IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Games Sederhana dengan HTML5

Yudha Yudhanto
yyudhanto@gmail.com
http://www.rumahstudio.com

HTML5 merupakan sebuah bahasa markah untuk menstrukturkan dan menampilkan isi dari World Wide Web, sebuah teknologi inti dari Internet. HTML5 adalah revisi kelima dari HTML.

Dimana tujuan utama pengembangan HTML5 adalah untuk memperbaiki teknologi HTML agar mendukung teknologi multimedia terbaru, mudah dibaca oleh manusia dan juga mudah dimengerti oleh mesin.

HTML5 merupakan hasil proyek dari W3C (World Wide Web Consortium dan WHATWG *(Web Hypertext Application Technology Working Group)*. Dimana WHATWG bekerja dengan bentuk web dan aplikasi dan W3C merupakan pengembang dari XHTML 2.0 pada tahun

2006, kemudian mereka memutuskan untuk bekerja sama dan membentuk versi baru dari HTML.

Berikut tujuan dibuatnya HTML5 :

- Fitur baru harus didasarkan pada HTML, CSS, DOM, dan JavaScript
- Mengurangi ketergantugan untuk plugin eksternal (Seperti Flash)
- Penanganan kesalahan yang lebih baik
- Lebih markup untuk menggantikan scripting
- HTML5 merupakan perangkat mandiri
- Proses pembangunan dapat terlihat untuk umum

Fitur baru dalam HTML5 :

- Unsur kanvas untuk menggambar
- Video dan elemen audio untuk media pemutaran
- Dukungan yang lebih baik untuk penyimpanan secara offline
- Elemen konten yang lebih spesifik, seperti artikel, footer, header, nav, section
- Bentuk kontrol form seperti kalender, tanggal, waktu, email, url, search.

Beberapa browser sudah mendukung HTML5 seperti safari, chrome, firefox, dan opera. Kabarnya IE9 (Internet Explorer) akan mendukung beberapa fitur dari HTML5. Pembuatan HTML5 juga di karenakan Standard HTML4 yang dijumpai banyak memiliki kelemahan untuk mendukung aplikasi web yang interaktif. Akibat hal ini banyak orang menambahkan fitur baru baik disisi aplikasi web ataupun disisi browser. Solusi ini dikenal dengan plugin dan salah satunya adalah Flash dan Silverlight.

Semakin menjamurnya plugin didalam aplikasi atau browser membuat aplikasi web ini susah untuk menembus banyak browser. Hal ini dikarenakan setiap plugin mempunyai cara yang berbeda-beda, sebagai contoh kita ingin memasang plugin flash untuk sharing video maka pada halaman web kita harus ditulis sebagai berikut

```
<object type="application/x-shockwave-flash" width="400" height="220" wmode="transparent"
data="flvplayer.swf?file=movies/holiday.flv">
<param name="movie" value="flvplayer.swf?file=movies/holiday.flv" />
<param name="wmode" value="transparent" />
</object>
```

Contoh diatas menggunakan plugin Flash dari Adobe untuk menjalankan aplikasi web pada browser maka lain caranya bila kita menggunakan Silverlight. Teknologi Silverlight dikembangkan oleh Microsoft. Contoh penggunaan Silverlight pada halaman web dapat dilihat pada HTML dibawah ini

```
<object width="300" height="300" data="data:application/x-silverlight-2," type="application/x-
silverlight-2" >
<param name="source" value="SilverlightApplication1.xap"/>
</object>
```

HTML5 ini dibuat menyederhanakan kompleksitas penggunaan media video dengan standard baru yaitu penggunaan tag <video>. Dengan fitur baru ini maka kita cukup menulis script untuk menjalankan file video sebagai berikut

```
<video src=tutorialku.mp4>
</video>
```

Isu bagaimana menjalankan file video pada aplikasi web merupakan salah satu contoh bagaimana HTML4 tidak dapat mencakup masalah ini dan masih banyak lagi isu pada HTML4. Oleh karena itu, kita sudah saatnya memanfaatkan HTML5 sebagai standard aplikasi web kita.

Testing Browser Support HTML5

Bagaimana caranya untuk menguji apakah browser yang anda install itu sudah support HTML5 atau tidak dan seberapa banyak fitur HTML5 yang disupport? Caranya cukup mudah, pertama-tama pastikan komputer anda sudah terhubung dengan internet dan arahkan ke alamat web sebagai berikut:

```
http://html5test.com
```

Dari data yang ada pada website itu browser Maxthon 3.4.1 merupakan browser terbaik dalam hal mendukung bahasa HTML5 dengan 422 total skor diikuti kemudian dengan google Chrome 20 dengan 414 total skor kemudian berturut-turut Opera 12.00 dengan 385, Firefox 13 dengan 345, Safari 5.1 dengan 317 dan Internet Explorer 9 dengan 138 point.

Ada beberapa hal yang perlu kita ketahui tentang HML 5. HTML 5 yang saat ini sudah mulai diimplementasikan oleh beberapa browser grade A akan membawa lebih dari sekedar fitur untuk layout dan format halaman. Beberapa di antaranya adalah Canvas dan Video.

Canvas

Dulu, untuk bisa memberikan interaksi menggambar di halaman web kita harus memakai applet Java atau Flash. HTML 5 akan memberikan satu opsi tambahan: *canvas*. Seperti namanya, canvas adalah media yang bisa dicorat-coret langsung. Tidak lagi perlu memuat plugin khusus. Cukup tambahkan <canvas> dan javascript maka kita sudah bisa menggambar langsung di halaman web. Sekarang Anda bisa berimaginasi sendiri, kira-kira apa saja yang orang lakukan dengan <canvas>. Apa yang sebelumnya jadi monopoli Flash dan aplet Java akan di-take-over oleh <canvas>.

Video dan Audio

Akan ada tag <audio> dan <video> di HTML 5. Jadi tidak perlu lagi menempelkan flash untuk sekedar memutar audio. Format video yang didukung akan bervariasi terhadap browser, kemungkinan besar codecnya adalah Ogg Theora (patent free) dan H.264. Sepertinya sampai sekarang codecnya masih jadi kontroversi.

Local Storage

Masih ingat Google Gears? Sekarang storage untuk browser akan diakomodasi sebagai standard dalam HTML 5. Aplikasi bisa menyimpan data dalam jumlah lebih besar dari biasanya tanpa harus mengimplementasikan trik dengan cookie atau Flash. Tentunya ini kabar baik bagi pengembang aplikasi web. Mungkin bisa meningkatkan performa aplikasi dengan menggunakan storage sebagai local cache. Coba liat detilnya di sini.

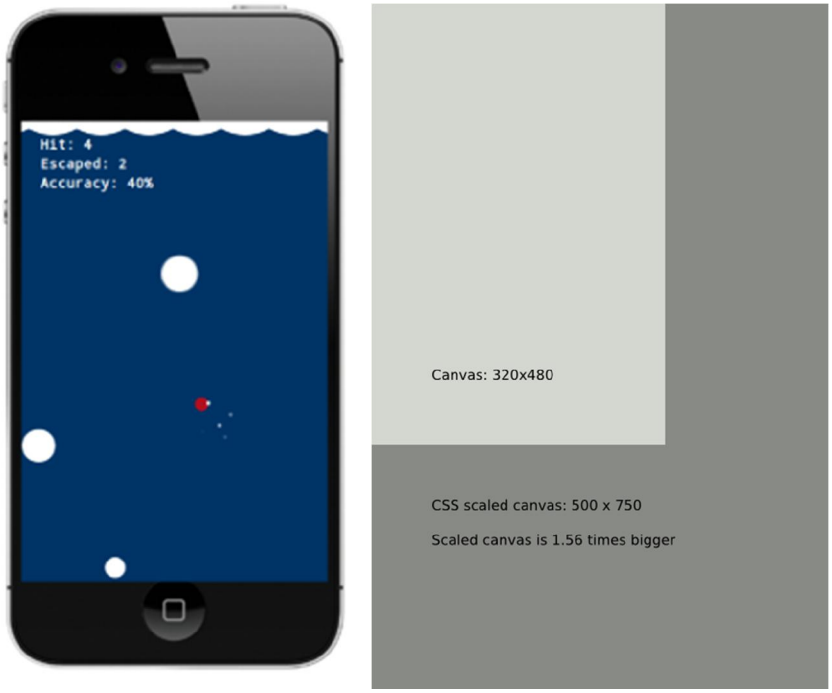
Web Workers

Yang ini juga sempat kita nikmati lewat Google Gears. Jika javascript biasanya yang kita nikmati di web kadangkala menyebabkan komputer kita melambat atau paling tidak membuat browser seperti sesak napas maka web worker akan bsia jadi pelega. Salah satu fitur web worker adalah threading. Kini javascript bisa dipakai untuk melakukan beberapa proses sekaligus tanpa harus menghambat proses terkait UI.

Semantics

Nah ini dia. Buat designer yang sering meng-abuse div dan span sebagia elemen nav, fret no more. Akan ada tag khusus untuk navigasi, section, footer, dll. Tag yang kaya semantic seperti ini pasti akan lebih bermanfaat dari pada tag yang hanya punya informasi format dan layout saja. Dan bagi mesin, HTML5 akan jadi lebih bisa dimengerti.

Games Sederhana HTML5



Berikut ini adalah script games sederhana dalam format HTML 5.

```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta name="viewport" content="width=device-width,
   user-scalable=no, initial-scale=1, maximum-scale=1, user-scalable=0" />
<meta name="apple-mobile-web-app-capable" content="yes" />
<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent" />

<style type="text/css">
body { margin: 0; padding: 0; background: #000;}
canvas { display: block; margin: 0 auto; background: #fff; }
</style>

</head>

<body>
<canvas></canvas>
<script>

// http://paulirish.com/2011/requestanimationframe-for-smart-animating
// shim layer with setTimeout fallback
window.requestAnimFrame = (function(){
  return  window.requestAnimationFrame       ||
          window.webkitRequestAnimationFrame ||
          window.mozRequestAnimationFrame    ||
          window.oRequestAnimationFrame      ||
          window.msRequestAnimationFrame     ||
          function( callback ){
            window.setTimeout(callback, 1000 / 60);
          };
})();

// namespace our game
var POP = {

   // set up some inital values
   WIDTH: 320,
   HEIGHT:  480,
   scale:  1,
   // the position of the canvas
   // in relation to the screen
   offset: {top: 0, left: 0},
   // store all bubble, touches, particles etc
   entities: [],
   // the amount of game ticks until
   // we spawn a bubble
   nextBubble: 100,
   // for tracking player's progress
   score: {
      taps: 0,
      hit: 0,
      escaped: 0,
      accuracy: 0
   },
   // we'll set the rest of these
   // in the init function
   RATIO:  null,
   currentWidth:  null,
   currentHeight:  null,
   canvas: null,
   ctx:  null,
   ua:  null,
   android: null,
   ios:  null,
```

```
init: function() {

    // the proportion of width to height
    POP.RATIO = POP.WIDTH / POP.HEIGHT;
    // these will change when the screen is resize
    POP.currentWidth = POP.WIDTH;
    POP.currentHeight = POP.HEIGHT;
    // this is our canvas element
    POP.canvas = document.getElementsByTagName('canvas')[0];
    // it's important to set this
    // otherwise the browser will
    // default to 320x200
    POP.canvas.width = POP.WIDTH;
    POP.canvas.height = POP.HEIGHT;
    // the canvas context allows us to
    // interact with the canvas api
    POP.ctx = POP.canvas.getContext('2d');
    // we need to sniff out android & ios
    // so we can hide the address bar in
    // our resize function
    POP.ua = navigator.userAgent.toLowerCase();
    POP.android = POP.ua.indexOf('android') > -1 ? true : false;
    POP.ios = ( POP.ua.indexOf('iphone') > -1 || POP.ua.indexOf('ipad') > -1  ) ? true : false;

    // set up our wave effect
    // basically, a series of overlapping circles
    // across the top of screen
    POP.wave = {
        x: -25, // x coord of first circle
        y: -40, // y coord of first circle
        r: 50, // circle radius
        time: 0, // we'll use this in calculating the sine wave
        offset: 0 // this will be the sine wave offset
    };
    // calculate how many circles we need to
    // cover the screen width
    POP.wave.total = Math.ceil(POP.WIDTH / POP.wave.r) + 1;

    // listen for clicks
    window.addEventListener('click', function(e) {
        e.preventDefault();
        POP.Input.set(e);
    }, false);

    // listen for touches
    window.addEventListener('touchstart', function(e) {
        e.preventDefault();
        // the event object has an array
        // called touches, we just want
        // the first touch
        POP.Input.set(e.touches[0]);
    }, false);
    window.addEventListener('touchmove', function(e) {
        // we're not interested in this
        // but prevent default behaviour
        // so the screen doesn't scroll
        // or zoom
        e.preventDefault();
    }, false);
    window.addEventListener('touchend', function(e) {
        // as above
        e.preventDefault();
    }, false);

    // we're ready to resize
    POP.resize();

    POP.loop();
```

```
},


resize: function() {

    POP.currentHeight = window.innerHeight;
    // resize the width in proportion
    // to the new height
    POP.currentWidth = POP.currentHeight * POP.RATIO;

    // this will create some extra space on the
    // page, allowing us to scroll pass
    // the address bar, and thus hide it.
    if (POP.android || POP.ios) {
        document.body.style.height = (window.innerHeight + 50) + 'px';
    }

    // set the new canvas style width & height
    // note: our canvas is still 320x480 but
    // we're essentially scaling it with CSS
    POP.canvas.style.width = POP.currentWidth + 'px';
    POP.canvas.style.height = POP.currentHeight + 'px';

    // the amount by which the css resized canvas
    // is different to the actual (480x320) size.
    POP.scale = POP.currentWidth / POP.WIDTH;
    // position of canvas in relation to
    // the screen
    POP.offset.top = POP.canvas.offsetTop;
    POP.offset.left = POP.canvas.offsetLeft;

    // we use a timeout here as some mobile
    // browsers won't scroll if there is not
    // a small delay
    window.setTimeout(function() {
            window.scrollTo(0,1);
    }, 1);
},

// this is where all entities will be moved
// and checked for collisions etc
update: function() {
    var i,
        checkCollision = false; // we only need to check for a collision
                        // if the user tapped on this game tick


    // decrease our nextBubble counter
    POP.nextBubble -= 1;
    // if the counter is less than zero
    if (POP.nextBubble < 0) {
        // put a new instance of bubble into our entities array
        POP.entities.push(new POP.Bubble());
        // reset the counter with a random value
        POP.nextBubble = ( Math.random() * 100 ) + 100;
    }

    // spawn a new instance of Touch
    // if the user has tapped the screen
    if (POP.Input.tapped) {
        // keep track of taps; needed to
        // calculate accuracy
        POP.score.taps += 1;
        // add a new touch
        POP.entities.push(new POP.Touch(POP.Input.x, POP.Input.y));
        // set tapped back to false
        // to avoid spawning a new touch
        // in the next cycle
        POP.Input.tapped = false;
```

```
            checkCollision = true;
        }

        // cycle through all entities and update as necessary
        for (i = 0; i < POP.entities.length; i += 1) {
            POP.entities[i].update();

            if (POP.entities[i].type === 'bubble' && checkCollision) {
                hit = POP.collides(POP.entities[i],
                                {x: POP.Input.x, y: POP.Input.y, r: 7});
                if (hit) {
                    // spawn an exposion
                    for (var n = 0; n < 5; n +=1 ) {
                        POP.entities.push(new POP.Particle(
                            POP.entities[i].x,
                            POP.entities[i].y,
                            2,
                            // random opacity to spice it up a bit
                            'rgba(255,255,255,'+Math.random()*1+')'
                        ));
                    }
                    POP.score.hit += 1;
                }

                POP.entities[i].remove = hit;
            }

            // delete from array if remove property
            // flag is set to true
            if (POP.entities[i].remove) {
                POP.entities.splice(i, 1);
            }
        }

        // update wave offset
        // feel free to play with these values for
        // either slower or faster waves
        POP.wave.time = new Date().getTime() * 0.002;
        POP.wave.offset = Math.sin(POP.wave.time * 0.8) * 5;

        // calculate accuracy
        POP.score.accuracy = (POP.score.hit / POP.score.taps) * 100;
        POP.score.accuracy = isNaN(POP.score.accuracy) ?
            0 :
            ~~(POP.score.accuracy); // a handy way to round floats

    },


    // this is where we draw all the entities
    render: function() {

        var i;


        POP.Draw.rect(0, 0, POP.WIDTH, POP.HEIGHT, '#036');

        // display snazzy wave effect
        for (i = 0; i < POP.wave.total; i++) {

            POP.Draw.circle(
                    POP.wave.x + POP.wave.offset +  (i * POP.wave.r),
                    POP.wave.y,
                    POP.wave.r,
                    '#fff');
        }

            // cycle through all entities and render to canvas
            for (i = 0; i < POP.entities.length; i += 1) {
```

```
                POP.entities[i].render();
            }

            // display scores
            POP.Draw.text('Hit: ' + POP.score.hit, 20, 30, 14, '#fff');
            POP.Draw.text('Escaped: ' + POP.score.escaped, 20, 50, 14, '#fff');
            POP.Draw.text('Accuracy: ' + POP.score.accuracy + '%', 20, 70, 14, '#fff');

        },


        // the actual loop
        // requests animation frame
        // then proceeds to update
        // and render
        loop: function() {

            requestAnimFrame( POP.loop );

            POP.update();
            POP.render();
        }


    };

    // checks if two entties are touching
    POP.collides = function(a, b) {

            var distance_squared = ( ((a.x - b.x) * (a.x - b.x)) +
                                ((a.y - b.y) * (a.y - b.y)));

            var radii_squared = (a.r + b.r) * (a.r + b.r);

            if (distance_squared < radii_squared) {
                return true;
            } else {
                return false;
            }
    };


    // abstracts various canvas operations into
    // standalone functions
    POP.Draw = {

        clear: function() {
            POP.ctx.clearRect(0, 0, POP.WIDTH, POP.HEIGHT);
        },


        rect: function(x, y, w, h, col) {
            POP.ctx.fillStyle = col;
            POP.ctx.fillRect(x, y, w, h);
        },

        circle: function(x, y, r, col) {
            POP.ctx.fillStyle = col;
            POP.ctx.beginPath();
            POP.ctx.arc(x + 5, y + 5, r, 0,  Math.PI * 2, true);
            POP.ctx.closePath();
            POP.ctx.fill();
        },


        text: function(string, x, y, size, col) {
            POP.ctx.font = 'bold '+size+'px Monospace';
            POP.ctx.fillStyle = col;
            POP.ctx.fillText(string, x, y);
```

```
    }

};


POP.Input = {

    x: 0,
    y: 0,
    tapped :false,

    set: function(data) {
        this.x = (data.pageX - POP.offset.left) / POP.scale;
        this.y = (data.pageY - POP.offset.top) / POP.scale;
        this.tapped = true;

    }

};

POP.Touch = function(x, y) {

    this.type = 'touch';    // we'll need this later
    this.x = x;             // the x coordinate
    this.y = y;             // the y coordinate
    this.r = 5;             // the radius
    this.opacity = 1;       // inital opacity. the dot will fade out
    this.fade = 0.05;       // amount by which to fade on each game tick
    // this.remove = false;    // flag for removing this entity. POP.update
                            // will take care of this

    this.update = function() {
        // reduct the opacity accordingly
        this.opacity -= this.fade;
        // if opacity if 0 or less, flag for removal
        this.remove = (this.opacity < 0) ? true : false;
    };

    this.render = function() {
        POP.Draw.circle(this.x, this.y, this.r, 'rgba(255,0,0,'+this.opacity+')');
    };

};

POP.Bubble = function() {

    this.type = 'bubble';
    this.r = (Math.random() * 20) + 10;
    this.speed = (Math.random() * 3) + 1;

    this.x = (Math.random() * (POP.WIDTH) - this.r);
    this.y = POP.HEIGHT + (Math.random() * 100) + 100;

    // the amount by which the bubble
    // will move from side to side
    this.waveSize = 5 + this.r;
    // we need to remember the original
    // x position for our sine wave calculation
    this.xConstant = this.x;

    this.remove = false;


    this.update = function() {

        // a sine wave is commonly a function of time
        var time = new Date().getTime() * 0.002;
```

```
            this.y -= this.speed;
            // the x coord to follow a sine wave
            this.x = this.waveSize * Math.sin(time) + this.xConstant;

            // if offscreen flag for removal
            if (this.y < -10) {
                POP.score.escaped += 1; // update score
                this.remove = true;
            }

        };

        this.render = function() {

            POP.Draw.circle(this.x, this.y, this.r, 'rgba(255,255,255,1)');
        };

    };

    POP.Particle = function(x, y,r, col) {

        this.x = x;
        this.y = y;
        this.r = r;
        this.col = col;

        // determines whether particle will
        // travel to the right of left
        // 50% chance of either happening
        this.dir = (Math.random() * 2 > 1) ? 1 : -1;

        // random values so particles do no
        // travel at the same speeds
        this.vx = ~~(Math.random() * 4) * this.dir;
        this.vy = ~~(Math.random() * 7);

        this.remove = false;

        this.update = function() {

            // update coordinates
            this.x += this.vx;
            this.y += this.vy;

            // increase velocity so particle
            // accelerates off screen
            this.vx *= 0.99;
            this.vy *= 0.99;

            // adding this negative amount to the
            // y velocity exerts an upward pull on
            // the particle, as if drawn to the
            // surface
            this.vy -= 0.25;

            // offscreen
            if (this.y < 0) {
                this.remove = true;
            }

        };


        this.render = function() {
            POP.Draw.circle(this.x, this.y, this.r, this.col);
        };

    };
```

```
window.addEventListener('load', POP.init, false);
window.addEventListener('resize', POP.resize, false);

</script>


</body>
</html>
```

Referensi :

- http://mkr-site.blogspot.com/2012/07/apa-itu-html5-dan-kelebihannya.html
- http://blog.politekniktelkom.ac.id/30210133/2012/07/17/apa-itu-html-5-kelebihan-html-5-video/
- http://rumahstudio.com/kuliah/games
- http://mobile.smashingmagazine.com/2012/10/19/design-your-own-mobile-game/

Biografi Penulis

Yudha Yudhanto, SKom

Alumni SMK N 2 Surakarta dan UNIKOM. Pernah merantau ke Bandung dan akhirnya pulang ke desa. Saat ini aktivitas utamanya belajar mulang di D3 TI Universitas Sebelas Maret (UNS).
Ketertarikan Ilmu dan Bisnis di bidang Solusi Barcode, RFID dan Fingerprint