Javascript Animation: Tutorial, Part 3

Creating Motion Tweens in Javascript

This is the long-overdue final piece of a series on creating animations in Javascript; see part 1 and part 2 for reference.

In the Flash world, objects can be animated or morphed between one state and the next over a period of time. This is where the term *tween* comes in. The developer does not have to write code to make the animation sequence, they simply specify the start and end points and Flash dynamically generates the frames in-be*tween* to make a smooth transition between states. With a tiny bit of math, you can easily make a reasonably-smooth tween generator in Javascript. This example will be very basic, but you could get much fancier with bezier and quadratic curves etc.

Full-featured animation APIs provided by the Yahoo! User Interface library, jQuery, Mootools and others allow you to specify start and end points, animation duration, tween effects and so on. In this article, I'll show a simple animation manager script which uses a few basic tween styles and a fixed frame count.

Less talk, and more demo:

- JS animation demo: Full-screen window + SFX (SoundManager 2)
- JS animation demo: Sequential + Simultaneous animation sequence

A Simple Motion Tween

Let's say we want to move an element from point A to point B. The tween in this case can be done in a linear fashion, or according to a curve or other function. A linear motion is easy, but doesn't look as nice as a fluid motion which mimics real-life physics. A simple way to do this is a "slow-fast-slow" curve; for example, where an object starts from 0 (not moving), accelerates to a peak and then slows down to 0 again.

Using a fixed number of frames (and thus, a non-fixed duration due to the asynchronous nature of Javascript), writing a linear tween is very simple:

```
var points = {
  // moving a box "from" and "to", eg. on the X coordinate
  'from': 200,
  'to': 300
}
var frameCount = 20; // move from X to Y over 20 frames
var frames = []; // array of coordinates we'll compute

// "dumb" tween: "move X pixels every frame"
var tweenAmount = (points.to - points.from)/frameCount;
for (var i=0; i<frameCount; i++) {
  // calculate the points to animate
  frames[i] = points.from+(tweenAmount*i);
}
// frames[] now looks like [205,210,215,220, ... ,300]; etc.
```

A more realistic tweening effect (including acceleration and deceleration) is achieved by changing the amount of motion made with each frame.

```
var points = {
  // moving a box "from" and "to", eg. on the X coordinate
  'from': 200,
  'to': 300
}
var animDelta = (points.to - points.from); // how far to move

// animation curve: "sum of numbers" (=100%), slow-fast-slow
var tweenAmount = [1,2,3,4,5,6,7,8,9,10,9,8,7,6,5,4,3,2,1];
// move from X to Y over frames as defined by tween curve
var frameCount = tweenAmount.length;
var frames = []; // array of coordinates we'll compute
var newFrame = points.from; // starting coordinate
for (var i=0; i<frameCount; i++) {
  // calculate the points to animate
  newFrame += (animDelta*tweenAmount[i]/100);
  frames[i] = newFrame;
}
// frames[] now looks like [201,203,206, ... ,228,236,245, ... ,297,299,300]; etc.
```

A linear acceleration and deceleration motion as above produces a visual improvement over a straight, linear motion. Each loop iteration adds a percentage of the total distance to move according to the tween amount, which adds up to 100%. Full-featured Javascript animation libraries go far beyond this in using bezier curves and fancier math functions, which provide much smoother motion tweens.

An Example Animation

Here, we'll create an animation object with some parameters and event handlers, and then start it.

```
var oAnim = new Animation({
  from: 0,
  to: 50,
  ontween: function(value) {
    writeDebug('oAnim.ontween(): value='+value);
  },
  oncomplete: function(value) {
    writeDebug('oAnim.oncomplete()');
  }
});
oAnim.start();
```

Nothing is actually animated on-screen in this case, but the relevant events fire and the tween value is updated as the animation runs. This could be used to move a box from 0 to 50 pixels across the screen, or from 0% to 50% of the screen or some other value.

Writing an Animation API in Javascript

Typically I like to have a single animator object which is responsible for managing all animations, following the idea that a single `setInterval` is regarded as the best way to do timing for Javascript animation loops. When an animation is started, a timer is created (if not already) and the collection of objects you have registered can then be animated. Despite the single timer and loop, each animation will effectively run on its own "timeline", and if you use different tweening functions, at its own pace.

For an individual animation effect from X to Y, an animation object is created with `from:` and `to:` properties, along with `ontween()` (called "per-frame") and `oncomplete()` event handlers/callbacks for the animation. The actual animation work is done within `ontween` in this example; most actual Javascript animation libraries allow you specify multiple HTML attributes to animate without any `ontween()` work needed, eg. `{marginLeft:{from:100,to:300},marginTop:{from:0,to:50}}` and so on. The syntax just shown is used by the animation component of the Yahoo! User Interface library.

```
function Animator() {
  // controller for animation objects.
  var self = this;
  var intervalRate = 20;
  this.tweenTypes = {
    // % of total distance to move per-frame, total always = 100
    'default': [1,2,3,4,5,6,7,8,9,10,9,8,7,6,5,4,3,2,1],
    'blast': [12,12,11,10,10,9,8,7,6,5,4,3,2,1],
    'linear': [10,10,10,10,10,10,10,10,10,10]
  }
  this.queue = [];
  this.queueHash = [];
  this.active = false;
  this.timer = null;
  this.createTween = function(start,end,type) {
    // return array of tween coordinate data (start->end)
    type = type||'default';
    var tween = [start];
    var tmp = start;
    var diff = end-start;
    var x = self.tweenTypes[type].length;
    for (var i=0; i<x; i++) {
      tmp += diff*self.tweenTypes[type][i]*0.01;
      tween[i] = {};
      tween[i].data = tmp;
      tween[i].event = null;
    }
    return tween;
  };

  this.enqueue = function(o,fMethod,fOnComplete) {
    // add object and associated methods to animation queue

    // writeDebug('animator.enqueue()');
    if (!fMethod) {
      // writeDebug('animator.enqueue(): missing fMethod');
    }
    self.queue.push(o);
    o.active = true;
  };

  this.animate = function() {
    // interval-driven loop: process queue, stop if done
    var active = 0;
    for (var i=0,j=self.queue.length; i<j; i++) {
      if (self.queue[i].active) {
        self.queue[i].animate();
        active++;
      }
    }
    if (active == 0 && self.timer) {
      // all animations finished

      // writeDebug('Animations complete');
      self.stop();
    } else {
      // writeDebug(active+' active');
    }
  };

  this.start = function() {
    if (self.timer || self.active) {
      // writeDebug('animator.start(): already active');
      return false;
    }
    // writeDebug('animator.start()');

    // report only if started
```

```
    self.active = true;
    self.timer = setInterval(self.animate,intervalRate);
  };

  this.stop = function() {
    // writeDebug('animator.stop()',true);

    // reset some things, clear for next batch of animations
    clearInterval(self.timer);
    self.timer = null;
    self.active = false;
    self.queue = [];
  };

};

var animator = new Animator();
```

Efficiency: Single Timer, Many Animations

The `Animation` object is a simple manager for multiple animations. It has a method for creating tweens, queueing for individual `Animation()` instances, and start and stop methods. When an animation is created and played for the first time, it adds itself via `animator.enqueue()` and then calls `animator.start()` if needed to kick things off. This way, the animator only ever has one single `setInterval` timer active - a good thing - and it only runs when animations are actively running. Once the last animation has finished, the animator shuts down until the next time it's needed.

```
function Animation(oParams) {
  // Individual animation sequence

  /*
    oParams = {
      from: 200,
      to: 300,
      tweenType: 'default', // see animator.tweenTypes (optional)
      ontween: function(value) { ... }, // method called each time (required)
      oncomplete: function() { ... } // when finished (optional)
    }
  */
  var self = this;
  if (typeof oParams.tweenType == 'undefined') {
    oParams.tweenType = 'default';
  }
  this.ontween = (oParams.ontween||null);
  this.oncomplete = (oParams.oncomplete||null);
  this.tween = animator.createTween(oParams.from,oParams.to,oParams.tweenType);
  this.frameCount = animator.tweenTypes[oParams.tweenType].length;
  this.frame = 0;
  this.active = false;

  this.animate = function() {
    // generic animation method
    if (self.active) {
      if (self.ontween && self.tween[self.frame]) {
        self.ontween(self.tween[self.frame].data);
      }
      if (self.frame++ >= self.frameCount-1) {
        // writeDebug('animation(): end');
        self.active = false;
        self.frame = 0;
        if (self.oncomplete) {
          self.oncomplete();
        }
        return false;
      }
```

```
      return true;
    } else {
      return false;
    }
  };

  this.start = function() {
    // add this to the main animation queue
    animator.enqueue(self,self.animate,self.oncomplete);
    if (!animator.active) {
      animator.start();
    }
  };

  this.stop = function() {
    self.active = false;
  };

};
```

Fun Stuff: Animation Sequences

Let's say you wanted a multi-part animation sequence, where a box resizes and moves in separate motions. A chain can easily be made using the `oncomplete()` method of separate animation objects.

In this case, a sequence of `animations[]` objects is made and a function simply iterates through them, increasing a counter each time an animation's `oncomplete()` fires. Animations can be fired and run simultaneously, as shown in the case of the last two.

```
var animations = [];
var animationCount = 0;

function nextAnimation() {
  if (animations[animationCount]) {
    // writeDebug('starting animation '+animationCount);
    animations[animationCount].start();
    animationCount++;
  }
}

// generic tween handlers
function animateBoxX(value) {
  document.getElementById('box2').style.left = value+'px';
}

function animateBoxH(value) {
  document.getElementById('box2').style.height = value+'px';
}

// motion #1
animations.push(new Animation({
  from: 0,
  to: 200,
  ontween: animateBoxX,
  oncomplete: nextAnimation
}));

// motion #2
animations.push(new Animation({
  from: parseInt(document.getElementById('box2').offsetHeight),
  to: 300,
  ontween: animateBoxH,
  oncomplete: nextAnimation
}));
```

```
// motion #3
animations.push(new Animation({
  from: 200,
  to: 400,
  ontween: animateBoxX,
  oncomplete: function() {
    // run last two animations simultaneously
    nextAnimation();
    nextAnimation();
  }
}));

// simultaneous #1
animations.push(new Animation({
  from: 400,
  to: 800,
  ontween: animateBoxX
}));

// simultaneous #2
animations.push(new Animation({
  from: 300,
  to: 600,
  ontween: animateBoxH
}));

nextAnimation(); // start the sequence
```

Javascript Animation API demo: Sequential + Simultaneous animation sequence

Demo Limitations, Annoyances etc.

While intended as examples, some shortcuts were taken in writing the demo animation code.

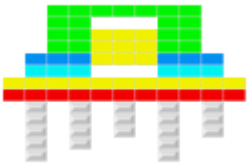
- Fixed frame count means animation duration is variable, and varies between browsers/platforms and CPU etc. (Ideally, duration is configurable and animation runs at best-possible speed, skipping frames if need be - eg., try to render 100 fps, calculate tween array of (frames * duration) up-front and skip to nearest frame based on Date() vs. time animation started while running.)
- One tween per animation, code doesn't conveniently handle animation of arbitrary (or multiple) HTML properties.
- `ontween()` is somewhat limited, doesn't provide access to all properties of the animation object etc.

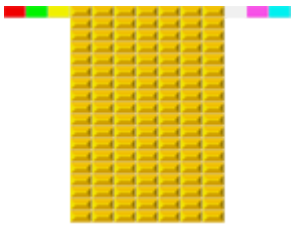
If for some reason you needed to write your own animation script, these would be items to consider. I also recommend the YUI animation library, which is quite solid. Bernie's Better Animation Class is a similar standalone script worth checking out.

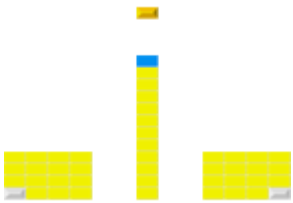
Related links

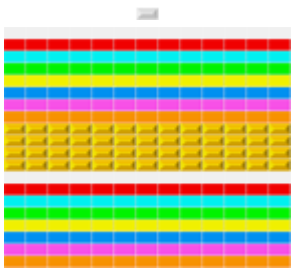
- Javascript Animation API demo: Full-screen window
- Javascript Animation API demo: Full-screen window + Sound
- Javascript Animation API demo: Sequential + Simultaneous animation sequence
- Bernie's Better Animation Class
- Javascript Animation: Tutorial, Part 1
- Javascript Animation: Tutorial, Part 2

- # Recently.
    - **Arkanoid: Recent Submissions**









[Create a level](#) | [More..](#)

Amateur Photography - Photos from Schill

Heavy Rotation - Recent top plays @last.fm

- Trust Me - Guru
- Down the Backstreets - Guru
- Sights In The City - Guru
- Slicker Than Most - Guru
- Looking Through Darkness - Guru

Reading.

Most Recently

Armor Alley (1990): Web Prototype
: A browser-based interpretation of the MS-DOS release of Armor Alley, a combat strategy game originally released in 1990 for MS-DOS PCs and the Macintosh. Written in HTML, JavaScript and CSS.

SURVIVOR: Remaking A C64 Game In HTML
: Remaking an Atari / Commodore 64-era space-based "shoot-'em-up" arcade game in HTML + CSS + JavaScript, including a level editor and design tool, thirty years after its release.

The Wheels Of Steel: An Ode To Turntables (in HTML)
: Building a browser-based turntable prototype using HTML, CSS, JavaScript and Flash: A bit of history, screenshots, explanation of the approach taken, limitations found and sample code.

How SoundManager 2 Works
: Background details to the JavaScript and Flash that makes SoundManager 2 work, plus technical notes and other findings about Flash's wacky ExternalInterface functionality.

CSS 3 and The Future: Image-free Rounded Corners, Drop Shadows and Gradients
: A brief history of rounded corner CSS 2 hacks and some examples of effects using border-radius, box-shadow and other fancy CSS 3 attributes (and vendor-specific extensions) allowing modern browsers to render shiny UI designs entirely from code.

The Cost of "Social Media" Javascript Overload

Security, load time and user experience are all affected by the amount of inline Javascript blocks and third-party script "includes" being crammed into some modern sites, particularly commercial and tech blogs. This entry is a rant, and some suggestions to improve performance.

HD Time-Lapse Photography: A How-To
: A hacker type can't simply plant basil and cilantro outside, and leave them alone to grow; of course there has to be something else, preferably nerdy, in the process. This is how I ended up getting into time-lapse photography and making videos from sequences of photos using a Canon camera, the free CHDK software, and QuickTime.

Javascript Malware Analysis: A Case Study
: When JS goes bad: Remote iFrame tricks on legitimate (or phishing) sites can load Javascript exploits or "shellcode", which can mean drive-by downloads and other risks. This is an example of some heavily-obfuscated code seen on a recent Facebook phishing site.

Javascript Animation: Tutorial, Part 3
: The final, and overdue, part in a series on Javascript animation techniques: Creating tweens, simultaneous animations and event handling are discussed. A simple animation API, demo and source code is included.

Extension Wars: Adblock Plus vs. NoScript
: Is it okay for one popular Firefox extension to alter the behaviour of another without asking for your permission? (PS: Internet drama/tempest in a teapot amusement goes here.)

Optimizing Your External Javascript References
: Traditionally, the script tag blocks parse, load and render of a web page. A few ideas around this include loading JS with JS, the defer attribute and caching locally.

Enhancing YUI-based Applications With Audio
: (yuiblog.com article) Thoughts about audio and how it can be applied online to user interface/experience design.

Flickr Web Uploadr: File uploads via YUI
: (code.flickr.com article) A brief history of how browser-based file uploads suck, and how we made them a lot better on Flickr.

A Snapshot of The Yahoo! Photos Beta (from 2006)
: Javascript performance learnings: Memory leaks and management, event delegation, performance tweaks, debugging, UI observations and side thoughts from a web development perspective from my time working on the Ajax-heavy redesign of Yahoo! Photos, between 2005 and 2006.

What I Did In 2008
: A productive year seeing the redesign of SoundManager 2, the Javascript Sound library, a refresh of Snowstorm (a free Javascript Snow effect), continued downtempo and trip-hop mixes, binaural audio recordings, time-lapse photography and video experiments.

On UI Quality (The Little Things)
: (code.flickr.com article) Observations of how browsers do linear and bicubic image resampling when resizing, and how to make them look better in IE 6 and 7.

Playing.

Entertainment: Games

DHTML Arkanoid
: An attempt at recreating the classic Arkanoid arcade game entirely in DHTML. Includes a level editor and a highscores list.

"Smash Christmas Lights"
: A seasonal twist on the 2007/08 web site theme, with animations and sound effects. There is a bonus sequence for those who are "thorough"..

Fireworks.js
: A goofy interactive Javascript Fireworks experiment, with animation and sound.

"Web 2.0 Awareness Test"

A Javascript/CSS experiment which gives you a score based on what Web 2.0-related sites you may have recently visited. (Written in 2006.)

- ### Code: F/OSS/Things You Can Use

SoundManager 2: Javascript Sound For The Web
: A new, fancier version of a Javascript sound project allowing people to add sound, or make sound apps using Javascript.

Javascript MP3 Player Demo
: An example of SoundManager 2, showing how you can make MP3 links play inline.

Even More Rounded Corners With CSS
: Another take at creating single-image, alpha-transparent, fluid, rounded corner dialogs with CSS.

Javascript Animation: Tutorial, Part 1
: Part one of a planned series: Theory behind Javascript/DHTML-based animation via setTimeout and setInterval.

Javascript Animation: Tutorial, Part 2
: Discussing efficiency regarding interval-based animation

Javascript Animation: Tutorial, Part 3
: The final, and overdue, part in a series on Javascript animation techniques: Creating tweens, simultaneous animations and event handling are discussed. A simple animation API, demo and source code is included.

SnowStorm: A Javascript Snow Effect
: A downloadable DHTML component that creates an animated snowfall effect. Uses PNG-based images where supported.

Thinking.

- Observations, Thoughts and Rants

Writing: Highlights

Non-technical

Top 5 favourite albums: Beck: Mellow Gold
: Comments on Beck's classic break-out album from 1994, with a few insights from this Beck fan.

What Turntables To Buy?
: My response to a forum post on choosing DJ equipment.

On DJing and Turntablism
: Why you should buy a pair of 1200s, a mixer and some records.

Nerd glasses required

Sound For The Web: SoundManager 2 (Article)
: A new, fancier version of a Javascript sound project allowing people to add sound, or make sound apps using Javascript.

How Web 2.0-aware Are You?
: A humorous look at Web 2.0-related sites, using a trick that tests your browser's visited URL history.

Writing HTML like it's 1993
: A new job update, California, Eric and Molly teach CSS.

Don't Believe The (Web 2.0) Hype!
: An attempt at popping the latest Internet 2.0 bubble and related technologies, along with a wishlist of ideals.

On SEO and Search Engine Spamming

From a post on theroot42.org originally entitled, 'SEO/How To Do Well On The Search Engines'

Application/XHTML+XML
: What I learned about XHTML when switching schillmania.com to application/xhtml+xml (forcing true XML validation.)

The Obligatory Standards Rant
: A rant advocating standards on a respected design-oriented forum.

Spam, spam, spam
: Spam: The scourge of the Internet. A quick overview-cum-rant, and some recommended tools to fight the spam war.

- # Listening.

  - Random soundbites + video clips

  ### Binaural Audio Recordings

  Hand/Eye Coordination
  : An attempt at 'Beat juggling' a seamless loop from a sample using turntables and two copies of the same record.

  Drumming with Vinyl
  : Correctly scratch a record with a drum sample on it, and you can make your own variation on a theme.

  Scratching .. with a mouse?
  : Combine a turntable with a computer mouse and a copy of the SoundCraft vinyl scratching emulation engine, and you have at least 30 seconds of entertainment.

  Chill with Schill
  : A two-hour Downtempo/Trip-Hop mix, recorded on a lazy afternoon while drinking a $4 mocha from Starbucks. How's that for detail?

- # Reacting.

  - Get In Touch.

  ### Interact

  Contact
  : How to get in touch with Scott Schiller.

  ### Elsewhere Online

  schill on Flickr
  : Scott's Flickr profile and photos

  chillwithschill on Last.fm
  : Scott's last.fm profile and music interests

  scottschiller.com: Portfolio
  : Scott Schiller's professional and personal portfolio site. (Last updated: 2004)

  freshly ground: Experimental sound/music
  : A side-project of experiments in music and binaural audio recordings

- # Digging.

  - The Archives: Design and content

Alternate Themes

2008 "Back To Basics" edition
: Schillmania! 2008 Edition: Back To Basics (CSS grid layout, minimal JS/images)

2007/08 "Night and Day" edition
: Schillmania! 2007/08 Edition: "Night and day" theme with time-sensitive UI/design, environmental sound effects and ambience

"Smash Christmas Lights" 2007
: Schillmania! 2007 Edition: "Smash Christmas Lights" seasonal feature

2006 "Work/life balance" edition
: Schillmania! 2006 Edition: Archived design (Sliding feature photo, two-column layout)

2005 "Bouncing Icon" edition
: Schillmania! 2005 Edition: Archived design (Horizon, bouncing icon layout)

2004 Edition: Now With More Funk
: Schillmania! 2004 Edition: Archived design (Multi-themed sliding photos+sound layout, may be buggy)

Design Archives

A Five-Year Retrospective
: A look back on the origins of schillmania.com and its previous incarnations.

2003 (Spring.03): Strictly XHTML edition
: Schillmania! 2003 Edition: Archived design ("Newspaper style", first major XHTML + heavy JavaScript rewrite)

News + Article Archives

Everything else not previously mentioned.

2005 - 2007

A Fresh Redesign: 2007/08 edition
: An aggressive new time-sensitive redesign renders the site with different visual and audible cues over the course of the day.

A Holiday Update (2006)
: The usual holiday festivities: Snow, good tunes, smashing christmas lights and other debauchery.

Sound For The Web: SoundManager 2 (Article)
: A new, fancier version of a Javascript sound project allowing people to add sound, or make sound apps using Javascript.
: The usual holiday festivities: Snow, good tunes, smashing christmas lights and other debauchery.

Beck Hacks Yahoo!
: Beck Hansen and his band play an unannounced gig at Yahoo! Hack Day.

How Web 2.0-aware Are You?
: A humorous look at Web 2.0-related sites, using a trick that tests your browser's visited URL history.

TJ Schiller Stomps US Open, X-Games
: My brother takes first place in two massive comps, throwing a never-done-before trick at one.

World Of Trip-Hop Mix
: A 160-minute mix of downtempo and trip-hop.

TJ Schiller wins 2005 WSI Slopestyle
: My brother wins at a popular freestyle skiing event.

A Piece of Okanagan Golf Paradise
: A site built for a tourist vacation company at Predator Ridge.

The Most Expensive Cab Ride. Ever.
: Regarding the purchase of a 2005 Toyota Celica.

Sugar Ray Rocks Yahoo! 10th Anniversary
: Yahoo celebrates 10 years with the musical talents of Sugar Ray.

Beck Live in Santa Cruz
: Meeting Beck Hansen, my favourite musician of all-time, after a concert in Santa Cruz

Writing HTML like it's 1993
: A new job update, California, Eric and Molly teach CSS.

Don't Believe The (Web 2.0) Hype!
: An attempt at popping the latest Internet 2.0 bubble and related technologies, along with a wishlist of ideals.

On SEO and Search Engine Spamming
: From a post on theroot42.org originally entitled, 'SEO/How To Do Well On The Search Engines'

What Turntables To Buy?
: My response to a forum post on choosing DJ equipment.

Way To D'oh!
: Some tech/geek thoughts on 'Way To Dough', a neat web-based promotion for the Subway chain of stores.

2004

The (almost) Five-Dollar Coffee
: Starbucks ups the cost of already-overpriced coffee; I've had enough.

Application/XHTML+XML
: What I learned about XHTML when switching schillmania.com to application/xhtml+xml (forcing true XML validation.)

The Obligatory Standards Rant
: A rant advocating standards on a respected design-oriented forum.

On DJing and Turntablism
: Why you should buy a pair of 1200s, a mixer and some records.

Spam, spam, spam
: Spam: The scourge of the Internet. A quick overview-cum-rant, and some recommended tools to fight the spam war.

Hand/Eye Coordination
: An attempt at 'Beat juggling' a seamless loop from a sample using turntables and two copies of the same record.

15 Minutes Of Fame
: The UK-based Magazine 'Web Designer,' writes some kind words about my work.

35mm Photo Viewer V5.0B
: A free, downloadable web-based photo viewer with optional XML templates.

Drumming with Vinyl
: Correctly scratch a record with a drum sample on it, and you can make your own variation on a theme.

The (almost) Five-Dollar Coffee
: Starbucks ups the cost of already-overpriced coffee; I've had enough.

Application/xhtml+xml
: What I learned about XHTML when switching schillmania.com to application/xhtml+xml (forcing true XML validation.)

TJ Schiller at Freshtival 04
: My younger brother TJ does the skiing thing at Calgary's Freshtival, a freestyle skiing movie release event and rail session, with Tanner Hall and others.

Javascript Sound API
: An API and flexible Flash template for adding sound to websites, controlled entirely by Javascript.

A Real-World Website Counter

A parallel-port-driven mileage-style counter is connected to schillmania.com. Meanwhile, rumors of geek-like antics abound.

Photo Viewer Update: v4.0a
: A DHTML-driven photo viewer application is revamped with a basic search engine and other nifty features.

When Lightning Strikes - Next Door
: A nasty storm in Calgary hits the neighbours', damaging all sorts of electronic equipment.

Montreal, Quebec, Toronto, Windsor
: Photos and commentary on my vacation to Ontario and Quebec, which have some great summer weather - both sunny and stormy.

The Obligatory Standards Rant
: A rant advocating standards on a respected design-oriented forum.

Critical Mass: Town Hall 2004
: The annual Town Hall event at Critical Mass happens on-time, but with a presentational twist.

Scottschiller.com v4: Portfolio Site Update
: Scott Schiller finally gets around to updating his technical/work experience portfolio site - now at version 4.

Flames win Game 3, city loses its mind
: The Calgary Flames win Game 3 of the Stanley Cup finals, prompting a massive and energetic crowd along the ever-popular 17th Avenue area downtown.

DJing, beer, fire and .. rollerskating?
: Urban nightlife activities keep one Scott Schiller away from the computer.

Josh Bibby, TJ Schiller stomp the WSI
: Two crazy freestyle skiiers from Vernon, BC turn heads at the 2004 Whistler Ski Invitational.

Okanagan Road Trip Photos
: Some new panoramic photos and other images from a trip to the Okanagan region of BC, Canada.

Scratching .. with a mouse?
: Combine a turntable with a computer mouse and a copy of the SoundCraft vinyl scratching emulation engine, and you have at least 30 seconds of entertainment.

Special Blend
: Vietnamese iced coffee, 'ca phe sua da' - an enjoyable caffeinated beverage.

Site Colophon
: A description of how the content for this site is generated and served.

Aquarium Time Lapse Video
: A time-lapse video (~12 days) of a fish tank.

Back in Black
: An overview of the site redesign.

TJ Schiller at the US Freeskiing Open
: TJ Schiller (my younger brother) wins the Slopestyle comp at the US Freeskiing open.

- **2003**

SnowStorm Update
: Changes and bug fixes relating to the DHTML SnowStorm component (Version 1.2a.)

SnowStorm Component
: Continued description of SnowStorm, now a downloadable component-based javascript snow library.

Bring the SnowStorm (DHTML, that is..)
: An introduction to SnowStorm, a DHTML component that provides a neat animated snow effect; uses PNG-based images where supported.

More Fish Stuff
: Some small additions (and consequent losses) involving a 90-gallon fish tank.

Ugly Hallowe'en Theme
: The one time of the year where the Spring.03 'Bold' theme actually suits the season.

I Love Spam

An experiment in spambots: 'How long until a new address is spammed?'