Maharishi University of Management

Section 7: DOM/jQuery

Exercise : Bouncing Ball

Create a page which contains an animated bouncing ball. You can view an example of this page here. You are given `ball.html` and `ball.css`, and you will write `ball.js`.

- Every frame of animation, apply a "gravity" to the ball and increase its downward speed by 1.
- If the ball hits the ground, make it "bounce" up at 90% the velocity it previously had.
- The function below will apply gravity and make the ball bounce when it hits the bottom of the window.
- Write the code needed to start the ball in the top middle of the screen and call the update function.

```
function update() {
  ballVelocity += 1;
  if (parseInt($("#ball").css('top')) > $(window).height()) {
    ballVelocity *= -.9;
  }
  $("#ball").css('top', function(idx, old) {
    return parseInt(old) + ballVelocity + 'px';
  });
}
```

Exercise Solution

```
var START = 0;
var ballVelocity = 0;

$(function() {
  $("#ball").css({
    'top': '0px',
    'left': $(window).width() / 2 + 'px'
  });
  setInterval(update, 20);
});
```

```
function update() {
  ballVelocity += 1;
  if (parseInt($("#ball").css('top')) > $(window).height()) {
    ballVelocity *= -.9;
  }
  $("#ball").css('top', function(idx, old) {
    return parseInt(old) + ballVelocity + 'px';
  });
}
```

Exercise : Cheerleader

Given cheerleader.html, write the JavaScript code `cheerleader.js` to echo typed characters to the screen as in this working example.

Inject the pressed keys as `li` elements inside `#cheers`. Each character should be in upper case, followed by an exclamation point. The lecture slides on key events will probably be useful to help you determine which key was pressed and convert it from a code to a character.

After you have made the pressed keys appear, modify your code so that each cheer removes itself from the page after two seconds (i.e., 2000 milliseconds).

Use the following code as a (big) hint. You just need to insert something in the indicated spots.

```
$(function() {
  $(document).keypress(cheer);
});

function cheer(e) {
  $("<li>")
    .text(String.fromCharCode(e.which).toUpperCase() + "!")
    .appendTo(REPLACE THIS WITH APPROPRIATE CODE);
  //setTimeout(removeCheer, 2000);
}

function removeCheer() {
  $(REPLACE THIS WITH APPROPRIATE CODE).first().remove();
}
```

Exercise Solution

```
$(function() {
  $(document).keypress(cheer);
});

function cheer(e) {
  $("<li>")
    .text(String.fromCharCode(e.which).toUpperCase() + "!")
    .appendTo($("#cheers"));
  setTimeout(removeCheer, 2000);
}

function removeCheer() {
  $("#cheers li").first().remove();
}
```

Exercise : Turtles All the Way Down

A well-known scientist (some say it was Bertrand Russell) once gave a public lecture on astronomy. He described how the earth orbits around the sun and how the sun, in turn, orbits around the center of a vast collection of stars called our galaxy. At the end of the lecture, a little old lady at the back of the room got up and said: What you have told us is rubbish. The world is really a flat plate supported on the back of a giant tortoise. The scientist gave a superior smile before replying, What is the tortoise standing on? You're very clever, young man, very clever, said the old lady. But it's turtles all the way down!

— *Stephen Hawking, A Brief History of Time*

(see also: Turtles all the way down on Wikipedia)

Exercise : Turtles All the Way Down

We will implement a version of the "turtles all the way down" model (commonly but falsely attributed to Hindu mythology) in which the earth rests on the back of an elephant, which in turn rests on the back of infinite tortoises.

Given turtles.html (containing the earth, elephant, and a single tortoise), write the necessary JavaScript code `turtles.js` to give the page infinitely-scrolling turtles as in this working example.

Exercise : Turtles All the Way Down

To solve this problem you will need to add a turtle to the bottom of the page every time the user scrolls to the bottom. You can attach an `onscroll` event handler to the `document`, so that every time the page is scrolled that event handler will be executed. Like this: $(document).scroll(myHandler)

You may also find these useful:

`$(window).scrollTop()`
: The portion of the document that has scrolled off-screen *above* the currently-visible portion.

`$(window).height()`
: The height of the visible area of the document on-screen.

`$(document.body).height()`
: The height of the `body` tag (i.e., the height of the entire page, on- and off-screen).

`$(window).scrollTop() + $(window).height() >= $(document).height()`
: Will be true whenever you have scrolled to where the document is smaller than the visible window.

`$("<div>").addClass('turtle')`
: This is how to create a div that will look like a turtle. (see the css file for details)

Exercise : Turtles All the Way Down

When the user has scrolled to the very bottom of the page, the sum of the window's `scrollTop()` (the off-screen portion above) and `height()` (the on-screen portion) will be equal to the `height()` of the body. That's when you'll want to add another turtle — which will grow the body and result in more scrolling.

To add a turtle, simply append to `document.body` a new `div` with the class of `turtle`.

The solution is actually quite simple given the previous hints. Fill in the indicated parts of the JS file shown below.

```
$(function() {
  //ASSIGN THE SCROLL EVENT HANDLER HERE
  // in case window height is initially taller than animals
  //turtles();
});

function turtles() {
  while ( HAVE SCROLLED BEYOND THE SCREEN) {
    $(document.body).append(
      ANOTHER TURTLE
    );
  }
}
```

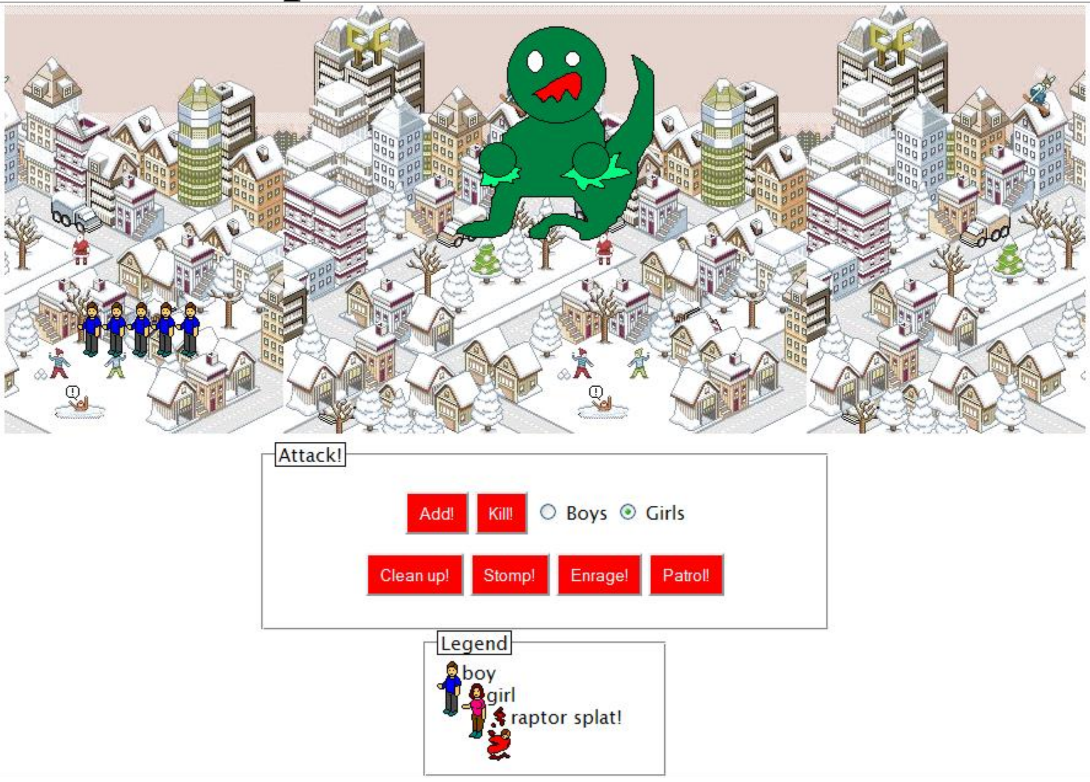
Exercise Solution

```
$(function() {
  $(document).scroll(turtles);
  // in case window height is initially taller than animals
  turtles();
});

function turtles() {
  while ($(window).scrollTop() + $(window).height() >= $(document).height())
{
    $(document.body).append(
      $('<div>')
```

```
        .addClass('turtle')
    );
  }
}
```

Exercise : Raptor



A raptor is on the loose. Rawr! He wants to stomp the townspeople. Write JavaScript code to allow the raptor to eat them. The HTML and CSS are already completely written; start from this skeleton of **attack.html**. (Click the image to run the sample solution.)

Exercise : Raptor

Make it so that when the page first appears, **5 boys are visible** in the town. There are already 5 persons in the HTML, but they have no gender. These are stored in the `div` with `id` of `people` as `div`s with the `class` of `person`. **Assign them the additional class `boy` when the page loads** (while retaining the class `person`).

```
<div id="people">
  <!-- give these 5 divs the class 'boy' -->
  <div class="person"></div>
  <div class="person"></div>
  <div class="person"></div>
  <div class="person"></div>
  <div class="person"></div>
</div>
```

- HINT 1 (hover)
- HINT 2 (hover)
- HINT 3: Add/remove CSS classes from an element with jQuery's `addClass()` function.

Exercise Solution

```
$(function() {
  prepopulate();
});
function prepopulate() {
  $("#people .person").addClass("boy");
}
```

Exercise : Raptor - Boys

Boys / **Girls**: Selects which gender to add or kill. Since our page will often need to know which radiobox is checked, let's make a function that returns the string "boy" if the Boys radio button is checked, and vise versa.

```
<label><input id="boys" type="radio"
    name="gender" value="boy" /> Boys</label>
<label><input id="girls" type="radio"
    name="gender" value="girl" checked /> Girls</label>
```

Exercise Solution

```
// Helper function to get which gender is currently selected.
// Use the return from this to parameterize populate() and kill().
function getGender() {
    return $('input:checked').val();
}
```

Exercise : Raptor - Add

Add! Adds 5 more people of the currently selected gender to the page. A person is a `div` with the classes of `person` and either `boy` or `girl`.

```
<button id="add">Add!</button>
```

- HINT 1
- HINT 2

Exercise Solution

```
$(function() {
    prepopulate();
    $("#add").click(populate);
});

// Add! button event handler; adds 5 people of current gender
function populate() {
    var gender = getGender();
    for (var i = 0; i < 5; i++) {
        $('#people').append($('<div>').addClass('person ' + gender));
    }
}
```

Exercise : Raptor - Kill

Kill! Randomly "kills" 1/5 of the people of the currently selected gender. Kill them by giving them a class of `splat` (in addition to their existing `person` class, but in place of their gender class such as `boy` or `girl`).

```
<button id="kill">Kill!</button>
```

Exercise Solution

```
$(function() {
    prepopulate();
    $("#kill").click(kill);
});
// Get all guys or girls and splat one fifth of them
function kill(gender) {
    var peeps = $('#people .' + gender);
    for (var i = 0; i < peeps.length / 5; i++) {
        var randomIndex = Math.floor(Math.random() * peeps.length);
        $(peeps[randomIndex]).removeClass(gender);
        // so future kills won't choose splat victims
        $(peeps[randomIndex]).addClass('splat');
    }
}
```

Exercise : Raptor - Clean Up

Clean Up! Removes any dead splatted people from the page (any `div`s with class `splat`).

```
<button id="cleanup">Clean up!</button>
```

- HINT: You can remove an element from the page by calling jQuery's `remove()` method.

Exercise Solution

```
$(function() {
    prepopulate();
    $("#cleanup").click(clearDead);
});

// Clean up the dead! button event handler
function clearDead() {
    $("#people .splat").remove();
}
```

Exercise : Raptor - Stomp

Stomp! Makes the raptor move up or down by 75px and also kills 1/5 of both genders. The raptor is an `img` tag with an `id` of `raptor`.

```
<button id="stomp">Stomp!</button>
```

- HINT 1: Move the raptor by setting his `top` style attribute to be either `10px` or `85px`.
- HINT 2: You can find out an object's existing style properties by calling jQuery's `css` method.

Exercise Solution

```
$(function() {
    prepopulate();
    $("#stomp").click(stomp);
});

// Stomp! button event handler
function stomp() {
    $('#raptor').css('top', function(idx, old) {
        return ((parseInt(old) + 75) % 150) + 'px';
    });
    splat('boy');
    splat('girl');
}
```

Exercise : Raptor - Enrage

Enrage! Applies the CSS class of `enrage` to the raptor and the page's top `h1` heading. In addition, the raptor should be made to be 50px wider than his current width. Clicking the button again removes the class from both elements and returns the width to its previous value.
The `h1` has an existing CSS class that should not be removed. You are guaranteed that there is exactly one `h1` element on the page.

```
<button id="enrage">Enrage!</button>
```

Exercise Solution

```
// Enrage! button event handler
function enrageRaptor() {
    // If enraged -- go back to normal, else get ENRAGED
    if ($('#raptor').hasClass('enrage')) {
        $('h1').first().removeClass('enrage');
        $('#raptor')
            .removeClass('enrage')
            .width(function(idx, old) {
                return old - 50;
            });
    } else {
        $('h1').first().addClass('enrage');
        $('#raptor')
            .addClass('enrage')
            .width(function(idx, old) {
                return old + 50
            });
    }
}
```

Exercise : Raptor - Patrol

Patrol! (advanced) Makes the raptor animate. He should move right by 4px every 20ms until his `left` position style is at least `300px`, he should change directions and start patrolling to the left until his `left` position is `10px` or less, at which point he stops patrolling.

```
<button id="patrol">Patrol!</button>
```

- HINT: Set a timer with the `setInterval` method.

Exercise Solution

```
// Patrol! event handler code (advanced)
var timer;

function patrol() {
    clearInterval(timer);
    timer = setInterval(patrolRight, 20);
}

function patrolRight() {
    $('#raptor').css('left', function(idx, old) {
        if (parseInt(old) >= 300) {
            clearInterval(timer);
            timer = setInterval(patrolLeft, 20);
        }
        return parseInt(old) + 4 + 'px'
    });
}
```

Exercise Solution

```
function patrolLeft() {
    $('#raptor').css('left', function(idx, old) {
        if (parseInt(old) <= 10) {
            clearInterval(timer);
            $(this).css({
                'top': '5px',
                'left': '10px'
            });
        }
        return parseInt(old) - 4 + 'px';
    });
}
```