# CS2204 Fundamentals of Internet Applications Development

## Lecture 10 JavaScript – Part5

*Computer Science, City University of Hong Kong*

*Semester B 2024-25*

# CW2 Reminder (Due on Apr 18th)

CW2 takes **13%** of the course assessment and will be due in two weeks on **April 18th, 23:59 PM**

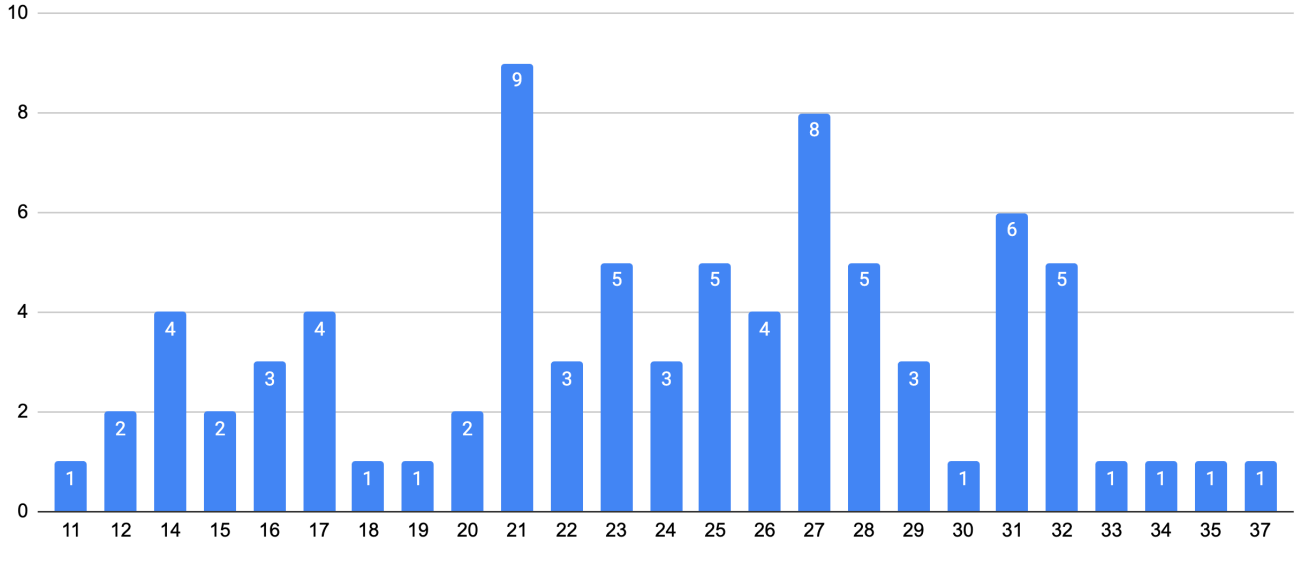
Please **start as soon as possible** as they are 6 tasks in this assignment!

If you have any questions about assignment requirement, please **contact the student helpers for clarification.**

If your question is about how to handle a specific task or debugging, the helpers are not supposed to directly help you with that, but they may provide some hints if applicable.

# Mid-term results are out on Canvas

The average score is 23.99 out of 40. For questions with high error rate, they will be covered in the last lecture



# Final Exam Schedule: May 10th (Sat)

Time: 14:00 -16:00 PM
Venue: Check your AIMS

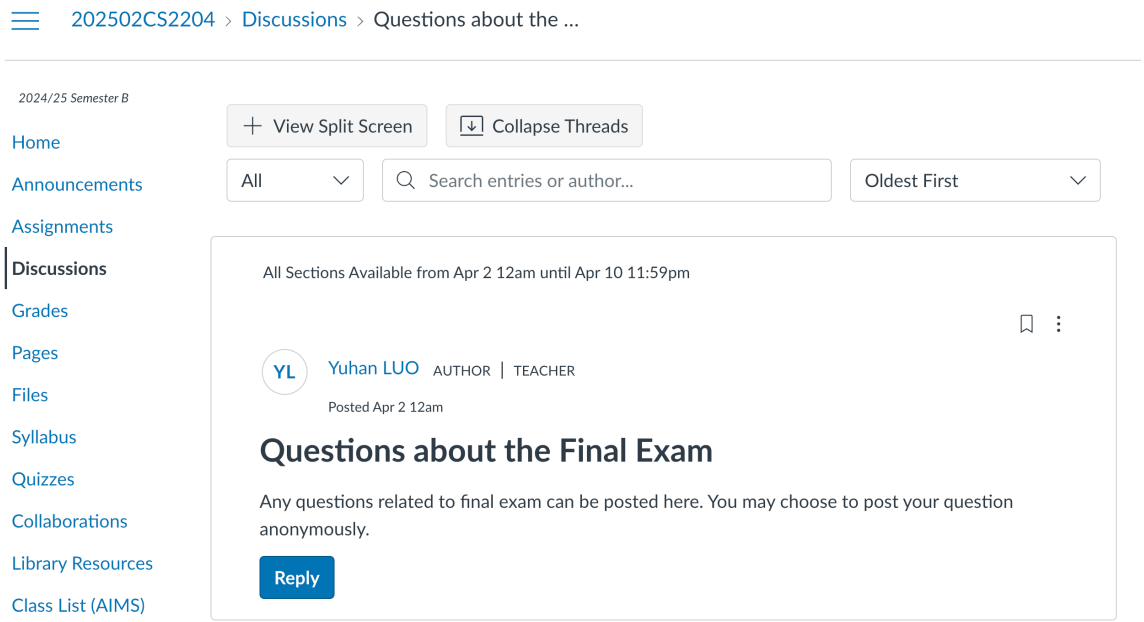
Coverage: all the lectures and labs
Form:

- Close-book
- All questions are open-ended - some will ask you to complete a piece of code or write the output of a program; for questions involving specific JS functions, there will be hints for the function usage.

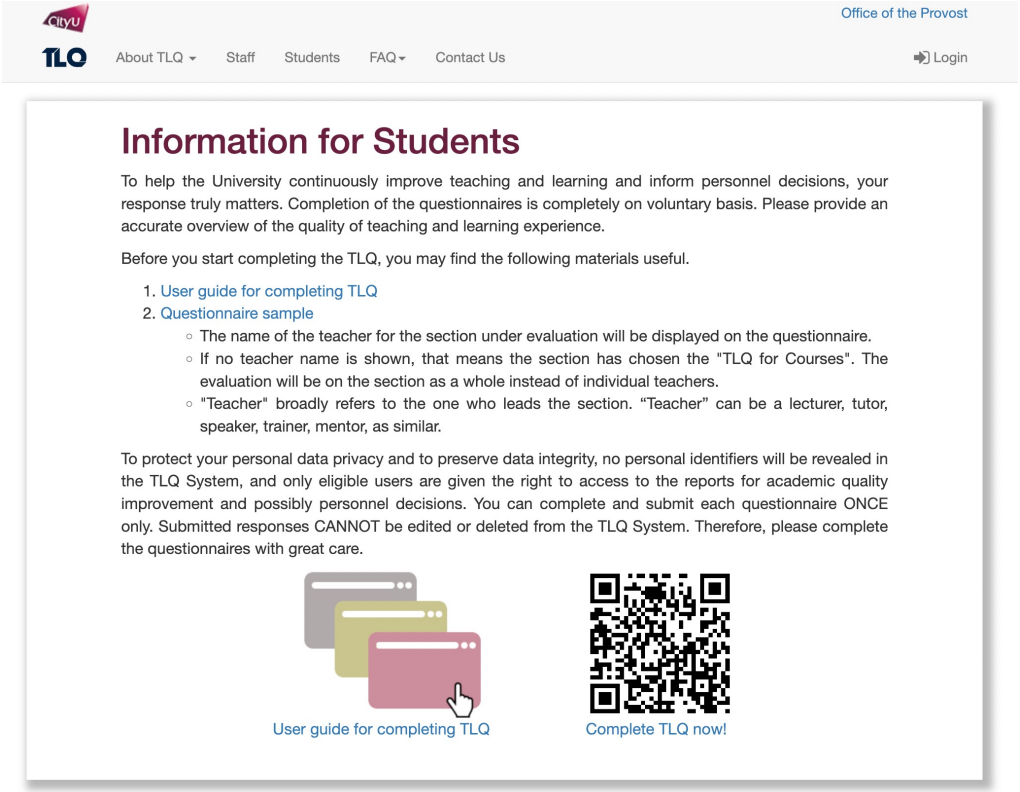
# Post your questions on Canvas before the last lecture (April 17th)

A discussion thread is created to collect your questions about the final exam.

The questions will be covered during the last lecture



# LOQ Reminder




https://onlinesurvey.cityu.edu.hk/loq/

# Post-lab Quiz 7 Review

Given the JavaScript code below, enter "true" or "false" in the following statements.

The statement "console.log (err);" within the catch block will display an error message

```
<script>
    const a = "My name is: ";
    var b = "Bob", c = "Ann", d = "";
    function f() {
      try {
        d = a + c;
        a += b;
      } catch (err) {
        alert(err);
      }
    }
    f();
    var s = d ? "My name is Ann" :
    "There is an error in the
    program";
    alert(s);
</script>
```

# Post-lab Quiz 7 Review

Given the JavaScript code below, enter "true" or "false" in the following statements.

If we switch the order of the statements "d = a + c; " and "a += b;", the output of this program remains unchanged.

```
<script>
    const a = "My name is: ";
    var b = "Bob", c = "Ann", d = "";
    function f() {
      try {
        d = a + c;
        a += b;
      } catch (err) {
        alert(err);
      }
    }
    f();
    var s = d ? "My name is Ann" :
    "There is an error in the
    program";
    alert(s);
</script>
```

# Question time: event handler

1. What are the major difference between regular `eventHandler` and `addEventListener`?

2. What are some different ways to cancel an event handler?

# Question time: dynamic content (1)

1. In creating dynamic content, what are the major difference between `createElement` and directly using `innerHTML`?

2. How to improve the efficiency when using `innerHTML` to generate dynamic content?

# Question time: dynamic content (2)

1. What are some different ways to control the visibility of HTML elements using JS?

2. How to periodically call a function?

# Agenda

Review: Dynamic Content

Multimedia:

- Video and audio
- Animation

Programming Exercise

# Agenda

Review: Dynamic Content

Multimedia:

- Video and audio
- Animation

Programming Exercise

# Review: Dynamic Content

Common ways to **dynamically control and manipulate contents**:

Add/delete elements

- createElement
- innerHTML property
- inline script

Show or hide

- display
- visibility
- z-index
- Positioning

`setInterval()` and `clearInterval()`

# Review: InnerHTML vs createElement

## Comparing the efficiency of three functions

```
function handler1() {
    var t1 = new Date();
    for (var i=0; i<500; i++) {
        var ls = document.createElement('li');
        o1.appendChild(ls);
    }
    var t2 = new Date();
    var time = t2 - t1;
    console.log(time);
    btn1.innerHTML = time;
}
```

Creating each of the 500 elements `<li>` one by one

`createElement('li')` is **faster** because it directly manipulates DOM without the need to locating and modifying existing elements

```
var btn2 = document.querySelector('#btn2');
var o2 = document.querySelector('#o2');
btn2.onclick = handler2;

function handler2() {
    var t1 = new Date();
    for (var i=0; i<500; i++) {
        o2.innerHTML += '<li></li>';
    }
    var t2 = new Date();
    var time = t2 - t1;
    console.log(time);
    btn2.innerHTML = time;

}
```

Inserting and displaying 500 `<li>` one by one

```
var btn3 = document.querySelector('#btn3');
var o3 = document.querySelector('#o3');
btn3.onclick = handler3;

function handler3() {
    var t1 = new Date();

    var arr = [];
    for (var i=0; i<500; i++) {
        arr.push('<li></li>');
    }
    o3.innerHTML = arr.join('');

    var t2 = new Date();
    var time = t2 - t1;
    console.log(time);
    btn3.innerHTML = time;
```

Adding 500 `<li>` to an array one by one and **display them all at once together**

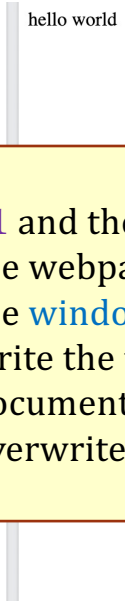
Code Example: lec10-01-JS-innerHTML3.html

# Review: Inline Script

HTML will be created at the location, where your scripts are running

It will cause the page to re-render if the page is fully loaded

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>
        Inline Script Example
    </h1>

    <button>Click</button>

    <script>
        var btn = document.querySelector('button');
        btn.onclick = function() {
            document.write('cs2204');
        }

        window.onload = function() {
            document.write('hello world');
        }
    </script>

</body>
</html>
```




h1 and the button “Click” is not shown on the webpage because the window.onload event listener is used to write the text "hello world" to the document after it has loaded, which overwrite the entire document



Code Example: Lec10-02-JS-inline-script2.html

# Review: Hide & Show (1)

JS allows you to change the CSS properties of an element

- You can make objects appear or disappear by changing the display property, e.g., display
- There are other properties that we can change, e.g., backgroundcolor, etc.

```
63  <!-- Dynamic content one -->
64      <div id="sweet">
65      <p>
66          <img src="../images/1Nail.jpg" alt="">
67          <img src="../images/2Nail.jpg" alt="">
68          <img src="../images/3Nail.jpg" alt="">
69      </p>
70      </div>
71  <!-- Dynamic content two -->
72      <div id="sour">
73      <p> <img src="../images/4Nail.jpg" alt="">
74          <img src="../images/5Nail.jpg" alt="">
75      </p>
76      </div>
```

```
36  <script type="text/javascript">
37  function show(index) {
38      if (index == 1) {
39          document.getElementById("sweet").style.display="block";
40          document.getElementById("sour").style.display="none";
41      }
42      else {
43          document.getElementById("sweet").style.display="none";
44          document.getElementById("sour").style.display="block";
45      }
46  }
47  </script>
```

Show as a block element

Code Example: Lec10-03-JS-dynamic-content.html



Display different contents when selecting a different link

# Review: Hide & Show (2)

## Other methods

**Visibility property**

```
object.style.visibility="hidden"
object.style.visibility="visible"
```

**`Z-index`** - prepare all elements and stack them up with overlapping; change z-index to show either one by putting it on top.


```
<!DOCTYPE html>
<html>
<head>
<style>
#img1 {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1
}
</style>
</head>
<body>

<h1>This is a Heading</h1>

<img id="img1" src="book.png" width="100" height="180">

<button type="button" onclick="changeStack()">Change stack order</button>

<p>Default z-index is 0. Z-index -1 has lower priority.</p>

<p id="myP">This is a p element.</p>

<button type="button" onclick="hide()">Hide img</button>
<button type="button" onclick="show()">Display img</button>

<script>
function hide() {
  document.getElementById("img1").style.visibility = "hidden";
}

function show() {
  document.getElementById("img1").style.visibility = "visible";
}

function changeStack() {
  if (document.getElementById("img1").style.zIndex != "1")
    document.getElementById("img1").style.zIndex = "1";
  else
    document.getElementById("img1").style.zIndex = "-1";
}
</script>

</body>
</html>
```

Code Example: Lec10-04-JS-hidet.html

# Review: `setInterval()` Example

Can we move this variable declaration `var myInterval;` into `function startTimer()`?

```
<script>
var myInterval;

function startTimer() {
   myInterval = setInterval(myTimer, 1000);
}


function myTimer() {
  const date = new Date();
  document.getElementById("demo").innerHTML =
  date.toLocaleTimeString();
}

function stopTimer(){
  clearInterval(myInterval);
}
</script>
```


```
<!DOCTYPE html>
<html>
<body>

<h2>setInterval() and clearInterval()</h2>

<p id="demo">Time will show up here</p>
<button onclick="startTimer()">Start</button>
<button onclick="stopTimer()">stop</button>
```

Code Example: Lec10-05-JS-interval.html

# Agenda

Review: Dynamic Content

Multimedia:

- Video and audio
- Animation

Mid-term Question Review

# Multimedia Programming

Before HTML5, Image is the only media that can be scripted

In HTML5, **video & audio scripting APIs** are provided in addition to the usual dynamic content techniques

In addition, we can create more complicated animation effects (e.g., creating slide show, movement and layering)

# Video

Video & audio are timed media: have to be played over time

HTML5 provides many useful video properties &

```
.duration
.control
.oncanplay
.onended
```

And methods:

```
load()
play()
pause() // preferred over stop() as a W3C standard
```

# Video example

**Oncanplay** attribute of video element execute a JavaScript code when a video is ready to start playing

```
<script>
    var v;

    function init() {
        v = document.getElementById("v");
    }

    function initButton() {
        document.getElementById('b').style.visibility = 'visible';
    }

    function vplay() {
        v.play();
    }

    function vpause() {
        v.pause();
    }

    function vstop() {
        v.currentTime = 0;
        v.pause();
    }

    function vff() {
        v.currentTime += v.duration / 10;
        if (v.currentTime >= v.duration) {
            v.currentTime = 0;
        }

    }

    function vfb() {
        v.currentTime -= v.duration / 10;
        if (v.currentTime < 0) {
            v.currentTime = 0;
        }
    }
</script>
```

```
<body onload="init();">
    <div id="container">
        <video id="v" oncanplay="initButton();">
            <source src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.mp4" type="video/mp4">
            <source src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.ogg" type="video/ogg">
        </video>
        <div id="b">
            <button onclick="vplay();">play</button>
            <button onclick="vstop();">stop</button>
            <button onclick="vpause();">pause</button>
            <button onclick="vff();">fast forward</button>
            <button onclick="vfb();">fast backward</button>
</body>
```

use methods to play and pause the video

stop is to set the play time to the beginning and pause

**fast forward:** add `v.duration/10` to current play time, reset to 0 if larger than duration

**fast backward**: minus `v.duration/10` to the current play time, reset to 0 if smaller than 0


Code Example: Lec10-06-JS-script-video.html

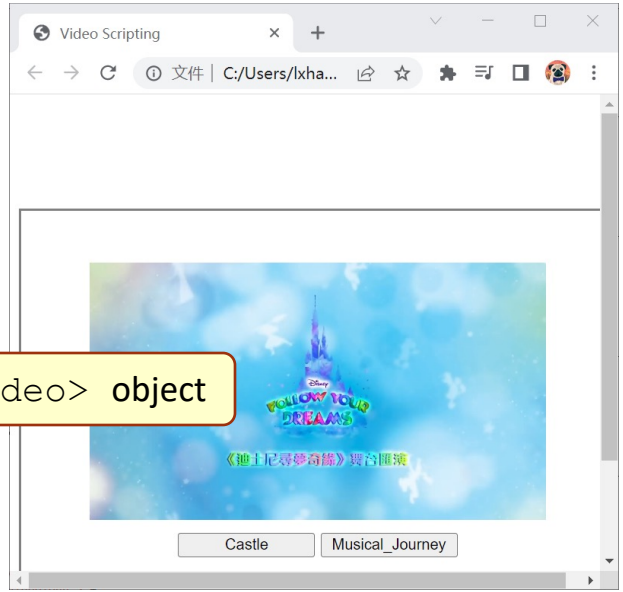
# Video example (Cont.)

How to **switch** videos by JS?

- by manipulating different HTML and their properties/attributes

```
        function init() {
            v = document.getElementById("v");
        }
        function initButton() {
            document.getElementById("b").style.visibility = "visible";
        }

        function switchVideo(n) {
            if (n%2 == 0) {
                document.getElementById('vdiv').innerHTML = '<video id="v"
oncanplay="initButton();" autoplay>' + '<source
src="https://personal.cs.cityu.edu.hk/~cs2204/video/Musical_Journey.mp4"
type="video/mp4">' + '<source
src="https://personal.cs.cityu.edu.hk/~cs2204/video/Musical_Journey.ogg"
type="video/ogg">' + '</video>';
            } else {
                document.getElementById('vdiv').innerHTML = '<video id="v"
oncanplay="initButton();" autoplay>' + '<source
src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.mp4" type="video/mp4">' +
'<source src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.ogg"
type="video/ogg">' + '</video>';
            }
        }
    </script>
</head>

<body onload="init();">
    <div id="container">
        <div id="vdiv">
            <video id="v" oncanplay="initButton();" autoplay>
                <source src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.mp4"
type="video/mp4">
                <source src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.ogg"
type="video/ogg">
            </video>
        </div>
        <div id="b">
            <button onclick="switchVideo(1);">Castle</button>
            <button onclick="switchVideo(2);">Musical_Journey</button>
        </div>
    </div>
</body>
```

Update the `innerHTML` arrtibute of the `<video>` object




Code Example: Lec10-07-JS-switch-video.html

# Audio

**Audio** is similar to video, except that it does not have a visual component

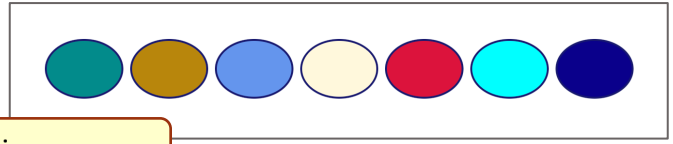
- audio can exist as a JS object **without** the `<audio>` tag
  - e.g., `var a= new Audio();`
  - especially useful for storing multiple audios for different events

```
41    <script type="text/javascript">
42    var audio=new Array();
43    var audioFile= new Array("a.wav","b.wav","c.wav","d.wav","e.wav","f.wav","g.wav");
44    //pre-load audio files
45    for (i=0; i<audioFile.length; i++) {
46      audio[i]= new Audio("../audio/"+audioFile[i]);
47      audio[i].load();
48    }
49    function init() {
50      //play a round of notes
51      for (i=0; i<audioFile.length; i++) {
52        audio[i].play();
53      }
54    }
55    function playAudio(m) {
56      audio[m].play();
57    }
58    </script>
```

storing many audio objects in an array

creating the audio object with a URL

Pre-loading the audio files so they could be played faster



```
60    <body onload="init();">
61      <div id="container">
62        <div id="a" onmouseover="playAudio(0)" class="note">
63        </div>
64        <div id="b" onmouseover="playAudio(1)" class="note">
65        </div>
66        <div id="c" onmouseover="playAudio(2)" class="note">
67        </div>
68        <div id="d" onmouseover="playAudio(3)" class="note">
69        </div>
70        <div id="e" onmouseover="playAudio(4)" class="note">
71        </div>
72        <div id="f" onmouseover="playAudio(5)" class="note">
73        </div>
74        <div id="g" onmouseover="playAudio(6)" class="note">
75        </div>
76      </div>
77    </body>
78  </html>
```

Code Example: Lec10-08-JS-switch-audio.html

# Basic slide show (1)

Showing images one by one at a certain interval of time

Two ways to execute JavaScript periodically

- **`setTimeOut(function, interval)`**
  - execute the function/code after interval millisecs
  - to execute repeatedly, need to use the method recursively in a function

```
3   window.onload = rotate;
4
5   var adImages = new Array("../images/1Nail.jpg","../images/2Nail.jpg","../images/3Nail.jpg",
6                            "../images/4Nail.jpg", "../images/5Nail.jpg");
7   var thisAd = 0;
8
9   function rotate() {
10      thisAd++;
11      if (thisAd == adImages.length) {
12          thisAd = 0;
13      }
14      document.getElementById("adBanner").src = adImages[thisAd];
15  //using recursive, can be done using setInterval as well
16      setTimeout("rotate()", 3 * 1000);
17  }
```

1.1. Store all the images in an array

1.2. Start with the first image

2.1. Iterate over all the images in the array; if it reaches the end of the array, starts from the first image

2.2. Replace the image source during each iteration

2.3. Periodically call rotate () every 3 secs

Code Example: Lec10-09-JS-slide-show-1.html

# Basic slide show (2)

Showing images one by one at a certain interval of time

Two ways to execute JavaScript periodically

- **`setInterval(function, interval)`**
  - e.g., `setInterval("myfunction( )", 500)` executes the function `myfunction` every 500 milliseconds

```
<script>
    window.onload = rotate;

    var adImages = new Array("./images/1Nail.jpg", "./images/2Nail.jpg", "./images/3Nail.jpg",
        "./images/4Nail.jpg", "./images/5Nail.jpg");
    var thisAd = 0;

    setInterval('rotate()', 1*1000);

    function rotate() {

        thisAd++;
        if (thisAd == adImages.length) {
            thisAd = 0;
        }

        document.getElementById('adBanner').src = adImages[thisAd];
    }

</script>
```

Code Example: Lec10-10-JS-slide-show-2.html

# Sequenced slide show

Show a sequence of images at a certain speed

- same as slide show
- require a sequence of action images
- `setInterval` or `setTimeout` again

Java Duke images from M. Hall, L. Brown - Core Web Programming (2nd Ed.)

```
20  <script type="text/javascript">
21  //similar to body onload but it is pure javascript event handling using window.onload
22  window.onload = rotate;
23  var thisAd = 0;
24  function rotate() {
25      thisAd++;
26      if (thisAd >14) {
27          thisAd = 0;
28      }
29      document.getElementById("adBanner").src = "../images/T" + thisAd + ".gif";
30  }
31  //using setInterval call function every 1000/5 sec
32  setInterval("rotate()", 1000/5);
33  </script>
34  </head>
35  <body>
36  <img src="../images/T0.gif" id="adBanner" alt="Ad Banner" >
37  <h4>Java Duke images from M. Hall, L. Brown - Core Web Programming (2nd Ed.)</h4>
38  </body>
39  </html>
```

**Question time:**
What if we want to make the character move faster?

Animation speed (frames per second) determined by the time interval

Code Example: Lec10-11-JS-animation.html

# Dynamic Styling

Update the the element styling, including background color, text color, and font size, in Radom values


```
<body>
    <div id="dynamicDiv">Click the button to change styles!</div>
    <button id="changeStyleBtn">Change Style</button>

    <script>
        document.getElementById('changeStyleBtn').addEventListener('click', function() {
            const div = document.getElementById('dynamicDiv');
            // Generate random color and front sizes, and assign them as new styles
            div.style.backgroundColor = '#' + Math.floor(Math.random()*16777215).toString(16);
            div.style.color = '#' + Math.floor(Math.random()*16777215).toString(16);
            div.style.borderRadius = Math.floor(Math.random() * 50) + 'px';
            div.style.fontSize = Math.floor(Math.random()*30+10) + 'px'; // Change font size
        });
    </script>
</body>
```

How to change the click event to a hover effect?

What does **16777215** stands for?



Code Example: Lec10-12-JS-dynamic-styling.html

# Dynamic Styling: changing event

We can use `addEventListener('mouseover', function(){…});` to change the trigger event

Click the button to change styles!

This has the same effects as `div.onmouseover = functionName;` or adding an attribute `"onmouseover = functionName();"` in HTML

```
<bod
    <div id="dynamicDiv">Click the button to change styles!</div>

    <script>
        document.getElementById('dynamicDiv').addEventListener('mouseover', function() {
            const div = document.getElementById('dynamicDiv');
            // Generate random color and front sizes, and assign them as new styles
            div.style.backgroundColor = '#' + Math.floor(Math.random()*16777215).toString(16);
            div.style.color = '#' + Math.floor(Math.random()*16777215).toString(16);
            div.style.borderRadius = Math.floor(Math.random() * 50) + 'px';
            div.style.fontSize = Math.floor(Math.random()*30+10) + 'px'; // Change font size
        });
    </script>
</body>
```

What if we want the div to change back to its previous state when mouse is out?



Code Example: Lec10-12-JS-dynamic-styling.html

# Dynamic Styling: changing event

To restore the previous states, we need to save the initial state value to some temporary variables, and then set the values back when **'mouseout'**

Hover over me to change styles!

```
<script>
    const div = document.getElementById('dynamicDiv');

    // Store initial styles
    const initialStyles = {
        backgroundColor: div.style.backgroundColor,
        color: div.style.color,
        borderRadius: div.style.borderRadius,
        fontSize: div.style.fontSize
    };

    div.addEventListener('mouseover', function() {
        // Generate random color and font sizes, and assign them as new styles
        div.style.backgroundColor = '#' + Math.floor(Math.random() * 16777215).toString(16);
        div.style.color = '#' + Math.floor(Math.random() * 16777215).toString(16);
        div.style.borderRadius = Math.floor(Math.random() * 50) + 'px';
        div.style.fontSize = Math.floor(Math.random() * 30 + 10) + 'px'; // Change font size
    });

    div.addEventListener('mouseout', function() {
        // Restore initial styles
        div.style.backgroundColor = initialStyles.backgroundColor;
        div.style.color = initialStyles.color;
        div.style.borderRadius = initialStyles.borderRadius;
        div.style.fontSize = initialStyles.fontSize;
    });
```

We can create individual variables or use an object to store these variables, as they conceptually belong to **"initialStyles"**

Re-assign the initial styles.

Code Example: Lec10-13-JS-dynamic-styling-2.html

# Movement & Layering

**Movement** is a continuous change of positions. Positioning schemes other than normal flow use CSS properties: left, right, top, bottom for precise control of positions.

Different scenes can be stacked up with overlapping using z-index and non-static positioning schemes. They can then changed rapidly by changing the z-index.

```
<script>
    var up, current_xPos, count, dot, moveright;
    function init() {
        up = -80;
        moveup = -80;
        current_xPos = 1;
        moveright = 1;
        dot = document.getElementById("dot");
        setInterval("move()", 25);
    }
    function move() {
        current_xPos = current_xPos + moveright;
        if (current_xPos > 400)
            moveright = -1;
        }
        if (current_xPos < 1) {
            moveright = 1;
        }
        if (current_xPos > 400 || current_xPos < 1) {
            up = up + moveup;
            if (up < -400) moveup = 80;
            if (up > -80) moveup = -80;
        }
        dot.style.top = up + "px";
        dot.style.left = current_xPos + "px";
    }
</script>
```

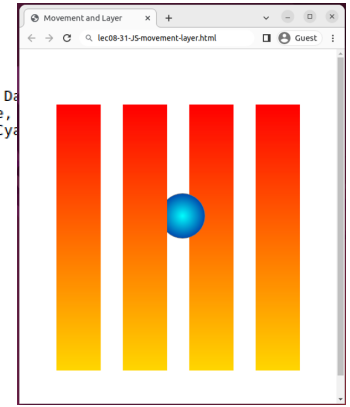
Periodically moving

In every iteration, `current_xPos` either increases by 1 or decreases by 1

```
.odd {
    position: relative;
    z-index: 0;
}
.even {
    position: relative;
    z-index: 2;
}
#dot {
    border-radius: 80px;
    -webkit-border-radius: 80px;
    -moz-border-radius: 80px;
    border: solid 1px grey;
    background: radial-gradient(circle, Cyan, Da
    background: -webkit-radial-gradient(circle,
    background: -moz-radial-gradient(circle, Cya
    clear:both;
    width: 80px;
    height: 80px;
    z-index: 1;
    position: relative;
}
```

the dot and the alternate vertical bars have different z-index

relative positioning is used for the circle




The dot moves from left to right, bottom to top, passing on top and below of vertical bars

Code Example: Lec10-14-JS-movement-layer.html

# Movement Tracking

Movement using position schemes

- fixed, absolute & relative

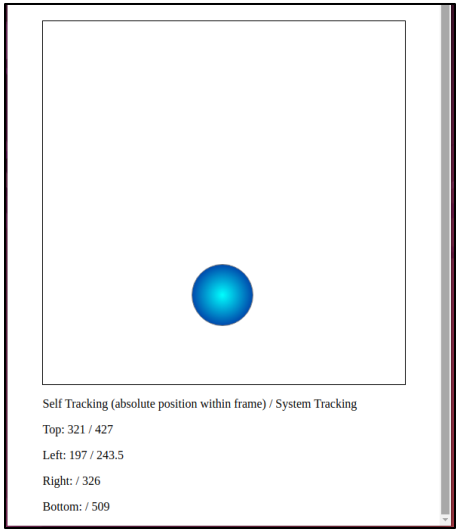
All involve manipulation of top, left, right & bottom properties

keep track of these values can determine object location

usually done with self-defined JavaScript variables

```
<script type="text/javascript">
var up, current_xPos, count, dot, moveright;
function init() {
    up=1;
    moveup=80;
    current_xPos=1;
    moveright=1;
    dot=document.getElementById("dot");
    setInterval("move()", 25);
}
function move() {
    current_xPos=current_xPos+moveright;

    if (current_xPos > 400) {
        moveright=-1;
    }
    if (current_xPos < 1) {
        moveright=1;
    }
    if (current_xPos > 400 || current_xPos < 1) {
        up=up+moveup;
        if (up < 80) moveup=80;
        if (up > 400) moveup=-80;
    }
    dot.style.top=up+"px";
    dot.style.left=current_xPos+"px";
    box=dot.getBoundingClientRect();
    document.querySelector('#top').innerHTML=up+" / "+Math.round(box.top,1);
    document.querySelector('#left').innerHTML=current_xPos+" / "+box.left;
    document.querySelector('#right').innerHTML=" / "+Math.round(box.right,1);
    document.querySelector('#bottom').innerHTML=" / "+Math.round(box.bottom,1);
}
</script>
```

```
#container {
   border: solid 1px Black;
   margin: 10px auto;
   width: 480px;
   height: 480px;
   text-align: center;
   position: relative;
}
#position {
   width: 480px;
   margin: 5px auto;
}
#dot {
    border-radius: 80px;
    -webkit-border-radius: 80px;
    -moz-border-radius: 80px;
    border: solid 1px grey;
   background: radial-gradient(circle, Cyan, Darkblue);
   background: -webkit-radial-gradient(circle, Cyan, Darkblue);
   background: -moz-radial-gradient(circle, Cyan, Darkblue);
    clear:both;
    width: 80px;
    height: 80px;
    z-index: 1;
    position: absolute;
}
```




- `getBoundingClientRect()` return box as an object with properties `box.top`, `box.left`, `box.right` & `box.bottom` `relative` to the viewport
- values are relative to window, require adjustment if page scrolled

Code Example: Lec10-15-JS-movement-tracking.html

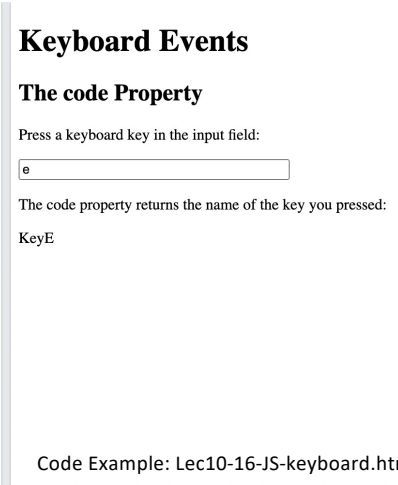
# Keyboard Handling

Frequently used in gaming: which key has been **pressed** or **released**

More than one events are triggered

- `onkeydown`: triggered when a key **is first pressed down**
- `onkeypress` : triggered **between the key is pressed down and released**
- `onkeyup`: triggered when a key is **released**
- In some browsers, some keys won't fire this event: alt, ctrl, shift, esc

```
<!DOCTYPE html>
<html>
<body>
<h1>Keyboard Events</h1>
<h2>The code Property</h2>

<p>Press a keyboard key in the input field:</p>

<input type="text" size="40"
onkeypress="myFunction(event)">

<p>The code property returns the name of the key you
pressed:</p>

<p id="demo"></p>

<script>
function myFunction(event) {
  let key = event.code;
  document.getElementById("demo").innerHTML = key;
}
</script>

</body>
</html>
```




Code Example: Lec10-16-JS-keyboard.html

# Collision Detection

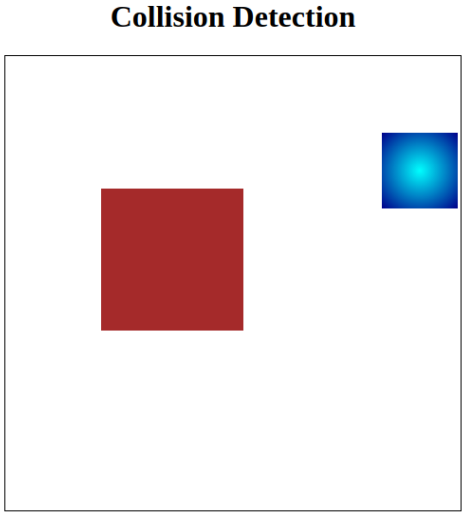
Object collision is when two moving objects start to overlap, could be complicated because it can occur in different directions and objects can have different shapes

```
function insidebigRect(x,y) {
    if ((x > bigRectleft && x < bigRectleft+150) &&
      (y > bigRecttop && y < bigRecttop+150)) {
      return true
    }
    else {
      return false
    }
}
function move() {
    current_xPos=current_xPos+moveright;

    if (current_xPos > 400) {
        moveright=-1;
    }
    if (current_xPos < 1) {
        moveright=1;
    }
    if (current_xPos > 400 || current_xPos < 1) {
        up=up+moveup;
        if (up < 80) moveup=80;
        if (up > 400) moveup=-80;
    }
    dot.style.top=up+"px";
    dot.style.left=current_xPos+"px";
    //show tracking
    box=dot.getBoundingClientRect();
    document.querySelector('#top').innerHTML=up+" / "+Math.round(box.top,1);
    document.querySelector('#left').innerHTML=current_xPos+" / "+box.left;
    document.querySelector('#right').innerHTML=" / "+Math.round(box.right,1);
    document.querySelector('#bottom').innerHTML=" / "+Math.round(box.bottom,1);
    //check collision
    if (insidebigRect(current_xPos,up) || insideb
       || insidebigRect(current_xPos,up+80) || in
      bangsound.play();
      bigRect.style.opacity=0.5;
      clearInterval(moveid);
    }
}
```

check whether a particular position (x, y) is within the bigRect's area

use the tracked locations of the small rect to do the comparison

Pass all the four corners of the small rect to `insidebigRect ()`




Code Example: Lec10-17-JS-collision-detection.html

# Drag & Drop

Two key components: what to drag and where to drop

- `ondragstart` attribute specifies what data to be dragged and calls a function
- `dataTransfer.setData()` sets the data type and the value of the dragged data
- `ondragover` event specifies where the dragged data can be dropped
- `event.preventDefault()` used for prevent the default (not allow for drag)

```
<script type="text/javascript">
    //function called when drag starts
    function drag(event) {
        theEvent = event;
        //tell the browser what to drag - id of the element
        theEvent.dataTransfer.setData("Text", theEvent.target.id);
    }
    //function called when element drops
    function drop(event) {
        theEvent = event;
        //get a reference to the element being dragged - id
        var theData = theEvent.dataTransfer.getData("Text");
        //get the element
        var theDraggedElement = document.querySelector("#" +
        //add it to the drop element
        theEvent.target.appendChild(theDraggedElement);
    }
    //function called to allow drop ondragover
    function allowDrop(event) {
        theEvent = event;
        //allow drop - default is not
        theEvent.preventDefault();
    }
    function init() {
        var which = Math.floor(Math.random() * 3);
        document.querySelectorAll(".drop")[which].ondragover = allowDrop;
        document.querySelectorAll(".drop")[which].ondrop = drop;
    }
</script>
```

```
<div id="container">
    <div id="drag">
        <img id="dragimg" alt="card to drag" src="./images/s1.gif"
draggable=true ondragstart="drag(event);">
    </div>
    <div class="drop" id="drop1">
    </div>
    <div class="drop" id="drop2">
    </div>
    <div class="drop" id="drop3">
    </div>
```

the format parameter is always set to "text" by default is that this is the most widely supported format across different browsers and platforms. The parameter could also be "image" or "url", etc

which is a randomly generated integer between 0 and 2 that determines which of the three drop target elements will be allowed to accept the dragged element during the drag and drop operation.

Guess which box can take the card ?

Code Example: Lec10-18-JS-drag-and-drop.html

# JS Summary (1)

JavaScript is an object-oriented programming languages. Objects can be the primitive data types or complex data structures.

Each object has its properties and methods (i.e., functions). Likewise, there are also built-in objects and self-created objects.

`this` refers to an object depending on how the object is being invoked

JavaScript interacts with the HTML elements through DOM (Document Object Model) by assigning/changing its properties and call its methods. We can select HTML elements in JavaScript by ID, tag names, CSS queries, and combinations of these methods

EventHandlers are a special type of functions that tells the web object how to react when an event occurs.

# JS Summary (2)

In JS, we can **self-define** functions or use **built-in** functions to implement the program. Commonly used built-in functions include `alert(), console.log()`

JS can dynamically create, edit, remove, and change the style and display of HTML elements on a website, either through directly manipulating the DOM or modifying the innerHTML property of the elements

To modify an element, JS first need to select the element. This can be be done by `getElementById()or` CSS queries. Both can be combined to select multiple elements

To execute a function at regular intervals, we can use bulit-in JS function `setInterval()` and cancel it by `clearInterval()`

In HTML5, videos and audios can be directly manipulated by JS

# Agenda

Review: Dynamic Content

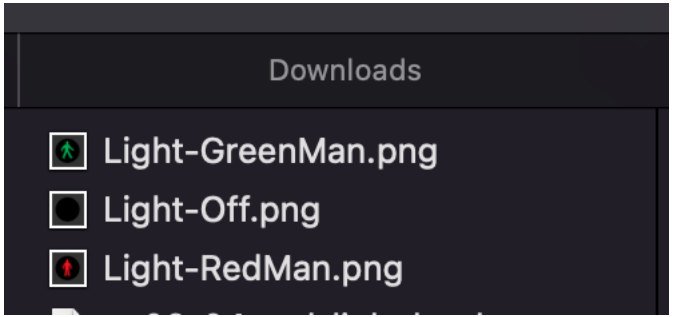
Multimedia:

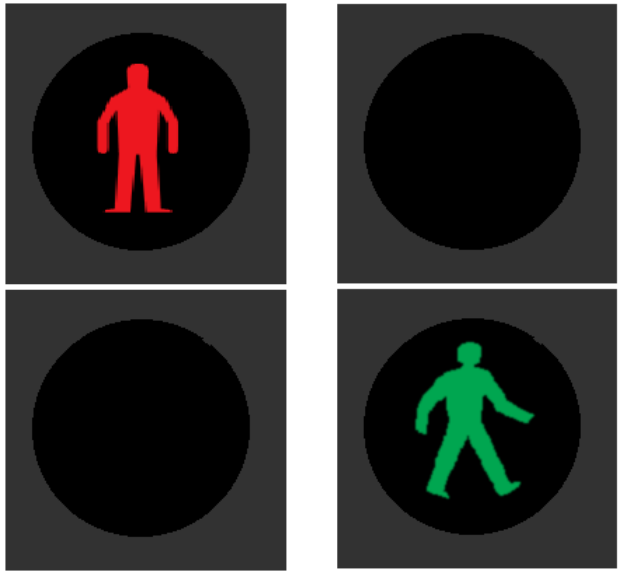
- Video and audio
- Animation

Programming Exercise

# Programming practice (1)

**Traffic light problem** We will create a webpage to show a pedestrian traffic light and write JavaScript programs to control it so that the traffic light can run automatically according to some timing events and the user can have some control over its status



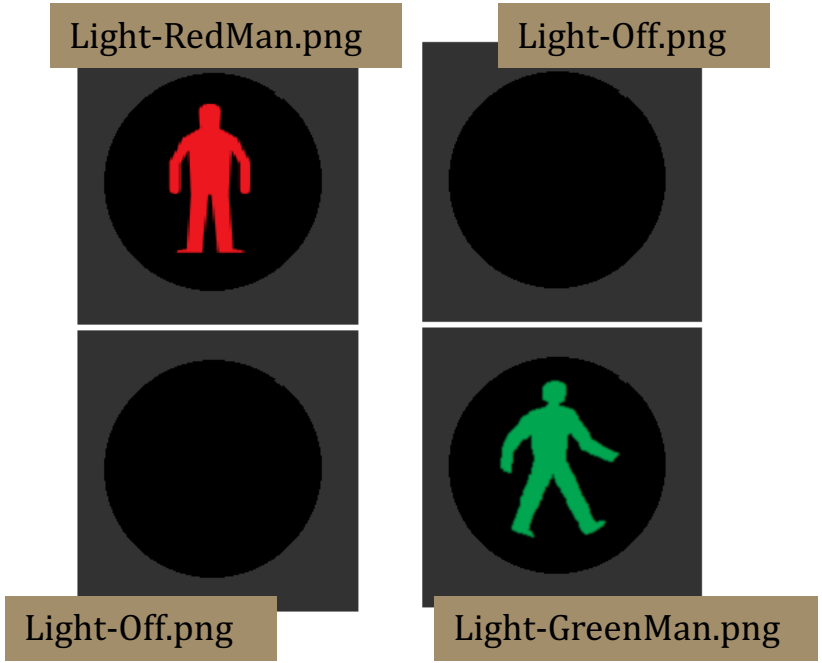



# Programming practice (2)

**Decompose the problem**

The question basically asks us to

1. Light-RedMan.png: to be shown on top when the red light is on
2. Light-GreenMan.png: to be shown at the bottom when the green light is on
3. Light-Off.png: to be shown on top or at the bottom when its corresponding light is not on



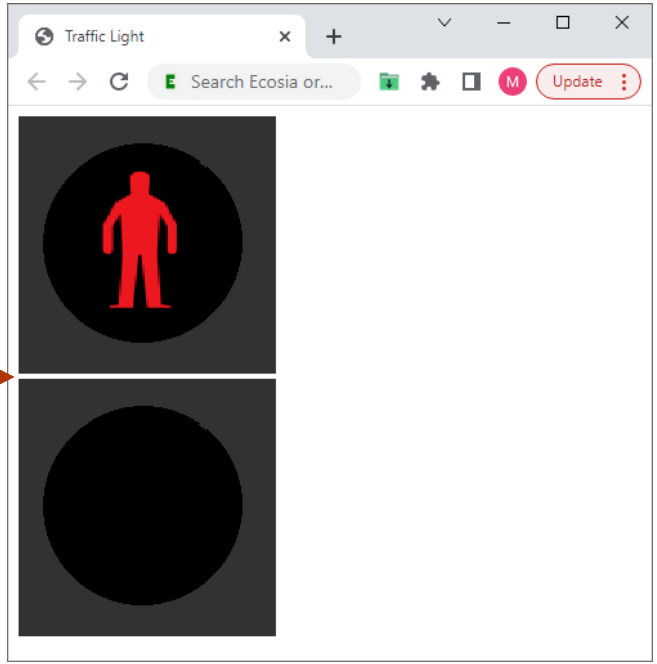
# Programming practice (3): Red light

- The following webpage shows a red light

```
<!DOCTYPE html>
<html>
  <head>
    <title>Traffic Light</title>
  </head>
  <body>
  <!-- Page content begins here -->
    <img id="toplight" src="Light-RedMan.png" /> <br/>
    <img id="bottomlight" src="Light-Off.png"  /> <br/>
  <!-- Page content ends here -->
  </body>
</html>
```

Code Example: red-light.html

A line break is added so that the next image will be displayed in a new line instead of on the right, since the default display style of img element is inline (instead of block)

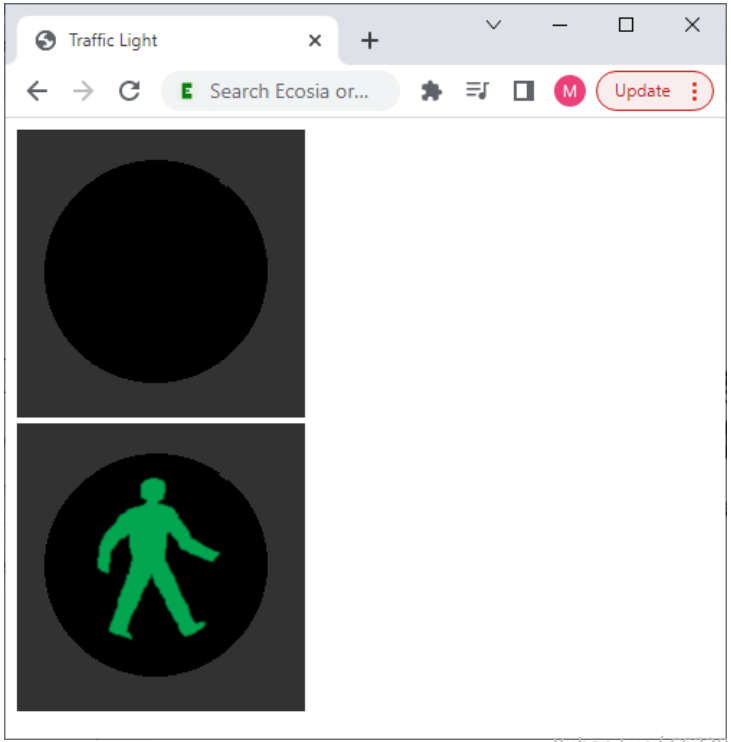


# Programming practice (4): Green Light

- Similarly, the following webpages show a green light

```
<!DOCTYPE html>
<html>
  <head>
    <title>Traffic Light</title>
  </head>
  <body>
  <!-- Page content begins here -->
    <img id="toplight" src="Light-Off.png" /> <br/>
    <img id="bottomlight" src="Light-GreenMan.png"  /> <br/>
  <!-- Page content ends here -->
  </body>
</html>
```

Code Example: green-light.html



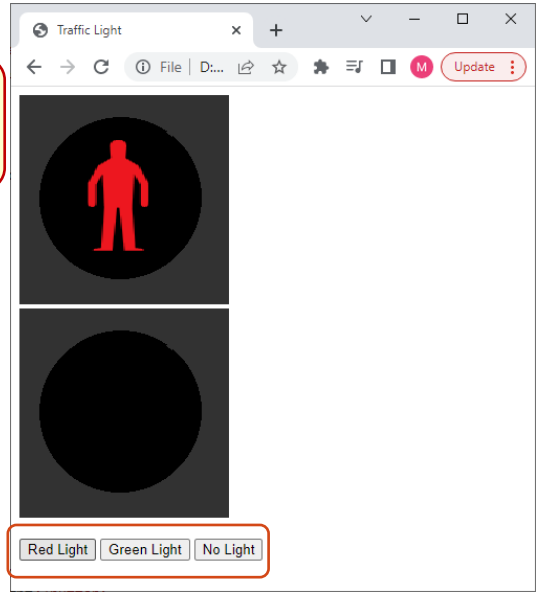
# Programming practice (5): User Choice

- We modify the webpage to include buttons to let user choose red/green/no light to be shown

```
<!DOCTYPE html>
<html>
  <head>
    <title>Traffic Light</title>
    <script>
        function showLight(top,bottom) {
          document.getElementById("toplight").src = top;
          document.getElementById("bottomlight").src = bottom;
        }
        function showRedLight() {
          showLight("Light-RedMan.png","Light-Off.png");
        }
        function showGreenLight() {
          showLight("Light-Off.png","Light-GreenMan.png");
        }
        function showNoLight() {
          showLight("Light-Off.png","Light-Off.png");
        }
    </script>
  </head>
  <body>
  <!-- Page content begins here -->
    <img id="toplight" src="Light-RedMan.png" /> <br/>
    <img id="bottomlight" src="Light-Off.png"  /> <br/>
    <p>
    <button id="buttonRedLight" onclick="showRedLight();">Red Light</button>
    <button id="buttonGreenLight" onclick="showGreenLight();">Green Light</button>
    <button id="buttonNoLight" onclick="showNoLight();">No Light</button>
    </p>
  <!-- Page content ends here -->
  </body>
</html>
```

This function changes the `src` attributes of the `img` elements whose file names are provided as the function parameters

Each of these 3 functions call the same function `showLight` but pass different arguments specifying the file names of the `img` elements on top and at the bottom



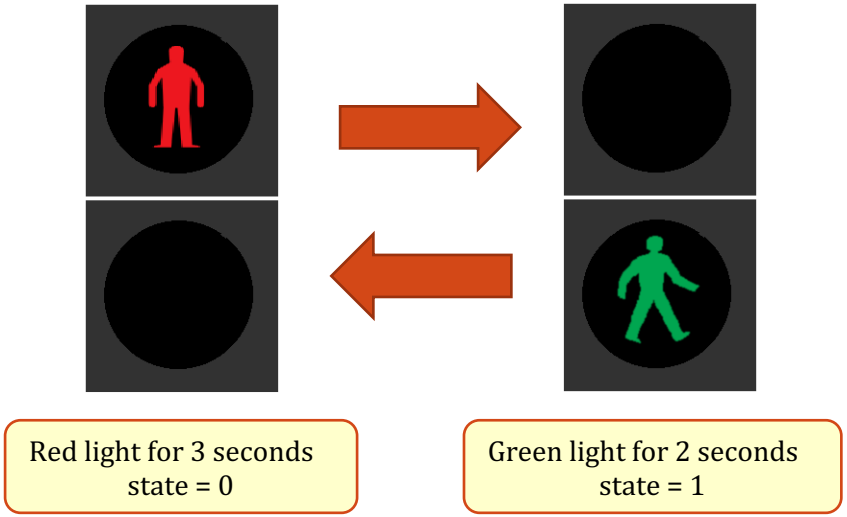

3 buttons are created at the bottom and the corresponding function will be called when each button is clicked to show red/green/no light



Code Example: user-choice.html

# Programming practice (5): how to make the light automatically switch?

- We will modify the webpage to let the light changes status automatically after some time and include buttons to enable/disable this feature
- For now we only consider the red light and green light status (no green light flashing)
- We will assign a variable `state` with value 0 denoting red light and value 1 denoting green light



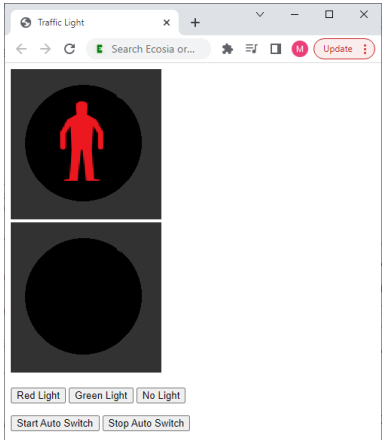
# Programming practice (6): Auto-switch implementation

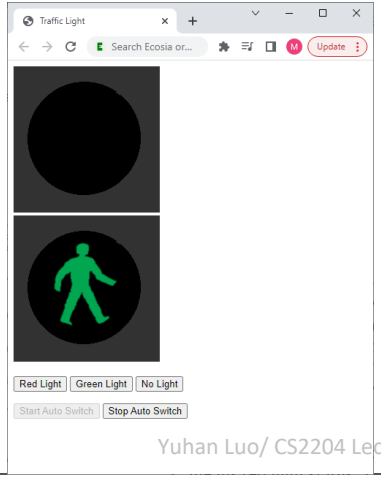
```
<!DOCTYPE html>
<html>
  <head>
    <title>Traffic Light</title>
    <script>
        var timerIDauto;
        var timeRed = 3000;
        var timeGreen = 2000;
        var state=-1;
        function showLight(top,bottom) {
          document.getElementById("toplight").src = top;
          document.getElementById("bottomlight").src = bottom;
        }
        function showRedLight() {
          showLight("Light-RedMan.png","Light-Off.png");
        }
        function showGreenLight() {
          showLight("Light-Off.png","Light-GreenMan.png");
        }
        function showNoLight() {
          showLight("Light-Off.png","Light-Off.png");
        }
        function startAutoSwitch() {
          document.getElementById("buttonStartAuto").disabled = true;
          state++;
          if (state > 1)
              state = 0;
          if (state==0) {
            showRedLight();
            timerIDauto = setTimeout(startAutoSwitch,timeRed);
          }
          else  {
            showGreenLight();
            timerIDauto = setTimeout(startAutoSwitch,timeGreen);
          }
        }
        function stopAutoSwitch() {
          document.getElementById("buttonStartAuto").disabled = false;
          clearTimeout(timerIDauto);
        }
    </script>
  </head>
  <body>
  <!-- Page content begins here -->
    <img id="toplight" src="Light-RedMan.png" /> <br/>
    <img id="bottomlight" src="Light-Off.png"  /> <br/>
    <p>
    <button id="buttonRedLight" onclick="showRedLight();">Red Light</button>
    <button id="buttonGreenLight" onclick="showGreenLight();">Green Light</button>
    <button id="buttonNoLight" onclick="showNoLight();">No Light</button>
    </p>
    <p>
    <button id="buttonStartAuto" onclick="startAutoSwitch();">Start Auto Switch</button>
    <button id="buttonStopAuto" onclick="stopAutoSwitch();">Stop Auto Switch</button>
    </p>
  <!-- Page content ends here -->
  </body>
</html>
```

Disable the Start Auto Switch button when it has already been clicked to avoid starting multiple timing (**`setTimeout`**) events

The variable **`state`** is increased by 1 each time when it is called, but will be reset back to 0 when it is larger than the possible value (1 in this case as there are only 2 values: 0 means red light status; 1 means green light status)

The Start Auto Switch button can be enabled back after the Stop Auto Switch button has been clicked when the previous timing event is cancelled (**`clearTimeout`**)




Code Example: auto-switch.html

# Reflect on the labs: Programming logic

**Lab 8 Task 2.1** Write a function swap the image sources

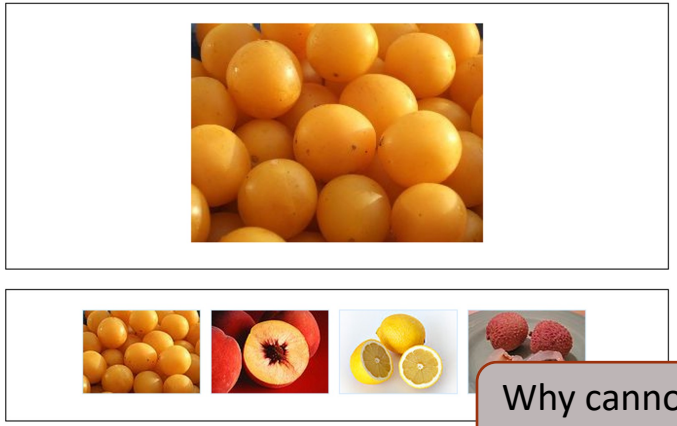
```
<div id="top">
    <img src="2Big.jpg" alt="b2" />
</div>
<div id="nails">
    <img src="1Nail.jpg" alt="n1"/>
    <img src="2Nail.jpg" alt="n2"/>
    <img src="3Nail.jpg" alt="n3"/>
    <img src="4Nail.jpg" alt="n4"/>
    <p onclick="shuffleImg();">Shuffle</p>
</div>
<script type="text/javascript">
    //add code here
    var bigImg = document.querySelector("#top img");
    var smallImg = document.querySelectorAll("#nails img");

    for(var i = 0; i < smallImg.length; i++) {
        smallImg[i].onmouseover = function(){swap(this);}
    }
    function swap(x) {
        bigImg.src = x.alt[1] + "Big.jpg";
        bigImg.alt = "b" + x.alt[1];
    }
```

The continuation condition of the for loop: when should we make `i < array.length` vs. `i < array.length-1`

Depends on what we need to output and whether we are using `i <` or `i < =`

# Reflect on the labs: Programming logic

**Lab 8 Task 2.1** Write a function swap the image sources



Why cannot we use `x.src[0]`?

x.src will return the absolute path instead of relative path, therefore the first char in the string is “f” instead of the number of the image

```
<div id="top">
    <img src="2Big.jpg" alt="b2" />
</div>
<div id="nails">
    <img src="1Nail.jpg" alt="n1"/>
    <img src="2Nail.jpg" alt="n2"/>
    <img src="3Nail.jpg" alt="n3"/>
    <img src="4Nail.jpg" alt="n4"/>
    <p onclick="shuffleImg();">Shuffle</p>
</div>
<script type="text/javascript">
    //add code here
    var bigImg = document.querySelector("#top img");
    var smallImg = document.querySelectorAll("#nails img");

    for(var i = 0; i < smallImg.length; i++) {
        smallImg[i].onmouseover = function(){swap(this);}
    }
    function swap(x) {
        bigImg.src = x.alt[1] + "Big.jpg";
        bigImg.alt = "b" + x.alt[1];
    }
```

# Notes on web dev

So far, we have finished all the new content regarding basic web concepts, HTML CSS, and JS.

What you learned in this course is just the fundamentals of web dev, but they are important skills to learn if you are interested in learning more advanced web programming

Particularly, JS is a powerful programming language that is widely used in not only front-end web dev, but also back-end servers (Node.js).

- There are many useful JS libraries to achieve fancy web effects (e.g., D3)
- JS can also but used for mobile programming (React.js)