Scripting for Multimedia

LECTURE 4: WORKING WITH OBJECTS

Working with arrays

- An array is a collection object that has a sequence of items
- Creating and populating an array
  - Inserting items with the indexer
  - Empty items will be added when index # is higher than the quantity of existing items

```
var pizzaParts = new Array();
pizzaParts[0] = 'pepperoni'
pizzaParts[1] = 'onion'
pizzaParts[2] = 'bacon'
```

  - Condensed array

```
var pizzaParts = new Array('pepperoni', 'onion', 'bacon');
```

  - Literal array

```
var pizzaParts = ['pepperoni', 'onion', 'bacon'];
```

Working with arrays

- Accessing the array items using the indexer
  - The array is zero-based

```
var secondItem = pizzaParts[1];
```

- Modifying the array items

```
pizzaParts[1] = 'cheese';
```

- Understanding array properties
  - Length property

```
for(var i=0; i < pizzaParts.length; i++) {
   alert(pizzaParts[i]);
};
```

Working with arrays

- Using array methods (see related materials for more array methods)
  - concat

```
var pizzaMeatParts = ['pepperoni', 'ham', 'bacon'];
var pizzaVegetableParts = ['pepper', 'onion'];
var pizzaParts = pizzaMeatParts.concat(pizzaVegetableParts);
```

  - indexOf

```
var pizzaMeatParts = ['pepperoni', 'ham', 'bacon'];
var baconIndex = pizzaMeatParts.indexOf('bacon');
```

  - join

```
var pizzaMeatParts = ['pepperoni', 'ham', 'bacon'];
var meatParts = pizzaMeatParts.join();
```

Working with arrays

- Using array methods (see related materials for more array methods)
  - pop

```
var pizzaMeatParts = ['pepperoni', 'ham', 'bacon'];
var lastItem = pizzaMeatParts.pop();
```

  - push

```
var pizzaMeatParts = ['pepperoni', 'ham', 'bacon'];
var newLength = pizzaMeatParts.push('prosciutto');
```

  - reverse

```
var pizzaMeatParts = ['pepperoni', 'ham', 'bacon'];
pizzaMeatParts.reverse();
```

  - …

Accessing DOM objects

- DOM (Document Object Model) represents a hierarchy of objects which represents the HTML doc
  - Retrieve elements from the DOM using document variable
  - Some search methods return a single element whereas other return an array of elements
    - *live NodeList*
    - *static NodeList*

Accessing DOM objects

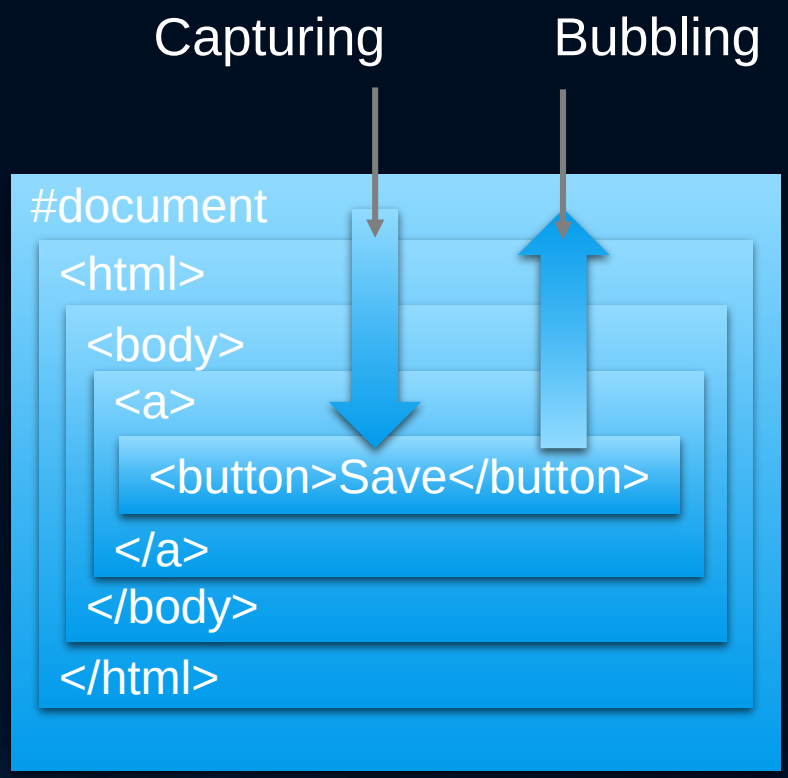
- DOM search methods (see related materials for more search methods)
  - getElementById
    ```
    var btn = document.getElementById('btnSave');
    ```
  - getElementsByTagName
    ```
    var images = document.getElementByTagName('img');
    ```
  - getElementsByName
    ```
    var pizzaSizes = document.getElementByName('pizzaSize');
    ```
  - getElementsByClass
    ```
    var pizzaParts = document.getElementByClass('pizzaPart');
    ```
  - …

Accessing DOM objects

- An event most commonly occurs with user interaction
- It also occurs when something changes state
- DOM provide events to be subscribed
- Events are based on the publisher-subscriber design pattern
  - When an event is triggered, all the event subscribers are notified by executing the event handler function
- When an event is triggered, an Event object is passed to the event handler function

Accessing DOM objects

- Event capturing and event bubbling



Accessing DOM objects

- Subscribe to an event using addEventListener
  - Three parameters
  - Example (preferred way)

```
var btn = document.getElementById('btnSave');
btn.addEventListener('click', saveData, false);
```

  - Inline method (the oldest way) for the same purpose

```
<button id='btnSave' onclick='saveData();'>Save</button>
```

  - A variation of the inline method (*traditional* subscription)

```
var btn = document.getElementById('btnSave');
btn.onclick = saveData;
```

Accessing DOM objects

- Unsubscribing from an event using the removeEventListener
  - Example

```
var btn = document.getElementById('btnSave');
btn.removeEventListener('click', saveData, false);
```

Accessing DOM objects

- Canceling event propagation using stopPropagation on the Event object
  - Example

```
var btn = document.getElementById('btnSave');
btn.addEventListener('click', saveData, false);
function saveData(e) {
//save the data
e.stopPropagation();
}
```

Accessing DOM objects

- Preventing the default operation of objects using preventDefault
  - Example

```
var hyperlink = document.getElementById('lnkSave');
hyperlink.addEventListener('click', saveData, false);
function saveData(e) {
//save the data
e.preventDefault();
}
```

Accessing DOM objects

- Working with "this"
  - The "this" keyword references the obj that caused the event
  - It provides a reference to the owner of the function

Some Window object events

- The built-in *window* variable is an instance of the Window object
- The following events can be applied to the <body> tag by adding the "on" prefix
  - focus
  - blur
  - beforeonload
  - load
  - haschange
  - error
  - …

```
window.addEventListener('load', winEvent, false);
function winEvent(e){
    alert('Window Load');
}
```

Some form events

- The following events are triggered by actions inside and HTML form
  - blur
  - change
  - focus
  - formchange
  - forminput
  - input
  - submit
  - …

```
var lastName = document.getElementById('txtLastName');
lastName.addEventListener('focus', gotFocus, false);
function gotFocus(e) {
    alert('last name has focus');
}
```

Keyboard events

- The following events are triggered by the keyboard and apply to all HTML5 elements
  - keydow
  - keypress
  - keyup

```
lastName.addEventListener('keypress', keyGotPressed, false);
function keyGotPressed (e) {
    var charCode = e.which;
    var charStr = String.fromCharCode(charCode);
    alert(charStr);
}
```

Some mouse events

- The following events are triggered by a mouse or similar user actions
  - click
  - dblclick
  - drag
  - drop
  - mousedown
  - mouseover
  - mousewheel
  - …

```
lastName.addEventListener('focus', gotFocus, false);
function gotClicked(e) {
    alert('Got Clicked');
}
```

Some media events

- The following events are triggered by media such as videos, images, and audio
  - ended
  - emptied
  - pause
  - play
  - waiting
  - …

```
var video = document.getElementById('video');
video.addEventListener('play', playing, false);

function playing(e) {
    alert('Playing');
}
```