**TÌM HIỂU CÁC THÀNH PHẦN HTML5 HỖ TRỢ PHÁT TRIỂN GAME**

1. **Giới thiệu về HTML5**

HTML5 là một chuẩn mới của HTML (HyperText Markup Language), là thế hệ tiếp theo của chuẩn HTML4.01 vốn đã được sử dụng từ năm 1999.

Sau một khoảng thời gian dài, công nghệ web cần phải có những thay đổi để đáp ứng nhu cầu của con người ngày càng tăng đã vượt xa ra ngoài khả năng của HTML4 đơn giản. HTML5 được hình thành để tập hợp các tính năng, công nghệ và các API, những thứ sẽ mang sức mạnh của một ứng dụng trên nền tảng PC và trải nghiệm đa phương tiện lên nền tảng WEB – trong khi tăng cường sức mạnh cốt lõi của công nghệ WEB là sự tương tác và kết nối.

HTML5 kết hợp với CSS3, và một loạt các JavaScript API. Cùng với nhau, các công nghệ này cho phép bạn tạo ra các ứng dụng phức tạp mà trước đây chỉ có thể được tạo ra cho các nền tảng PC.

Được phát triển bởi sự cộng tác của World Wide Web Consortium (W3C) và Web Hypertext Application Technology Working Group (WHATWG) HTML5 không thuộc về một công ty hoặc một trình duyệt cụ thể. Nó được thử nghiệm và hổ trợ bởi một cộng đồng của những người quan tâm đến việc phát triển các trang web, các tập đoàn dẫn đầu về công nghệ bao gồm Google, Microsoft, Apple, Mozilla, Facebook, IBM, HP, Adobe và nhiều công ty khác.

Cộng đồng và các tập đoàn tiếp tục hợp tác trên các tiêu chuẩn trình duyệt phổ biến để thúc đẩy mở rộng các khả năng của công nghệ web. Thế hệ kế tiếp của ứng dụng web có thể chạy đồ họa hiệu suất cao, làm việc ngoại tuyến, lưu trữ một lượng lớn dữ liệu trên máy khách, thực hiện các phép tính nhanh chóng, và có thể tương tác, phối hợp tạo ra ứng dụng web ở một tầm cao mới.

HTML5 dự kiến sẽ ra mắt chính thức vào năm 2012, và đạt đến mức tương thích giữa các trình duyệt vào năm 2022. Dù chặng đường có thể còn dài, nhưng những lợi ích mà HTML5 đem lại có thể đảm bảo cho nó một thành công không kém các phiên bản trước. HTML5 được dự đoán sẽ phổ biến như HTML4 hiện nay vào một ngày không xa. Hiện tại HTML5 chưa thể thay thế các công nghệ trước nó nhưng tương lai sẽ là của HTML5 vì những nền tảng ứng dụng cạnh tranh với nó đang bắt đầu bị ngừng phát triển (Flash ngừng phát triển trên trình duyệt di động, Microsoft cũng ngưng phát triển mạnh Silverlight mà chuyển sang tập trung HTML5…).

1. **Các tính năng mới**
2. **Đa phương tiện (Video, Audio)**
3. *Tại sao lại cần chúng?*

Trước đây, nếu người lập trình web muốn đặt video vào trang của họ, họ phải sử dụng đến <object>. Do sự khác biệt của trình duyệt, họ phải dùng đến các phần tử <embed> và lặp lại nhiều thông số. Điều này dẫn đến một đoạn mã rắc rối như thế này:

*<object width=”425” height=”344”>*

*<param name=”movie” value=”http://www.youtube.com/9sEI1AUFJKw&hl=en\_GB&fs=1&”></param>*

*<param name=”allowFullScreen”*

*value=”true”></param>*

*<param name=”allowscriptaccess”*

*value=”always”></param>*

*<embed src=”http://www.youtube.com/9sEI1AUFJKw&hl=en\_GB&fs=1&”*

*type=”application/x-shockwave-flash”*

*allowscriptaccess=”always”*

*allowfullscreen=”true” width=”425”*

*height=”344”></embed>*

*</object>*

Như trên, chúng ta có thể thấy đoạn mã đó khá là “xấu xí” và “phức tạp”. Tệ hơn nữa, thực tế là các trình duyệt vượt qua nhiều điều nữa, đó là người dùng có cài plugin video hay chưa, rồi phải hy vọng là plugin đó đúng phiên bản để nó có thể làm việc tốt. Mà không phải người dùng nào cũng có kiến thức đó (lựa chọn đúng plugin, tải plugin về và cài đặt,…), rồi hàng loạt các rắc rối khác kéo theo điển hình như người dùng phải thường xuyên cập nhật plugin, quá nhiều rào cản cho đến khi trình duyệt có thể chạy được một video. Thật rắc rối và phức tạp cho cả người lập trình lẫn người sử dụng.

Một trong những lợi thế chính của HTML5 là phần tử video, cuối cùng video là một thành phần chính thức trên trang web. Không cần phần tử <object> đại diện, cũng không cần các yếu tố <embed> nữa.

Khả năng hỗ trợ đa phương tiện trong HTML5 được hiện thực qua việc bổ sung hai thẻ mới là <video> và <audio>. Hãy thử tưởng tượng đây là tất cả những gì cần làm để thêm một đoạn phim vào trang web:

*<video src=“myVideo.mp4”></video>*

Vì vậy, phần tử <video> có thể định dạng theo kiểu CSS. Chúng có thể được thay đổi kích cỡ bằng cách chuyển đổi CSS. Chúng cũng có thể được tinh chỉnh và hiển thị lại vào <canvas> với JavaScript.

Các thẻ video và audio của HTML5 được thiết kế để làm cho việc nhúng một video dễ dàng như việc nhúng một hình ảnh. Chúng cũng được thiết kế để cung cấp cho người dùng một trải nghiệm nhanh hơn so với các plugin như Adobe Flash.

1. *Các thành phần đa phương tiện và phát triển game*
2. Thẻ <audio>

Âm thanh là góp phần rất lớn tạo nên cái hay của các bộ phim cũng như các trò chơi. Hãy thử coi một bộ phim hay chơi một trò chơi mà tắt âm thanh đi, chúng ta sẽ nhận ra được vai trò quan trọng của nó.

1. Web Audio API.

Trước đây, Flash hoặc các plugin khác được sử dụng để tạo ra âm thanh trên các trang web. Từ khi tag <audio> HTML5 ra đời, âm thanh trên web không còn đòi hỏi plugin nữa, tuy nhiên tag <audio> cũng có những hạn chế đáng kể để thực hiện các trò chơi phức tạp và các ứng dụng tương tác.

Web Audio API là JavaScript API cấp cao để xử lý và tổng hợp âm thanh trong các ứng dụng web. Mục tiêu của API này là bao gồm khả năng nhầm thay thế các công cụ âm thanh hiện đại cũng như xử lý, pha trộn, và lọc các nhiệm vụ được tìm thấy trong các ứng dụng về âm thanh của máy tính hiện nay. Sau đây là một hướng dẫn cơ bản để sử dụng API mạnh mẽ này.

* Bắt đầu với AudioContext

AudioContext để quản lý và chơi tất cả các âm thanh. Tạo ra một âm thanh bằng cách sử dụng các trang Web Audio API nghĩa tạo ra một hoặc nhiều nguồn âm thanh và kết nối chúng đến đích âm thanh được cung cấp bởi thể hiện AudioContext. Kết nối này không cần trực tiếp, và có thể đi qua bất kỳ số lượng AudioNodes trung gian nào thứ mà thực hiện xử lý các mô-đun cho các tín hiệu âm thanh. Phần này sẽ được mô tả chi tiết hơn ở các đặc điểm kỹ thuật của Audio Web.

Một thể hiện AudioContext có thể hỗ trợ nhiều âm thanh đi vào và đồ thị âm thanh phức tạp, vì vậy chúng ta chỉ cần một thể hiện duy nhất cho mỗi ứng dụng âm thanh chúng ta tạo ra.

Sau đây là cách tạo ra một AudioContext:

var context;

window.addEventListener('load', init, false);

function init() {

try {

context = new webkitAudioContext();

}

catch(e) {

alert('Web Audio API is not supported in this browser');

}

}

* Tải âm thanh

Web Audio API sử dụng một AudioBuffer cho âm thanh có độ dài ngắn đến trung bình. Cách cơ bản là sử dụng XMLHttpRequest để lấy các tập tin âm thanh.

API hỗ trợ chạy các tập tin âm thanh ở định dạng khác nhau, chẳng hạn như WAV, MP3, AAC, OGG và một số định dạng khác. Các trình duyệt cũng có một vài khác biệt trong việc hỗ trợ các định dạng âm thanh khác nhau.

Đoạn sau đây là ví dụ cho việc tải một âm thanh:

var dogBarkingBuffer = null;

var context = new webkitAudioContext();

function loadDogSound(url) {

var request = new XMLHttpRequest();

request.open('GET', url, true);

request.responseType = 'arraybuffer';

*// Decode asynchronously*

request.onload = function() {

context.decodeAudioData(request.response, function(buffer) {

dogBarkingBuffer = buffer;

}, onError);

}

request.send();

}

Các tập tin âm thanh là dạng nhị phân (không phải văn bản), vì vậy chúng ta thiết lập các responseType là 'arraybuffer'.

Một khi dữ liệu tập tin âm thanh đã được nhận, nó có thể được giữ lại giải mã sau, hoặc nó có thể được giải mã ngay lập tức bằng cách sử dụng phương thức decodeAudioData() của AudioContext. Phương thức này lấy ArrayBuffer dữ liệu tập tin âm thanh được lưu trữ trong request.response và giải mã không đồng bộ.

Khi decodeAudioData () được hoàn thành, nó gọi một hàm callback thứ mà cung cấp các gải mã PCM dữ liệu âm thanh thành một AudioBuffer.

* Phát âm thanh



Một biểu đồ âm thanh đơn giản

Khi một hoặc nhiều AudioBuffers được nạp, sau đó chúng đã sẵn sàng để phát âm thanh. Giả sử một AudioBuffer với âm thanh của một con chó sủa đã tải hoàn tất. Sau đó, chúng ta có thể phát nó với mã sau đây.

var context = new webkitAudioContext();

function playSound(buffer) {

var source = context.createBufferSource(); *// creates a sound source*

source.buffer = buffer; *// tell the source which sound to play*

source.connect(context.destination); *// connect the source to the context's destination (the speakers)*

source.noteOn(0); *// play the source now*

}

Hàm playSound() có thể được gọi là mỗi khi có sự kiện từ bàn phím hay nhấp chuột.

Hàm noteOn() để dễ dàng sắp xếp phát lại âm thanh một cách chính xác. Tuy nhiên, để nó làm việc chính xác, cần phải đảm bảo rằng bộ đệm âm thanh đã được nạp sẵn.

Một lưu ý quan trọng nữa là đối với iOS, Apple hiện nay đã tắt tất cả âm thanh phát ra cho tới khi có sự tương tác của người dùng. Nghĩa là khi gọi hàm playSound() khi mới vào game có thể nó sẽ ko phát âm thanh, để khắc phục điều này, chúng ta có thể hướng người dùng vào sự tương tác sớm ví dụ như tạo 1 button “Chạm vào đây để chơi game”.

* Trừu tượng hóa

Tất nhiên, sẽ hay hơn nếu chúng ta tạo ra một hệ thống load tổng quát nhiều âm thanh để sử dụng sau này, đặc biệt là với game, có rất nhiều âm thanh cần quản lý. Một trong những cách giải quyết là sử dụng một lớp BufferLoader.

Sau đây là một ví dụ về cách sử dụng lớp BufferLoader. Trước tiên, tạo ra hai AudioBuffers, và ngay sau khi chúng được nạp, chúng ta có thể chơi chúng cùng một lúc.

window.onload = init;

var context;

var bufferLoader;

function init() {

context = new webkitAudioContext();

bufferLoader = new BufferLoader(

context,

[

'../sounds/hyper-reality/br-jam-loop.wav',

'../sounds/hyper-reality/laughter.wav',

],

finishedLoading

);

bufferLoader.load();

}

function finishedLoading(bufferList) {

*// Create two sources and play them both together.*

var source1 = context.createBufferSource();

var source2 = context.createBufferSource();

source1.buffer = bufferList[0];

source2.buffer = bufferList[1];

source1.connect(context.destination);

source2.connect(context.destination);

source1.noteOn(0);

source2.noteOn(0);

}

* Xử lý thời gian: phát âm thanh có nhịp điệu

Bằng việc xử lí thời gian, chúng ta có thể tạo ra 1 bài nhạc phức tạp hợp thành từ những âm thanh đơn giản.

Để chứng minh điều này, chúng ta hãy thiết lập một ca khúc nhịp điệu đơn giản. Đó là mẫu Drumkit khá phổ biến:



Một mẫu drum đơn giản

trong khi hihat được chơi tất cả các nốt thứ tám, kick and snare được chơi xen kẽ mỗi quý, 4/4 thời gian.

Đoạn mã này thực hiện điều đó:

var RhythmSample = {};  
  
RhythmSample.play = function() {  
    function playSound(buffer, time) {  
        var source = context.createBufferSource();  
        source.buffer = buffer;  
        source.connect(context.destination);  
        source.noteOn(time);  
    }  
  
    var kick = BUFFERS.kick;  
    var snare = BUFFERS.snare;  
    var hihat = BUFFERS.hihat;  
  
    // We'll start playing the rhythm 100 milliseconds from "now"  
    var startTime = context.currentTime + 0.100;  
    var tempo = 80; // BPM (beats per minute)  
    var eighthNoteTime = (60 / tempo) / 2;  
  
    // Play 2 bars of the following:  
    for (var bar = 0; bar < 2; bar++) {  
        var time = startTime + bar \* 8 \* eighthNoteTime;  
        // Play the bass (kick) drum on beats 1, 5  
        playSound(kick, time);  
        playSound(kick, time + 4 \* eighthNoteTime);  
  
        // Play the snare drum on beats 3, 7  
        playSound(snare, time + 2 \* eighthNoteTime);  
        playSound(snare, time + 6 \* eighthNoteTime);  
  
        // Play the hi-hat every eighthh note.  
        for (var i = 0; i < 8; ++i) {  
            playSound(hihat, time + i \* eighthNoteTime);  
        }  
    }  
};

* Thay đổi âm lượng

Một trong những thao tác cơ bản nhất mà người dùng muốn làm với một âm thanh thay đổi âm lượng của nó. Sử dụng API Audio Web, chúng ta có thể dùng AudioGainNode để điều khiển âm lượng:



Đồ thị âm thanh với GainNode

Thiết lập kết nối này có thể đạt được như sau:

*// Create a gain node.*

var gainNode = context.createGainNode();

*// Connect the source to the gain node.*

source.connect(gainNode);

*// Connect the gain node to the destination.*

gainNode.connect(context.destination);

tiếp theo, chúng ta đã có thể thay đổi âm lượng bởi thao tác gainNode.gain.value như sau:

*// Reduce the volume.*

gainNode.gain.value = 0.5;

Ngoài ra, chúng ta cũng có thể sử dụng AudioGainNode để trộn các âm thanh lại với nhau (tạo ra ứng dụng chơi nhạc cho DJ trên ngay trình duyệt).

Chúng ta cũng có thể lọc và tạo ra các hiệu ứng âm thanh đơn giản bằng đối tượng BiquadFilterNode



Biểu đồ âm thanh với BiquadFilterNode

Đoạn mã sau đây là ví dụ đơn giản để lọc low-bass từ một âm thanh:

*// Create the filter*

var filter = context.createBiquadFilter();

*// Create the audio graph.*

source.connect(filter);

filter.connect(context.destination);

*// Create and specify parameters for the low-pass filter.*

filter.type = 0; *// Low-pass filter. See BiquadFilterNode docs*

filter.frequency.value = 440; *// Set cutoff to 440 HZ*

*// Playback the sound.*

source.noteOn(0);

1. Nhạc nền

Khi tạo một trò chơi, một trong những yếu tố quan trọng đó chính là nhạc nền, chúng ta có thể vận dụng những gì đã tìm hiểu về Web Audio API ở phần trước đề tạo ra nhạc nền hấp dẫn, không đơn điệu. Nhạc nền, không có nghĩa chỉ là lặp đi lặp lại đoạn nhạc trong suốt trò chơi, nó dễ dàng tạo nên cảm giác nhàm chán cho người chơi.

Trong từng hoàn cảnh khác nhau, đòi hỏi nhạc nền cũng khác nhau, ví dụ lúc chiến đấu bình thường, lúc đánh với boss, lúc sắp mất hết máu,… Để giải quyết điều này, ta có thể dùng lớp BufferLoader để load các nhạc nền sẵn sàng lúc bắt đầu game.

function BufferLoader(context, urlList, callback) {

this.context = context;

this.urlList = urlList;

this.onload = callback;

this.bufferList = new Array();

this.loadCount = 0;

}

BufferLoader.prototype.loadBuffer = function(url, index) {

var request = new XMLHttpRequest();

request.open("GET", url, true);

request.responseType = "arraybuffer";

var loader = this;

request.onload = function() {

loader.context.decodeAudioData(request.response, function(buffer) {

if (!buffer) {

alert('error decoding file data: ' + url);

return;

}

loader.bufferList[index] = buffer;

if (++loader.loadCount == loader.urlList.length) loader.onload(loader.bufferList);

}, function(error) {

console.error('decodeAudioData error', error);

});

}

request.onerror = function() {

alert('BufferLoader: XHR error');

}

request.send();

}

BufferLoader.prototype.load = function() {

for (var i = 0; i < this.urlList.length; ++i)

this.loadBuffer(this.urlList[i], i);

}

Tiếp theo, mỗi nhạc nền ta tạo cho nó một GainNode tương ứng, ví dụ:

*// Assume gains is an array of AudioGainNode, normVal is the intensity*

*// between 0 and 1.*

var value = normVal \* (gains.length - 1);

*// First reset gains on all nodes.*

for (var i = 0; i < gains.length; i++) {

gains[i].gain.value = 0;

}

*// Decide which two nodes we are currently between, and do an equal*

*// power crossfade between them.*

var leftNode = Math.floor(value);

*// Normalize the value between 0 and 1.*

var x = value - leftNode;

var gain1 = Math.cos(x \* 0.5\*Math.PI);

var gain2 = Math.cos((1.0 - x) \* 0.5\*Math.PI);

*// Set the two gains accordingly.*

gains[leftNode].gain.value = gain1;

*// Check to make sure that there's a right node.*

if (leftNode < gains.length - 1) {

*// If there is, adjust its gain.*

gains[leftNode + 1].gain.value = gain2;

}

Như vậy, chúng ta có thể dễ dàng quản lý toàn bộ nhạc nền, rộng hơn nữa là quản lý các âm thanh của toàn bộ trò chơi một cách dễ dàng.

1. Hiệu ứng âm thanh

Một phần quan trọng của trò chơi nữa là hiệu ứng âm thanh, hàng loạt các hiệu ứng như vụ cháy nổ, tiếng súng, tiếng bước chân, chạy,… Nó cũng phải phát đi phát lại tương tự như âm thanh nền, nhưng có vài điểm khác cơ bản là:

+ Có hàng loạt các hiệu ứng tương tự nhau, chỉ khác nhau một chút về âm thanh cũng như tốc độ phát ra như tiếng các loại súng, tiếng bước chân từ chậm đến nhanh dần. Nói chung, chúng là những biến thể nhẹ của nhau.

+ Các âm thanh này phải lặp đi lặp lại nhiều lần và có thể phát cùng lúc hàng loạt các hiệu ứng khác nhau, trong thời gian chính xác. Hãy tưởng tượng đến một cuộc đấu súng giữa một nhóm người, tiếng các loại súng, tiếng người, tiếng bước chân, tiếng va chạm,…

Hiệu ứng âm thanh của Web Audio API thì thật sự tỏa sáng trong việc giải quyết các vấn đề trên.

Ví dụ sau đây tạo ra một vụ bắn súng máy từ một tiếng đạn được phát lại dồn dập trong một thời gian.

var time = context.currentTime;

for (var i = 0; i < rounds; i++) {

var source = this.makeSource(this.buffers[M4A1]);

source.noteOn(time + i \* interval);

}

Sau khi tạo ra được các tiếng súng đơn giản, ta nghĩ đến việc tiếng súng thay đổi tùy vào khoảng cách từ các mục tiêu cũng như vị trí tương đối. May mắn thay, Web Audio API cung cấp cho chúng ta có thể tùy chỉnh nó một cách dễ dàng theo hai cách:

+ Với một sự thay đổi tinh tế trong thời gian bắn ra của các viên đạn.

+ Bằng cách thay đổi playbackRate của âm thanh một cách ngẫu nhiên giống như ngoài đời thực.

Ví dụ:

function MachineGun(context) {

var ctx = this;

var loader = new BufferLoader(context, ['sounds/m4a1.mp3', 'sounds/m1-garand.mp3'], onLoaded);

function onLoaded(buffers) {

ctx.buffers = buffers;

};

loader.load();

}

MachineGun.prototype.shootRound = function(type, rounds, interval, random, random2) {

if (typeof random == 'undefined') {

random = 0;

}

var time = context.currentTime;

for (var i = 0; i < rounds; i++) {

var source = this.makeSource(this.buffers[type]);

source.playbackRate.value = 1 + Math.random() \* random2;

source.noteOn(time + i \* interval + Math.random() \* random);

}

}

MachineGun.prototype.makeSource = function(buffer) {

var source = context.createBufferSource();

var compressor = context.createDynamicsCompressor();

var gain = context.createGainNode();

gain.gain.value = 0.2;

source.buffer = buffer;

source.connect(gain);

gain.connect(compressor);

compressor.connect(context.destination);

return source;

};

1. Định vị âm thanh 3D

Khi tạo ra một trò chơi, ta thường phải tuân thủ các tính chất vật lý, hình học, của không gian 2D cũng như 3D. Nếu ta cũng tuân thủ điều đó đối với âm thanh thì quả là một điều tuyệt vời. Hãy nghĩ đến việc tiến lại gần hoặc ra xa mục tiêu, người chơi nhận được các âm thanh lớn nhỏ khác nhau một cách chân thật. Và tất nhiên, Web Audio API hỗ trợ ta làm điều đó một cách dễ dàng thông qua việc sử dụng AudioPannerNode. Ý tưởng cơ bản là hoạt động bằng cách thay đổi thiết lập vị trí các nguồn âm thanh như sau:

PositionSample.prototype.changePosition = function(position) {

*// Position coordinates are in normalized canvas coordinates*

*// with -0.5 < x, y < 0.5*

if (position) {

if (!this.isPlaying) {

this.play();

}

var mul = 2;

var x = position.x / this.size.width;

var y = -position.y / this.size.height;

this.panner.setPosition(x \* mul, y \* mul, -0.5);

} else {

this.stop();

}

};

1. Kết luận

Đó là tất cả những khía cạnh quan trọng, cần thiết để phát triển âm thanh cho trò chơi bằng cách sử dụng Web Audio API. Với kĩ thuật này, chúng ta có thể tạo ra những trải nghiệm âm thanh thật sự hấp dẫn ngay trên trình duyệt. Bên cạnh đó, một điều quan trọng mà người lập trình cần phải chú ý, đó là phải đảm bảo rằng âm thanh phải tạm dừng khi người dùng chuyển tab, chuyển trang hay cho trình duyệt chạy nền. Nếu không làm như vậy, chắc chắn nó sẽ tạo ra cảm giác vô cùng bực bội cho người dùng.

1. Thẻ <video>

Các thẻ <video> là một trong những tính năng của HTML5 được rất nhiều sự chú ý. Thường được sử dụng thay thế cho flash, bên cạnh đó, nó còn có những tính năng thật sự mạnh mẽ hơn rất nhiều. Mặc dù gần đây đã tham gia phần còn lại của các thẻ HTML phổ biến, khả năng của mình và hỗ trợ trên các trình duyệt đã tăng lên tại một tốc độ đáng kinh ngạc. Nó có thể tích hợp chồng với các lớp khác của web như CSS và JavaScript cũng như các thẻ HTML khác điển hình như thẻ <canvas>.

1. Các đánh dấu.

Để nhúng video vào trang web, chỉ cần các dòng sau đây là đủ:

<video>

<source src="movie.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"' />

<source src="movie.webm" type='video/webm; codecs="vp8, vorbis"' />

</video>

Đoạn này sử dụng thẻ <source> cho phép bao gồm nhiều định dạng khác nhau phòng trường hợp trình duyệt của người dùng trình duyệt không hỗ trợ một trong số đó.

Tất nhiên, chúng ta vẫn có thể sử dụng một định dạng video duy nhất làm cho cú pháp giống với các thẻ hình ảnh. Cú pháp này sẽ được sử dụng phổ biến hơn trong tương lai, khi mà tất cả các trình duyệt đều hỗ trợ các định dạng video phổ thông.

<video src="movie.webm"></video>

Để cải thiện hiệu suất phía người dùng điều quan trọng là xác định các loại định dạng trong các thẻ <source> trước tiên. Bằng cách này, trình duyệt có thể quyết định có nên tải về và chơi một video nào đó hay không. Nghĩa là, nó sẽ không tải về những loại video mà nó không thể chơi, bằng cách đó, chúng ta có thể cải thiện được hiệu suất của trang web.

1. Các định dạng Video

Hãy nghĩ một định dạng video như là một tập tin zip chứa các dòng video mã hóa và dòng âm thanh. Ba định dạng chúng ta cần quan tâm để phát triển web là (webm, mp4 và ogv):

* .mp4 = H.264 + AAC
* .ogg/.ogv = Theora + Vorbis
* .webm = VP8 + Vorbis

Tốc độ phát triển của thẻ <video> thật sự rất mạnh mẽ. Chỉ hai năm trước, khi mà thẻ <video> chỉ được hỗ trợ trên trình duyệt Safari, thì giờ đây hầu như tất cả các trình duyệt hiện tại đều hỗ trợ nó. Ngoài ra định dạng Webm càng trở nên phổ biến hơn thông qua dự án WebM. Internet Explore cũng hỗ trợ nó nếu codes này được cài đặt trên máy tính.

1. Các tính năng quan trọng với phát triển game

Như đã nói trong phần giới thiệu lợi thế chính của các thẻ <video> là một thành phần của HTML5, vì vậy nó có thể tích hợp với các lớp khác của web. Sau đây là những tính năng quan trọng đó:

* Video + các thành phần HTML khác

Tất cả các thuộc tính HTML thông thường bây giờ đã có thể được sử dụng trong thẻ <video>. Ví dụ, trong đoạn mã sau, sử dụng tabindex để bắt sự kiện từ bần phím. Có những thuộc tính mới có thể được sử dụng trong thẻ <video> cũng như thẻ <audio> chẳng hạn như “loop” và “autoplay”. Thuộc tính “poster” xác định hình ảnh nào sẽ được hiện khi video đang được load và khi đã kết thúc, “control” được sử dụng để chỉ ra rằng thay vì xây dựng các điều khiển tùy chỉnh, chúng tôi muốn trình duyệt tự động phát sinh nó. Ngoài ra còn có thuộc tính “preload”, chúng ta có thể sử dụng để tải về các video ở nền bên dưới ngay khi tải trang, ngay cả khi nó chưa bắt đầu phát.

<video *poster="star.png" autoplay loop controls tabindex="0"*>

<source src="movie.webm" type='video/webm; codecs="vp8, vorbis"' />

<source src="movie.ogv" type='video/ogg; codecs="theora, vorbis"' />

</video>

Kể từ khi video không còn là một đối tượng nhúng từ bên ngoài, có một số lợi ích khác cho trải nghiệm người dùng. Ví dụ, ngay cả khi nó đang được “focus” thì người dùng vẫn có thể scroll trang web hay sử dụng bàn phím bình thường.

* Video + JavaScript

Các thẻ video đi kèm với một tập hợp các thuộc tính và các phương thức, cho phép chúng ta có điều khiển tốt video từ mã JS.

Như bất kỳ phần tử HTML khác, chúng ta có thể gắn các sự kiện phổ biến cho thẻ <video> như “drag”, “mouse”,” focus”,… Tuy nhiên, điều này kèm theo hàng loạt các sự kiện mới phát sinh khi video đang phát, tạm dừng hoặc kết thúc. Khi bắt đầu tải một tài nguyên video về thì có rất nhiều thứ cần phải cẩn trọng, cũng như có rất nhiều sự kiện cần phải kiểm soát tốt trong quá trình tải, ở tầng mạng (loadstart, progress, suspend, abort, error, emptied, stalled) lẫn ở bộ đệm (loadedmetadata, loadeddata, waiting, playing, canplay, canplaythrough).

Ở cấp độ đơn giản nhất, bạn có thể gắn sự kiện “canplay” để bắt đầu làm việc với video.

video.addEventListener('canplay', function(e) {

this.volume = 0.4;

this.currentTime = 10;

this.play();

}, false);

* Video + CSS

Các thẻ <video> có thể được sử dụng theo CSS truyền thống (ví dụ như “border”, “opacity”,…) vì nó là một phần tử class đầu tiên trong DOM. Nhưng điều thú vị là ở bản mới nhất CSS3 thành phần video cũng có các thuộc tính như “reflections”, “masks”, “gradients”, “transforms”, “transitions” and “animations”.

*#video-css.enhanced {*

border: 1px solid white;

-moz-box-shadow: 0px 0px 4px *#ffffff; /\* FF3.5+ \*/*

-webkit-box-shadow: 5px 44px 28px *#333; /\* Saf3.0+, Chrome \*/*

box-shadow: 0px 0px 4px *#ffffff; /\* Opera 10.5, IE 9.0 \*/*

-moz-transform: translate(0, 10px); */\* FF3.5+ \*/*

-o-transform: translate(0, 10px); */\* Opera 10.5 \*/*

-webkit-transform: translate(0, 10px); */\* Saf3.1+, Chrome \*/*

}

* Video + Canvas

Canvas là một thành phần của HTML5 có nhiều tiềm năng khi được sử dụng kết hợp với các thẻ <video>.

Chúng ta có thể tận hai tính năng của phần tử <canvas> là nhập và xuất hình ảnh. Đầu tiên là phương thức drawImage cho phép nhập hình ảnh từ ba nguồn khác nhau: phần tử hình ảnh, phần tử <canvas> và phần tử <video>! Điều này có nghĩa rằng mỗi lần chúng ta chạy phương thức này, frame hiện tại trong đoạn video sẽ được nhập và trả lại vào canvas.

Đặc trưng thứ hai của thẻ <canvas> là chúng ta có thể sử dụng phương thức toDataURL để xuất nội dung của canvas thành một hình ảnh.

Đoạn mã sau đây sử dụng hàm drawImage cứ sau một khoảng thời gian 1,5 giây sẽ vẽ một hình ảnh từ video nguồn.

video\_dom.addEventListener('play', function() {

video\_dom.width = canvas\_draw.width = video\_dom.offsetWidth;

video\_dom.height = canvas\_draw.height = video\_dom.offsetHeight;

var ctx\_draw = canvas\_draw.getContext('2d');

draw\_interval = setInterval(function() {

*// import the image from the video*

ctx\_draw.drawImage(video\_dom, 0, 0, video\_dom.width, video\_dom.height);

*// export the image from the canvas*

var img = new Image();

img.src = canvas\_draw.toDataURL('image/png');

img.width = 40;

frames.appendChild(img);

}, 1500)

}, false);

* Video + SVG

SVG giúp chúng ta render và thao tác đồ họa vector, ngoài ra nó cũng đi kèm với nhiều tính năng như hiệu ứng lọc SVG. Với các bộ lọc, bạn có thể nhắm mục tiêu một phần tử DOM cụ thể và áp dụng một số các hiệu ứng như cảnh mờ, pha trộn, gạch,...

Ví dụ:

<svg id='image' version="1.1" xmlns="http://www.w3.org/2000/svg">

<defs>

<filter id="myblur">

<feGaussianBlur stdDeviation="1" />

</filter>

</defs>

</svg>

<style>

video { filter:url(#myblur); border: 2px solid red; }

</style>

1. *Thu âm và quay phim bằng HTML5*

Trước đây, để truy cập tới máy ảnh hay micro của người dùng chúng ta đã phải dựa vào các plugin (Flash hoặc Silverlight).

Giờ đây nhờ có HTML5 mang lại một sự đột phá về quyền truy cập vào thiết bị phần cứng cũng như định vị (GPS), API định hướng (gia tốc), WebGL (GPU), và Web Audio API (phần cứng) là những ví dụ điển hình.

Một API mới, navigator.getUserMedia(), cho phép các ứng dụng web để truy cập vào máy ảnh và microphone của người dùng.

1. Cơ bản

Giờ đây, với navigator.getUserMedia(), chúng ta đã có thể truy cập vào webcam và micro của người dùng mà không cần đến plugin. Thực hiện một cuộc gọi ngay trên trình duyệt, không cần phải cài đặt thêm gì cả, thật tuyệt vời!

* Kiểm tra sự hỗ trợ của trình duyệt

Tính năng này cho phép người lặp trình kiểm tra trình duyệt của người dùng có hỗ trợ chức năng này hay không:

function hasGetUserMedia() {

*// Note: Opera is unprefixed.*

return !!(navigator.getUserMedia || navigator.webkitGetUserMedia ||

navigator.mozGetUserMedia || navigator.msGetUserMedia);

}

if (hasGetUserMedia()) {

*// Good to go!*

} else {

alert('getUserMedia() is not supported in your browser');

}

* Tiếp cận với thiết bị đầu vào

Để sử dụng webcam hay microphone, chúng ta cần phải yêu cầu sự cho phép. Tham số đầu tiên getUserMedia () là một đối tượng xác định kiểu của thiết bị mà chúng ta muốn truy cập. Ví dụ, nếu muốn truy cập vào webcam, tham số đầu tiên là {video: true}. Để sử dụng cả hai micro và máy ảnh, pass {video: true, audio: true}:

<video autoplay></video>

<script>

var onFailSoHard = function(e) {

console.log('Reeeejected!', e);

};

*// Not showing vendor prefixes.*

navigator.getUserMedia({video: true, audio: true}, function(localMediaStream) {

var video = document.querySelector('video');

video.src = window.URL.createObjectURL(localMediaStream);

*// Note: onloadedmetadata doesn't fire in Chrome when using it with getUserMedia.*

*// See crbug.com/110938.*

video.onloadedmetadata = function(e) {

*// Ready to go. Do some stuff.*

};

}, onFailSoHard);

</script>

* Bảo mật

Một số trình duyệt trực tiếp hoạt động khi có lệnh gọi getUserMedia(), mà trước tiên nó cho phép người dùng tùy chọn cấp quyền hoặc từ chối quyền vào máy ảnh / micro của họ. Ví dụ, đây là hộp thoại confirm của Chrome:



* Xử lí khi trình duyệt không hỗ trợ

Đối với trường hợp trình duyệt người dùng không có hỗ trợ getUserMedia(), dưới đây là một trong những cách xử lí trường hợp này:

*// Not showing vendor prefixes or code that works cross-browser:*

function fallback(e) {

video.src = 'fallbackvideo.webm';

}

function success(stream) {

video.src = window.URL.createObjectURL(stream);

}

if (!navigator.getUserMedia) {

fallback();

} else {

navigator.getUserMedia({video: true}, success, fallback);

}

1. Chụp hình

Với phương thức ctx.drawImage(video, 0, 0) của thẻ <canvas> ta có thể dễ dàng vẽ frame <video> lên <canvas>. Chúng ta cũng đã có đầu vào video thông qua phương thức getUserMedia(), chúng ta có thể tạo ra hàng loạt các ứng dụng cũng như các trò chơi từ những thứ trên:

<video autoplay></video>

<img src="">

<canvas style="display:none;"></canvas>

var video = document.querySelector('video');

var canvas = document.querySelector('canvas');

var ctx = canvas.getContext('2d');

var localMediaStream = null;

function snapshot() {

if (localMediaStream) {

ctx.drawImage(video, 0, 0);

// "image/webp" works in Chrome 18. In other browsers, this will fall back to image/png.

document.querySelector('img').src = canvas.toDataURL('image/webp');

}

}

video.addEventListener('click', snapshot, false);

// Not showing vendor prefixes or code that works cross-browser.

navigator.getUserMedia({video: true}, function(stream) {

video.src = window.URL.createObjectURL(stream);

localMediaStream = stream;

}, onFailSoHard);

1. Sử dụng hiệu ứng

* CSS Filters

Sử dụng bộ lọc CSS, chúng ta có thể áp dụng một số hiệu ứng cho <video>, ví dụ:

<style>

video {

width: 307px;

height: 250px;

background: rgba(255,255,255,0.5);

border: 1px solid #ccc;

}

.grayscale {

[+filter: grayscale(1);](http://sass-lang.com/docs/yardoc/file.SASS_REFERENCE.html#including_a_mixin)

}

.sepia {

[+filter: sepia(1);](http://sass-lang.com/docs/yardoc/file.SASS_REFERENCE.html#including_a_mixin)

}

.blur {

[+filter: blur(3px);](http://sass-lang.com/docs/yardoc/file.SASS_REFERENCE.html#including_a_mixin)

}

...

</style>

<video autoplay></video>

<script>

var idx = 0;

var filters = ['grayscale', 'sepia', 'blur', 'brightness', 'contrast', 'hue-rotate',

'hue-rotate2', 'hue-rotate3', 'saturate', 'invert', ''];

function changeFilter(e) {

var el = e.target;

el.className = '';

var effect = filters[idx++ % filters.length]; *// loop through filters.*

if (effect) {

el.classList.add(effect);

}

}

document.querySelector('video').addEventListener('click', changeFilter, false);

</script>

* WebGL Textures

Chúng ta có thể chụp hình ảnh video và render nó vào texture WebGL bằng cách sử dụng getUserMedia() và Three.js.

Hướng dẫn: <http://learningthreejs.com/blog/2012/02/07/live-video-in-webgl/>

Demo: <http://learningthreejs.com/data/live-video-in-webgl/>

1. Thu âm

Sử dụng getUserMedia() + Web Audio API chúng ta hoàn toàn có thể thu lại âm thanh ngay trên trình duyệt.

Với phiên bản Chrome 24, chúng ta có thể kích hoạt cờ "Web Audio Input" trong tab about:flags để thử nghiệm với getUserMedia() + Web Audio API. Tuy nó vẫn trong giai đoạn phát triển, nhưng hiện đang hoạt động khá tốt.

var context = new window.webkitAudioContext();

navigator.webkitGetUserMedia({audio: true}, function(stream) {

var microphone = context.createMediaStreamSource(stream);

var filter = context.createBiquadFilter()

*// microphone -> filter -> destination.*

microphone.connect(filter);

filter.connect(context.destination);

}, onFailSoHard);

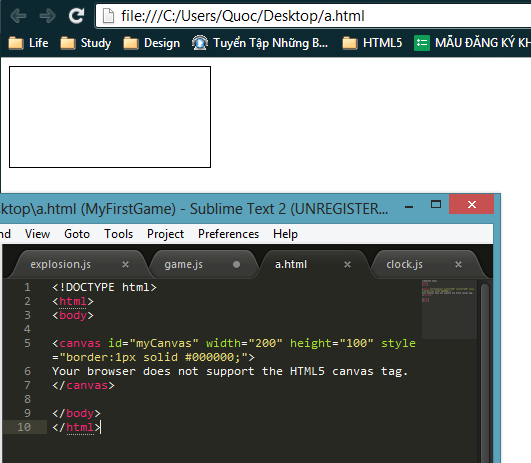
1. **Canvas**

Một trong những thành phần quan trọng nhất tạo nên sự khác biệt của HTML5. Canvas cung cấp một vùng trên giao diện dựa trên bitmap ( khác với Vector – SVG) có thể được dùng để vẽ đồ thị, hiển thị đồ họa game hoặc các hình ảnh trực quan khác thông qua *rendering context*(2D hoặc 3D với WebGL – WebGL không thuộc chuẩn HTML5). Canvas loại bỏ ràng buộc cần phải cài đặt Plugin của Flash

Các trình duyệt hổ trợ:

* Chrome
* Firefox
* Safari
* Opera
* Internet Explorer 9+

Khai báo 1 canvas và kết quả:

Các thuộc tính:

* + width – chiều rộng của canvas
  + height – chiều cao của canvas

typedef ([CanvasRenderingContext2D](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#canvasrenderingcontext2d) or [WebGLRenderingContext](http://www.whatwg.org/specs/web-apps/current-work/multipage/infrastructure.html#webglrenderingcontext)) **RenderingContext**;

interface **HTMLCanvasElement** : [HTMLElement](http://www.whatwg.org/specs/web-apps/current-work/multipage/elements.html#htmlelement) {

attribute unsigned long [width](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-width);

attribute unsigned long [height](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-height);

[RenderingContext](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#renderingcontext)? [getContext](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-getcontext)(DOMString contextId, any... arguments);

boolean [supportsContext](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-supportscontext)(DOMString contextId, any... arguments);

void [setContext](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-setcontext)([RenderingContext](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#renderingcontext) context);

[CanvasProxy](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#canvasproxy) [transferControlToProxy](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-transfercontroltoproxy)();

DOMString [toDataURL](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-todataurl)(optional DOMString type, any... arguments);

DOMString [toDataURLHD](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-todataurlhd)(optional DOMString type, any... arguments);

void [toBlob](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-toblob)([FileCallback](http://www.whatwg.org/specs/web-apps/current-work/multipage/infrastructure.html#filecallback)? \_callback, optional DOMString type, any... arguments);

void [toBlobHD](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-toblobhd)([FileCallback](http://www.whatwg.org/specs/web-apps/current-work/multipage/infrastructure.html#filecallback)? \_callback, optional DOMString type, any... arguments);

};

Kích thước của canvas có thể được thay đổi theo style đã được định nghĩa trong CSS.

Kích thước của canvas phải là số nguyên dương. Kích thước mặc định là width = 300 và height = 150.

Để thực hiện vẽ đồ họa trên Canvas ta cần phải phải có CanvasRenderingContext2D.

**context = canvas .**[**getContext**](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-getcontext)**(contextId [, ... ])**

Trả về một đối tượng rendering context API để vẽ trên canvas.

Tham số đầu tiên xác định API muốn dùng để vẽ: “2d” (đồ họa hình ảnh 2 chiều) hoặc “webgl” (đồ họa 3 chiều với thư viện webGL).

Trả về null nếu contextId không được hổ trợ hoặc canvas đã được khởi tạo trước với kiểu context khác với contextId.

Tạo ra lỗi InvalidStateError nếu phương thức setContext() hoặc [transferControlToProxy()](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-transfercontroltoproxy) đã được sử dụng trước đó.

* **supported*=*canvas*.***[**supportsContext**](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-supportscontext)**(contextId [, ... ])**

Trả về false nếu khi gọi phương thức [getContext()](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-getcontext) với cùng tham số trả về null, trả về true cho các trường hợp còn lại.

Giá trị trả về của phương thức này không đảm bảo phương thức getContext() sẽ return hoặc không return về đối tượng tương ứng, điều kiện (khả năng của hệ thống) có thể thay đổi theo thời gian.

Tạo ra lỗi InvalidStateError nếu phương thức setContext() hoặc [transferControlToProxy()](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-transfercontroltoproxy) đã được sử dụng trước đó.

* **canvas*.***[**setContext**](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-setcontext)**(context)**

Set đối tượng context tham số đầu vào cho canvas

Tạo ra lỗi InvalidStateError nếu phương thức setContext() hoặc [transferControlToProxy()](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-transfercontroltoproxy) đã được sử dụng trước đó.

* 3 hàm trên loại trừ lẫn nhau khi được gọi. Nếu gọi đã 1 hàm thì khi gọi bất kì hàm nào trong 3 hàm này đều bị tạo ra lỗi InvalidStateError.
* Có 2 cách để có được rendering context cho canvas: setContext() có sẳn cho canvas hoặc getContext() từ đối tượng canvas.
* **url*=*canvas.** **toDataURL ( [type, ... ] )**
* **url*=*canvas .**[**toDataURLHD**](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-todataurlhd)**( [type,... ] )**

Trả về “URL” trỏ đến dữ liệu hình ảnh trên canvas.

Tham số đầu tiên (nếu có), sẽ quyết định kiểu hình ảnh được trả về (PNG hay JPG). Mặc định sẽ là PNG. Các tham số khác sẽ kiểm soát các chi tiết và cách hình ảnh được tạo ra.

Phương thức [toDataURL()](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-todataurl) trả về hình ảnh với mật độ điểm ảnh 96dpi.  
Phương thức [toDataURLHD()](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-todataurlhd) trả hình ảnh với mật độ điểm ảnh giống với mật độ trong canvas.

* **canvas .**[**toBlob**](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-toblob)**(callback [, type, ... ])**
* **canvas .**[**toBlobHD**](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-toblobhd)**(callback [, type, ... ])**

Tạo ra một đối tượng Blob (File API) đại diện cho tập chứa hình ảnh trong canvas, và gọi một hàm callback xữ lí đối tượng tạo ra.

Các tham số sau đó giống với tham số của các phương thức toURL(), toURLHD()

Phương thức [[toBlob](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-toblob) ()](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-todataurl) trả về hình ảnh với mật độ điểm ảnh 96dpi.  
Phương thức [[toBlob](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-toblob)HD ()](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-canvas-todataurlhd) trả hình ảnh với mật độ điểm ảnh giống với mật độ trong canvas.

**RENDERING CONTEXT 2D**

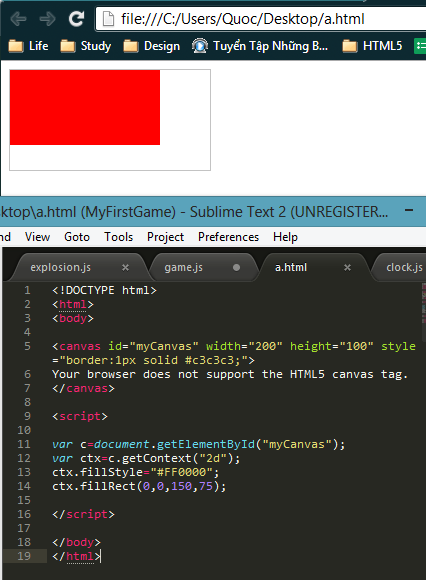
[CanvasRenderingContext2D](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#canvasrenderingcontext2d) là đối tượng dùng để vẽ trên Canvas được API cung cấp. Tất cả thao tác vẽ đồ họa trên Canvas đều thông qua rendering context.

Có 2 cách để có được rendering context 2d của canvas:

* gọi phương thức canvas.getContext(“2d”) trả về đối tượng context
* khởi tạo một unbound context (chưa được gắn với canvas) bằng phương thức khởi tạo của context [CanvasRenderingContext2D([width, height])](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#dom-context-2d) với width, height là kích thước sẽ sử dụng để vẽ ( với cách này thì phải gán context vào canvas bằng phương thức canvas.setContext(context).

Ta có thể lấy đối tượng canvas, kích thước từ đối tượng context nếu context đã được gắn với canvas: context.canvas, context.width, context.height.

Hệ tọa độ trong canvas được xác định từ trên xuống và trái qua phải với góc trái trên của canvas có tọa độ (0,0).

Vẽ một hình chữ nhật đơn giản:  


## Các thuộc tính và phương thức đồ họa của context 2D:

## Màu, Style, and Bóng

|  |  |
| --- | --- |
| **Property** | **Description** |
| fillStyle | Sets hoặc returns màu, gradient hoặc mẫu dùng để tô màu cho hình vẽ.  Context.fillStyle = “red” ;  Context.fillStyle = gradient; (gradient được tạo trước)    Context.fillStyle = pattern; (pattern được tạo trước lấy từ một tag img và lặp lại theo 2 hướng X,Y) |
| strokeStyle | Sets hoặc returns màu, gradient hoặc mẫu dùng để tô màu cho các đường.  Tương tự như fillStyle: ví dụ pattern vẽ vòng tròn |
| shadowColor | Sets hoặc returns màu dùng để vẽ bóng.  Ví dụ vẽ bóng với hình tròn được có bóng và độ mờ 10: |
| shadowBlur | Sets hoặc returns mức độ làm mờ cho bóng. |
| shadowOffsetX | Sets hoặc returns độ lệch theo phương nằm ngang của bóng. |
| shadowOffsetY | Sets hoặc returns độ lệch theo phương thẳng đứng của bóng.  Ví dụ: hình tròn với độ lệch 3 theo cả 2 phương X, Y và độ mờ bóng là 10, bóng màu đỏ. |

|  |  |
| --- | --- |
| **Method** | **Description** |
| createLinearGradient() | Tạo Linear Gradient để dùng trên canvas. *context*.createLinearGradient(*x0,y0,x1,y1*);  x0,y0: tọa độ điểm bắt đầu gradient  x1,y1: tọa độ điểm kết thúc gradient |
| createPattern() | Tạo một mẫu để có thể lập lại trên hình vẽ theo một hướng nhất định  *context*.createPattern(*image*,"repeat|repeat-x|repeat-y|no-repeat");  *image*: hình ảnh sử dụng làm pattern  kiểu lập lại: 2 chiều - “repeat”, chiều ngang: “repeat-x”, chiều dọc - “repeat-y”, không lặp: “no-repeat”. |
| createRadialGradient() | Tạo Radial Gradient để dùng trên canvas *context*.createRadialGradient(*x0,y0,r0,x1,y1,r1*);  x0,y0: tọa độ tâm của vòng tròn bắt đầu radient.  r0: bán kính vòng tròn bắt đầu radient.  x1,y1: tọa độ tâm của vòng tròn kết thúc radient.  r1: bán kính vòng tròn kết thúc radient.  Gradient với tâm của các hình tròn bắt đầu và kết thúc trùng nhau |
| addColorStop() | Thêm một màu vào gradient. *gradient*.addColorStop(*stop*,*color*);  stop: giá trị giữa 0.0 đến 1.0 xác định vị trí bắt đầu và kết thúc của radient thêm vào.  color: giá trị màu theo CSS. |

## Style cho đường thẳng

|  |  |
| --- | --- |
| **Property** | **Description** |
| lineCap | Sets hoặc returns kiểu kết thúc của 1 đường ( butt - tiết diện thẳng , round – tròn, square – hình vuông) |
| lineJoin | Sets hoặc returns kiểu kết nối khi 2 đường thẳng nối nha (bevel – góc bị vát, round – tròn, miter – góc nhọn) |
| lineWidth | Sets hoặc returns độ rộng của đường |
| miterLimit | Sets or returns độ dài của phần nhọn khi nối nhau kiểu miter |

## Hình chữ nhật

|  |  |
| --- | --- |
| **Method** | **Description** |
| rect() | Tạo một hình chữ nhật nằm trong path hiện tại.  *context*.rect(*x,y,width,height*);  x,y: tọa độ điểm trái trên của hình chữ nhật,  width,height: chiều dài, chiều rộng hình chữ nhật.  Sẽ không có đường nối giữa điểm đầu của hình chữ nhật với điểm cuối cùng trong path trước đó nhưng sẽ có đường nối với điểm được thêm vào (lineTo) sau hình chữ nhật.  Ví dụ: |
| fillRect() | Vẽ một hình chữ nhật được tô bằng fillStyle.  Tham số tương tự rect(). |
| strokeRect() | Vẽ một hình chữ nhật chỉ có cạnh theo strokeStyle mà không tô.  Tham số tương tự rect(). |
| clearRect() | Xóa tất cả pixel trong một hình chữ nhật xác định.  Tham số tương tự rect(). |

## Các phương thức về Paths – Hình phức tạp gồm nhiều điểm nối lại

|  |  |
| --- | --- |
| **Method** | **Description** |
| fill() | Tô màu path đang vẽ, nếu Path không kín thì sẽ có đường thẳng ảo nối từ điểm đầu đến điểm cuối path được thêm vào. |
| stroke() | Vẽ path đã được tạo.  Ví dụ: stoke màu đen path gồm ½ hình tròn và 2 điểm phía trên ½ hình tròn. Ta có hình sẽ không kín nhưng vẫn được ”tô màu”. |
| beginPath() | Bắt đầu một path mới hoặc reset path hiện tại. |
| moveTo() | Chuyển path đến một tọa độ mà không có đường nối giữa điểm ở tọa độ cũ và điểm ở tọa độ mới.  Tham số: x,y : tọa độ điểm kế tiếp. |
| closePath() | Tạo Path từ điểm hiện tại đến điểm đầu tiên của path. ( kết thúc để bắt đầu path khác). |
| lineTo() | Giống moveTo nhưng sẽ có đường nối giữa điểm mới và điểm trước đó.  Tham số: x,y : tọa độ điểm kế tiếp. |
| clip() | Xóa bỏ và giữ lại một phần hình ảnh trong canvas theo hình dạng path.  Ví dụ: |
| quadraticCurveTo() | *context*.quadraticCurveTo(*cpx,cpy,x,y*);  Thêm một đường cong Bézier bậc 2 vào path với điểm control point (cpx,cpy) và điểm kết thúc (x,y).  Ví dụ: |
| bezierCurveTo() | *context*.bezierCurveTo(*cp1x,cp1y,cp2x,cp2y,x,y*);  Thêm một đường cong Bézier bậc 3 vào path với 2 điểm control point (cpx1,cpy1), (cpx2,cpy2) và điểm kết thúc (x,y).  Ví dụ: |
| arc() | Thêm một hình tròn hoặc cung tròn vào path. *context*.arc(*x,y,r,sAngle,eAngle,counterclockwise*);  x,y: tọa độ tâm hình tròn.  r: bán kính.  sAngle: Góc bắt đầu (radian). 0 sẽ là góc 3 giờ.  eAngle: Góc kết thúc.  counterclockwise: cách tính góc theo chiều kim đồng hồ hoặc ngược lại. Mặc định nếu không có tham số thì tính theo chiều kim đồng hồ.  Một số ví dụ ở trên đã sử dụng arc(). |
| arcTo() | Thêm một cung tròn nối 2 điểm.  *context*.arcTo(*x1,y1,x2,y2,r*);  x1,y1: điểm bắt đầu.  x2,y2: điểm kết thúc.  r: bán kính.  Ví dụ: |
| isPointInPath() | *context*.isPointInPath(*x,y*); Kiểm tra điểm (x,y) có nằm trong path hiện tại không. |

## Các phép biến đổi

|  |  |
| --- | --- |
| **Method** | **Description** |
| scale() | *context*.scale(scalewidth,scaleheight);  Thay đổi hình vẽ theo tỉ lệ chiều rộng và chiều ngang: 1.0 = 100%, 2.0 = 200%, 0.5 = 50%.  Những hàm vẽ gọi sau khi gọi phương thức này sẽ có tác dụng |
| rotate() | *context*.rotate(angle);  Xoay hình đang vẽ một góc angle (radian).  Ví dụ dưới đây được chỉnh sửa từ một ví dụ đã sử dụng ở trên. |
| translate() | *context*.translate(x,y);  Chuyển gốc tọa độ (0,0) của canvas về tọa độ (x,y) |
| transform() | *context*.transform(a,b,c,d,e,f);  Thay phép biến đổi ma trận hiện tại cho hình ảnh bằng ma trận (nhân với ma trận hiện tại): a c e  b d f  0 0 1  Trong đó: a, d : tỉ lệ thu phóng hình ảnh theo ngang, dọc.  b, c : độ làm xiên hình theo chiều ngang, dọc.  e, f : di chuyển hình theo ngang, dọc.  Ví dụ: Hình màu đỏ là sau khi áp dụng transform. Kéo xiên theo phương ngang với tỉ lệ 0.5.  Dịch chuyển đến tọa độ (30, 10). |
| setTransform() | Chuyển ma trận hiện tại thành ma trận đơn vị I. Sau đó tranform(). |

## Text

|  |  |
| --- | --- |
| **Property** | **Description** |
| font | Sets hoặc returns thông số font cho việc vẽ text. |
| textAlign | Sets hoặc returns cách canh lề cho text. |
| textBaseline | Sets hoặc returns vị trí. Giá trị có thể nhận: alphabetic|top|hanging|middle|ideographic|bottom |

|  |  |
| --- | --- |
| **Method** | **Description** |
| fillText() | Vẽ chuỗi được tô màu. |
| strokeText() | Vẽ chuỗi nhưng không được tô. |
| measureText() | Trả về đối tượng chứa độ rộng theo pixel của một chuỗi text.  *context*.measureText(*text*).width;  text: chuỗi Ví dụ: Chuỗi “width …” là chuỗi được vẽ bằng fillText.  “Hello World” được vẽ bằng strokeText. |

## Image Drawing

|  |  |
| --- | --- |
| **Method** | **Description** |
| drawImage() | Vẽ một hình ảnh, một canvas khác, video lên canvas. *context*.drawImage(*img,x,y*);  Vẽ với kích thước dài rộng:  *context*.drawImage(*img,x,y,width,height*);  Vẽ một phần được cắt từ hình ảnh nguồn:  *context*.drawImage(*img,sx,sy,swidth,sheight,x,y,width,height*);  img: Chứa hình ảnh nguồn từ hình ảnh, canvas, video  sx,sy tọa độ bắt đầu cắt hình từ nguồn  swidth, sheight: chiều dài, cao phần sẽ cắt từ nguồn  x,y: Tọa độ hình ảnh sẽ được vẽ trên canvas  width, height: chiều rộng , chiều cao hình ảnh sẽ vẽ |

## Pixel Manipulation

|  |  |
| --- | --- |
| **Property** | **Description** |
| width | *imgData*.width: Chiều rộng của một đối tượng ImageData |
| height | *imgData*.height: Chiều cao của một đối tượng ImageData |
| data | *imgData*.data: Trả về mảng chứa dữ liệu của đổi tượng ImageData |

|  |  |
| --- | --- |
| **Method** | **Description** |
| createImageData() | Tạo một đối tượng ImageData rỗng với chiều rộng và chiều cao, hoặc từ kích thước của một ImageData khác  *context*.createImageData(*width,height*);  *context*.createImageData(*imgData*); |
| getImageData() | Trả về đối tượng ImageData là bản sao các pixel theo 1 hình chữ nhật từ canvas  *context*.getImageData(*x,y,width,height*);  x,y: tọa độ điểm trái trên của hình chữ nhật,  width,height: chiều cao, chiều rộng hình chữ nhật. |
| putImageData() | Lấy dữ liệu từ một ImageData vẽ lại trên canvas. *context*.putImageData(*imgData,x,y,dirtyX,dirtyY,dirtyWidth,dirtyHeight*);  imgData: dữ liệu ImageData nguồn. x,y: tọa độ sẽ đưa dữ liệu từ ImageData vào dirtyX, dirtyY: tọa độ phần dữ liệu sẽ lấy từ ImageData dirtyWidth, dirtyHeight: Chiều rộng, cao phần dữ liệu được vẽ lên canvas |

## Compositing

|  |  |
| --- | --- |
| **Property** | **Description** |
| globalAlpha | Sets hoặc returns độ mờ của hình vẽ.  *context*.globalAlpha=*number*; - number [0.0 ; 1.0] |
| globalCompositeOperation | Sets hoặc returns cách hình ảnh mới được vẽ trên canvas.  *context*.globalCompositeOperation=value; |

Một số cách cải thiện hiệu suất của Canvas

* + Thao tác render trên một canvas đệm không được hiển thị lên màn hình

no pre-rendering:

*// canvas, context are defined*

function render() {

drawMario(context);

requestAnimationFrame(render);

}

pre-rendering:

var m\_canvas = document.createElement('canvas');

m\_canvas.width = 64;

m\_canvas.height = 64;

var m\_context = m\_canvas.getContext(‘2d’);

drawMario(m\_context);

function render() {

context.drawImage(m\_canvas, 0, 0);

requestAnimationFrame(render);

}

* + Gộp chung một số thao tác

Thao tác vẽ rất tốn kém. Vì vậy cần giảm đến mức tối đa.

Chẳng hạn:

Thay vì vẽ nhiều line nối lại với nhau

for (var i = 0; i < points.length - 1; i++) {

var p1 = points[i];

var p2 = points[i+1];

context.beginPath();

context.moveTo(p1.x, p1.y);

context.lineTo(p2.x, p2.y);

context.stroke();

}

Có thể vẽ 1 line liên tục cho ra kết quả tương tự

context.beginPath();

for (var i = 0; i < points.length - 1; i++) {

var p1 = points[i];

var p2 = points[i+1];

context.moveTo(p1.x, p1.y);

context.lineTo(p2.x, p2.y);

}

context.stroke();

* + Tránh thay đổi trạng thái canvas

Thay vì thay đổi trong từng vòng lặp

for (var i = 0; i < STRIPES; i++) {

context.fillStyle = (i % 2 ? COLOR1 : COLOR2);

context.fillRect(i \* GAP, 0, GAP, 480);

}

Ta có thể dùng 2 vòng lặp chẳn lẻ

context.fillStyle = COLOR1;

for (var i = 0; i < STRIPES/2; i++) {

context.fillRect((i\*2) \* GAP, 0, GAP, 480);

}

context.fillStyle = COLOR2;

for (var i = 0; i < STRIPES/2; i++) {

context.fillRect((i\*2+1) \* GAP, 0, GAP, 480);

}

* + Chỉ vẽ lại những phần canvas thay đổi thay vì toàn bộ canvas

context.fillRect(0, 0, canvas.width, canvas.height);

Nên dùng:

context.fillRect(last.x, last.y, last.width, last.height);

* + Dùng nhiều canvas chồng lên nhau cho những hình ảnh phức tạp

<canvas id="bg" width="640" height="480" style="position: absolute; z-index: 0">

</canvas>

<canvas id="fg" width="640" height="480" style="position: absolute; z-index: 1">

</canvas>

* + Giảm dùng hàm làm mờ bóng
  + Tránh dùng tọa độ dấu chấm động
  + Biết nhiều cách để xóa trong Canvas (tùy theo trình duyệt mà hiệu suất khác nhau)

clearRect() hoặc canvas.width = canvas.width

* + Dùng hàm requestAnimationFrame để tối ưu hóa chuyển động.

1. **Geolocation**

Xác định ví trí của người dùng

HTML5 Geolocation API được dùng để xác định vị trí địa lí cuản gười dùng

Do chức năng này cần sự cho phép của người dùng nên nó sẽ không hoạt động khi người dùng không chấp thuận sử dụng.

Trình duyệt hổ trợ

Internet ExplorerFirefoxOperaGoogle ChromeSafari

**Lưu ý: Geolocation sẽ chính xác hơn trên thiết bị có GPS như một số Smart Phone…**

Cách sử dụng Geolocation

Sử dụng phương thức getCurrentPosition() để lấy thông tin vị trí người dùng.

Ví dụ bên dưới lấy thông tin vĩ độ và kinh độ của người dùng:

<script>  
var x=document.getElementById("demo");  
function getLocation()  
  {  
  if (navigator.geolocation)//Kiểm tra trình duyệt có hổ trợ không  
    {  
    navigator.geolocation.getCurrentPosition(showPosition);  
    }  
  else{x.innerHTML="Geolocation is not supported by this browser.";}  
  }  
function showPosition(position)  
  {  
  x.innerHTML="Latitude: " + position.coords.latitude +   
  "<br>Longitude: " + position.coords.longitude;   
  }  
</script>

Phương thức getCurrentPosition() có thể được thêm 1 hàm để handle error với tham số Error đầu vào có thể là các giá trị

error.PERMISSION\_DENIED //Người dùng từ chối sử dụng  
error.POSITION\_UNAVAILABLE //Vị trí không tìm được  
error.TIMEOUT //time out  
error.UNKNOWN\_ERROR //lỗi khác

Dữ liệu được phương thức getCurrentPosition() trả về

Phương thức getCurrentPosition()returns một đối tượng nếu chạy thành công. Latitude, longitude and accuracy sẽ luôn được trả về. Các thuộc tính khác sẽ được trả về nếu có.

|  |  |
| --- | --- |
| **Property** | **Description** |
| coords.latitude | Vĩ độ - số thập phân |
| coords.longitude | Kinh độ - số thập phân |
| coords.accuracy | Độ chính xác của vị trí được lấy |
| coords.altitude | Độ cao so với mực nước biển |
| coords.altitudeAccuracy | Độ chính xác của độ cao so với mực nước biển |
| coords.heading | Độ lệch hướng so với hướng Bắc theo chiều kim đồng hồ |
| coords.speed | Tốc độ di chuyển theo m/s |
| timestamp | Ngày giờ của kết quả trả về. |

Một số phương thức khác của Geolocation

watchPosition() – Trả về vị trí hiện tại của người dùng và liên tục cập nhật vị trí khi người dùng di chuyển

clearWatch() – ngừng việc cập nhật vị trí của watchPosition

Ví dụ: (cần thiết bị có GPS chính xác để kiểm tra)

<script>  
var x=document.getElementById("demo");  
function getLocation()  
  {  
  if (navigator.geolocation)  
    {  
    navigator.geolocation.watchPosition(showPosition);  
    }  
  else{x.innerHTML="Geolocation is not supported by this browser.";}  
  }  
function showPosition(position)  
  {  
  x.innerHTML="Latitude: " + position.coords.latitude +   
  "<br>Longitude: " + position.coords.longitude;   
  }  
</script>

1. **Server-Sent Event**

HTML5 Server-Sent Events cho phép trang web cập nhật thay đổi từ server.

Server-Sent Events – Tin nhắn một chiều

Server-sent event là khi trang web tự động nhận cập nhật từ server.

Việc này trước đây có thể làm được nhưng trang web cần phải hỏi server xem có cập nhật nào mới không. Với Server-Sent Events thì việc này được thực hiện tự động.

Trình duyệt hổ trợ

Internet ExplorerFirefoxOperaGoogle ChromeSafari

Phần lớn trình duyệt phổ biến trừ Internet Explorer

Kiểm tra Trình duyệt có hổ trợ không

Đoạn code kiểm tra xem trình duyệt có hổ trợ Server-Sent Event không:

if(typeof(EventSource)!=="undefined")  
  {  
  // Yes! Server-sent events support!  
  // *Some code.....*  
  }  
else  
  {  
  // Sorry! No server-sent events support..  
  }

Cách nhận thông báo Server-Sent Event

Đối tượng EventSource được dùng để nhận Server-Sent Event từ server:

Ví dụ:

var source=new EventSource("demo\_sse.php");  
source.onmessage=function(event)  
  {  
  document.getElementById("result").innerHTML+=event.data + "<br>";  
  };

Các Event của đối tượng EventSource

|  |  |
| --- | --- |
| **Events** | **Description** |
| onopen | Kết nối đến server được mở |
| onmessage | Nhận được message từ server |
| onerror | Lỗi xảy ra |

1. **Drag & Drop**

Trong những năm trước đây, chúng ta phải sử dụng những thư viện như JQuery và Dojo để đơn giản hóa các thành phần giao diện người dùng phức tạp như hình ảnh động, góc tròn, và kéo và thả. Giờ đây, kéo và thả đã là một phần tử chính thức trong HTML5! Nó bao gồm các sự kiện, JavaScript API, và đánh dấu bổ sung để cho phép bất kì thành phần nào trên trang web cũng có thể kéo thả được. Đặc biệt nó còn nhanh hơn, đáp ứng tốt hơn cho các ứng dụng web.

1. Kiểm tra hỗ trợ

Giống như những phần trước đã giới thiệu, khi sử dụng một API nào đó trước tiên ta cần kiểm tra xem trình duyệt hiện tại có hỗ trợ không. Thư viện tốt nhất để kiểm tra đó là “Modernizr”:

if (Modernizr.draganddrop) {

*// Browser supports HTML5 DnD.*

} else {

*// Fallback to a library solution.*

}

1. Tạo đối tượng kéo thả

Việc kéo đối tượng chỉ đơn giản là thiết lập thuộc tính draggable=true đối với các thành phần muốn làm cho nó có thể di chuyển. Bất cứ thành phần nào, miễn là nó có thể kéo đi bao gồm hình ảnh, liên kết, các tập tin, hoặc các nút DOM (Document Object Model - Mô hình Đối tượng Tài liệu).

Ví dụ:

<div id="columns">

<div class="column" draggable="true"><header>A</header></div>

<div class="column" draggable="true"><header>B</header></div>

<div class="column" draggable="true"><header>C</header></div>

</div>

1. Lắng nghe sự kiện kéo thả

Sau đây là các sự kiện trong quá trình kéo thả:

* *dragstart*
* *drag*
* *dragenter*
* *dragleave*
* *dragover*
* *drop*
* *dragend*

Đối với việc kéo thả, ta cần hiểu rõ 2 khái niệm, yếu tố nguồn – cái mà người dùng nắm kéo, ví dụ như hình ảnh, danh sách, đối tượng tập tin, liên kết,… Yếu tố đích đến – nơi mà người dùng thả đối tượng được kéo, vì vậy, không phải phần tử nào cũng có thể là đích đến.

* Dragstart

Một khi đã gắn thuộc tính draggable="true" thì kèm theo đó là xử lý ở sự kiện dragstart.

Sau đây là ví dụ làm đục đối tượng 40% khi người dùng bắt đầu kéo đối tượng:

function handleDragStart(e) {

this.style.opacity = '0.4'; *// this / e.target is the source node.*

}

var cols = document.querySelectorAll('#columns .column');

[].forEach.call(cols, function(col) {

col.addEventListener('dragstart', handleDragStart, false);

});

* Dragenter, dragover, dragleave, dragend

Sự kiện dragenter, dragover, dragleave có thể được sử dụng để xử lý đối tượng trong quá trình kéo. Ví dụ, khi một đối tượng được kéo ngang qua một đối tượng khác, hoặc khi kéo rời khỏi một đối tượng khác,…

<style>

.column.over {

border: 2px dashed #000;

}

</style>

function handleDragStart(e) {

this.style.opacity = '0.4'; *// this / e.target is the source node.*

}

function handleDragOver(e) {

if (e.preventDefault) {

e.preventDefault(); *// Necessary. Allows us to drop.*

}

e.dataTransfer.dropEffect = 'move'; *// See the section on the DataTransfer object.*

return false;

}

function handleDragEnter(e) {

*// this / e.target is the current hover target.*

this.classList.add('over');

}

function handleDragLeave(e) {

this.classList.remove('over'); *// this / e.target is previous target element.*

}

var cols = document.querySelectorAll('#columns .column');

[].forEach.call(cols, function(col) {

col.addEventListener('dragstart', handleDragStart, false);

col.addEventListener('dragenter', handleDragEnter, false);

col.addEventListener('dragover', handleDragOver, false);

col.addEventListener('dragleave', handleDragLeave, false);

});

1. Kéo tập tin

* Kéo từ máy tính vào trình duyệt

Kéo một file từ máy tính vào trình duyệt bằng cách sử dụng các sự kiện kéo thả loại khác. Sự khác biệt chính là trong việc xử lý xử lý thả. Thay vì sử dụng dataTransfer.getData() để truy cập các tập tin, dữ liệu sẽ được chứa trong thuộc tính dataTransfer.files:

function handleDrop(e) {

e.stopPropagation(); *// Stops some browsers from redirecting.*

e.preventDefault();

var files = e.dataTransfer.files;

for (var i = 0, f; f = files[i]; i++) {

*// Read the File objects in this FileList.*

}

}

* Kéo từ trình duyệt ra máy tính

Hướng dẫn: <http://www.thecssninja.com/javascript/gmail-dragout>

1. **Storage**

HTML5 Web Storage là gì?

Với HTML5, trang web có thể được lưu trữ dữ liệu nội bộ trong trình duyệt của người dùng.

Trước đây, việc lưu trữ được thực hiện bởi cookies.Tuy nhiên, Web Storage giúp việc lưu trữ bảo mật hơn, nhiều hơn (5Mb hoặc nhiều hơn so với 4Kb của Cookie) và nhanh hơn. Dữ liệu không nằm trong các server request nhưng sẽ được sử dụng khi cần. Ngoài ra nó có thể lưu số lượng lớn dữ liệu mà không ảnh hưởng đến hiệu suất của website.

Dữ liệu được lưu theo cặp key/value, và một trang web chỉ được truy xuất đến dữ liệu của chính nó. Dữ liệu trong localStorage sẽ không bị expire như cookie.

## Trình duyệt hỗ trợ

Internet ExplorerFirefoxOperaGoogle ChromeSafari

Lưu ý: Internet Explorer chỉ hổ trợ Web Storage từ phiên bản 8 trở lên.

## localStorage và sessionStorage

Có 2 loại đối tượng để lưu trữ dữ liệu ở máy người dùng:

* localStorage – Dữ liệu sẽ được lưu vô thời hạn.
* sessionStorage – Dữ liệu được lưu trong một session

Trước khi sử dụng localStorage và sessionStorage ta phải kiểm tra trình duyệt có hổ trợ không:

if(typeof(Storage)!=="undefined")  
  {  
  // Yes! localStorage and sessionStorage support!  
  // *Some code.....*  
  }  
else  
  {  
  // Sorry! No web storage support..  
  }

## localStorage

Dữ liệu được lưu với localStore sẽ không bị hết hạn. Dữ liệu sẽ không bị xóa khi trình duyệt bị tắt, và sẽ luôn sẳn sàng để sử dụng ở những lần kế tiếp.

Cú pháp: localStorage.<key> = value Với key là khóa sẽ dùng để truy xuất hoặc chỉnh sửa. value là giá trị được lưu.

## Giá trị value sẽ được lưu dưới dạng String. Vì vậy khi trước khi lưu hoặc sau lấy ra cần phải chuyển kiểu dữ liệu. (dùng JSON.stringtify() và JSON.parse() cho đối tượng phức tạp)

## Ví dụ: đếm số lần click chuột của người dùng.

if (localStorage.clickcount)  
  {  
  localStorage.clickcount=Number(localStorage.clickcount)+1;  
  }  
else  
  {  
  localStorage.clickcount=1;  
  }  
document.getElementById("result").innerHTML="You have clicked the button " + localStorage.clickcount + " time(s).";

## sessionStorage

sessionStorage cũng giống với localStorage ngoại trừ việc nó được lưu chỉ trong một session. Dữ liệu trong sessionStorage sẽ bị xóa khi người dùng tắt trình duyệt.

1. **Web Workers & Web Sockets**
2. Web Workers

Với công nghệ phần cứng hiện nay, việc sử dụng đa luồng đã trở nên một phần không thể thiếu trong các phần mềm. Tuy nhiên, công nghệ thiết kế web vẫn chưa tận dụng được sức mạnh này. Với các công việc đòi hỏi một quá trình xử lý lâu, lập trình viên thường phải sử dụng những thủ thuật như setTimeout(), setInterval(),… để thực hiện từng phần công việc. Hiện nay, để giải quyết vấn đề này, một API mới đã xuất hiện với tên gọi Web Worker.

Đối tượng Web Worker được tạo ra sẽ thực thi trong một thread độc lập và chạy ở chế độ nền nên không ảnh hưởng đến giao diện tương tác của trang web với người dùng. Với đặc điểm này, chúng ta có thể sử dụng Web Worker trong các công việc đòi hỏi thời gian xử lý lâu nạp dữ liệu, tạo cache,…

Điểm hạn chế của Web Worker là không thể truy xuất được thành phần trên DOM, và cả các đối tượng **window**, **document** hay **parent**. Mã lệnh các công việc cần thực thi cũng phải được cách ly trong một tập tin script.

* Kiểm tra hỗ trợ

Trước khi tạo một web worker, kiểm tra xem trình duyệt của người dùng có hỗ trợ:

if(typeof(Worker)!=="undefined")

{

// Yes! Web worker support!

// Some code.....

}

else

{

// Sorry! No Web Worker support..

}

* Tạo một tệp Web Worker

Ví du, tạo một tập tin “demo\_workers.js”. Trong đó là đoạn mã để đếm tăng dần theo thời gian:

var i=0;

function timedCount()

{

i=i+1;

postMessage(i);

setTimeout("timedCount()",500);

}

timedCount();

Tất nhiên chúng ta không sử dụng web workers để làm chuyện đơn giản như vậy, thường nó được sử dụng cho các tác vụ chuyên sâu hơn.

* Tạo một đối tượng Web Worker

Khi đã tập tin web worker đã sẵn sàng, bây giờ ta gọi nó hoạt động từ trang web. Tạo một đối tượng web workers, kiểm tra xem nó tồn tại hay chưa, nếu chưa thì tạo ra và chạy đoạn mã trong tập tin web workers:

if(typeof(w)=="undefined")

{

w=new Worker("demo\_workers.js");

}

Khi tạo xong, bắt đầu nó bằng cách gọi phương thức postMessage():

worker.postMessage(); // Start the worker.

Sau đó chúng ta có thể nhận và gởi tin nhắn đến web workers, thêm sự kiện “onmessage” để lắng nghe web workers:

w.onmessage=function(event){

document.getElementById("result").innerHTML=event.data;

};

* Hủy Web Worker

Khi một đối tượng web workers được tạo ra, nó sẽ lắng nghe các tin nhắn cho tới khi nó được hủy. Để hủy một web workers, giải phóng tài nguyên, ta dùng phương thức terminate():

w.terminate();

1. Web Sockets

WebSocket là một API xây dựng kết nối "socket" giữa trình duyệt và máy chủ. Nói một cách đơn giản: có một kết nối liên tục giữa máy client và máy chủ vì vậy cả hai bên có thể bắt đầu gửi dữ liệu bất cứ lúc nào.

Sử dụng WebSocket có thể tạo ra độ trễ thấp hơn, kết nối thì gần như là kết nối thời gian thực. Nhờ vào các đặt điểm nổi bật đó mà WebSocket thường được sử dụng để:

* **Tạo các trò chơi trực tuyến nhiều người.**
* Tạo ứng dụng chat qua mạng.
* Đặt vé trực tuyến.
* Cập nhật tin tức mạng xã hội.
* Tạo kết nối

Để mở một kết nối websocket, chỉ đơn giản bằng cách gọi:

var connection = new WebSocket('ws://html5rocks.websocket.org/echo', ['soap', 'xmpp']);

* Thêm xử lý sự kiện

Sau khi kết nối vừa tạo xong, tiếp theo, thêm các xử lý sự kiện, các sự kiện như kết nối được mở, nhận tin nhắn đến, xảy ra lỗi, …

*// When the connection is open, send some data to the server*

connection.onopen = function () {

connection.send('Ping'); *// Send the message 'Ping' to the server*

};

*// Log errors*

connection.onerror = function (error) {

console.log('WebSocket Error ' + error);

};

*// Log messages from the server*

connection.onmessage = function (e) {

console.log('Server: ' + e.data);

};

* Giao tiếp với server

Ngay khi có một kết nối đến máy chủ, chúng ta có thể bắt đầu gửi dữ liệu đến máy chủ bằng cách sử dụng phương thức send(‘your message’) trên đối tượng kết nối. Nó hỗ trợ gởi các chuỗi duy nhất, nhưng trong phiên bản mới nhất, đã có thể gửi tin nhắn dạng nhị phân. Để gửi dữ liệu nhị phân, ta có thể sử dụng một trong hai đối tương Blob hoặc ArrayBuffer.

*// Sending String*

connection.send('your message');

*// Sending canvas ImageData as ArrayBuffer*

var img = canvas\_context.getImageData(0, 0, 400, 320);

var binary = new Uint8Array(img.data.length);

for (var i = 0; i < img.data.length; i++) {

binary[i] = img.data[i];

}

connection.send(binary.buffer);

*// Sending file as Blob*

var file = document.querySelector('input[type="file"]').files[0];

connection.send(file);

Tương tự như vậy, server cũng có thể gởi messages cho client bất cứ khi nào.

Để xác định các định dạng nhị phân nhận được, thiết lập thuộc tính binaryType của WebSocket là 'blob' hoặc 'arraybuffer'. Định dạng mặc định là 'blob':

*// Setting binaryType to accept received binary as either 'blob' or 'arraybuffer'*

connection.binaryType = 'arraybuffer';

connection.onmessage = function(e) {

console.log(e.data.byteLength); *// ArrayBuffer object if binary*

};

1. **Working Offline**

Một trong những tính năng mới không kém phần quan trọng của HTML5 đó là sử dụng bộ nhớ cache của trang web để làm việc ngoại tuyến. Nó có những lợi thế nhất định như sau:

* Duyệt web ngoại tuyến – người dùng có thể duyệt toàn bộ trang web hay sử dụng ứng dụng web ở chế độ ngoại tuyến.
* Các tài nguyên cache lưu trữ cục bộ nên giúp load nhanh hơn.
* Giảm tải server, duyệt chỉ tải tài nguyên về khi nó bị thay đổi.

Application Cache (hoặc AppCache) cho phép một lập trình viên xác định các tập tin mà trình duyệt sẽ đưa vào bộ nhớ cache vì vậy người dùng có thể sử dụng khi ngoại tuyến. Ứng dụng sẽ hoạt động chính xác, ngay cả khi người dùng nhấn nút refresh trong khi khi họ đang ngoại tuyến.

1. Tập tin Cache Mainfest

Tập tin cache manifest là một tập tin văn bản đơn giản liệt kê các nguồn tài nguyên trình duyệt sẽ lưu vào bộ nhớ đệm để truy xuất offline.

* Tham chiếu tới một tập tin mainfest

Để sử dụng được nó, trước hết ta cần include thuộc tính mainfest vào HTML:

<html manifest="example.appcache">

...

</html>

Thuộc tính manifest nên được đưa vào mỗi trang mà chúng ta muốn lưu trữ. Trình duyệt không lưu vào bộ nhớ đệm một trang nếu nó không chứa thuộc tính này (ngoại trừ nó được liệt kê rõ ràng trong tập tin manifest ). Điều này có nghĩa rằng bất kỳ trang nào có thuộc tính mainfest sẽ được mặc nhiên thêm vào bộ nhớ cache của ứng dụng.., không cần phải liệt kê tất cả các trang trong tập tin mainfest.

Thuộc tính manifest cũng có thể trỏ đến một URL tuyệt đối hoặc đường dẫn tương đối, nhưng một URL tuyệt đối phải dưới cùng gốc như trang hiện tại. Một tập tin manifest có thể có bất kỳ phần mở rộng nào, chỉ cần nội dung của nó đúng định dạng.

<html manifest="http://www.example.com/example.mf">

...

</html>

Một tập tin mainfest cần được cung cấp kèm theo mime-type text/cache-manifest, thêm nó và tập tin tùy chỉnh tới server hoặc tập tin cấu hình .htaccess:

AddType text/cache-manifest .appcache

Hoặc:

- url: /mystaticdir/(.\*\.appcache)

static\_files: mystaticdir/\1

mime\_type: text/cache-manifest

upload: mystaticdir/(.\*\.appcache)

* Cấu trúc của một tập tin mainfest

Một tập tin mainfest đơn giản thì có dạng như thế này:

CACHE MANIFEST

index.html

stylesheet.css

images/logo.png

scripts/main.js

Trình duyệt sẽ sử dụng tài nguyên cache tới khi nó được thay đổi, vì vậy, một khi các nguồn tài nguyên đó thay đổi, ta cần tạo một đánh dấu thay đổi trong tập tin cache để trình duyệt nhận biết và cập nhật lại tài nguyên đó. Việc cập nhật cache sẽ được nói đến trong phần sau.

Một tập tin mainfest có 3 phần riêng biệt: CACHE, NETWORK và FALLBACK.

CACHE: tất cả những tập tin được liệt kê trong phần này, sẽ được lưu vào bộ nhớ cache sau khi chúng được tải xuống lần đầu tiên.

NETWORK: ngược lại với CACHE, các tài nguyên liệt kê trong phần này phải được kết nối mạng, nó sẽ ko lưu vào bộ nhớ cache ngay cả khi người dùng ngoại tuyến.

FALLBACK: trang dự phòng trong trường hợp một tài nguyên không thể truy cập. URI đầu tiên là tài nguyên chính, URI thứ hai là cái dự phòng. Cả hai đều phải là địa chỉ tương đối và nằm cùng gốc với tập tin mainfest.

CACHE MANIFEST

*# 2010-06-18:v2*

*# Explicitly cached 'master entries'.*

CACHE:

/favicon.ico

index.html

stylesheet.css

images/logo.png

scripts/main.js

*# Resources that require the user to be online.*

NETWORK:

login.php

/myapi

http:*//api.twitter.com*

*# static.html will be served if main.py is inaccessible*

*# offline.jpg will be served in place of all images in images/large/*

*# offline.html will be served in place of all other .html files*

FALLBACK:

/main.py /static.html

images/large/ images/offline.jpg

\*.html /offline.html

1. Cập nhật Cache

Các ứng dụng web sẽ sử dụng bộ nhớ cache tới khi có một trong các điều sau đây xảy ra:

* Người dùng xóa dữ liệu trình duyệt của trang web.
* Tập tin mainfest bị thay đổi.
* Bộ nhớ ứng dụng cache được cập nhật
* Trang thái Cache

Đối tượng window.applicationCache truy cập tới bộ nhớ cache của trình duyệt, vì vậy ta sử dụng trạng thái trả về của nó để kiểm tra tình trạng của cache:

var appCache = window.applicationCache;

switch (appCache.status) {

case appCache.UNCACHED: *// UNCACHED == 0*

return 'UNCACHED';

break;

case appCache.IDLE: *// IDLE == 1*

return 'IDLE';

break;

case appCache.CHECKING: *// CHECKING == 2*

return 'CHECKING';

break;

case appCache.DOWNLOADING: *// DOWNLOADING == 3*

return 'DOWNLOADING';

break;

case appCache.UPDATEREADY: *// UPDATEREADY == 4*

return 'UPDATEREADY';

break;

case appCache.OBSOLETE: *// OBSOLETE == 5*

return 'OBSOLETE';

break;

default:

return 'UKNOWN CACHE STATUS';

break;

};

Để tiến hành cập nhật bộ nhớ cache, đầu tiên gọi applicationCache.update(). Nó sẽ cố gắng cập nhật bộ nhớ cache của người dùng (đòi hỏi các tập tin manifest đã thay đổi). Cuối cùng, khi applicationCache.status ở trạng thái UPDATEREADY, gọi applicationCache.swapCache() để đổi thay bộ nhớ cache cũ.

var appCache = window.applicationCache;

appCache.update(); *// Attempt to update the user's cache.*

...

if (appCache.status == window.applicationCache.UPDATEREADY) {

appCache.swapCache(); *// The fetch was successful, swap in the new cache.*

}

Như vậy, chúng ta chỉ tải lại các tài nguyên cache khi nó thật sự thay đổi, vì vậy ở sự kiện tải trang ta cài đặt lắng nghe sự kiện “updateready”. Nhờ vậy, nếu tài nguyên không có thay đổi gì thì trình duyệt không cần tải về ngay cả khi người dùng tải lại trang:

*// Check if a new cache is available on page load.*

window.addEventListener('load', function(e) {

window.applicationCache.addEventListener('updateready', function(e) {

if (window.applicationCache.status == window.applicationCache.UPDATEREADY) {

*// Browser downloaded a new app cache.*

*// Swap it in and reload the page to get the new hotness.*

window.applicationCache.swapCache();

if (confirm('A new version of this site is available. Load it?')) {

window.location.reload();

}

} else {

*// Manifest didn't changed. Nothing new to server.*

}

}, false);

}, false);

* Sự kiện AppCache

Chúng ta có thể thêm vào xử lý các sự kiện trong quá trình làm việc với bộ nhớ cache, tùy vào từng sự kiện cụ thể mà ta có những xử lí phù hợp. Sau đây là ví dụ thêm lắng nghe các sự kiện cache:

function handleCacheEvent(e) {

*//...*

}

function handleCacheError(e) {

alert('Error: Cache failed to update!');};

*// Fired after the first cache of the manifest.*

appCache.addEventListener('cached', handleCacheEvent, false);

*// Checking for an update. Always the first event fired in the sequence.*

appCache.addEventListener('checking', handleCacheEvent, false);

*// An update was found. The browser is fetching resources.*

appCache.addEventListener('downloading', handleCacheEvent, false);

*// The manifest returns 404 or 410, the download failed,*

*// or the manifest changed while the download was in progress.*

appCache.addEventListener('error', handleCacheError, false);

*// Fired after the first download of the manifest.*

appCache.addEventListener('noupdate', handleCacheEvent, false);

*// Fired if the manifest file returns a 404 or 410.*

*// This results in the application cache being deleted.*

appCache.addEventListener('obsolete', handleCacheEvent, false);

*// Fired for each resource listed in the manifest as it is being fetched.*

appCache.addEventListener('progress', handleCacheEvent, false);

*// Fired when the manifest resources have been newly redownloaded.*

appCache.addEventListener('updateready', handleCacheEvent, false);

1. **Tại sao là HTML5**

Nhanh. An toàn. Đáp ứng hầu hết các nhu cầu của một trang web hiện đại. Tương tác cao. Hiệu ứng tuyệt vời. Đây là những tính năng mạnh mẽ, đón đầu xu hướng chỉ có ở HTML5.

HTML5 với nội lực mạnh mẽ mang theo nhiều tính năng tiên tiến cho phép các nhà phát triển tạo ra các ứng dụng và các trang web với các chức năng, tốc độ, hiệu suất, các thể thay thế, thách thức các ứng dụng máy tính để bàn. Tuy nhiên, không giống như các ứng dụng máy tính để bàn, các ứng dụng được xây dựng trên nền tảng web có thể đạt được một lượng người dùng lớn hơn nhiều bằng cách các đặt tính nổi bật như gọn nhẹ, không cần cài đặt, không đòi hỏi khắt khe về phần cứng,…. HTML5 tiện dụng trong việc cập nhật, nó cho phép dễ dàng tung ra sản phẩm mới nhất cho tất cả người dùng cùng một lúc.

HTML5 cũng giải phóng chúng ta khỏi các phức tạp cài đặt ứng dụng trên nhiều thiết bị. Người dùng chỉ cần bấm vào một liên kết hoặc một biểu tượng là có thể bắt đầu chạy một ứng dụng mới. Họ không còn bị làm phiền với các chi tiết như tải về các bản cập nhật mới nhất, đảm bảo rằng họ đang sử dụng đúng phiên bản. Tất cả dữ liệu, công việc, các công cụ, tính năng giải trí của người dùng theo họ bất cứ nơi nào, họ không còn bị ràng buộc với một thiết bị cụ thể. Tất cả đều trên website, một máy tính, một trình duyệt hỗ trợ HTML5 (hầu hết các trình duyệt đều hỗ trợ HTML5, nếu bạn đang sử dụng một trình duyệt không hỗ trợ HTML5 thì đã đến lúc bạn nên nghĩ tới việc cập nhật trình duyệt của mình).

Một trong những khó khăn ngăn cản việc phổ biến HTML5 là trình duyệt Internet Explorer, hiện chiếm thị phần lớn nhất thế giới, vốn không có mấy quan tâm đến việc tương thích chuẩn này. Tuy nhiên, với việc Microsoft chính thức tuyên bố hỗ trợ HTML5 trong IE9 (các kết quả thử nghiệm gần đây cho thấy tốc độ HTML5 trên IE9 rất khả quan), có thể nói rằng tất cả các trình duyệt lớn đều đã sẵn sàng chào đón HTML5. Điều đó vừa cho thấy sức cuốn hút công nghệ mới này, đồng thời hứa hẹn một tương lai tốt đẹp cho HTML5 cất cánh.

Với tầm với của hàng trăm triệu người sử dụng (Google Chrome có hơn 200 triệu người sử dụng) và phát triển nhanh chóng. Với xu thế mới hiện nay, với việc các website phát triển ngày càng mạnh mẽ và hầu như chiếm toàn bộ lượng người dùng.

Vậy tại sao lại là HTML5? Câu trả lời sau đây sẽ giải thích tất cả:

1. **Đa phương tiện và đồ họa**

Trên thực tế, hầu hết tất cả mọi người đều thích các hiệu ứng 3D, các vụ cháy nổ, các hiệu ứng mượt mà sinh động kèm theo âm thanh,….

Kể từ những ngày “sơ khai” của công nghệ web, các trang web đã là một phương tiện hình ảnh, mặc dù có hạn chế. Các nhà phát triển nếu muốn tạo ra một trang web với trò chơi hấp dẫn, các hiệu ứng hình ảnh tinh tế, mượt mà trên trang web của họ thì phải chuyển sang một nền tảng khác hoặc sử dụng plug-in.

Với HTML5, trình duyệt đã trở thành một nền tảng đầy đủ cho các trò chơi, hiệu ứng hoạt hình, phim đồ họa, bất cứ điều gì. Các chi tiết như ánh sáng và bóng tối, phản chiếu, và họa tiết phong phú tạo ra các thành phần như thật. Các tính năng hiệu suất cao như CSS 3D, đồ họa vector (Canvas và SVG), WebGL giúp tạo nên ứng dụng web với đồ họa 3D và các hiệu ứng đặc biệt tuyệt vời. API âm thanh phong phú và kết nối mạng WebSockets độ trễ thấp cùng với các API và công nghệ đồ họa cho phép tạo ra trải nghiệm hấp dẫn cho người sử dụng.

Tất nhiên, các công nghệ mới này sẽ không hữu ích nếu không có khả năng chạy nhanh. Rất may, các JavaScript đã trở nên đủ nhanh để chạy hiệu năng đồ họa cao và thao tác video trong thời gian thực. Hơn nữa, tăng tốc phần cứng đang được thực hiện trên các trình duyệt hiện đại, có nghĩa là trình duyệt hiện nay sử dụng đơn vị xử lý đồ họa (GPU) để tăng tốc độ tính toán cần thiết để hiển thị mượt mà sự chuyển động của hình ảnh và trong dựng hình 3D.

1. **Hoạt động ngoại tuyến và lưu trữ thông minh**

Web ngoại tuyến? Sẽ không còn là một nghịch lý nữa.

"Web" và "ngoại tuyến", hai thuật ngữ này khiến nhiều người cảm thấy sự mâu thuẫn, không liên kết với nhau. Nhưng chẳng bao lâu chúng sẽ phối hợp hỗ trợ nhau chặt chẽ, với HTML5 API cho phép chúng ta tạo ra các ứng dụng làm việc ngay cả khi nó không có kết nối. Ứng dụng bộ nhớ cache, sessionStorage localStorage IndexedDB File System, và các event trực tuyến và ngoại tuyến cho phép các ứng dụng để làm việc "lưỡng cư"-có hoặc không có kết nối trực tuyến. Người dùng thậm chí có thể tải về các tập tin lớn (lớn hơn 1 GB) toàn bộ hoặc một phần để xem offline sau.

Đi trên máy bay, đi du lịch đến bất cứ nơi nào mà không có Wi-Fi hoặc 3G (boondocks), hoặc nơi có kết nối chập chờn, không đáng tin cậy cũng không thể ngăn cản người dùng làm việc hay giải trí, người dùng hoàn toàn có thể sử dụng trang web hoạt động ngoại tuyến. Nếu để tài nguyên và nội dung trong ổ đĩa cục bộ, ứng dụng của tạo ra có thể hoạt động ngoại tuyến, không có vấn đề gì hay trở ngại nào, không quan tâm người sử dụng đang ở đâu.

Ngoài việc cho phép các ứng dụng giữ lại trạng thái của chúng và giữ các dữ liệu mà không cần một máy chủ, các tính năng ngoại tuyến có ưu điểm nữa là cải thiện hiệu suất của một ứng dụng bằng cách lưu trữ dữ liệu trong bộ nhớ cache hoặc làm cho dữ liệu được đảm bảo giữa session của người dùng và việc tải lại trang.

1. **Hiệu suất**

Nhanh hơn, cao hơn, mạnh mẽ hơn!

HTML5 cho phép các ứng dụng web đáp ứng tốt hơn, tạo ra một trải nghiệm người dùng thứ là đối thủ nặng kí của các ứng dụng máy tính để bàn.

Các API ngoại tuyến không chỉ dành để lưu các tập tin lưu trữ cục bộ, chúng còn có thể cải thiện hiệu suất. Nó cho phép ứng dụng của chúng ta vừa nhanh chóng truy cập dữ liệu được lưu trữ cục bộ vừa giảm thiểu số lần ứng dụng cần phải thực hiện các yêu cầu đến máy chủ. Chúng ta có thể đánh dấu các trang mà người dùng thường bấm vào và lưu trữ tài nguyên cần thiết đó cho nhiệm vụ tiếp theo hoặc level kế tiếp của trò chơi. Kết quả là thời gian tải rất nhanh.

Nhưng ngay cả khi không thể lưu lại tài nguyên trước, chúng ta vẫn có thể tạo ra các ứng dụng chất lượng với các công nghệ mới như Web Workers, cho phép chạy nhiều tiến trình ở nền bên dưới.

Trên hết, JavaScript đã được phát triển rất nhiều. Chúng đã được điều chỉnh để tối ưu hóa tốc độ xữ lí JavaScript. Một số thử nghiệm thậm chí còn cho thấy JavaScript trên V8 engines – trình thông dịch mã nguồn JavaScript mở trên Google Chrome - chạy nhanh hơn so với Python.

Ngoài các công nghệ mới, một loạt các kỹ thuật, chẳng hạn như giảm thiểu việc sử dụng băng thông và thời gian kết nối với các máy chủ, tập tin nén, callbacks có thể bất đồng bộ, thêm zip vào các ứng dụng. Chúng ta cũng có thể truy cập một loạt các thư viện và các công cụ tối ưu hóa hiệu suất. Dễ dàng trong việc phát triển

1. **Dễ dàng phát triển**

Hãy tưởng tượng rằng chúng ta chỉ cần viết ra trò chơi một lần duy nhất và nó có thể chạy trên nhiều thiết bị khác nhau.

Trước tiên, HTML5 cho phép chúng ta hướng tới mục tiêu số lượng lớn nhất của các thiết bị với số tiền phát triển ít nhất. Thứ hai, các trình duyệt hiện đại và các kỹ thuật khác nhau giảm thiểu sự phân tán. Cuối cùng, HTML5 dễ tiếp cận hơn với một tập thể lớn các nhà phát triển.

1. *Sự tương thích*

Các thiết bị chạy nền tảng khác nhau đang được ra mắt liên tục. Vì vậy, khi chúng ta tạo một ứng dụng, chúng ta nên nhắm tới nền tảng nào? Windows, Mac OS, Unix và các biến thể của nó, Chrome OS, iOS, Android, BlackBerry OS, Windows Mobile, hoặc bất cứ hệ điều hành mới sẽ phát triển trong tương lai? Làm thế nào về tối ưu hóa cho một mảng rộng các thiết bị đó, có thể là nhỏ như một điện thoại hoặc một cái gì đó lớn hơn như là một chiếc TV đầy đủ tính năng? Tất cả đều tuyệt vời cho người sử dụng, nhưng thiếu tài nguyên thì phát triển để làm gì?

Mỗi nền tảng đòi hỏi bạn phải nắm vững SDK, các công cụ và ngôn ngữ của nó, nhưng phải mất rất nhiều thời gian. Có thể chọn để hỗ trợ chỉ có một vài nền tảng đang chạy trong một tập hợp tất cả các nền tảng hiện có, nhưng như vậy nghĩa là chúng ta đang bỏ qua một số lượng lớn người dùng tiềm năng.

Với HTML5, nó đơn giản hoá chu kỳ phát triển bằng cách cho phép chúng ta sử dụng cùng một công nghệ trên nhiều nền tảng và thiết bị.

Chúng ta có thể triển khai trên nhiều thiết bị hơn nhanh hơn và dễ dàng hơn bằng cách phát triển cho các trình duyệt web, hơn là bỏ công sức ra phát triển các ứng dụng cho một danh sách ngày càng tăng của các nền tảng.

1. *Một cho tất cả*

Sự thật là cho dù ứng dụng trên máy bàn hay trên trình duyệt thì sự phân khúc luôn là vấn đề (với máy bàn thì chúng ta có khá nhiều hệ điều hành phổ biến, trình duyệt cũng thế!). Tuy nhiên, với HTML5 – một công nghệ duy nhất, thứ mà có thể chạy tốt trên các trình duyệt khác nhau. Chúng ta không cần phải viết lại mọi thứ từ đầu, bất kì cổng gì, hoặc việc sử dụng các đoạn mã để chạy trên các thiết bị riêng biệt là điều dễ dàng. HTML5 gần như có thể nhận được một nguồn duy nhất. Vâng, dĩ nhiên chúng ta vẫn cần phải tùy chỉnh một vài yếu tố để cố gắng làm sao để chúng phù hợp hơn với từng trình duyệt khác nhau, dù sao trong hoàn cảnh nào, chúng ta cũng cần làm thế, hiển nhiên rồi.

Ngoài ra, "sự phân khúc" trong HTML5 tương đối thấp. Đầu tiên, trái ngược với quá khứ, các nhà cung cấp trình duyệt đã nỗ lực đưa trình duyệt của họ vào một tiêu chuẩn chung, góp phẩn giảm thiểu tối đa sự khác biệt giữa các trình duyệt. Thứ hai, lập trình web đã ngày càng “trưởng thành” hơn thông qua việc đối phó với sự phân khúc của các trình duyệt trong quá khứ, nó có thư viện phát triển tốt, kỹ thuật và thực tiễn hoạt động tốt, mượt mà, thống nhất trên các trình duyệt khác nhau – xử lý khác nhau.

Với HTML5, chúng ta có thể tạo các ứng dụng cho duy nhất một trình duyệt trước, sau đó tăng dần thêm lớp tinh chỉnh cho từng trình duyệt cụ thể cho đến khi ứng dụng web đó mở rộng “tầm hoạt động” trên phần lớn các thiết bị.

1. *Khả năng tiếp cận*

Công nghệ HTML5, JavaScript, CSS, và HTML thì dễ dàng tìm hiểu hơn cho những người phát triển. JavaScript thì linh hoạt, sự thích nghi cao, hoạt động tốt trong số lượng lớn các môi trường.

Chúng ta không cần phải làm mọi thứ từ đầu bởi vì có rất nhiều framework, các công cụ, và các thư viện, như Sencha, Dojo, và jQuery có thể giúp ta làm được rất nhiều thứ từ cơ bản đến nâng cao.

Nếu lập trình viên nào không thấy ấn tượng hay hứng thú gì với JavaScript, hãy xem xét lại. Thực tế là có hàng loạt các thư viện, bộ công cụ và framework, đã phát triển nhanh chóng và tiếp tục cải thiện.

1. **Tầm hoạt động không giới hạn**

Không có nền tảng khác có thể phù hợp với người dùng cơ sở của HTML5.

Kết nối với người sử dụng ở bất cứ nơi nào bằng cách tạo ra các ứng dụng HTML5 có thể được triển khai trên nhiều nền tảng và một loạt các thiết bị. Cho dù bạn đang là một doanh nghiệp, phát triển trò chơi, hoặc phát triển ứng dụng cá nhân, HTML5 cho phép người dùng nhanh chóng truy cập vào các ứng dụng quan trọng.

HTML5 được hỗ trợ rộng rãi trên tất cả các trình duyệt máy tính để bàn hiện đại và các thiết bị di động lớn. Không có công nghệ nào khác có thể so sánh được.

1. **Bảo mật**

Tại sao Apple cương quyết không cho phép Plugin Flash tồn tại trên nền iOS của họ? Phải chăng là do iPhone không đủ sức mạnh để xử lí Flash. Câu trả lời là Không. Apple là một hãng rất đề cao sự hoàn hảo của mỗi sản phẩm họ tạo ra. Ở đây, Flash nổi tiếng là một cánh cổng dành cho hacker xâm nhập vào thiết bị của người dùng với số lượng lớn lỗ hỗng bảo mật và sự hoạt động không ổn định của nó. Vì thế Apple quyết định loại bỏ Flash mà tập trung hổ trợ HTML5 ngay từ phiên bản đầu tiên của iPhone – Thiết bị khởi đầu cho thời đại di động hiện nay – để tránh khỏi một trong những nguy cơ lớn về bảo mật.

HTML5 thực sự giúp bạn có thể lướt Web an toàn hơn mà không phải sợ rằng ai đó đang lén lút làm gì đó bên dưới trình duyệt của bạn. HTML5 và các trình duyệt hiện đại đã giới thiệu nhiều tính năng tuân theo các tiêu chuẩn hiện đại về bảo mật. Những tính năng hổ trợ sẳn theo chuẩn HTML5 của trình duyệt về đa phương tiện và tính năng mở rộng khác (cần đến plugin) sẽ giúp giảm sự cần thiết của việc bổ sung các Plugin, mà trong đó như đã nói ở trên, một số Plugin được biết đến như là lỗ hổng của các trình duyệt: Flash, mới đây là Java …

Những tiêu chuẩn của HTML5 giúp trang web an toàn hơn. Việc chuẩn hóa các thuật toán parsing giúp loại bỏ sự khác biệt giữa các trình duyệt, thứ không chỉ làm cản trở khả năng tương tác với các trang HTML không hợp lệ mà còn tạo nên lỗ hổng cho việc tấn công XSS. Một số hành vi có thể gây hại cũng được khóa lại chẳng hạn như block các URL có javascript: trong một số tình huống. Một số cải tiến về bảo mật được thêm vào element iframe –một trong những nơi được dùng để tấn công. iframe hiện tại bao gồm một số thuộc tính mới như sandbox, seamless và srcdoc những thứ mà khi được thông qua bởi các trình duyệt thì sẽ giúp việc duyệt trang web có nội dung không đáng tin cậy được an toàn hơn.

Một số API không liên quan đến bảo mật cũng được thiết kế an toàn hơn. Chẳng hạn như API thông báo và lấy thông tin địa lý yêu cầu người dùng cho phép mỗi lần sữ dụng. API lưu trữ dữ liệu chỉ được thao tác với dữ liệu trong cùng tên miền.

1. **Giảm chi phí và bảo trì dễ dàng hơn**

Những tính năng đã nói trước đây: như về hiệu suất, về hoạt động ngoại tuyến và lưu trữ, bảo mật, những tính năng đó cho phép tạo ra các ứng dụng web mạnh mẽ cấp độ doanh nghiệp. Ứng dụng web rẻ hơn nhiều vì có chạy trên nhiều nền tảng, chúng dễ dàng hơn để bảo trì và làm việc tốt trên các thiết bị cụ thể không liên quan. Tất cả các tính năng này không chỉ giảm chi phí, mà còn nâng cao năng suất.

1. *Không đòi hỏi phần cứng cao*

Trong khi các ứng dụng máy tính để bàn phải được cài đặt một cách riêng biệt trên mỗi máy tính, ứng dụng web chỉ là một cú nhấn chuột. Việc cập nhật trên hoàng loạt máy tính chỉ là điều đơn giản đối với các ứng dụng web, hãy nghĩ đến việc phải cập nhật một ứng dụng máy tính cho 1 hệ thống nhiều máy tính – quả là cơn ác mộng! Người dùng thậm chí không cần phải bận tâm tới những lời nhắc nhở dai dẳng để cập nhật vô tận và sự phức tạp của khởi động lại vào những thời điểm không thích hợp. Khi thiết bị trở nên lỗi thời và đã được thay thế, thì viễn cảnh dã tràng xe cát lại tái hiện, cài đặt và cài đặt, nâng cấp và nâng cấp các ứng dụng máy tính để bàn được lặp đi lặp lại cho thiết bị mới. Nhưng với các ứng dụng web, nhóm nghiên cứu chỉ khởi động lên các thiết bị mới, cài đặt các trình duyệt mới nhất và bắt đầu sử dụng, thế thôi, không thể đơn giản hơn!

1. *Trải nghiệm người dùng tốt hơn*

Từ quan điểm của người dùng, họ được đảm bảo rằng ứng dụng của họ luôn được cập nhật. Họ không phải lo lắng về khả năng tương thích, và cũng không cần phải lo ngại về những phiên bản đồng nghiệp của họ đang sử dụng có thể ảnh hưởng tới dữ liệu chung trong quá trình làm việc nhóm. Trong tương lai không xa, với HTML5, các khái niệm về các phiên bản phần mềm trở nên hoàn toàn vô nghĩa. Nhớ lại xem, lần cuối cùng bạn quan tâm đến số phiên bản của ứng dụng webmail là khi nào?

1. *Độc lập với thiết bị*

Với HTML5 các thiết bị không liên quan nữa. Người sử dụng có thể chỉ cần mở trình duyệt trên thiết bị khác và ngay lập tức truy cập các ứng dụng cốt lõi và các dữ liệu quan trọng. Nếu họ rời khỏi các thiết bị của họ đến một nơi nào đó, cũng không có vấn đề gì, họ vẫn có thể truy cập vào công việc của họ và cuộc sống điện tử từ các thiết bị khác trong một vị trí khác.

Vì vậy, khi bị mất, bị đánh cắp hay bị phá hủy, hư hại các thiết bị,… các điều đó không hoàn toàn là 1 thảm họa.

Người dùng không chỉ được giải phóng khỏi sự gò bó khi phải “chôn chân, gắn chặt” với các thiết bị cụ thể, mà họ còn được giải thoát khỏi những thứ lặt vặt làm mất thời gian chẳng hạn như việc sao lưu dữ liệu và lo ngại một khi ổ cứng bị hư hại.

1. *Giải pháp cho các doanh nghiệp*

Phát triển một lần cho nhiều thiết bị, dễ dàng triển khai và nâng cấp đồng thời, di chuyển đơn giản hóa sao lưu dữ liệu và truy cập vào các ứng dụng cốt lõi từ nhiều thiết bị làm cho HTML5 là một nền tảng tuyệt vời cho các doanh nghiệp.

HTML5 mang lại lợi ích đáng kể cho các công ty mà không có mô hình công nghệ nào khác có thể mang lại như: chi phí, an ninh, tính linh hoạt, tính di động và tốc độ đổi mới.

1. **Truy cập dữ liệu cục bộ**

“Sandbox” của các trình duyệt bảo vệ hệ thống tập tin của người sử dụng từ các trang web độc hại, nhưng nó cũng giúp các ứng dụng web có thể truy cập hệ thống tập tin để đọc và ghi các tập tin. Điều này giúp các ứng dụng web có thể lưu trữ các tập tin hoặc thao tác với chúng. Với File System API, chúng ta hoàn toàn có thể trực tiếp thao tác với các dữ liệu nhị phân như âm nhạc, hình ảnh, và video trên các trang web, những tương tác này không còn là riêng trên ứng dụng máy tính nữa. Các ứng dụng web có thể tạo, đọc, điều hướng cũng như gởi tin nhắn đến một phần sandbox của hệ thống tập tin cục bộ của người sử dụng.

Vậy lợi ích là gì? Vâng, chúng ta có thể tạo ra các ứng dụng web thứ mà có thể giao tiếp tuyệt với với các tập tin cục bộ của người sử dụng. Ví dụ, tạo ra sự thuận tiện cho người dùng trong việc trao đổi tập tin. Hoặc có thể tạo ra các ứng dụng web hiển thị xem trước hình ảnh khi chúng được gửi đến máy chủ hoặc việc khôi phục tải lên hay tải xuống các tập tin khi bị gián đoạn mạng hoặc trình duyệt bị treo.

Ngoài ra, với HTML5 ta còn có thể làm được rất nhiều từ phía người dùng như việc kiểm tra thứ mà trước đây phải thực hiện trên máy chủ. Ví dụ, ứng dụng của bạn có thể sử dụng các tập tin trên máy người dùng để đăng tải lên. Các ứng dụng có thể hạn chế kích thước của tập tin khi tải lên, vì vậy có thể tránh khỏi được sự phiền toái khi người dùng phải ngồi chờ đợi thời gian upload lâu để rồi nhận được thông báo là việc đăng tải không thành công do kích thước tập tin quá lớn.

File System API không chỉ đơn giản là truy cập vào hệ thống tập tin cục bộ, mà nó còn xa hơn, đó là: sự tương tác trực tiếp giữa các tập tin web và tập tin cục bộ, có nghĩa là ta có thể tạo các ứng dụng chỉnh sửa hình ảnh hoặc phim, chỉnh sửa nhạc, phát nhạc, và các công cụ trực quan phương tiện truyền thông bằng cách truy cập trực tiếp vào các tập tin nhạc, hình ảnh, video, và các tập tin khác trên ổ đĩa cứng của người sử dụng.

1. **Sự trình diễn**

Với sự ra đời của CSS3, các ứng dụng có thể có một thiết kế thanh lịch, biểu cảm và linh hoạt. Chúng ta có thể tạo các hiệu ứng đẹp mắt như các góc tròn, gradients và bóng tối. Các thiết kế có thể chuyển đổi một văn bản đơn giản thành thứ gì đó sống động, phong phú hơn như hình ảnh động hoặc các biến đổi 3D.

Đơn giản hóa màu sắc và quản lý hình ảnh cho phép tải nhanh hơn (ít yêu cầu HTTP) và thiết kế lại đơn giản. Chúng ta cũng có thể thao tác trực tiếp các yếu tố, xoay, phóng to hầu như bất cứ thứ gì mà không cần đổi hình ảnh khác.

Tất cả sự trình bày có thể được thực hiện chỉ với CSS3, không cần phải đụng tới một tí JavaScript, làm cho bảo trì dễ dàng hơn, bởi vì không phải tất cả người phát triển đều phải biết cả CSS lẫn JavaScript.

1. **Tương tác người dùng**

Tương tác theo kiểu click chuột là vào thế kỷ trước.

Trong web cũ, trình duyệt đầu vào chỉ chấp nhận những cú click chuột và các mục bàn phím. Sau đó, di chuyển theo chuột, như vậy thôi mọi người đã cảm thấy thú vị rồi.

Ngày nay, HTML5 là tất cả về làm web tốt hơn, nó mang đến một luồng gió mới. Vì vậy, phương pháp mới để tương tác với các ứng dụng web đã được giới thiệu: kéo và thả, định vị, định hướng thiết bị, và các sự kiện liên lạc.

Việc sử dụng web trên trên thiết bị truy cập đang được phát triển, có nghĩa là không quá xa trong tương lai, các ứng dụng web sẽ có thể chấp nhận đầu vào từ micro, máy ảnh, máy ghi video, và các thiết bị USB.

1. **Kết nối, giao tiếp thời gian thực**

Các tính năng của HTML5 có thể chia thành hai loại: các tính năng giúp ứng dụng web ngang bằng với ứng dụng trên máy tính và các tính năng cung cấp cho ứng dụng web lợi thế hơn cả ứng dụng ứng dụng máy tính. Giao tiếp thời gian thực thuộc loại thứ hai, và nó mang lại cho các ứng dụng web một thế mạnh rất lớn.

WebSockets và WebRTC thực sự có thể thay đổi các trò chơi giao tiếp thời gian thực bằng cách phát triển dễ dàng hơn và trải nghiệm người dùng tốt hơn. Chúng loại bỏ các plug-in và công nghệ độc quyền.

WebSockets cung cấp các kênh song song và đầy đủ. Bởi vì nó cho phép máy tính giao tiếp với nhau theo cả hai hướng, các ứng dụng web không đòi hỏi sự chấp thuận của các máy chủ với yêu cầu lặp đi lặp lại, tăng tốc độ hiệu suất. Hơn nữa, người dùng khác nhau có thể chia sẻ tài nguyên với nhau từ các máy tính khác nhau ở cùng thời gian như chia sẻ một màn hình hoặc âm thanh đến một máy tính khác. WebSockets cũng là công cụ tuyệt vời để trò chuyện hoặc gởi bất kỳ dữ liệu theo thời gian thực.

WebRTC hỗ trợ cho thảo luận, họp nhóm trực tiếp qua hình ảnh và âm thanh.

1. **Sự hỗ trợ của các trình duyệt**

Đối với nhiều nhà phát triển web, điều này có thể là vấn đề lớn khi mà HTML5 không được hỗ trợ bởi các trình duyệt cũ như IE6. Nhưng không cần phải lo lắng về điều đó, HTML5 hiện đang trong quá trình phát triển và đến nay nó đã được xác nhận là thể hỗ trợ tốt trên nhiều trình duyệt cũ.

Tất cả các trình duyệt phổ biến hiện nay đều hỗ trợ HTML5 (Chrome, Firefox, Safari IE9 và Opera) và DOCTYPE của HTML5 được tạo ra để chạy trên tất cả các trình duyệt, ngay cả những trình duyệt khó chịu như IE6 cũng có thể sử dụng nó. Nhưng chỉ vì trình duyệt cũ không nhận loại tài liệu đó không có nghĩa nó không thể sử dụng tất cả các thẻ HTML5 và các tính năng mới! May mắn thay, HTML5 đang được xây dựng để làm cho mọi việc dễ dàng hơn và nhiều hơn nữa, với những trình duyệt IE cũ, không tương thích các thẻ mới, chúng ta có thể chỉ đơn giản là thêm một Shiv Javascript thì nó sẽ cho phép họ sử dụng các yếu tố mới:

<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>

1. **Đó là sản phẩm của tương lai**

Trên tất cả, lý do số một lý do tại sao nên bắt đầu sử dụng HTML5 ngay hôm nay là: đó là tương lai, bắt đầu sử dụng nó ngay bây giờ để không bị bỏ lại phía sau. HTML5 sẽ không bị lãng quên khi mà ngày càng nhiều các yếu tố được áp dụng nhiều hơn và nhiều hơn nữa, công ty sẽ bắt đầu phát triển trong HTML5. Về cơ bản, nó chỉ là HTML, nó không đáng sợ, nó không phải là điều gì ghê gớm khiến ta phải bỏ công sức ra mày mò, học mới lại - nếu bạn đang phát triển XHTML, hiện giờ nó đã có trong HTML5! Vậy tại sao không tận dụng lợi thế đầy đủ các năng lực hiện tại đó?

1. **Tham khảo**

[4.8.11 The canvas element — HTML Standard](http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#the-canvas-element)

[Improving HTML5 Canvas Performance - HTML5 Rocks](http://www.html5rocks.com/en/tutorials/canvas/performance/)

[HTML Canvas Reference](http://www.w3schools.com/tags/ref_canvas.asp)

[HTML5 Server-Sent Events](http://www.w3schools.com/html/html5_serversentevents.asp)

[HTML5 Geolocation](http://www.w3schools.com/html/html5_geolocation.asp)

1. **a**