# Introduction to JavaScript – Part Two

Website Development 2

# Lecture 2 Outline

- Date Object
- Functions
- Arrays

# Date object

- **Date()**
- Constructs an empty date object.

  For example: var now=new Date();

## getDate()

Returns the day of the month.

```
var dayNum =
now.getDate();
```

## getDay()

Returns an integer representing the day of the week, Sunday is 0 and Saturday is 6.

```
var day =
now.getDay();
```

## getMonth()

Returns the month field of the Date object, represented by an integer, January is 0 and December is 11.

```
var month = now.getMonth();
```

## getFullYear()

Returns the year as a four digit number.

```
var thisyear = now.getFullYear();
```

# Date: Other Retrieval methods

getHours()
getMinutes()

getSeconds()
getMilliseconds()

getTime()

# Lecture 2 Outline

- Date Object
- Functions
- Arrays

# Functions

- Functions:
  - consist of one or more *statements* (i.e., lines of program code that perform some operation).
  - are separated in some way from the rest of the program, for example, by being enclosed in curly brackets, {.....}
  - are given a unique name, so that they can be *called* from elsewhere in the program.
- Functions are used where the same operation has to be performed many times within a program.

# Functions

In JavaScript, functions are created in the following way:

```
function name()
{
    statement;
    statement;
}
```

# Functions

However, it is often necessary to supply information to a function so that it can carry out its task.

```
function addVAT(price)

{
    price *= 1.21;
    alert(price);
}
```

We would call this function in the following way:

```
addVAT(costPrice);
```

# Functions

Sometimes we also need to get some information back from a function.

```
function addVAT(price)

{
   price *= 1.21;
   return(price);
}
```

We would call this function in the following way:

```
var newPrice = addVAT(costPrice);
```

# Lecture 2 Outline

- Date Object
- Functions
- Arrays

# Arrays

- The Array object is used to store a set of values in a single variable name.

```
var data= new Array();
data[0] = "Hurling" ;
data[1] = "Rugby";
data[2] = "Football";
data[3] = "Soccer";
data[4] = "Tennis";

document.write(data);
```

# Array Manipulation

```
for (count=0; count< data.length; count++) {
  document.write(data[count] + "<br>");
}
```

```
for (x in data) {
  document.write(data[x] + "<br>");
}
```

# Object based array functions

- Arrays have lots of nifty built in functions such as:
  - join(),
  - push(),
  - pop(),
  - sort(),
  - slice(),
  - splice(),
  - and more…

# join()

The **join()** method is used to put all the elements of an array into a string. The elements will be separated by a specified separator.

```
data.join(', ');
data.join(‘<br> ');
```

## push()

The **push()** method adds one or more elements to the end of an array and returns the new length.

```
data.push('golf');
```

## unshift()

The **unshift()** method adds one or more elements to the start of an array and returns the new length.

```
data.unshift('golf');
```

## pop()

The **pop()** method is used to remove and return the last element of an array.

```
data.pop();
```

## shift()

The **shift()** method is used to remove and return the first element of an array.

```
data.shift();
```

# splice()-delete

The **splice()** command must specify where it should begin deleting (index number of first item to be deleted) and how many items it should delete.

data.splice(2,2);

# splice()-add

The splice() command must specify where the new items should be located, 0 to indicate that you do not want to delete any items, then the list of items to be inserted: one or more values separated by commas.

**data.splice(2,0, "Cricket", "Snooker");**

# splice()-replace

The process is the same as adding an item, but instead of specifying 0 for the second piece of information, you supply the number of items to be replaced. This is followed by the list of items that are replacing the deleted (replaced) items.

```
data.splice(2,2, "Cricket", "Snooker");
```

## reverse()

The **reverse()** method is used to reverse the order of the elements in an array.

```
data.reverse();
```

## concat()

The **concat()** method is used to join two or more arrays.

```
data.concat(data);
```

| sort() | sort() - numeric |
| --- | --- |
| The sort() method is used to sort the elements of an array. | To sort numbers, you must add a function that compare numbers. |
| data.sort(); | data.sort(function(a,b){return a - b}); |