Contents

jQuery 2

jQuery Magazinfy 2

Decisions take during the development 3

Box model 3

jQuery Mobile 3

Considerations 4

Box model 4

Layer movement 4

Touch compatibility 4

Multi-touch devices 5

Selectable text 5

Other approaches 5

Compatibility and known issues 6

# jQuery

jQuery is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML. It was released in January 2006 at BarCamp NYC by John Resig. Used by over 46% of the 10,000 most visited websites, jQuery is the most popular JavaScript library in use today.

jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets. The modular approach to the jQuery framework allows the creation of powerful and dynamic web pages and web applications

# jQuery Magazinfy

jQuery Magazinfy is a plugin that creates a book from a simple HTML markup, with an flip page effect and compatible with all touch devices.

The needed markup is as follows. Keep in mind that it doesn’t matter if the number of pages is not even, the plugin will add an extra blank page at the end.

<div id="myId">

<div><!-- Página 1 --></div>

<div><!-- Página 2 --></div>

<div><!-- Página 3 --></div>

<div><!-- Página 4 --></div>

</div>

In order to work, you have to import (or copy the content of) magazinify.css. This plugin depends on jQuery 1.5.1 and jQuery Mobile (better the reduced version).

To create a magazine you need to type the following line:

$('#myId').magazinify();

You can modify the by default values using an options object as a parameter to the magazinify call:

var options = {

duration: 1000, // transition time for a page in ms

width: 350, // Width in px of a page

height: 400, // Height in px of a page

pagePdding: 20, // Space in px between the content and the border

pageShadowWidth:10, // Width in px of the projected shadow of the pages

contentShadowWidth: 15, // Internal shadow width in px of each page classes: {

//Class that indicates the plugin activation

magazinify: 'magazinify',

// Class related to the layer that contains the shadow of each page

shadow: 'magazine\_shadow',

// Class for the initial container of each page (with the shadow)

content: 'magazine\_content',

// Class assigned to the container thas has 2 pages

sheet: 'magazine\_sheet',

// Class for the page container (contains content)

page: 'magazine\_page',

}

};

$('#myId').magazinify(options);

# Decisions take during the development

## Box model

During the development, we had to decide between the classic box-model and box-sizing.

content-box Default

This is the behavior of width and height as specified by CSS2.1. The specified width and height (and respective min/max properties) apply to the width and height respectively of the content box of the element. The padding and border of the element are laid out and drawn outside the specified width and height.

### border-box

The specified width and height (and respective min/max properties) on this element determine the border box of the element. That is, any padding or border specified on the element is laid out and drawn inside this specified width and height. The content width and height are calculated by subtracting the border and padding widths of the respective sides from the specified 'width' and 'height' properties. As the content width and height cannot be negative ([CSS21], section 10.2), this computation is floored at 0.

**Note** This is the behavior of width and height as commonly implemented by legacy HTML user agents for replaced elements and input elements.

We decided to use the classic content-box to remain as compatible as possible with browsers that do not support the new model.

## jQuery Mobile

We decided to use jQuery Mobile (although not completely) because of its integration with jQuery and its touch events. Because it doesn’t have a tool to select the functionality you want a manual clean was done reducing the size from 208KB to 34KB.

The kept modules are listed later in this document.

# Considerations

## Box model

When you are working with widths and paddings you need to keep have the box model in your mind at any time. Even if the CSS3 properties allow us to change the size of the box with box-sizing to facilitate the operations.

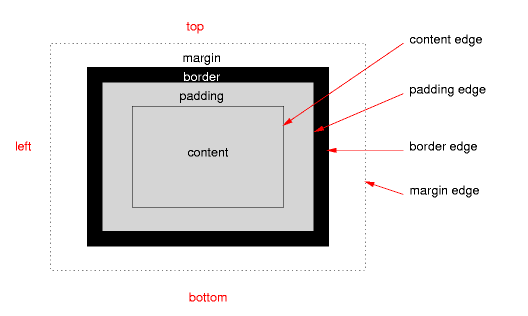


Figure 1: Different areas and borders of a common box

## Layer movement

Each of the layers has a different movement and not all of them have the same transition. Some can move a certain amount of pixel while others can shrink another different amount. There has to be independent formulas for each one of the properties and the layers.

## Touch compatibility

Mouse events are different from touch events so we have to establish equivalence. jQuery Mobile adds the property touch to its variable $.support. We don’t need all the jQuery Mobile library so we will borrow only the following:

* jQuery UI Widget
* jQuery Mobile Framework:
  + widget factory extentions for mobile
  + a workaround for window.matchMedia
  + support tests
  + "mouse" plugin
  + events

Using this approach we can use the jQuery touch events and establish a relationship between mouse and touch as follows:

var supportTouch = $.support.touch,

scrollEvent = "touchmove scroll",

touchStartEvent = supportTouch ? "touchstart" : "mousedown",

touchStopEvent = supportTouch ? "touchend" : "mouseup",

touchMoveEvent = supportTouch ? "touchmove" : "mousemove";

## Multi-touch devices

Multi-touch devices capture an array of events in each touch and we have to keep this in mind while hooking up an event with a DOM element. Because in this application we are supposed to use only one finger to flip the page, we will only take into account the first one of the events, verifying that it really exists:

var \_event = event.originalEvent.touches && event.originalEvent.touches[0] ? event.originalEvent.touches[0] : event;

## Selectable text

The flip movement can trigger the text selection so this behavior has to be blocked. To prevent this we do this:

$('#magazine').bind(

touchStart,

function(event)

{

event.preventDefault();

// Code...

}

);

# Other approaches

Different tests were performed during the development to achieve more realistic effects (flipping the upper corner for example) but the implementation using the mouse events needed of complicated formulas that take into account not only x and y but also the movements and speed. Although it is possible to develop this effects, we considered it out of the scope of this project.

# Compatibility and known issues

|  |  |
| --- | --- |
| Compatibility | Known issues |
| * Internet Explorer 9+ * Google Chrome 12+ * Firefox 4+ * Safari 5+ | Safari iOS:   * The outside shadow is not displayed * The transition never ends * The page goes further than expected   Firefox 3.6   * The outside shadow is not displayed |