JavaScript animations and JQuery

cs380
DIFFERENCE BETWEEN ‘==‘ AND ‘===‘

By- Gulesh Shukla
Why two operators for comparison: ‘==’ and ‘===’?

- JavaScript allows users the flexibility to make data types optional, while performing equivalence comparisons.
- Therefore, to implement these two different type of equivalence comparisons, JavaScript has two operators: “==” and “===”.
- Other languages do not provide users with this flexibility. Hence, there is no requirement to have an additional comparison operator.
‘==’ Operator

- This operator compares the values without caring about their types.
- For example, if we have 2 == “2”, we will get true because this operator did not worry about the type but the value itself.
- Similarly, if we have 0 == false, we get true as 0 is associated with false.
- We can write 2==2 and get true. But with this kind of comparisons (strict comparison), we would want to use ===.
‘===’ operator

- This operator is equivalent to our conventional equivalence comparison operator which is ‘==’
- We use === to compare the variables along with their data types
- For example, if we have 2 ===“2”, we will get false because we are comparing an integer with string.
- Similarly, if we have 0 === false, we will get false as these two are not the same: one is an integer while other is a Boolean.
Sources

Presentations

• Scoping in JavaScript
  • https://docs.google.com/presentation/d/1GW1pH19opLh5rMZ0YBJphW3YbcMDbAuftxnHgM71PLc/edit#slide=id.p

• Var, Let and Const
  • https://docs.google.com/presentation/d/1vSGR2S_uAYfHCAo1KzerXLwdvp7X_ZykJyorloDflc/edit#slide=id.p
Javascript issue #4

Watching closely, the expected text shows up, briefly.
Why did it go away?

file:formprob.html
Promises

- fetch('products.json')
  - Puts a promise into the promise cloud which will result in putting the “then” into the Job Queue
  - First return value is effectively a new Promise
    - when it completes new then then goes into Job Queue.

```javascript
fetch('products.json').then(function(response) {
  return response.json();
}).then(function(json) {
  products = json;
  initialize();
}).catch(function(err) {
  console.log('Fetch problem: ' + err.message);
});
```
Promise Chaining

- initial promise executes `resolve(1)` (asynch)
- `resolve` puts `then` onto JobQueue
- “then” execution acts like a new promise, putting next `then` onto job queue.
- Etc
- This example all could have been done in single `then`, but what if you want to load a page, then load another based on the contents of the first
- In second example, `JSON.parse` .. is async so `then` gets executed when parse complete
More Promising

Two functions in then depending on call to resolve or reject in promise
file:eventloop3.html
“Drawing” with divs

idea: make a div and put it where you want it

```html
<html>
<head>
    <script src="../JQ/jquery-1.9.1.min.js"></script>
    <style>
        .maind {
            width: 90%;
            height: 90%;
            margin-left: 50px;
            margin: auto;
            border: 3px solid #73AD21;
        }
        .balld {
            position: absolute;
            text-align: center;
            border: 3px solid black;
        }
        .btnn {
            height: 45px
        }
    </style>
</head>
```
DwD, part 2

The Layout

```html
<body>
  <div id="mdiv" class="maind"></div>
  <table width="100%">
    <tr><td width="50%"><center><button class="btn" onclick="makeBall()">Add A Ball</button></center></td>
        <td><center><button class="btn" onclick="clearBalls()">Clear</button></center></td>
  </tr>
</table>
```
DwD, part 3
Javascript (using JQuery)

```javascript
function makeBall() {
    counter = counter + 1; // get the place where the ball will
    cnvas = $("#mdiv"); // get the place where the ball will
    tx = cnvas.width(); // get the place where the ball will
    ty = cnvas.height(); // get the place where the ball will
    radius = 14 + Math.random() * (0.4 * (tx < ty ? tx : ty)); // ball size
    x = Math.random() * (tx - radius); // ball location
    y = Math.random() * (ty - radius); // ball location
    // make the ball
    jelem = $('\'<div>' + counter + '\'</div>'); //document.createElement("div");
    jelem.addClass("balld");
    jelem.css( {
        'line-height':radius+'px',
        'margin-left':x+'px',
        'margin-top':y+'px',
        'height': radius+'px',
        'width':radius+'px',
        'border-radius':radius+'px',
        'background-color': randomColor() });
    jelem.hover(function(){ //mouseover and mouseout
        $(this).css('border', '3px solid yellow');
    }, function(){
        $(this).css('border', '3px solid black');
    }); //put the ball into the target div
    cnvas.append(jelem);
}

function randomColor() {
    return ("rgba(" +
            Math.round(Math.random() * 250) + "," +
            Math.round(Math.random() * 250) + "," +
            Math.round(Math.random() * 250) + "," +
            Math.ceil(Math.random() * 10) / 3 + ")"); //The number of balls created.
var counter = 0;
/**
 * Create a random color. Actually this returns a string
 * which can be evaluated into a random color
 */
function randomColor() {
    return ("rgba(" +
            Math.round(Math.random() * 250) + "," +
            Math.round(Math.random() * 250) + "," +
            Math.round(Math.random() * 250) + "," +
            Math.ceil(Math.random() * 10) / 3 + ")");
}
/**
 * Clear all of the balls
 */
function clearBalls() {
    "balld".remove();
}
```
Putting things in Motion

CSS transforms

- transform allow you to do simple things to an html element
  - translate
  - scale
  - rotate
  - skew
  - matrix
- all of the above

Button causes one 10 degree rotation. How do you make it do more rotation on each tap?

How to you make it do more than one move?

file:rotate0.html
JQuery Animations

• JQuery animate function lets you animate transition of several properties.
• Things transforms work on
• When complete, perform completion function

```html
<html>
<head>
  
  <script src="../JQ/jquery-1.9.1.min.js"></script>
  
  <style>
  .maind {
    ...
  }
  .rdiv {
    ...
  }
  </style>
</head>
<body>
  
  <div class="maind">
    
    <div class="rdiv" id="rdiv">DIV2</div>
  </div>
  
  <button class="bttn" onclick="rotate()">Rotate</button>

  var openn=true;
  function rotate() {
    let div=$(".rdiv");
    let wid = parseInt(div.css("width"));
    let marl = parseInt(div.css("margin-left"));
    div.animate({width:0, marginLeft:(marl+wid/2)}, 500, function() {
      div.css("background-color", (openn?"red":"gold");
      openn=!openn;
      div.animate({width: wid, marginLeft:marl}, 500);
    });
  }
</body>
</html>
```

This could use a skew also to create some forced perspective

Question: make the door rotate 3 times
Rotate in Other divs

• Rather than just resizing a div, one goes away a new appears

• Get a classic “slideshow” effect

• Again, how to make it continuous?

Show in VS code: rotatingdiv4.html
URL  rotatingDiv4.html
DwD, adding animation

Adjustments to CSS and html layout

file:b2anim.html, b2animu.html
DwD, adding animation

Javascript changes

```javascript
// unchanged above here

function moveOneStep() {
    for (ball of balls) {
        let cnvas = $("#mdiv"); // get the place where the ball will be put
        let tx = cnvas.width();
        let ty = cnvas.height();
        ball.xloc += ball.addx;
        ball.yloc += ball.addy;
        if (ball.xloc >= (tx-ball.radius)) {
            ball.xloc = tx-ball.radius;
            ball.addx = -ball.addx;
        }
        if (0>ball.xloc) {
            ball.xloc=0;
            ball.addx = -ball.addx;
        }
        if (ball.yloc >= (ty-ball.radius)) {
            ball.yloc = ty-ball.radius;
            ball.addy = -ball.addy;
        }
        if (0 > ball.yloc) {
            ball.yloc = 0;
            ball.addy = -ball.addy;
        }
        ball.element.css({
            'margin-left': ball.xloc+'px',
            'margin-top':ball.yloc+'px',
        });
    }
}
```

```javascript
var balls = [];
function addBall() {
    balls.push(makeBall());
}
```
setInterval and setTimeout

• Do Once in future:
  • handle = setTimeout(callback, time)
    • puts callback func into timerTable
  • clearTimeout(handle)
    • removes from timerTable
• Do repeatedly in future:
  • handle = setInterval(callback, time)
  • clearInterval(handle);

• Write setInterval using setTimeout() — including the ability to stop!
Clicks in a Canvas

```javascript
$(document).ready(function() {
    var ww = Math.floor(0.9*window.innerWidth);
    var wh = Math.floor(0.9*window.innerHeight);
    var cnvas = document.getElementById("canvas");
    cnvas.width = ww;
    cnvas.height = wh;

    cnvas.addEventListener("mousedown", function(e) {
        getMousePosition(cnvas, e);
    });

    function getMousePosition(canvas, event) {  
        var rect = canvas.getBoundingClientRect();  
        var x = event.clientX - rect.left;
        var y = event.clientY - rect.top;
        console.log("Coordinate x: ", x, "Coordinate y: ", y);
    }
});
```
Drawing with Canvas

- canvas is an html element that you can literally draw on.
- Just doing circles so everything here could be done with divs.
- Diagonal lines, etc not so much.
- canvas and jquery do not talk well so using base javascript for canvas.
var ballcount = 0;
var balls = [];
var animating = 0;
$(document).ready(
    function() {
        console.log("A");
        console.log("b");
        ww = Math.floor(0.9*window.innerWidth);
        wh = Math.floor(0.9*window.innerHeight);
        cnvas = document.getElementById("canvas");
        cnvas.width = ww;
        cnvas.height = wh;
    }
);

function randomColor() {
    return ("rgba(" +
    Math.round(Math.random() * 250) + "," +
    Math.round(Math.random() * 250) + "," +
    Math.round(Math.random() * 250) + "," +
    Math.ceil(Math.random() * 10) / 3 + ")"
function makeBall() {
    ballcount = ballcount + 1;
    cnvas = document.getElementById("canvas");
    tx = cnvas.clientWidth;
    ty = cnvas.clientHeight;
    ball = new Object();
    ball.radius = Math.random() * (tx * 0.1) + 14;
    ball.x = Math.random() * (tx - 2 * ball.radius) + ball.radius;
    ball.y = Math.random() * (ty - 2 * ball.radius) + ball.radius;
    ball.color = randomColor();
    ball.speed = Math.random() * ball.radius * 0.33 + 1;
    ball.counter = ballcount;
    return ball;
}

function startBall() {
    balls.push(makeBall());
    if (animating == 0) {
        console.log("Interval start");
        animating = 1;
        window.requestAnimationFrame(drawBalls);
    }
}

DwC, javascript part 2
function drawBall(ctx, ball) {
    ctx.beginPath();
    ctx.arc(ball.x, ball.y, ball.radius, 0, 2 * Math.PI);
    console.log(ball.counter + " " + ball.x + " " + ball.y + " " + ball.radius);
    ctx.fillStyle = ball.color;
    ctx.fill();
    ctx.stroke();
    ball.x = ball.x + ball.speed;
}

function drawBalls() {
    var tx = cnvas.clientWidth;
    var ty = cnvas.clientHeight;
    var ctx = cnvas.getContext("2d");
    ctx.clearRect(0, 0, tx, ty);
    for (i = balls.length - 1; i >= 0; i--) {
        drawBall(ctx, balls[i]);
        if ((balls[i].x - balls[i].radius) > tx) {
            balls.splice(i, 1);
        }
    }
    if (balls.length > 0)
        window.requestAnimationFrame(drawBalls);
    else
        animating = 0;
}