CS2204 Fundamentals of Internet Applications Development

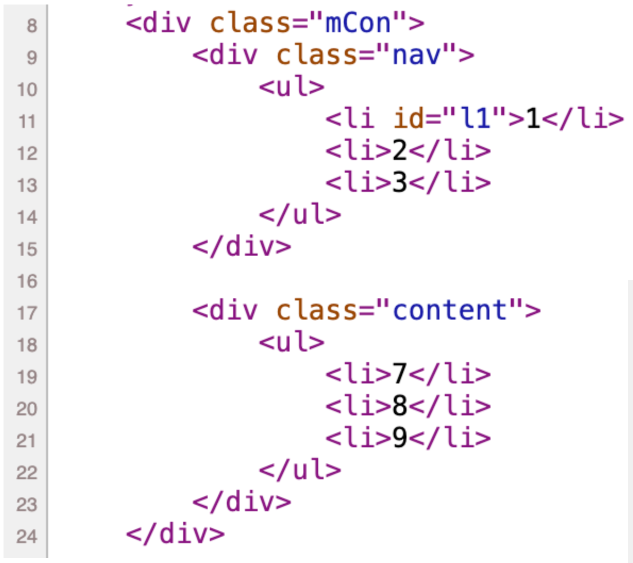
Lecture 11&12 JavaScript – Part 4 & Revision

Computer Science, City University of Hong Kong

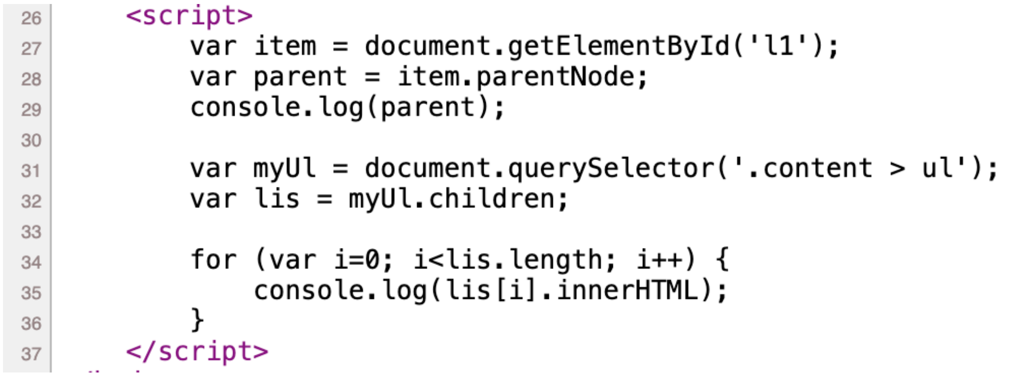
Semester A 2023-24

Select Based on Node Relationship

- Select the parent node of an element or node
  - `element.parentNode;`
- Select the children elements
  - `element.children`



```html
<div class="mCon">
    <div class="nav">
        <ul>
            <li id="l1">1</li>
            <li>2</li>
            <li>3</li>
        </ul>
    </div>

    <div class="content">
        <ul>
            <li>7</li>
            <li>8</li>
            <li>9</li>
        </ul>
    </div>
</div>
```

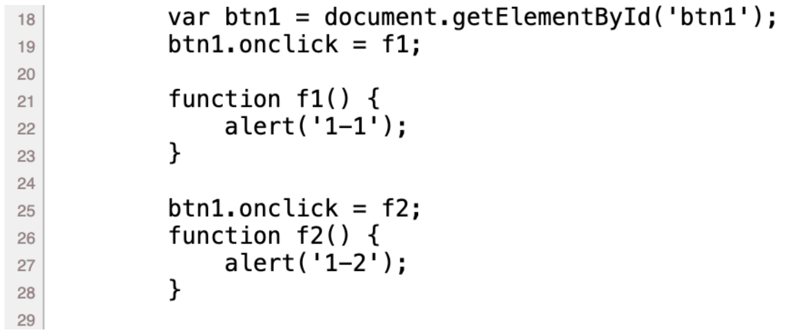


```html
<script>
    var item = document.getElementById('l1');
    var parent = item.parentNode;
    console.log(parent);

    var myUl = document.querySelector('.content > ul');
    var lis = myUl.children;

    for (var i=0; i<lis.length; i++) {
        console.log(lis[i].innerHTML);
    }
</script>
```

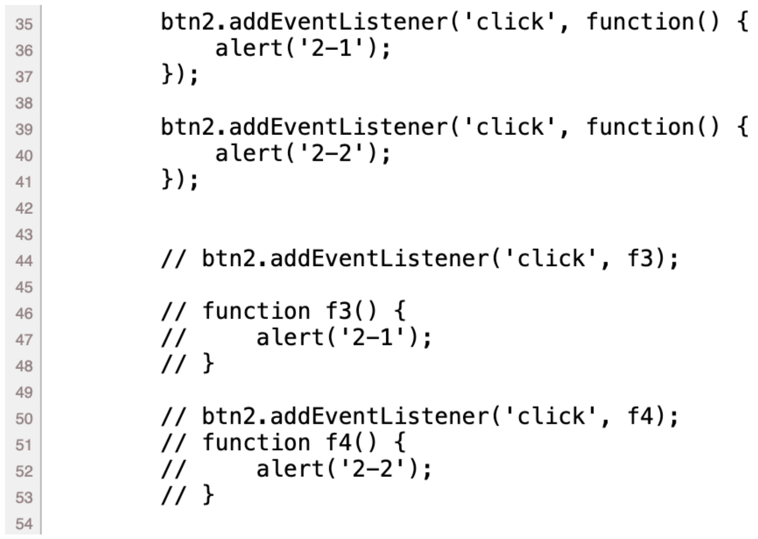
Code Example: lec10-15-JS-relation.html

Event Handler

- The second way to add an event handler
  - Why we need it?
- Syntax `element.addEventListener(type, listener[, useCapture])`
  - type: a **string** to represent an event (no prefix *on*)
    - e.g., 'click', 'mouseover', etc.
  - listener: **function** to handle this event



```
var btn1 = document.getElementById('btn1');
btn1.onclick = f1;

function f1() {
    alert('1-1');
}

btn1.onclick = f2;
function f2() {
    alert('1-2');
}
```

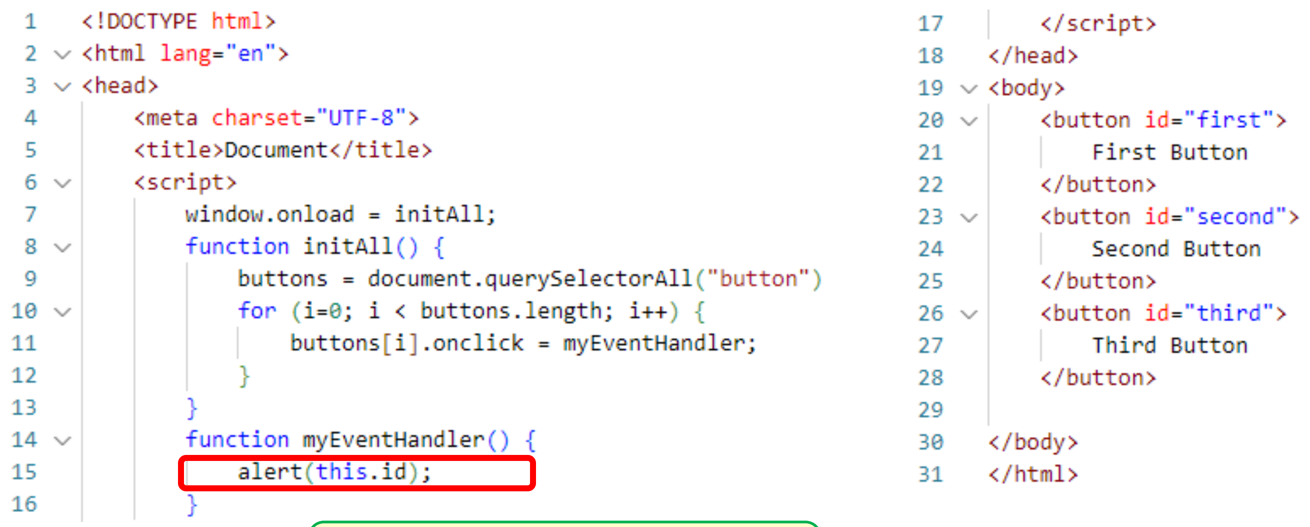


```
btn2.addEventListener('click', function() {
    alert('2-1');
});

btn2.addEventListener('click', function() {
    alert('2-2');
});


// btn2.addEventListener('click', f3);

// function f3() {
//     alert('2-1');
// }

// btn2.addEventListener('click', f4);
// function f4() {
//     alert('2-2');
// }
```

Code Example: lec10-16-JS-event-listener.html

Object This

- **this** can be used in the event handler to refer to the assigned object
  - Benefit: same event handler may be used for many similar objects



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        window.onload = initAll;
        function initAll() {
            buttons = document.querySelectorAll("button")
            for (i=0; i < buttons.length; i++) {
                buttons[i].onclick = myEventHandler;
            }
        }
        function myEventHandler() {
            alert(this.id);
        }
    </script>
</head>
<body>
    <button id="first">
        First Button
    </button>
    <button id="second">
        Second Button
    </button>
    <button id="third">
        Third Button
    </button>

</body>
</html>
```

Refer to the object assigned with this event handler

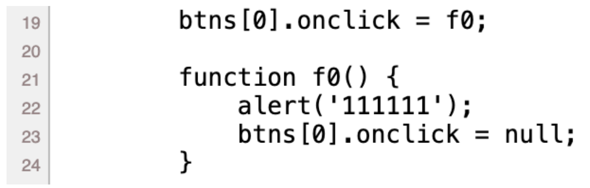
Code Example: Lec10-17-JS-object-this.html

Event Cancelling

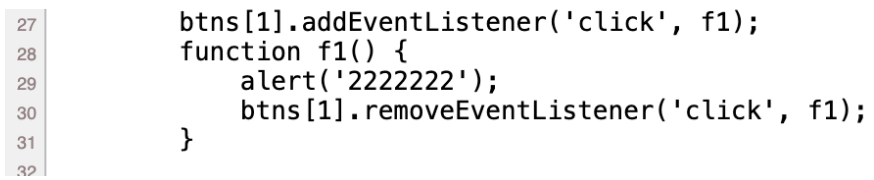
- After event handling, the event should not go on
- If an event is added as follows
  - `element.onclick = eventHandler;`
  - we can cancel it by `element.onclick = null;`

- If an event is added using `addEventListener`
  - we can cancel it by

```
element.removeEventListener(type, listener[, useCapture])
```



```
btns[0].onclick = f0;

function f0() {
    alert('111111');
    btns[0].onclick = null;
}
```

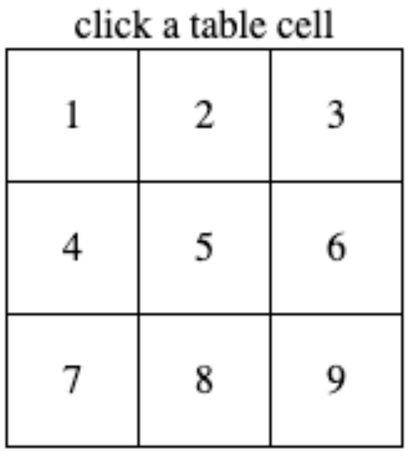


```
btns[1].addEventListener('click', f1);
function f1() {
    alert('2222222');
    btns[1].removeEventListener('click', f1);
}
```

Code Example: lec11-01-JS-delete.html

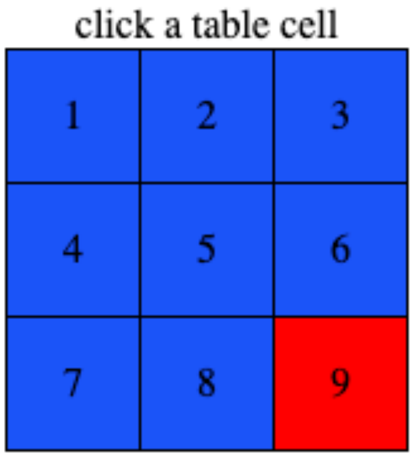
Example

- Given a table of 3x3 cells, click one cell
  - change this cell's background color to red
  - change all other cells' background color to blue



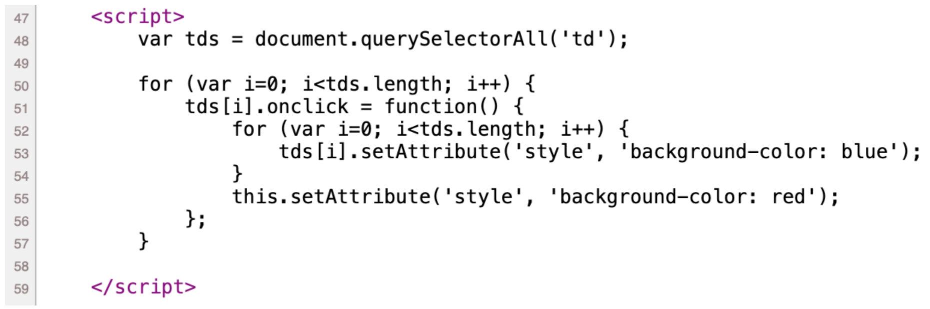
click a table cell

1	2	3
4	5	6
7	8	9



click a table cell

1	2	3
4	5	6
7	8	9



```
<script>
    var tds = document.querySelectorAll('td');

    for (var i=0; i<tds.length; i++) {
        tds[i].onclick = function() {
            for (var i=0; i<tds.length; i++) {
                tds[i].setAttribute('style', 'background-color: blue');
            }
            this.setAttribute('style', 'background-color: red');
        };
    }

</script>
```

Code Example: lec11-02-JS-table.html

Topics

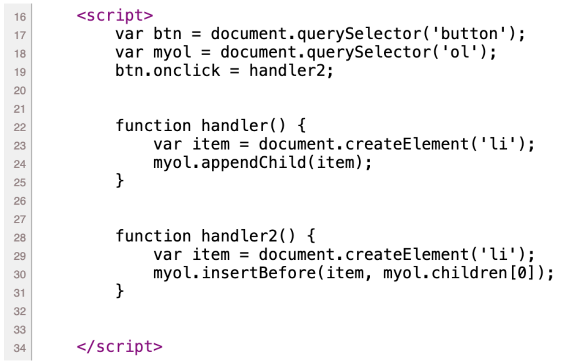
- Applications of JavaScript:
  - Dynamic content
  - Multimedia programming
  - JS library

Dynamic Content

- Common ways to build **dynamic contents**:
  - Add/delete elements
    - createElement
    - innerHTML property
  - Show or hide

createElement

- Create an element node
  - Syntax `document.createElement('elementName');`
- Add an element
  - Add a child node at the end of a parent node
    - `element.appendChild('childNode');`
  - Add a child node before another specified child node
    - `element.insertBefore('childNode', 'specifiedNode');`
    - e.g., `specifiedNode` can be `element.children[0]`
- Two steps
  - 1) create an element
  - 2) add this element



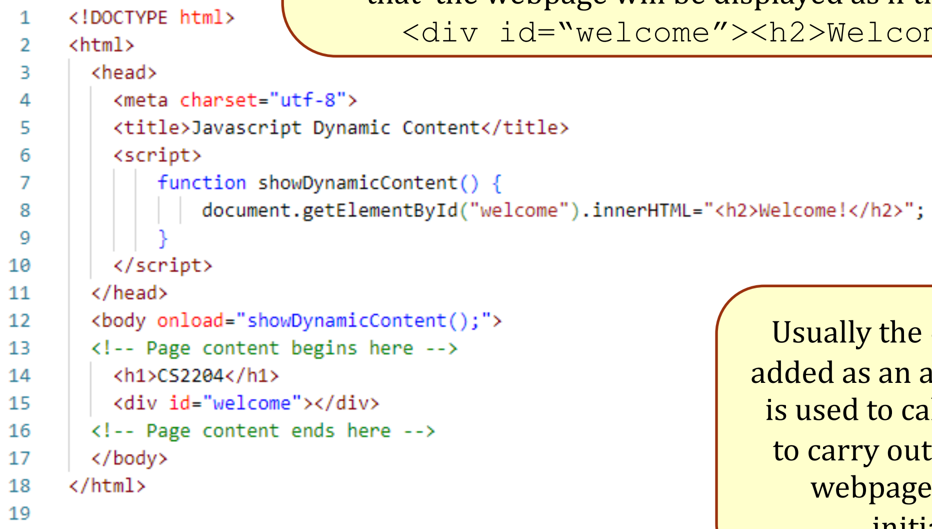
```
<script>
    var btn = document.querySelector('button');
    var myol = document.querySelector('ol');
    btn.onclick = handler2;


    function handler() {
        var item = document.createElement('li');
        myol.appendChild(item);
    }


    function handler2() {
        var item = document.createElement('li');
        myol.insertBefore(item, myol.children[0]);
    }


</script>
```

Code Example: lec11-03-JS-create.html

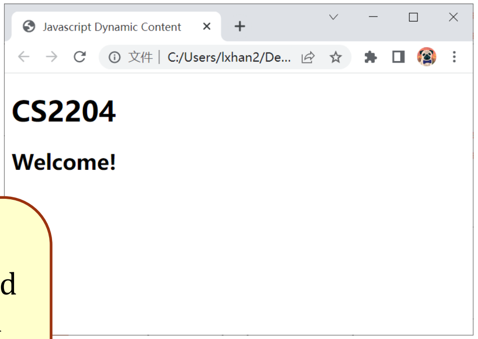
InnerHTML

- It allows access of the content of an element (or actual HTML) as a string, e.g.,

1. Originally the “welcome” div has no content: `<div id=“welcome”></div>`
2. After the page has finished loading, the `onload` function is invoked which will call the `showDynamicContent()` function
3. The following statement
   `document.getElementById(“welcome”).innerHTML = “<h2>Welcome!</h2>”`
   replaces the current content of the “welcome” div with the string `“<h2>Welcome!</h2>”` such that the webpage will be displayed as if the html of the “welcome” div is
   `<div id=“welcome”><h2>Welcome!</h2></div>`



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Javascript Dynamic Content</title>
    <script>
      function showDynamicContent() {
        document.getElementById("welcome").innerHTML="<h2>Welcome!</h2>";
      }
    </script>
  </head>
  <body onload="showDynamicContent();">
  <!-- Page content begins here -->
    <h1>CS2204</h1>
    <div id="welcome"></div>
  <!-- Page content ends here -->
  </body>
</html>

```



CS2204

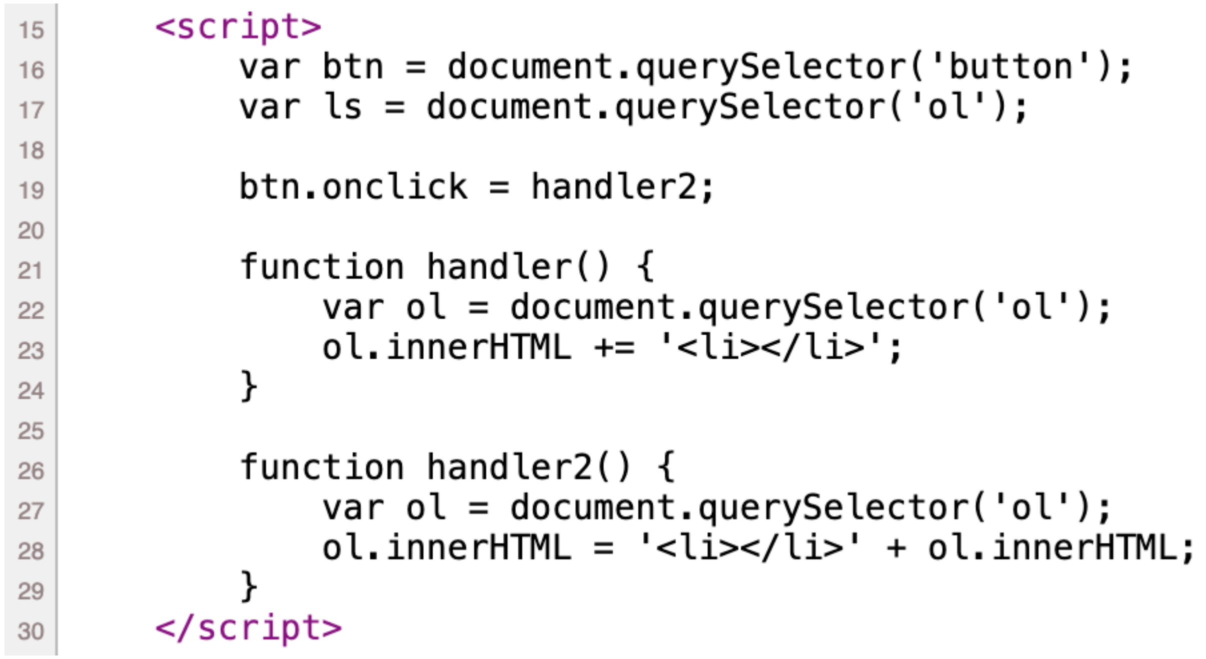
Welcome!

Usually the `onload` **event handler** is added as an attribute in the body tag and is used to call some JavaScript function to carry out some tasks right after the webpage has finished loading to initialize some settings

Code Example: lec11-04-JS-innerHTML.html

Critical Thinking

- Can we add `<li>` into `<ol>` using `innerHTML`?



```
<script>
    var btn = document.querySelector('button');
    var ls = document.querySelector('ol');

    btn.onclick = handler2;

    function handler() {
        var ol = document.querySelector('ol');
        ol.innerHTML += '<li></li>';
    }

    function handler2() {
        var ol = document.querySelector('ol');
        ol.innerHTML = '<li></li>' + ol.innerHTML;
    }
</script>
```

Code Example: lec11-05-JS-innerHTML2.html

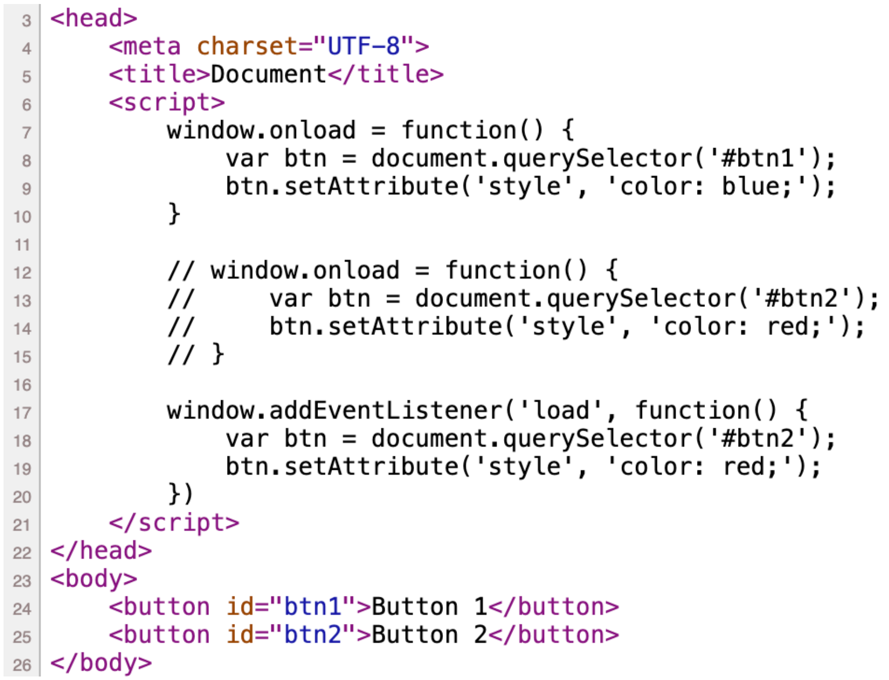
InnerHTML and createElement

- Difference is the efficiency
  - The efficiency of innerHTML depends on how to use it

Code Example: lec11-06-JS-innerHTML3.html

Where to put <script> </script>

- If we want to place `<script></script>` before html codes, we can use one of them below:
  - `window.onload`
  - `window.addEventListener('load’, f(){})`

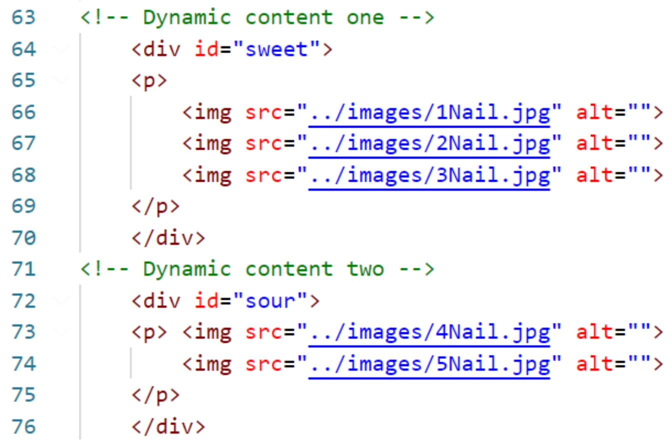


```
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        window.onload = function() {
            var btn = document.querySelector('#btn1');
            btn.setAttribute('style', 'color: blue;');
        }

        // window.onload = function() {
        //     var btn = document.querySelector('#btn2');
        //     btn.setAttribute('style', 'color: red;');
        // }

        window.addEventListener('load', function() {
            var btn = document.querySelector('#btn2');
            btn.setAttribute('style', 'color: red;');
        })
    </script>
</head>
<body>
    <button id="btn1">Button 1</button>
    <button id="btn2">Button 2</button>
</body>
```

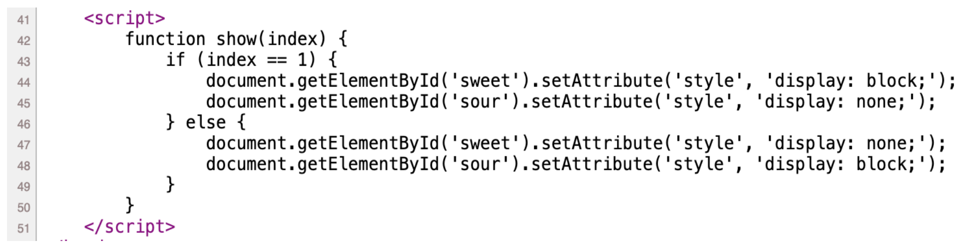
Code Example: lec11-07-JS-load.html

Hide & Show

- 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.



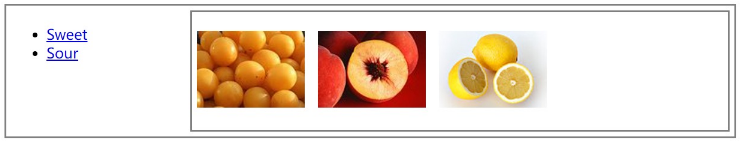
```html
<!-- Dynamic content one -->
    <div id="sweet">
    <p>
        <img src="../images/1Nail.jpg" alt="">
        <img src="../images/2Nail.jpg" alt="">
        <img src="../images/3Nail.jpg" alt="">
    </p>
    </div>
<!-- Dynamic content two -->
    <div id="sour">
    <p> <img src="../images/4Nail.jpg" alt="">
        <img src="../images/5Nail.jpg" alt="">
    </p>
    </div>
```

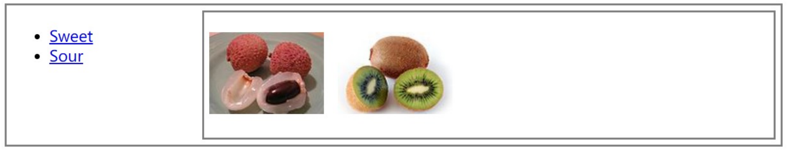


```html
<script>
    function show(index) {
        if (index == 1) {
            document.getElementById('sweet').setAttribute('style', 'display: block;');
            document.getElementById('sour').setAttribute('style', 'display: none;');
        } else {
            document.getElementById('sweet').setAttribute('style', 'display: none;');
            document.getElementById('sour').setAttribute('style', 'display: block;');
        }
    }
</script>
```

- Show as a block element
- Hide

Code Example: Lec11-08-JS-dynamic-content.html





Display different contents when selecting a different link

Multimedia Programming

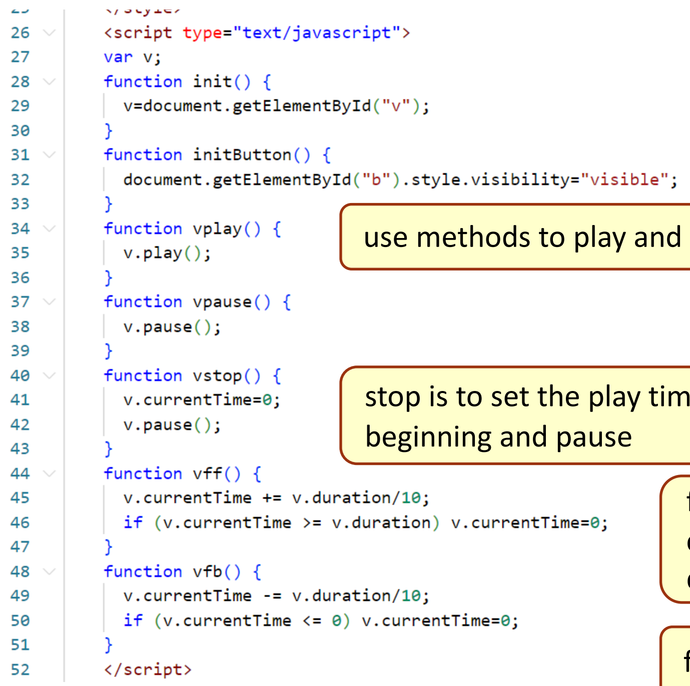
- Before HTML5, although video and audio are supported with the <object> tag, there was no scripting ability
  - Image is the only media that can be scripted
- In HTML5, video & audio scripting APIs are provided in addition to the usual dynamic content techniques for:
  - video

Video

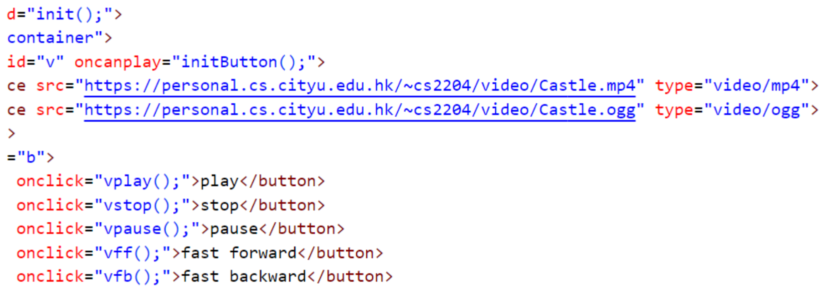
- Video & audio are timed media, i.e., have to be played according to a time interval
- HTML5 provides many useful properties, methods and events in JS, e.g.,
  - load()
  - play()
  - pause() - no stop method
  - .duration
  - .control
  - .oncanplay
  - .onended

Video

- Programming the video can best be illustrated with the VCR (video camera and recorder) operations in the example below
  - `oncanplay`: execute a JavaScript code when a video is ready to start playing



```
</style>
<script type="text/javascript">
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>
```



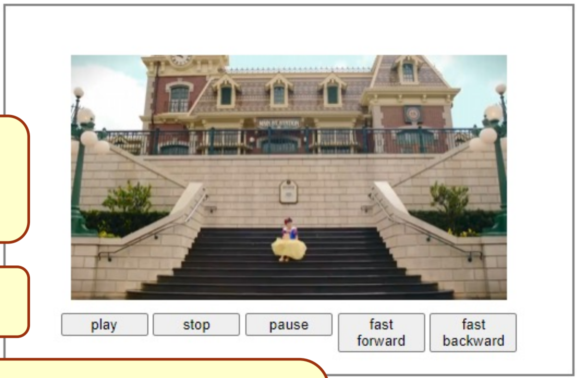
```
d="init();">
container">
id="v" oncanplay="initButton();">
ce src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.mp4" type="video/mp4">
ce src="https://personal.cs.cityu.edu.hk/~cs2204/video/Castle.ogg" type="video/ogg">
>
="b">
 onclick="vplay();">play</button>
 onclick="vstop();">stop</button>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

use methods to play and pause the video

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

fast forward is add video duration/10 to current play time, reset to 0 if larger than duration

fast backward is minus current play time

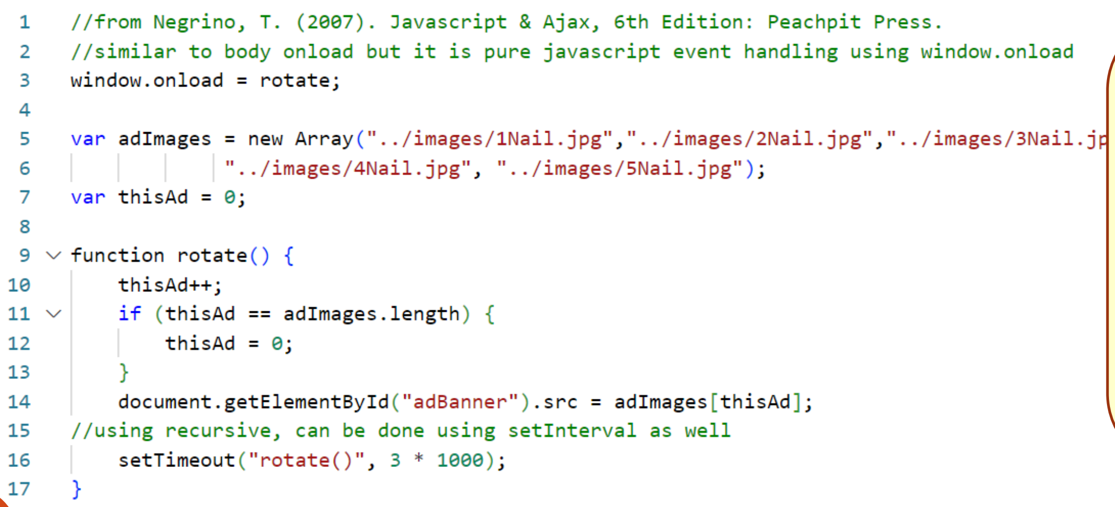


Code Example: lec11-09-JS-script-video.html

Also check out :
- how to set up various events for different operations?
- how to avoid timing problem - use of buttons before the video is ready?

Slide Show

- **Slide show** is in fact 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)`
    - to execute the function `myfunction` every 500 milliseconds
  - setTimeOut(function, interval)
    - execute the function/code after interval millisec
    - to execute repeatedly, need to use the method recursively in a function



```
//from Negrino, T. (2007). Javascript & Ajax, 6th Edition: Peachpit Press.
//similar to body onload but it is pure javascript event handling using window.onload
window.onload = rotate;

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

function rotate() {
    thisAd++;
    if (thisAd == adImages.length) {
        thisAd = 0;
    }
    document.getElementById("adBanner").src = adImages[thisAd];
//using recursive, can be done using setInterval as well
    setTimeout("rotate()", 3 * 1000);
}
```

To stop execution after the JavaScript is started

- clearInterval(id) or clearTimeOut(id) can stop the execution
- where id is the return value of the setInterval( ) or setTimeout( ) methods
- e.g., id = setTimeout(function, interval); clearTimeout(id); there may be more than one setTimeOut/setInterval in your code and their return ids are different

Code Example: lec11-10-JS-slide-show.html

Topics

- Applications of JavaScript:
  - Dynamic content
  - Multimedia programming
  - JS library

Use Of Library

- External **scripts**
  - when some of your codes are placed externally and used repeatedly - you have created a library
  - many people provide codes for others to use - 3rd party library
- API - application programming interface
  - expose objects and methods for users to use your library
- Different kinds of library
  - help to write **DOM** accessing JS easier - jQuery, YUI
  - widget/GUI library - jQuery UI
  - specialized library - Google Map, … & others

Use of Library

- jQuery
  - a very popular and powerful library
  - requires strong background in basic JS
- Get started with jQuery
  - download jQuery (https://jquery.com/)
  - store it in a **.js** file, e.g., myjquery.js
- Try jQuery

```
<script src=‘myjquery.js’></script>
```

```
<script>
        jQuery codes
</script>
```

Code Example: lec12-01-JS-jQuery.html

Launching Code

- Launch code when document is ready
  - syntax:

```
$(document).ready(function() {
          jQuery codes;
});
```

```
$(function() {
          jQuery codes;
});
```

  - meaning of $
    - stands for jQuery
  - why can we use methods?
    - convert a DOM object to a jQuery object

Code Example: lec12-02-JS-launch.html

Selecting Elements and Styling

- jQuery with CSS selector
  - `$('CSS_Selector')`
  - CSS_Selector
    - classical selectors: #id, .class, group selector
    - contextual selector, e.g., ul li, ol > li, etc.
    - advanced selectors, e.g., :first, :last, :eq(index), :odd, :even, etc.
- Styling
  - `$('CSS_Selector').css('attr', 'value');`
  - Use an object for multiple attributes, e.g.,

```
$('selector').css({
          'width': '100px',
          'height': '200px',
          'backgroundColor': 'red'
});
```

Code Example: lec12-03-JS-selector.html

Selecting Elements and Contents

- jQuery with contents
- Element's content (**including** tags)
  - get the context: `.html()`
  - change its context: `.html('new content')`
- Element's textual content (**without** tags)
  - get the context: `.text()`
  - change its context: `.text('new content')`
- Form element's value
  - get the value: `.val()`
  - change its value: `.val('new value')`

Code Example: lec12-04-JS-content.html

jQuery with Events

- Add an event
  - syntax

```
$('selector').event( function() {
        function code;
});
```

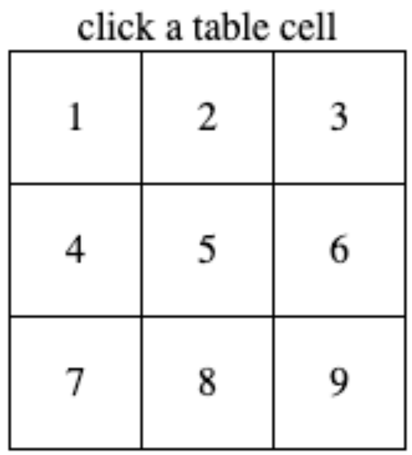
  - examples of `event` above are `click`, `mouseenter`, `mouseleave`, etc.

- Implicit iteration
  - When multiple elements are selected, a style or event can be applied to all these elements

Code Example: lec12-05-JS-iteration.html

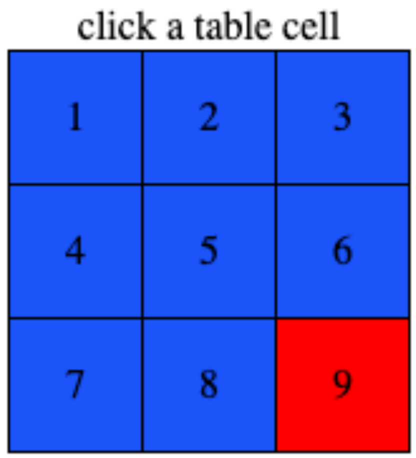
jQuery with Events

- Given a table of 3x3 cells, click one cell
  - change this cell's background color to red
  - change all other cells' background color to blue



click a table cell

1	2	3
4	5	6
7	8	9



click a table cell

1	2	3
4	5	6
7	8	9

Code Example: lec12-06-JS-table.html

jQuery with Events

- Add event using `.on()`
  - syntax

```
$('selector').on( {
         event1: function() { handler1},
         event2: function() { handler2},
         …
});
```

Code Example: lec12-07-JS-on.html

  - if events use the same handler

```
$('selector').on('event1 event2 …', function() {
         handler code
});
```

Code Example: lec12-08-JS-on-in-one.html