ЛИТЕРАТУРА

1. Бородаев Д.В. Web-сайт как объект графического дизайна. Монография. – Х.: "Септима ЛТД", 2006. – С. 288.

2. Грибов Д.Е. Macromedia Flash 4. Интерактивная web-анимация. - М. ДМК. 2000. - 672 с.

3. Франклин Д., Паттон Б. Flash 4. Анимация в интернете. - СПб. Символ Плюс. 2000.-464 с.

4. Энди Андерсон, Марк Дел Лима, Стив Джонсон. Macromedia Flash MX 2004 Show Me: Macromedia Flash MX 2004. – М.:Вильямс, 2005. – С. 544

5. http://club.chelbis.ru/articles/design/design_67.html

УДК 004.5

Студ. М. Д. Ковтик, А. С. Байденок
Науч. рук. асс. Н. И. Потапенко
(кафедраинформатики и веб-дизайна, БГТУ)

РЕАЛИЗАЦИЯ ИГРОВОЙ СТРАТЕГИИ СРЕДСТВАМИ JQUERY И JS

Цель работы: ознакомиться с ходом разработки программного кода для браузерной игры, рассмотреть различия в реализации этого кода средствами JavaScript и JQuery.

В данном проекте рассматривается выполнение задания для Winter Simulation Conference 2015. В данном проекте была дана графическая часть задания, а целью было – разработать программную.

В ходе данной игры пользователь управляет олимпийским бегуном, который бежит по одной из трёх дорожек. На дорожках каждый раз случайным образом генерируются препятствия, которые пользователь может перепрыгивать или обегать. Из предоставленных изображений необходимо создать анимацию бега и прыжка. В техническом задании описаны подробные требования. В рамках данной работы они рассматриваться не будут.

Игра начинается со стартового экрана, на котором пользователю объясняются правила игры и управление. После нажатия на кнопку «Старт» бегун начинает бежать. Пользователь побеждает, когда достигает олимпийского огня, и проигрывает, когда столкнётся с препятствием.

Программная составляющая подчиняется алгоритму, расположенному на третьем слайде. На этапе подготовки объявляются и/или инициализируются переменные, необходимые в дальнейшем.

Для бегуна:

• line – содержит номер дорожки, на которой находится бегун в данный момент;

• isjump – логическая переменная, которая отвечает на вопрос «находится ли бегун в прыжке сейчас?»;

• phase – определяет текущее изображение бегуна в соответствии с его анимацией;

• phaseVector – определяет, какое изображение бегуна будет следующим в соответствии с его анимацией.

Для препятствий:

• runwayNObstacles – массив, содержащий координаты препятствий на N-ой дорожке;

• obstaclesCount – количество препятствий, которые необходимо сгенерировать;

• maxObstacleDistance – дистанция, после которой препятствия генерироваться не будут, альтернатива для предыдущей переменной.

Далее представлено сравнение реализации одних и тех же функций средствами JS и JQ.

Функция: Получение блока с id = runner.

JS: var runner = document.getElementById(“runner”)

JQ: var runner = $(“#runner”)

Комментарий: JQuery выигрывает по длине кода.

Функция: Привязка события по нажатию на кнопку.

JS:

в html: <button id=start onclick=start()>start<button>

в js: function start(){}

JQ:

в js: $(‘#start’).click(function(){})

Комментарий: JQuery выигрывает по удобству написания кода. Нет необходимости в переключении документов.

Функция: Добавление нового препятствия. В переменной str содержится код препятствия. Переменная runwayn – *n*-я беговая дорожка.

JS: runway2.innerHTML += str

JQ: $(“runway”).append(str)

Комментарий: Реализации обладают разной логикой действия. Других существенных отличий нет.

Функция: Изменении CSS

JS: runner.style.marginLeft += runSpeed

JQ: runner.css(‘margin-left’, ‘+=runSpeed’)

Комментарий: в JQ понадобится меньше кода, если понадобится поменять более одного свойства.

Функция: Заменакартинки.

JS: runner.innerHTML = “<img src=’../runner/runner” + number + “.png’ alt=’runner’>”

JQ: $(“#runner img”).attr(‘src’, “imgs/runner/runner” + number + “.png”);

Комментарий: в JQ нет необходимости заменять целиком весь блок – достаточно поменять лишь атрибут нужного тега. В JS такой возможности.

Функция: Создание CSS-анимации

JS:

В CSS: transition: marginLeft 12s;

В JS: runner.style.marginLeft = 4400px;

JQ: $(‘html, body’).animate({scrollLeft: 4400}, 12000);

Комментарий: в данном случае код JQ написать быстрее и удобнее, чем JS.

Функция: Скрыть HTML.

JS:

document.getElementById(“success”).classList.remove(“List”);

JQ: $(“#success”).removeClass(“hide”);

Комментарий: JQ незначительнокороче.

Рассмотрев эти ситуации можно сделать вывод о том, что главным отличием JQ от JS является синтаксис. В некоторых ситуациях он может быть короче и удобнее, в некоторых одни и те же функции обладают разной логикой исполнения, а во всех остальных случаях возможности и удобство JQ и JS почти одинаковы. Для максимальной гибкости следует ориентироваться по ситуации и уметь использовать все доступные инструменты.