JavaScript and jQuery Course

Session 04

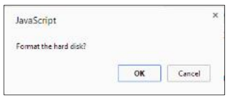
Objects

Window Object - top level object

- alert() method
- prompt() method
- confirm() method   new
- parseInt() – global function
- parseFloat() – global function

Window Object

confirm() method


JavaScript

Format the hard disk?

```
var response = confirm('Format the hard disk?');

// user clicks OK
if(response) {
    alert('You clicked okay!');
}
// user clicks Cancel
else {
    alert('You clicked cancel!');
}
```

Window Object

parseInt() global function

```
var x = '10';
X = parseInt(x);   // returns a whole number 10
```

parseFloat() global function

```
var y = '0.5';
y = parseFloat(y); // returns a decimal number .5
```

Window Object

parseInt() global function

```
var x = '10xxx';
X = parseInt(x);   // returns a whole number 10

var y = 'xxx10';
y = parseInt(y);   // returns a NaN

var z = 'xxx';
z = parseInt(z);   // returns a NaN
```

Not a number

Not a number

5

Document Object

Document Object - top level object in DOM

- write() method
- writeLn() method – adds a new line character in the source code
- document.getElementById(id) method

6

Objects

Document Object Model

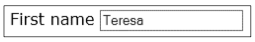
- allows access to any HTML element by making an object reference using the value of the 'id' attribute
- document.getElementById(id) method

7

Textbox Object

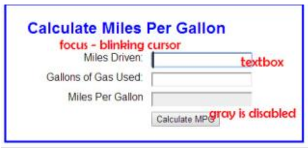
```
<label for="fname">First name</label>

<input type="text" id="fname" name="fname">
```



8

Textbox Object


Calculate Miles Per Gallon

focus - blinking cursor

Miles Driven:

textbox

Gallons of Gas Used:

Miles Per Gallon

Calculate MPG

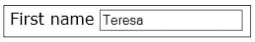
gray is disabled

- Method – focus()
- Property – value
- Property - disabled

9

Textbox Object - retrieve the value

```
<label for="fname">First name</label>
<input type="text" id="fname" name="fname">
```


First name Teresa

```
var firstName = document.getElementById("fname");

firstName = firstName.value;

alert(firstName);  // displays Teresa
```

10

Textbox Object - retrieve the value (Chaining)

```
<label for="fname">First name</label>
<input type="text" id="fname" name="fname">
```

First name Teresa

```
var firstName = document.getElementById("fname").value;

alert(firstName);  // displays Teresa
```

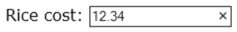
11

Textbox - always returns a string value

To Convert to an integer or float (for calculations)

```
  parseInt()
  parseFloat()
```

```
<label for="riceCost">Rice cost:</label>
<input type="text" id="riceCost" name="riceCost">
```


Rice cost: 12.34

```
var riceCost = document.getElementById("riceCost").value;
// value of riceCost is "12.34" (text)

var iRiceCost = parseInt(riceCost);   // value is 12
var fRiceCost = parseFloat(riceCost); // value is 12.34
```

12

Textbox - always returns a string value

```
<label for="riceCost">Rice cost:</label>
<input type="text" id="riceCost" name="riceCost">
```

Rice cost: 12

```
var riceCost = document.getElementById("riceCost").value;
// value of riceCost is "12" (text)

var iRiceCost = parseInt(riceCost);   // value is 12
var fRiceCost = parseFloat(riceCost); // value is 12
```

13

Textbox - always returns a string value

```
<label for="riceCost">Rice cost:</label>
<input type="text" id="riceCost" name="riceCost">
```

Rice cost: 12 34 ×

```
var riceCost = document.getElementById("riceCost").value;
// value of riceCost is "12 34" (text)

var iRiceCost = parseInt(riceCost);   // value is 12
var fRiceCost = parseFloat(riceCost); // value is 12
```

14

Textbox - always returns a string value

```
<label for="riceCost">Rice cost:</label>
<input type="text" id="riceCost" name="riceCost">
```

Rice cost: $12.34

```
var riceCost = document.getElementById("riceCost").value;
// value of riceCost is "$12.34" (text)

var iRiceCost = parseInt(riceCost);   // value is NaN
var fRiceCost = parseFloat(riceCost); // value is NaN
```

15

isNaN() – JS global function

```
<label for="riceCost">Rice cost:</label>
<input type="text" id="riceCost" name="riceCost">
```

Rice cost: $12.34

```
var riceCost = document.getElementById("riceCost").value;
// value of riceCost is "$12.34" (text)

var iRiceCost = parseInt(riceCost);   // value is NaN
var fRiceCost = parseFloat(riceCost); // value is NaN
```

✓ **Test that it is a valid number before parsing**

```
if(isNaN(riceCost){
   // returns true or false
}
```

16

isNaN() – JS global function

```
isNaN(123)          //false
isNaN(-1.23)        //false
isNaN(5-2)          //false
isNaN(0)            //false
isNaN('123')        //false
isNaN('Hello')      //true
isNaN('2005/12/12') //true
isNaN('')           //false
isNaN(true)         //false
isNaN(undefined)    //true
isNaN('NaN')        //true
isNaN(NaN)          //true
isNaN(0 / 0)        //true
```

17

Empty textbox

empty

A textbox without any user entered value returns “”

It is empty – it is not null

To validate that the user entered a value

```
if(txtVariableValue == “”){
  // returns true or false
}
```

18

Undefined vs empty vs null in JavaScript

undefined

A **variable without a value**, has the value undefined.
The typeof is also undefined.

The typeof operator returns type information as a literal.
There are six possible values that typeof returns:

- "number“
- "string“
- "boolean“
- "object“
- "function“
- "undefined“

empty

An empty string variable has both a value and a type.

Undefined vs empty vs null in JavaScript

null

In JavaScript null is "**nothing**". It is supposed to be something that doesn't exist.
Unfortunately, in JavaScript, the data type of null is an object.

You can consider it **a bug in JavaScript** that typeof null is an object. It should be null. this is a bug which may be fixed at some point.

Therefore: do not use null when comparing variable values!

20

Validate textbox contains a number

empty

```
A textbox without any user entered value returns “”

It is empty – it is not null
```

```
To validate that the user entered a value

if(txtVariableValue == “”){
  // returns true or false
}
```

Validate textbox contains a number

NaN

```
A textbox with an invalid number returns “NaN”

isNaN() – global function
        - will return true if not a valid number
```

```
To validate that the user entered a value

if(isNaN(txtVariableValue)){
  // returns true or false
}
```

Date Object

- JS Object
- `Date()` – uppercase “D”
- Returns: day, number, month, year, hour, minutes, seconds, milliseconds
- to retrieve, set, perform calculations **– EBAY / Bb**

Date Object

- Begins January 1, 1970 at midnight
- **epoch or Unix epoch**
- 86,400,000 milliseconds in one day

Date() Object – create it

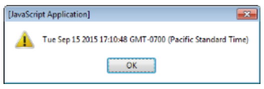
instance of object

```
var today = new Date();
```

JS object

JS keyword / operator

```
alert(today);
```



Tue Sep 15 2015 17:10:48 GMT-0700 (Pacific Standard Time)

Date() Object – create it

Holds number of milliseconds since January 1, 1970 at midnight

```
var today = new Date();
```

Methods of Date() Object

`var today = new Date();`

`today.getDate()`	Returns the day number of the month (1 - 31) `var monthDay = today.getDate();`

`today.getDay()`	Returns an integer for the day of the week
	Sunday = 0, Monday = 1, … Saturday = 6
	`var dayOfWeek = today.getDay();`

`today.getMonth()`	Returns an integer for the month
	January = 0, … December = 11
	`var month = today.getMonth();`

`today.getFullYear()`	Returns the year as 4 digits `var fullYear = today.getFullYear();`
`today.getYear()`	Returns the year as 2 digits `var year = today.getYear();`

Methods of Date() Object

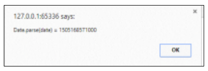
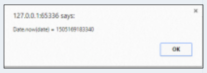
`var today = new Date();`

`today.getTime()`	Returns milliseconds since midnight 1/1/1970 `var time = today.getTime();`
`today.getHours()`	Returns the current hour, 0 to 23, universal time `var currentHour = today.getHours();`
`today.getMinutes()`	Returns the current minute, 0 to 59 `var currentMinute = today.getMinutes();`
`today.getSeconds()`	Returns the current second, 0 to 59 `var currentSecond = today.getSeconds();`

`today.getMilliseconds()`	Returns the current millisecond, 0 to 999
	`var currentMilliseconds =`
	`today.getMilliseconds();`

Methods of Date() Object

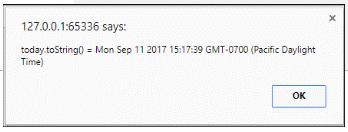
`var today = new Date(); // using Sept 11, 2017`

`Date.parse()`  127.0.0.1:65336 says: Date.parse(date) = 1505168571000	Returns: milliseconds since 1/1/1970 Local time `var milsec = Date.parse(today);`
`Date.now()`  127.0.0.1:65336 says: Date.now(date) = 1505169183340	Returns: milliseconds since 1/1/1970 UTC `var milsec2 = Date.now(today);`
`today.toDateString()`	Returns: Mon Sep 11 2017 `var str1 = today.toDateString();`

Methods of Date() Object

`var today = new Date(); // using Sept 11, 2017`

`today.toLocaleDateString()`	Returns: 9/11/2017 `var str2 = toLocaleDateString();`
`today.toLocalString()`	Returns: 9/11/2017, 3:17:39 PM `var str3 = today.toLocalString();`
`Today.toString()`	`var str4 = today.toString();`



127.0.0.1:65336 says:

today.toString() = Mon Sep 11 2017 15:17:39 GMT-0700 (Pacific Daylight Time)

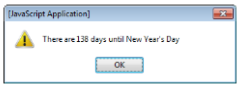
Date calculations

```
// create new date for next New Year's day
var nextYear    = new Date().getFullYear() + 1;
var daysNewYear = new Date(nextYear, 0, 1); // next Jan 1

// get today's date
var today = new Date();

// calculation
var days = Math.round((daysNewYear - today) / 86400000);

alert("There are " + days + " days until New Year's day");
```



[JavaScript Application]

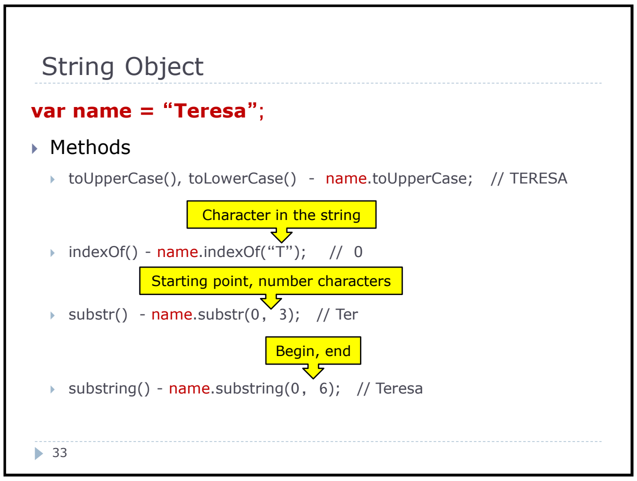
There are 138 days until New Year's Day

String Object

Object is automatically created when you assign a string value to a variable

var name = "Teresa";

- Property
  - length - name.length; // 6

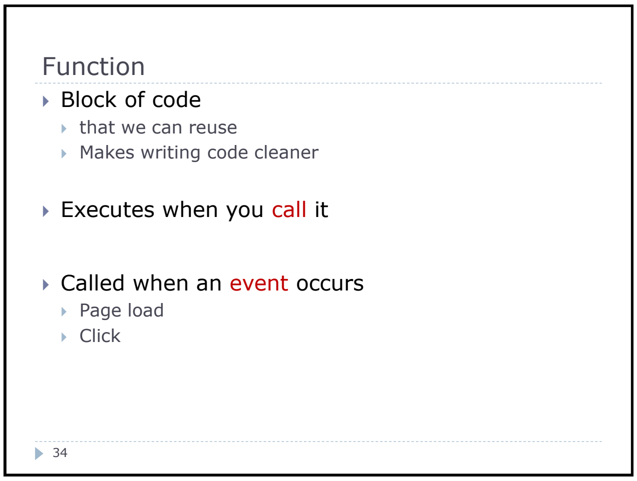


String Object

var name = “Teresa”;

- Methods
  - toUpperCase(), toLowerCase() - name.toUpperCase; // TERESA
  - Character in the string
  - indexOf() - name.indexOf(“T”); // 0
  - Starting point, number characters
  - substr() - name.substr(0, 3); // Ter
  - Begin, end
  - substring() - name.substring(0, 6); // Teresa

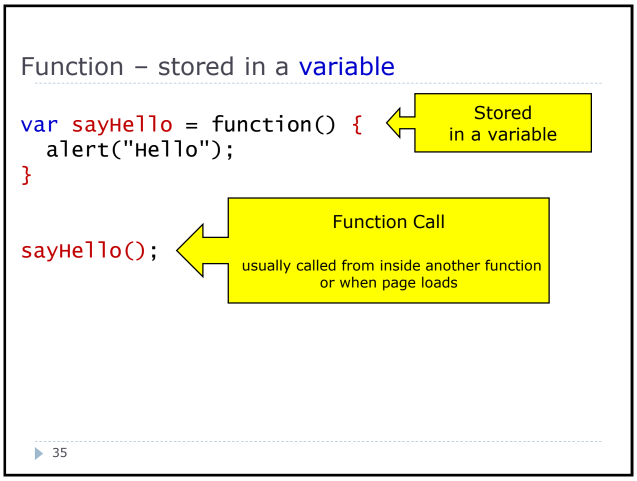
33



Function

- Block of code
  - that we can reuse
  - Makes writing code cleaner
- Executes when you call it
- Called when an event occurs
  - Page load
  - Click

34



Function – stored in a variable

```
var sayHello = function() {
  alert("Hello");
}
```

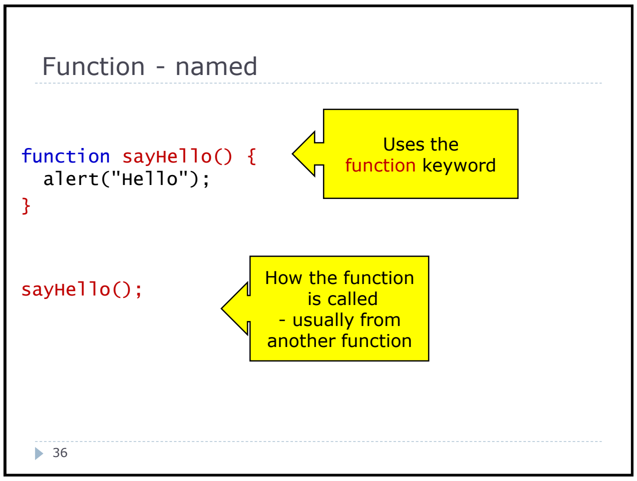
Stored in a variable

```
sayHello();
```

Function Call

usually called from inside another function or when page loads

35



Function - named

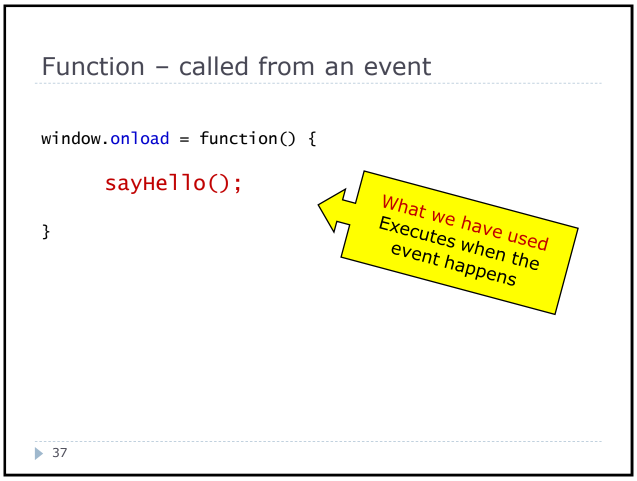
```
function sayHello() {
  alert("Hello");
}
```

Uses the function keyword

```
sayHello();
```

How the function is called - usually from another function

36



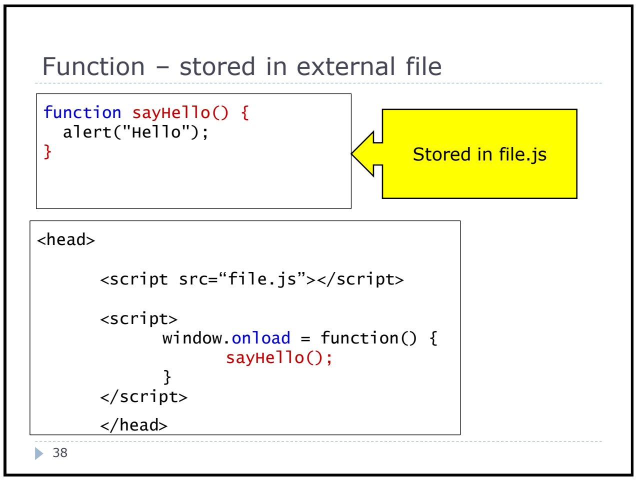
Function – called from an event

```
window.onload = function() {

      sayHello();

}
```

What we have used
Executes when the event happens



Function – stored in external file

```
function sayHello() {
  alert("Hello");
}
```

Stored in file.js

```
<head>

      <script src="file.js"></script>

      <script>
            window.onload = function() {
                  sayHello();
            }
      </script>
      </head>
```