Михаил Ботов

Интерактивная анимация HTML5

Методические указания

Михаил Ботов

Интерактивная анимация HTML5. Методические указания

«Издательские решения»

Ботов М.

Интерактивная анимация HTML5. Методические указания / М. Ботов — «Издательские решения»,

ISBN 978-5-44-856305-8

Данная методичка раскрывает основные подходы к созданию интерактивных анимаций в рамках базового стека веб-технологий, опирающихся на стандарт HTML5. В частности рассмотрены: прямое манипулирование элементами DOM при помощи javascript, работа с тегами canvas и svg, в т. ч. с использованием SMIL.

ISBN 978-5-44-856305-8

Содержание

Интерактивная анимация HTML5
Методические указания

Михаил Ботов

ISBN 978-5-4485-6305-8
Создано в интеллектуальной издательской системе Ridero

1 Анимация с помощью функции setinterval

Простую анимацию можно организовать при помощи функции *setInterval ()*; Она циклически вызывает переданный ей метод через заданные интервалы времени до тех пор, пока не будет остановлена функцией *clearInterval ()*;

Пример – смещение блока разметки вправо:

```
<script>

function move (elem) {

var left = 0; // начальное значение

function frame () {// функция для отрисовки
left++
elem.style. left = left + ’px’

if (left == 100) {
clearInterval (timer); // завершить анимацию
}
}

var timer = setInterval (frame, 100) // рисовать каждые 100мс
}
</script>

<div onclick="move(this.children [0])» class=«example_path»>
<div class=«example_block»> </div>
</div>
```

2 Анимация на jquery

2.1 События

Для реакции на действия пользователя и внутреннего взаимодействия элементов скриптов существует механизм событий.

Событие – это сигнал от браузера о том, что что-то произошло. Существует возможность определить реакцию скрипта на возникновение того или иного события, назначив ему функцию-обработчик. Данная функция (или функции, т. к. можно привязать несколько обработчиков к одному событию) будет вызываться всякий раз, когда нужное событие произойдет (например, пользователь кликнет по изображению). На вход функции обработчику передается специальный объект с информацией о произошедшем событии.

Существует много видов событий. Примеры функций библиотеки jQuery, позволяющие удобно привязывать обработчики к определенным событиям, показаны в приведенных ниже четырех таблицах.

.on()	Универсальный метод для установки обработчиков событий на выбранные элементы страницы.
.off()	Удаляет обработчики, установленные с помощь .on().
.bind()	Устанавливает обработчик события на выбранные элементы страницы. Обработчик не сработает на элементах, появившихся после его установки.
.live()	Устанавливает обработчик события на выбранные элементы страницы. Обработчик сработает и на элементах, появившихся после его установки.
.delegate()	Устанавливает обработчик события на выбранные элементы страницы. Элементы выбираются с помощью уточняющего селектора. Обработчик будет действовать и на элементах, появившихся после его установки.
.one()	Устанавливает обработчик события на выбранные элементы страницы, который сработает только по одному разу, на каждом из элементов.
.unbind()	Удаляет обработчик событий у выбранных элементов.
.die()	Удаляет обработчик событий, который был установлен с помощью live().
.undelegate()	Удаляет обработчик событий, который был установлен с помощью delegate().
.trigger()	Выполняет указанное событие и запускает его обработчик.
.triggerHandler()	Запускает обработчик указанного события, без его выполнения.
jQuery.proxy()	По заданной функции, создает другую, внутри которой переменная this будет равна заданному значению.
event	Объект, содержащий данные о текущем событии. Передается всем обработчикам событий.

Базовые события

.click()	Устанавливает обработчик "клика" мышью по элементу, либо, запускает это событие.
.dblclick()	Устанавливает обработчик двойного "клика" мышью по элементу, либо, запускает это событие.
.hover()	Устанавливает обработчик двух событий: появления/исчезновения курсора над элементом.
.mousedown()	Устанавливает обработчик нажатия кнопки мыши, либо, запускает это событие.
.mouseup()	Устанавливает обработчик поднятия кнопки мыши, либо, запускает это событие.
.mouseenter()	Устанавливает обработчик появления курсора в области элемента, либо, запускает это событие. Появление этого события, отработано лучше, чем стандартного mouseover.
.mouseleave()	Устанавливает обработчик движения курсора в области элемента, либо, запускает это событие.
.mouseout()	Устанавливает обработчик выхода курсора из области элемента, либо, запускает это событие.
.mouseover()	Устанавливает обработчик появления курсора в области элемента, либо, запускает это событие.
.toggle()	Поочередно выполняет одну из двух или более заданных функций, в ответ на "клик" по элементу. События клавиатуры
.keydown()	Устанавливает обработчик перехода клавиши клавиатуры в нажатое состояние, либо, запускает это событие.
.keyup()	Устанавливает обработчик возвращение клавиши клавиатуры в ненажатое состояние, либо, запускает это событие.
.keypress()	Устанавливает обработчик ввода символа с клавиатуры, либо, запускает это событие. События формы
.focus()	Устанавливает обработчик получения фокуса, либо, запускает это событие.
.blur()	Устанавливает обработчик потери фокуса, либо, запускает это событие.
.focusin()	Устанавливает обработчик получения фокуса самим элементом или одним из его дочерних.
.focusout()	Устанавливает обработчик потери фокуса самим элементом или одним из его дочерних.
.select()	Устанавливает обработчик выделения текста, либо, запускает это событие.
.submit()	Устанавливает обработчик отправки формы, либо, запускает это событие.
.change()	Устанавливает обработчик изменения элемента формы, либо, запускает это событие.

События мыши

.ready()	Устанавливает обработчик готовности дерева DOM.
.load()	Устанавливает обработчик завершения загрузки элемента.
.unload()	Устанавливает обработчик ухода со страницы (при переходе по ссылке, закрытии браузера и.т.д.).

События загрузки страницы

.error()	Устанавливает обработчик ошибки при загрузке элементов (например отсутствие необходимой картинки на сервере).
.resize()	Устанавливает обработчик изменения размеров окна браузера, либо, запускает это событие.
.scroll()	Устанавливает обработчик "прокрутки" элементов документа, либо, запускает это событие.

События браузера

Всплывание события и его остановка

Нужно сказать, что при наступлении события обработчики сначала срабатывают на самом вложенном элементе, затем на его родителе, затем выше и так далее, вверх по цепочке вложенности. Это называется всплыванием события.

Всплытие идёт вверх по иерархии DOM. Обычно событие будет всплывать наверх и наверх, до элемента <html>, а затем до *document*, а иногда даже до *window*, вызывая все обработчики на своем пути. Но любой промежуточный обработчик может решить, что событие полностью обработано, и остановить всплытие.

Для остановки всплытия нужно вызвать метод *event.stopPropagation ()*.

2.2 Функция animate

Jquey функция *animate ()* выполняет определенную анимацию на заданном наборе элементов. Анимация происходит за счет плавного изменения их CSS-свойств. Функция имеет два варианта использования:

1).***animate (properties, [duration], [easing], [callback])***

где

properties – список CSS-свойств, участвующих в анимации и их конечных значений. (см. описание ниже)

duration – продолжительность выполнения анимации. Может быть задана в миллисекундах или строковым значением ’fast’ или ’slow’ (200 и 600 миллисекунд).

easing – изменение скорости анимации (будет ли она замедляется к концу выполнения или наоборот ускорится). (см. описание ниже)

callback – функция, которая будет вызвана после завершения анимации.

2).***animate (properties, options)***

где

properties – список CSS-свойств, участвующих в анимации и их конечных значений. (см. описание ниже)

options – дополнительные опции. Должны быть представлены объектом, в формате опция: значение.

Варианты опций:

duration – продолжительность выполнения анимации (см. выше).

easing – изменение скорости анимации (будет ли она замедляется к концу выполнения или наоборот ускориться). (см. описание ниже)

complete – функция, которая будет вызвана после завершения анимации.

step – функция, которая будет вызвана после каждого шага анимации.

queue – булево значение, указывающее, следует ли помещать текущую анимацию в очередь анимаций. В случае false (по-умолчанию), анимация будет запущена сразу же, не вставая в очередь. Иначе она будет ждать завершения предыдущей анимации, стоящей в очереди.

specialEasing – позволяет установить разные значения easing, для разных CSS-параметров. Задается объектом, в формате параметр: значение.

Пример – изменение группы css-свойств по клику на кнопку:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
background-color:#bca;
width:100px;
border:1px solid green;
}
</style>
<script src="http://code.jquery.com/jquery-latest.min.js"></script>
</head>
<body>

<button id=«go»> Полить </button>
```

```
<div id=«block»> Газон </div>

<script>
// Произведем изменение нескольких css-величин в ходе одной анимации.
$("#go").click (function () {
$("#block").animate ({
width: «70%», // ширина станет 70%
opacity: 0.4, // прозрачность будет 40%
marginLeft: «0.6in», // отступ от левого края станет равным 6 дюймам
fontSize: «3em», // размер шрифта увеличится в 3 раза
borderWidth: «10px» // толщина рамки станет 10 пикселей
}, 1000); // анимация будет происходить 1 секунду
});
</script>

</body>
</html>
```

Задание «Скачущий мяч»

1) Сделать анимацию прыгающего мяча. Мяч начинает движение от верхней границы окна браузера, летит вниз, по достижении определенного уровня (пролететь должен не менее полэкрана) изменяет направление движения на противоположное – летит вверх. По достижении верхней границы окна – снова изменяет направление и летит вниз. И так в бесконечном цикле.

В точках разворота мяч должен немного сжиматься по вертикали.

2). По клику на мяч в точке экрана, на которой был совершен клик появляется текст, например, «Гол!», который уходит вправо, постепенно исчезая.

Элемент разметки для текста создается по клику, смещается вправо с изменением прозрачности и удаляется по достижении правого края экрана

3 canvas: рисование на холсте

Canvas – это HTML элемент, предназначенный для создания растровых изображений посредством javascript.

Важно отметить, что сам элемент canvas является частью DOM-модели документа, но все, что на нем изображается (фигуры, надписи) остается внутри него и в DOM не встраивается.

3.1 Нанесение изображений при помощи родного API

Canvas вставляется в разметку как и любой другой тег HTML. Работа с его содержимым осуществляется при помощи javascript.

Например:

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
</head>

<body>
<canvas id=«myCanvas» width=600px height=600px> … </canvas>

<script>
var canvas = document.getElementById («myCanvas»);
var context = canvas.getContext («2d»);

//Заливка
context.fillStyle = «#5F5»;
context.fillRect (10,10, 300, 300);

//надпись
context.font = «50px serif»
context.fillStyle = «#FF0000»;
context.fillText («Hello People!», 30, 90);

//фиолетовый прямоугольник
context.fillStyle = «violet»;
context.fillRect (25,25,150,150);

context.strokeStyle = «gray»; // цвет линии
context.lineWidth = 6; // толщина линии
context.strokeRect (150,150,150,150); // прямоугольник

context.clearRect (100,100,150,150); // очищаем прямоугольную область
</script>
</body>
</html>
```

Пример градиентной заливки

```
<script>
$ (function () {
var canvas = document.getElementById («myCanvas»);
var context = canvas.getContext («2d»);

//Линейный градиент
context.fillStyle = «#000000»;
```

```
var gradient = context.createLinearGradient (5,45,150,150);
gradient.addColorStop (0.0,«#00FF00»);
gradient.addColorStop (0.471428571, «#FF0000»);
gradient.addColorStop (1.0,«#0000FF»);

context.fillStyle = gradient;
context.fillRect (25,25,150,150);

//Радиальный градиент
var rad_grad = context.createRadialGradient (250,250,1, 150,250,120);
rad_grad.addColorStop (0,«#F00»);
rad_grad.addColorStop (0.5,«#0F0»);
rad_grad.addColorStop (1,«#00F»);
context.fillStyle = rad_grad;
context.fillRect (150,150,200,200);
});
</script>
```

Пример отрисовки фигуры «звезда»

```
//star
context.strokeStyle = «red»;
context.lineWidth = 10;
context.beginPath ();
context.moveTo (50,100);
context.lineTo (240,100);
context.lineTo (70,230);
context.lineTo (140,30);
context.lineTo (220,230);
context.closePath ();
context.stroke ();
```

Пример рисования кривой Безье

```
context.strokeStyle = «red»;
context.lineWidth = 7;
context.beginPath ();
context.moveTo (300, 400);
context. quadraticCurveTo (400, 500, 450,300);
//context.closePath ();
context.stroke ();
```

На «холст» можно загружать и уже имеющиеся растровые изображения из файлов. Для этого существует функция *drawImage ()*. Ниже ***пример*** ее использования.

Конец ознакомительного фрагмента.