School of Computer Science

Web and Database Computing 2019

Lecture 7: JavaScript Objects and Events

adelaide.edu.au

seek LIGHT

Bonus DOM

Manipulating DOM Tree

Create Element:

```
var newElement = document.createElement('P');
```

Add it to another element (with id 'parent'):

```
var parent = document.getElementById('parent');
parent.appendChild(newElement);
```

Remove that child element

```
parent.removeChild(newElement);
```

Javascript Objects

What are objects in Javascript?

Objects are collections of primitive values

Primitive types in Javascript

- string
- number
- boolean
- null
- undefined

Everything else is an object

In JavaScript, almost "everything" is an object.

- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

All JavaScript values, except primitives, are objects.

Booleans, Numbers and Strings can even be objects if defined using a constructor.

Defining objects

Objects are collections of primitive values

written as a series of property:value pairs

```
var object = { p1:'value 1', p2:2, p3:false };
```

Accessing objects

Get/set values using dot notation:

```
var object = { p1:'value 1', p2:2, p3:false };
var x = object.p1;
object.p4 = true;
```

Can also use array notation:

```
var object = { p1:'value 1', p2:2, p3:false };
var x = object['p1'];
```

The values can be anything that you would normally store in a variable:

```
object = { withinanobject:{ anobject:'bwaaah' } };
```

But objects should also have behaviour?

- Functions are always objects

JavaScript Result Edit in JSFiddle

```
var object = {
                age:42,
                alertAge: function() {
                            alert(this.age);
                          }
                         };

object.alertAge();
```

Properties can be dynamically added to objects

JavaScript Result Edit in JSFiddle

```
var object = {
                age:42
             };

object.colour = 'green';
document.body.innerText = object.age+' '+object.colour;
```

Object Constuctors

Making several of the same type of object

Constuctor definition & use

Constructors are a special type of function:

```
function Car(make, model, year) {
  this.make = make;
  this.model = model;
  this.year = year;
}
```

They can be called using the new keyword:

```
var car1 = new Car("Morris", "Mini Deluxe", 1967);
var car2 = new Car("Nissan", "Pulsar", 2013);
```

Modifying constructors using prototypes

Given constructor that we want to add to:

```
function Car(make, model, year) {
  this.make = make;
  this.model = model;
  this.year = year;
}
```

Use a prototype to modify it:

```
Car.prototype.transmission = "manual";
```

Arrays

Arrays are are a type of object used to store multiple values in the same variable.

```
var animals = ["Lion", "Tiger", "Bear"];
```

The values can be any object and do not have to be the same type.

```
var animals = ["Lion", "Tiger", 1924];
```

Individual elements accessed using indexes

```
var lion = animals[0];
```

Array functions & iteration

Arrays have methods to make standard array operations easier, e.g

- `push(var)`
- `pop()`
- `join()`
- `concat(array)`
- `slice(index1,index2)`

The preferred way of looping over elements in an Array is with an Iterator function:

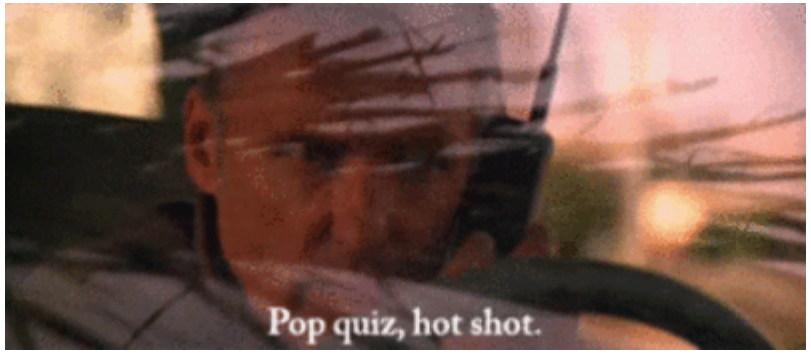
```
var animals = ["Lion", "Tiger", "Bear"];
animals.forEach(myFunction);

function myFunction(value, index, array) {
  console.log(value);
}
```

```
animals.forEach(function(value, index, array) {
    console.log(value);
}
```

There are also other iterator functions that perform specfic tasks.
See https://www.w3schools.com/js/js_array_iteration.asp

Quiz!



Javascript Events

Different events

We've already seen an exmaple of events:

- `click` Event (onclick attribute)

There are many types of events:

- `load` (onload attribute) Fires when an element finishes loading.
- `change` (onchange) Fires when the value of an input is changed.
- `mousedown` and `mouseup` Fire when the mouse button is pressed down and then when released respectively.
- `mouseover` and `mouseout` Fire when the mouse cursor enters or leaves an element respectively.
- And many more!
  - https://www.w3schools.com/jsref/dom_obj_event.asp

Using events

Events can be used to trigger our code by attaching our code to an event attribute:

- In the HTML

```
<button id="aButton" onmouseout="myfunction()">Triggers when the mouse leaves</button>
```

- Or in Javascript

```
document.getElementById("aButton").onmouseout = myfunction;
```

Multiple event handlers using listeners

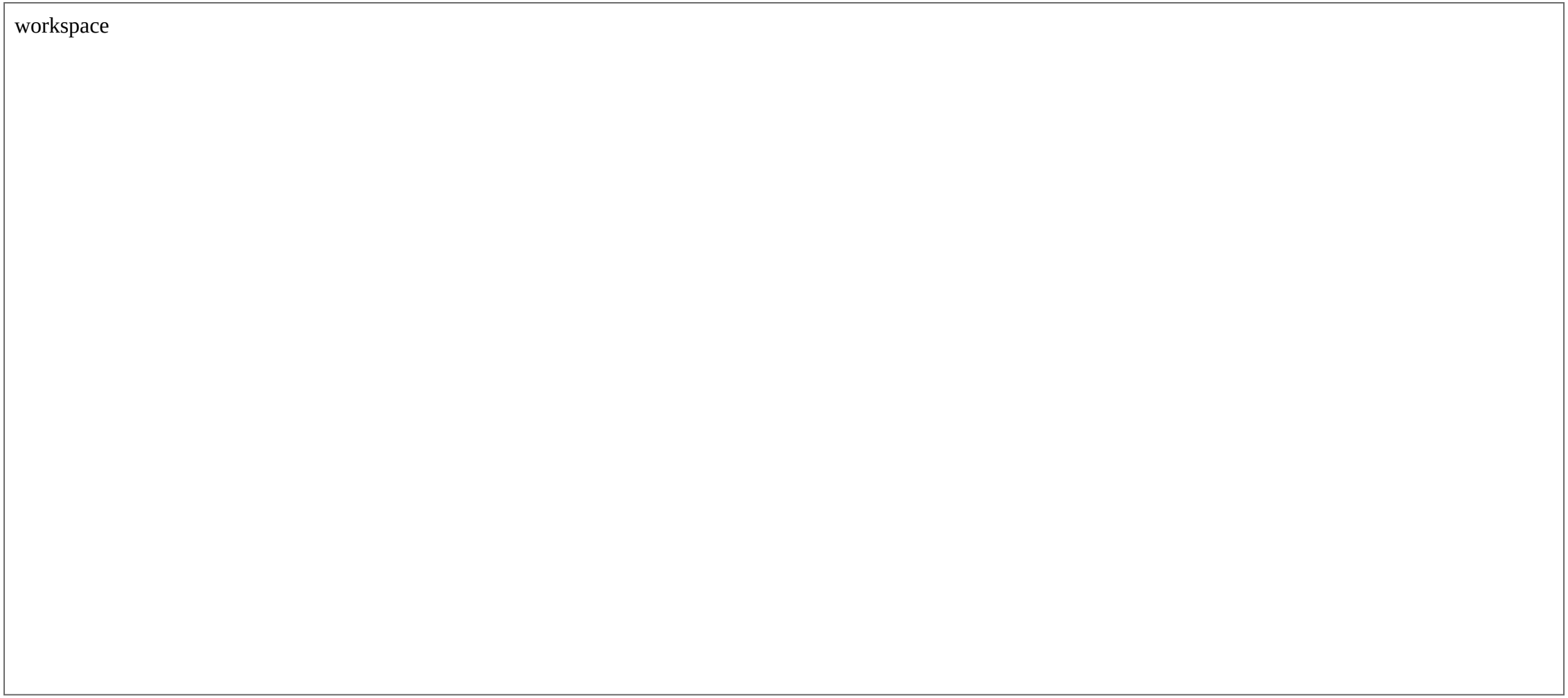
The previous examples only allowed 1 function to run when an event occurs, but what if we want to dynamically add or remove several?

Use an EventListener:

```
var myButton = document.getElementById("aButton");
myButton.addEventListener('mouseout', myfunction);
myButton.removeEventListener('mouseout', myfunction);
```

Debugging

Using your browser's devlopment tools



https://jsfiddle.net/ian_knight_uofa/fsk013uy/5/


THE UNIVERSITY
of ADELAIDE

What's happening

Due:

- Prac Exercise 2 try to complete before Monday if not already done.
- Prac Exercise 3 now available. Websub available soon.

Next week:

- Client-Server model
- Introduction to NodeJS & AJAX

Further learning:

- Keep working through the Javascript tutorial at https://www.w3schools.com/js/
  - Try the exercises included
- Keep using HTML and CSS in your forum posts.