольких накладываемых друг на друга слоев, которые при прокручивании двигаются с различными скоростями. Такая технология позволяет создать также искусственные трехмерные эффекты.

В ближайшее время более 80% всего интернет-трафика захватит видеоконтент, то есть видео будут чаще появляться на большинстве сайтов [7].

2. Инструменты и средства сайтостроения и анимации

Современные сайты имеют довольно сложную структуру, поэтому их разработка требует специальных знаний и опыта работы в области *html*, веб-программирования, веб-дизайна и сайтостроения.

Для быстрой разработки сайтов широко используются различные *CMS* (content management system – система управления контентом), среди которых наиболее популярной является *WordPress*. В настоящее время более 40% интернет-сайтов разработаны с использованием данной *CMS*, которая проста в использовании и легко настраиваема с помощью плагинов. Она позволяет публиковать новые сообщения и формировать страницы, имеет мощный редактор для редактирования и форматирования контента, поддерживает регистрацию пользователей [6].

Используются также такие *CMS*, как *Joomla*, *Drupal*, *Serendipity*, *DotClear*, *ImpressPages*, *Chamilo* и другие [8].

Joomla является менее дружелюбной к новичкам, но в некоторых аспектах может быть более гибкой, позволяет делать сложные динамические сайты. Первая версия вышла в 2005 г. Ее код является открытым.

Drupal содержит более гибкую, чем *Joomla*, систему для работы с настраиваемыми типами сообщений.

Serendipity – это удобная для пользователя *CMS*, которая идеально подходит для небольших блогов.

Dotclear – еще один отличный вариант для ведения блога. Позволяет добавлять новые функции на сайт с помощью плагинов и тем.

ImpressPages – малоизвестная *CMS*, лучше всего подходит для создания блогов, содержит простой редактор, оснащенный современным интерфейсом.

Chamilo –это так называемая система управления обучением (*LMS* – Learning Management System), являющаяся специфическим видом *CMS*.

Если сайт разработан на языке *PHP*, то браузер не сможет его интерпретировать, так как *PHP*-скрипты выполняются на сервере. В таком случае необходимо либо найти реальный удаленный сервер, либо создать локальный у себя на компьютере. Во втором случае огромный плюс в том, что работать будет проще, никто не увидит результаты экспериментов, потому что локальный сервер доступен только на компьютере разработчика сайта.

Самый известный локальный сервер на Windows – это *Denwer*, который уже не активно поддерживается. Известны также *OpenServer*, *WAMP*, *MAMP* и другие, которые позволяют открывать *php*-файлы и видеть результат их работы.

При создании веб-страниц широко используются различные анимации, которые являются одной из основ современного веб-дизайна [7].

Для создания анимированных веб-страниц используется *CSS*-анимация и *JavaScript*. *CSS*-анимация не позволяет создавать сложные физические эффекты и имитировать реалистичное движение. Поэтому такие анимации, как подпрыгивание, пауза, остановка и замедление лучше писать на *JavaScript* [9].

Для формирования сложных эффектов и 3D анимации для виртуальной реальности можно использовать библиотеку веб-графики *WebGL* (WebGraphicsLibrary), которая позволяет формировать рендеринг (визуализацию) графики 60 кадров в секунду.

3. Примеры создания анимированных сайтов на языке JavaScript

Для поддержки анимации современные браузеры не требуют установки дополнительных плагинов, таких, как *Flash*, а используют *JavaScript*, *CSS3* и *html5*.

В настоящей работе рассматриваются два примера анимации: покадровая (пример 1) и с использованием элемента *canvas* (пример 2).

Пример 1. Простейшая покадровая анимация “Бегущий кот” [8]. В папке данной задачи содержатся 13 файлов (*index.html*, содержащий *html* код задачи и функцию *anim*, а также 12 файлов образа кота в различных позициях – *ket01.png – ket012.png*).

Раздел <head> файла *index.html* содержит тег <title>, задающий название задачи в строке заголовка браузера и тег <meta charset="UTF-8">, задающий кодировку шрифта. Далее записаны две функции на языке *JavaScript – anim()* и *pusk()*, Функция *anim()* выполняет формирование массива *imAr* из 12 графических элементов размером 200 × 100 пикселей, в каждом из которых записываются имена графических файлов *ket01.png – ket012.png*. Функция *pusk()* используется для формирования анимации, в строке *document.images[0].src = imAr[is].src*, которой элементу коллекции присваивается *is*-й графический элемент из массива *imAr*. Строка *setTimeout("pusk()", 100)* – это внутренний таймер-планировщик, который задает вызов функции *pusk()* через каждые 100 миллисекунд. Полный текст кода анимации “Бегущий кот” приведен ниже (рис. 1).

```
<html>
    <head>
    <title>Бегущийкот</title>
    <meta charset = "UTF-8">
    <script language = "JavaScript">
      functionanim(){
        imAr = new Array();
        for(vari= 0; i< 12; i++){
              imAr[i] = new Image(200,100);
              imAr[i].src = "ket0" + (i+1).toString() +".png";
        }
         is = -1;
      }
      functionpusk(){
        is = is + 1;
        document.images[0].src = imAr[is].src;
        setTimeout("pusk()",100);
        if(is == 11) is = -1;
      }
  </script>
  </head>
  <body onload="anim()">
    <input type="button" name="button1" value="Start" onclick="pusk()"><hr>
    <img>
  </body>
</html>
```

Рисунок 1 – Текст кода анимации “Бегущий кот”

На рисунке 2 приведен фрагмент кадра анимации, которая запускается кнопкой *Start*.

Рисунок 2. Фрагмент кадра анимации “Бегущий кот”

Пример 2. Анимационная сцена в *canvas* "Прыгающий мяч" [7]. Для разработки данной сцены на *JavaScript* создается холст размером 250 × 150 пикселей, а на нем с помощью элемента *canvas* формируется анимационная сцена [9]. Данный элемент введен в раздел *body*, в котором записано также пять процедур *JavaScript* – два конструктора (*Ball* – задающего параметры мяча и *circle* – выполняющего его начальную отрисовку); и три функции (*Ball.prototype.draw* – отрисовки перемещающегося мяча; *Ball.prototype.mov* – формирования перемещения мяча; *Ball.prototype.cheCo* – отслеживания столкновения мяча с границами холста).

Конструктор *Ball* задает начальные координаты мяча $x = 5$ пикселей, $y = 5$ пикселей и начальные скорости по оси x – -3 пикселя/с – влево, и по оси y – +5 пикселей/с – вниз для формирования диагонального движения мяча вниз до соприкосновения его с границами холста и отскока.

В строке *var circle= function(x, y, rad, Circl)* создается метод отрисовки мяча. Здесь *Circl* – логическая переменная, определяющая необходимость заливки круга (мяча).

Метод *ctx.beginPath()* определяет начало вывода линии контура мяча, а метод *ctx.arc(x, y, radius, 0, Math.PI*2, false)* отрисовывает полный круг без заливки, то есть формирует мяч.

Метод *draw* отрисовки мяча добавлен к свойству *prototype* конструктора *Ball*, чтобы все объекты, созданные при помощи *Ball*, могли его использовать.

Далее элементу *Ball* добавлен метод *circle* отрисовки окружности, то есть отрисовки закрашенного мяча, радиус которого равен 7 пикселям:

```
Ball.prototype.draw=function() {
circle(this.x, this.y, 7, true)
}
```

Таким образом, создается функция отрисовки окружности либо залитого круга и добавляется метод *draw* с помощью свойства *prototype* для всех объектов *Ball*.

Для обновления координат мяча *x* и *y* построен метод, в котором они будут обновляться в соответствии с текущей скоростью *xSpeed* и *ySpeed*:

```
Ball.prototype.move = function(){
    this.x+=this.xSpeed;
    this.y+=this.ySpeed;
    };
```

Метод формирования отскоков мяча, в котором проверяется, столкнулся ли мяч с левой или правой, нижней или верхней границами холста, имеет вид:

```
Ball.prototype.cheCo = function(){
        if (this.x<5||this.x>245){
                this.xSpeed = -this.xSpeed;
        }
        if (this.y<5||this.y>145){
                this.ySpeed = -this.ySpeed;
        }
```

Если столкновение наступило, то знак соответствующей скорости мяча меняется на противоположный.

После этого формируется анимация мяча с использованием элемента *canvas*. Для этого добавлены две строки, чтобы получить элемент *canvas* и указать контекст рисования:

```
var canvas = document.getElementById("canvas");
varctx = canvas.getContext("2d");
```

Далее объявляется переменная *ball*:

```
varball = newBall();
```

После этого создается функция *setInterval*, первым атрибутом которой является функция, содержащая параметры: *ctx.clearRect(0, 0, 250, 150)* – метод очистки области холста; *ball.draw()* – метод отрисовки мяча; *ball.move()* – метод формирования движения мяча с помощью функции *move()*; *ball.checkCollision()*– метод проверки столкновения с помощью функции *checkCollision()*; *ctx.strokeRect(0, 0, 250, 150)*– метод отрисовки прямоугольника с границами. Вторым атрибутом функции *setInterval* является интервал анимации, равный 25 миллисекундам:

```
setInterval(function(){
        ctx.clearRect(0,0,250, 150);
        ball.draw();
        ball.move();
        ball.checkCollision();
        ctx.strokeRect(0,0,250, 150);
        }, 25);
```

Полный текст *html* и *JavaScript* кода приведен на рисунке 3:

```
<!DOCTYPE html>
<html long="en">
<head>
    <meta charset="UTF-8">
    <title>Мяч></title>
</head>
<body>
    <canvas id="canvas" width="250" height="150"></canvas>
    <script>
        var Ball = function(){
            this.x = 5;                    this.y = 5;
            this.xSpeed = -3;              this.ySpeed = 4;    };
        var circle = function(x, y, rad, Circl){
            ctx.beginPath();
            ctx.arc(x, y, rad, 0, Math.PI*2, false);
            if (Circl){
                ctx.fill();
            } else {
                ctx.stroke();          }
        };
        Ball.prototype.draw=function(){
            circle(this.x, this.y, 7, true);   };
        Ball.prototype.mov = function(){
                ctx.x += this.xSpeed;
                ctx.y += this.ySpeed;          };
        Ball.prototype.cheCo = function(){
                if (this.x< 5||this.x> 245){
                    this.xSpeed = -this.xSpeed;   }
```

```
                        if (this.y< 5||this.y> 145){
                                this.ySpeed = -this.ySpeed;    }
                };
        var canvas = document.getElementById("canvas");
        varctx = canvas.getContext("2d");
        var ball = new Ball();
        setInterval(function(){
                ctx.clearRect(0, 0, 250, 150);
                ball.draw();            ball.mov();
                ball.cheCo(); ctx.strokeRect(0, 0, 250, 150);
                },25);
    </script>
</body>
</html>
```

Рисунок 3 – Текст кода анимационной сцены “Прыгающий мяч”

Результаты работы *html* страницы “Прыгающий мяч” приведены на рисунке 4.

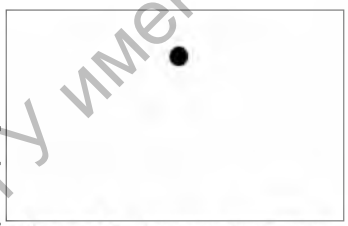

Рисунок 4 – Анимационная сцена “Прыгающий мяч”

Заключение

Таким образом, в настоящей работе проведен обзор и анализ современных технологий разработки веб-сайтов с анимацией, в том числе для мобильных устройств.

В качестве примеров приведены две сцены анимации: бегущий кот, построенной на покадровой анимации, и “Прыгающий мяч”, разработанной на основе элемента *canvas*.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://qwizz.ru/новые-технологии-сайтов/;
2. http://nonameno.com/creation-and-promotion-of-sites-the-basics.html/;
3. https://techrocks.ru/2018/11/16/web-animation-tools-for-web-app/;
4. https://www.kasper.by/blog/slaider-dlya-saita/;
5. http://isizov.ru/github-kak-hosting-dlya-sajtov/;
6. https://1ps.ru/blog/sites/2018/kakimi-budut-sajtyi-v-2019-godu/;
7. https://www.internet-technologies.ru/articles/sozdanie-animaciy-na-holste.html;
8. https://www.pvsm.ru/javascript/7177;
9. https://www.templatemonster.com/ru/blog/2015/05/12/html5-animation-tools/.

Поступила в редакцию 23.05.2019 г.

Контакты: ed50@tut.by; +375 222 77 13 26 (Ясюкович Эдвард Игнатьевич)

Yasyukovich E. SOME MODERN TECHNOLOGIES FOR THE DEVELOPMENT OF WEB SITES WITH ANIMATION.

The article reviews some of the modern front-end and back-end technologies for website development. Modern technologies and tools for constructing animated graphics, responsive web design and responsive layout of sites are considered. Two examples of building animated websites are given: one is based on a frame-by-frame animation and the other is based on the Canvas element.

Keywords: website, responsive web design, front-end technology, back-end technology, web animation, Canvas element, hypertext markup language, cascading style sheets, web programming languages, browser, content management systems, parallax effect.