Your fastest way to learn. Why wait?

FIREBRAND

Exam:70-480

Exam Prep Review Slides

Programming in HTML5 with JavaScript and CSS3

Courseware Version 1.0

KIT CODE: K-494-01

www.firebrandtraining.co.uk

Exam Prep 70-480

Programming in HTML5 with JavaScript and CSS3

Updated June 2017

JavaScript Method document.getElementById("demo");

- Returns the element that has the ID attribute with the specified value.
- Returns *null* if no elements with the specified ID exists.
- An ID should be unique within a page.
- If more than one element with the specified ID exists, the first element in the source code is returned.
- Supported in all browsers.

Parameter	Type	Description
elementID	String	Required. The ID attribute's value of the element you want to get

© Firebrand Training Ltd

HTML onclick Event Attribute

- The onclick attribute fires on a mouse click on the element.
- Syntax: <*element* onclick = *"script"*>
- Example:

Click on a <p> element to change its text color to red:

```
<p id="demo" onclick="myFunction()">Click me to change
my text color.</p>

<script>
function myFunction() {
    document.getElementById("demo").style.color =
"red";
}
</script>
```

- Attribute Values

Value	Description
script	The script to be run on onclick

3

JavaScript Prototype Property

- Every JavaScript object has a prototype.
- The prototype is also an object.
- All JavaScript objects inherit their properties and methods from their prototype.
- Objects created using an object literal, or with new Object(), inherit from a prototype called Object.prototype.
- Objects created with new Date() inherit the Date.prototype.
- The Object.prototype is on the top of the prototype chain.
- All JavaScript objects inherit from the Object.prototype.

4

© Firebrand Training Ltd

Jquery val() Method

- The val() method returns or sets the value attribute of the selected elements.
- **When returning a value:** This method returns the value of the value attribute of the FIRST matched element.
- **When setting a value:** This method sets the value of the value attribute for ALL matched elements.
- Mostly used with HTML form elements.
- Syntax
  - Return the value: $(*selector*).val()
  - Set the value: $(*selector*).val(*value*)
  - Set the value attribute using a function: $(*selector*).val(function*(index,currentvalue)*)

5

JQuery val() Method (Continued)

Parameter	Description
value	Required. Specifies the value of the value attribute.
function(index,currentvalue)	Optional. Specifies a function that returns the value to set. • index - Returns the index position of the element in the set • currentvalue - Returns the current value attribute of selected elements

6

© Firebrand Training Ltd

Style Position Property

- Sets or returns the type of positioning method used for an element.

```
document.getElementById("myDIV").style.position = "absolute";
```

- Syntax
  - Return: `object.style.position`
  - Set: `object.style.position = "static|absolute|fixed|relative|initial|inherit"`
- Property Values

Value	Description
static	Elements renders in order, a they appear in the document flow. This is default
Absolute	The element is positioned relative to its first positioned (not static) ancestor.
Fixed	The element is positioned relative to the browser window.
Relative	The element is positioned relative to its normal position, so "left:20" adds 20 pixels to the elements LEFT position.
Initial	Sets the property to its default value.
Inherit	Inherits this property from its parent element.

Style Top Property

- Sets or returns the top position of a positioned element.

```
document.getElementById("myBtn").style.top = "100px";
```

- Including: padding, scrollbar, border and margin.
- Syntax
  - Return: `object.style.top`
  - Set: `object.style.top = "auto|length|%|initial|inherit"`
- Property Values

Values	Description
auto	Lets the browser set the top position. This is default.
length	Defines the top position in length units. Negative values are allowed.
%	Sets the top position in % of the height of the parent element.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

8

© Firebrand Training Ltd

Style Border Property

- Sets or returns up to three separate border properties, in a shorthand form.
- With this property, you can set/return one or more of the following (in any order):
  - border-width
  - border-style
  - border-color

Property Values

Parameter	Description
width	Sets the width of the borders.
style	Sets the style of the borders.
color	Sets the color of the borders.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

- Syntax:
  - *Return the border property:* *object*.style.border
  - Set the border property: *object*.style.border = *"width style color*|initial|inherit"

9

Style Padding Property

- The padding property sets or returns the padding of an element.
- Padding inserts the space within the border of an element.
- Takes one to four values:
  - **One** value, like: div {padding: 50px} - all four sides will have a padding of 50px
  - **Two** values, like: div {padding: 50px 10px} - the top and bottom padding will be 50px, left and right padding will be 10px
  - **Three** values, like: div {padding: 50px 10px 20px} - the top padding will be 50px, left and right padding will be 10px, bottom padding will be 20px
  - **Four** values, like: div {padding: 50px 10px 20px 30px} - the top padding will be 50px, right padding will be 10px, bottom padding will be 20px, left padding will be 30px

10

© Firebrand Training Ltd

Style Padding Property (continued)

Syntax

- Return: *object*.style.padding
- Set: *object*.style.padding = *"%|length|initial|inherit"*

Property Values

Value	Description
%	Defines the padding in % of the width of the parent element.
length	Defines the padding in length units.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

11

Navigator online Property

The onLine property returns a Boolean value that specifies whether the browser is in online or offline mode.

This is a read-only property.

Browser Support:

				
14.0	Yes	3.5	5.0.4	Yes

Syntax: navigator.onLine

Example

Find out whether the browser is online:

```
var x = "Is the browser online? " + navigator.onLine;
```

The result of *x* will be:

```
Is the browser online? true
```

12

© Firebrand Training Ltd

JQuery Effect show() Method

- The show() method shows the hidden, selected elements.
- Syntax: $(*selector*).show(*speed*,*easing*,*callback*)

Parameter	Description
speed	Optional. Specifies the speed of the show effect. Default value is 400 milliseconds Possible values: • milliseconds • "slow" • "fast"
easing	Optional. Specifies the speed of the element in different points of the animation. Default value is "swing“ Possible values: • "swing" - moves slower at the beginning/end, but faster in the middle • "linear" - moves in a constant speed
callback	Optional. A function to be executed after the show() method is completed

13

FileReader Object

- Provides methods to asynchronously read a File or Blob, and events to obtain the results of these reads.
- The **FileReader** object has these types of members:
  - Events
  - Methods
  - Properties

Events

Event	Description
onabort	**ProgressEvent** that occurs when a read or creation operation is aborted by calling the **abort** method.
onerror	**ProgressEvent** that fires when an error occurs during a read operation by an **MSStreamReader** or **FileReader** object.
onload	**ProgressEvent** that occurs when a read operation by an **MSStreamReader** or **FileReader** object successfully completes.
onloadend	**ProgressEvent** that occurs when a read operation by an **MSStreamReader** or **FileReader** object completes, even if the request fails.
onloadstart	**ProgressEvent** that occurs when a read operation is started by an **MSStreamReader** or **FileReader** object.
onprogress	**ProgressEvent** that occurs when an **MSStreamReader** or **FileReader** object reports progress about a read operation.

14

© Firebrand Training Ltd

FileReader Object (continued)

Methods

Method	Description
abort	Aborts a read operation by an **MSStreamReader** or **FileReader** object.
readAsArrayBuffer	Reads a **File**, **Blob**, **MSStream** into memory as an ArrayBuffer object.
readAsBinaryString	Reads the contents of a **Blob** or **File** as raw binary.
readAsDataURL	Reads a **File** or **Blob** object into memory as a data URL string.
readAsText	Reads a **File**, **Blob**, or **MSStream** object into memory as a text string.

Properties

Property	Access type	Description
error	Read-only	The error that occurred while reading a **File**, **Blob**, or **MSStream** object.
readyState		Contains a constant indicating the current state of the **FileReader** or **MSStreamReader** object.
result		The result of the read operation.

1

JavaScript: The *this* Keyword

- In JavaScript, the thing called **this**, is the object that "owns" the JavaScript code.
- The value of **this**, when used in a function, is the object that "owns" the function.
- The value of **this**, when used in an object, is the object itself.
- The **this** keyword in an object constructor does not have a value. It is only a substitute for the new object.
- The value of **this** will become the new object when the constructor is used to create an object.
- Note that **this** is not a variable. It is a keyword. You cannot change the value of **this**.

16

© Firebrand Training Ltd

HTML5 Web Workers

- A web worker is a JavaScript running in the background, without affecting the performance of the page.
- Full Web Worker Example Code

```
<!DOCTYPE html>
<html>
<body>

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>

<script>
var w;

function startWorker() {
    if(typeof(Worker) !== "undefined") {
        if(typeof(w) == "undefined") {
            w = new Worker("demo_workers.js");
        }
        w.onmessage = function(event) {
            document.getElementById("result").innerHTML = event.data;
        };
    } else {
        document.getElementById("result").innerHTML = "Sorry! No Web Worker support.";
    }
}

function stopWorker() {
    w.terminate();
    w = undefined;
}
</script>

</body>
</html>
```

Since web workers are in external files, they do not have access to the following JavaScript objects:

- The window object
- The document object
- The parent object

17

HTML DOM setAttribute() Method

- The setAttribute() method adds the specified attribute to an element, and gives it the specified value.
- If the specified attribute already exists, only the value is set/changed.
- Syntax: *element*.setAttribute(*attributename*, *attributevalue*)
- Property Values

Parameter	Type	Description
attributename	String	Required. The name of the attribute you want to add
attributevalue	String	Required. The value of the attribute you want to add

18

© Firebrand Training Ltd

CSS Attribute Selectors

- It is possible to style HTML elements that have specific attributes or attribute values.

Attribute Selector	Description	Example
[attribute]	Selects elements with a specified attribute.	`a[target] { background-color: yellow; }`
[attribute="value"]	Selects elements with a specified attribute value.	`a[target="_blank"] { background-color: yellow; }`
[attribute~="value"]	Selects elements with an attribute value containing a specified word.	`[title~="flower"] { border: 5px solid yellow; }`
[attribute|="value"]	Selects elements with the specified attribute starting with a specified value.	`[class|="top"] { background: yellow; }`

CSS Attribute Selectors (continued)

Attribute Selector	Description	Example
[attribute^="value"]	Selects elements whose attribute value begins with a specified value.	`[class^="top"] { background: yellow; }`
[attribute$="value"]	Selects elements whose attribute value ends with a specified value.	`[class$="test"] { background: yellow; }`
[attribute*="value"]	Selects elements whose attribute value contains a specified value.	`[class*="te"] { background: yellow; }`

© Firebrand Training Ltd

Cross-Origin Resource Sharing (CORS)

- Lets JavaScript overcome the same-origin policy security restriction imposed by browsers.
- Same-origin policy: Means that your JavaScript can only make AJAX calls back to the same origin of the containing Web page.
- CORS lets servers indicate which origins are allowed to call them.
- CORS is enforced by browsers but must be implemented on the server.

21

HTML DOM value Property

- Sets or returns the value of the attribute.
- Syntax:
  - Returns: attribute.value
  - Sets: attribute.value = *value*
- Example:

```
var x = document.getElementsByTagName("IMG")[0];
x.getAttributeNode("src").value = "pic_bulbon.gif";
```

- Property Values

Value	Description
value	Specifies the value of the attribute

22

© Firebrand Training Ltd

isPrototypeOf JavaScript Method

- Determines whether an object exists in another object's prototype chain.
- Syntax: prototype.isPrototypeOf(object)
- Parameters:

Parameter	Description
prototype	Required. An object prototype.
object	Required. Another object whose prototype chain is to be checked.

- Example:

```
function Rectangle() {
}

var rec = new Rectangle();

document.write(Rectangle.prototype.isPrototypeOf(rec));

// Output: true
```

23

jQuery trigger() Method

- The trigger() method triggers the specified event and the default behavior of an event for the selected elements.
- Similar to the triggerHandler() method, except that triggerHandler() does not trigger the default behavior of the event.
- Syntax : $(*selector*).trigger(*event*,*eventObj*,*param1*,*param2*,...)

Parameter	Description
event	Required. Specifies the event to trigger for the specified element. Can be a custom event, or any of the standard events
param1,*param2*,...	Optional. Additional parameters to pass on to the event handler. Additional parameters are especially useful with custom events

- Example:

```
$("button").click(function(){
    $("input").trigger("select");
});
```

24

© Firebrand Training Ltd

HTML DOM Media Events

Event	Description	DOM
onabort	The event occurs when the loading of a media is aborted	3
oncanplay	The event occurs when the browser can start playing the media (when it has buffered enough to begin)	3
oncanplaythrough	The event occurs when the browser can play through the media without stopping for buffering	3
ondurationchange	The event occurs when the duration of the media is changed	3
onemptied	The event occurs when something bad happens and the media file is suddenly unavailable	3
onended	The event occurs when the media has reach the end	3
onerror	The event occurs when an error occurred during the loading of a media file	3
onloadeddata	The event occurs when media data is loaded	3

HTML DOM Media Events (continued)

Event	Description	DOM
onloadedmetadata	The event occurs when meta data (like dimensions and duration) are loaded	3
onloadstart	The event occurs when the browser starts looking for the specified media	3
onpause	The event occurs when the media is paused either by the user or programmatically	3
onplay	The event occurs when the media has been started or is no longer paused	3
onplaying	The event occurs when the media is playing after having been paused or stopped for buffering	3
onprogress	The event occurs when the browser is in the process of getting the media data (downloading the media)	3
onratechange	The event occurs when the playing speed of the media is changed	3
onseeked	The event occurs when the user is finished moving/skipping to a new position in the media	3

© Firebrand Training Ltd

HTML DOM Media Events (continued)

Event	Description	DOM
onseeking	The event occurs when the user starts moving/skipping to a new position in the media	3
onstalled	The event occurs when the browser is trying to get media data, but data is not available	3
onsuspend	The event occurs when the browser is intentionally not getting media data	3
ontimeupdate	The event occurs when the playing position has changed (like when the user fast forwards to a different point in the media)	3
onvolumechange	The event occurs when the volume of the media has changed (includes setting the volume to "mute")	3
onwaiting	The event occurs when the media has paused but is expected to resume (like when the media pauses to buffer more data)	3

27

jQuery text() Method

- The text() method sets or returns the text content of the selected elements.
- To **return** content, it returns the text content of all matched elements (HTML markup will be removed).
- To **set** content, it overwrites the content of ALL matched elements.

Parameter	Description
content	Required. Specifies the new text content for the selected elements **Note:** Special characters will be encoded
function(*index*,*currentcontent*)	Optional. Specifies a function that returns the new text content for the selected • elements*index* - Returns the index position of the element in the set • *currentcontent* - Returns current content of selected elements

- Syntax:
  - Return: $(*selector*).text()
  - Set: $(*selector*).text(*content*)
  - Set using a function: $(*selector*).text(function*(index,currentcontent)*)

28

© Firebrand Training Ltd

HTML DOM appendChild() Method

- The appendChild() method appends a node as the last child of a node.
- You can also use this method to move an element from one element to another.
- Syntax: *node*.appendChild(*node*)
- Example:

```
var node = document.getElementById("myList2").lastChild;
document.getElementById("myList1").appendChild(node);
```

- Parameter Values

Parameter	Type	Description
node	Node object	Required. The node object you want to append

29

Style display Property

- The display property sets or returns the element's display type.
- Elements in HTML are mostly "inline" or "block" elements:
  - An inline element has floating content on its left and right side.
  - A block element fills the entire line, and nothing can be displayed on its left or right side.
- The display property also allows the author to show or hide an element.
- Syntax:
  - Return: *object*.style.display
  - Set: *object*.style.display = *value*
- *Property Values*

Value	Description
block	Element is rendered as a block-level element
compact	Element is rendered as a block-level or inline element. Depends on context
flex	Element is rendered as a block-level flex box. New in CSS3

© Firebrand Training Ltd

Style display Property (continued)

Value	Description
inherit	The value of the display property is inherited from parent element
inline	Element is rendered as an inline element. This is default
inline-block	Element is rendered as a block box inside an inline box
inline-flex	Element is rendered as a inline-level flex box. New in CSS3
inline-table	Element is rendered as an inline table (like <table>), with no line break before or after the table
list-item	Element is rendered as a list
marker	This value sets content before or after a box to be a marker (used with :before and :after pseudo-elements. Otherwise this value is identical to "inline")
none	Element will not be displayed
run-in	Element is rendered as block-level or inline element. Depends on context
table	Element is rendered as a block table (like <table>), with a line break before and after the table

31

Style display Property (continued)

Value	Description
table-caption	Element is rendered as a table caption (like <caption>)
table-cell	Element is rendered as a table cell (like <td> and <th>)
table-column	Element is rendered as a column of cells (like <col>)
table-column-group	Element is rendered as a group of one or more columns (like <colgroup>)
table-footer-group	Element is rendered as a table footer row (like <tfoot>)
table-header-group	Element is rendered as a table header row (like <thead>)
table-row	Element is rendered as a table row (like <tr>)
table-row-group	Element is rendered as a group of one or more rows (like <tbody>)
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

32

© Firebrand Training Ltd

onMessage Event

- The onmessage event occurs when a message is received through an event source.
- The event object for the onmessage event supports the following properties:
  - data - Contains the actual message
  - origin - The URL of the document that invoked the event
  - lastEventId - the identifier of the last message seen in the event stream
- Related Events
  - onopen - Occurs when a connection to the server is open
  - onerror - Occurs when a problem occurs
- Syntax : *object*.onmessage = function(){*myScript*};
- Syntax using the addEventListener() method:
  object.addEventListener("message", *myScript*);

33

HTML DOM blur() & focus() Methods

- The blur() method is used to remove focus from an element.
- Syntax: *HTMLElementObject*.blur()
- Example: `document.getElementById("myAnchor").blur();`

- The focus() method is used to give focus to an element (if it can be focused).
- Syntax: *HTMLElementObject*.focus()
- Example: `document.getElementById("myAnchor").focus();`

34

© Firebrand Training Ltd

AJAX - The XMLHttpRequest Object

- The XMLHttpRequest object can be used to exchange data with a web server behind the scenes.
- Able update parts of a web page, without reloading the whole page.
- XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method*, *url*, *async*, *user*, *psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password

35

AJAX - The XMLHttpRequest Object (continued)

- XMLHttpRequest Object Methods (continued)

Method	Description
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

- XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready

36

© Firebrand Training Ltd

AJAX - The XMLHttpRequest Object (continued)

- XMLHttpRequest Object Properties

Property	Description
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

37

HTML DOM addEventListener() Method

- The addEventListener() method attaches an event handler to the specified element.
- Syntax: *element*.addEventListener(*event*, *function*, *useCapture*)
- Parameter Values

Parameter	Description
event	Required. A String that specifies the name of the event. **Note:** Do not use the "on" prefix. For example, use "click" instead of "onclick".
function	Required. Specifies the function to run when the event occurs. When the event occurs, an event object is passed to the function as the first parameter. The type of the event object depends on the specified event. For example, the "click" event belongs to the MouseEvent object.
useCapture	Optional. A Boolean value that specifies whether the event should be executed in the capturing or in the bubbling phase. Possible values: • true - The event handler is executed in the capturing phase • false- Default. The event handler is executed in the bubbling phase

© Firebrand Training Ltd

Button disabled Property

- The disabled property sets or returns whether a button is disabled, or not.
- A disabled element is unusable and un-clickable. Disabled elements are usually rendered in gray by default in browsers.
- Syntax:
  - Return: *buttonObject*.disabled
  - Set: *buttonObject*.disabled = true|false
- Property Values

Property	Description
true|false	Specifies whether a button should be disabled or not • true - The button is disabled • false - Default. The button is not disabled

39

CSS Pseudo-classes

- A pseudo-class is used to define a special state of an element.

Selector	Example	Example Description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over

40

© Firebrand Training Ltd

CSS Pseudo-classes (continued)

Selector	Example	Example Description
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent

41

CSS Pseudo-classes (continued)

Selector	Example	Example Description
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute

42

© Firebrand Training Ltd

CSS Pseudo-classes (continued)

Selector	Example	Example Description
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

43

CSS Pseudo-classes (continued)

All CSS Pseudo Elements

Selector	Example	Example Description
::after	p::after	Insert content after every <p> element
::before	p::before	Insert content before every <p> element
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

44

© Firebrand Training Ltd

jQuery ajax() Method

- Used to perform an AJAX (asynchronous HTTP) request.
- Mostly used for requests where the other methods cannot be used.
- Syntax : `$.ajax({name:value, name:value, ... })`
- The parameters specifies one or more name/value pairs for the AJAX request.
- Possible names/values in the table below:

Name	Value/Description
async	A Boolean value indicating whether the request should be handled asynchronous or not. Default is true
beforeSend(*xhr*)	A function to run before the request is sent
cache	A Boolean value indicating whether the browser should cache the requested pages. Default is true

45

jQuery ajax() Method (continued)

Name	Value/Description
complete(*xhr,status*)	A function to run when the request is finished (after success and error functions)
contentType	The content type used when sending data to the server. Default is: "application/x-www-form-urlencoded"
context	Specifies the "this" value for all AJAX related callback functions
data	Specifies data to be sent to the server
dataFilter(*data,type*)	A function used to handle the raw response data of the XMLHttpRequest
dataType	The data type expected of the server response.
error(*xhr,status,error*)	A function to run if the request fails.
global	A Boolean value specifying whether or not to trigger global AJAX event handles for the request. Default is true
ifModified	A Boolean value specifying whether a request is only successful if the response has changed since the last request. Default is: false.

© Firebrand Training Ltd

jQuery ajax() Method (continued)

Name	Value/Description
jsonp	A string overriding the callback function in a jsonp request
jsonpCallback	Specifies a name for the callback function in a jsonp request
password	Specifies a password to be used in an HTTP access authentication request.
processData	A Boolean value specifying whether or not data sent with the request should be transformed into a query string. Default is true
scriptCharset	Specifies the charset for the request
success(*result*,*status*,*xhr*)	A function to be run when the request succeeds
timeout	The local timeout (in milliseconds) for the request
traditional	A Boolean value specifying whether or not to use the traditional style of param serialization
type	Specifies the type of request. (GET or POST)
url	Specifies the URL to send the request to. Default is the current page
username	Specifies a username to be used in an HTTP access authentication request
xhr	A function used for creating the XMLHttpRequest object

CSS3 text-shadow Property

- The text-shadow property adds shadow to text.
- This property accepts a comma-separated list of shadows to be applied to the text.

Default Value:	none
Inherited:	yes
Animatable:	yes
Version:	CSS3
Javascript syntax:	*object*.style.textShadow="2px 5px 5px red"

- CSS Syntax:

```
text-shadow: h-shadow v-shadow blur-radius color|none|initial|inherit;
```

48

© Firebrand Training Ltd

CSS3 text-shadow Property (continued)

- Property Values

Value	Description
h-shadow	Required. The position of the horizontal shadow. Negative values are allowed
v-shadow	Required. The position of the vertical shadow. Negative values are allowed
blur-radius	Optional. The blur radius. Default value is 0
color	Optional. The color of the shadow.
none	Default value. No shadow
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

- Example: Basic text-shadow

```
h1 {
    text-shadow: 2px 2px #ff0000;
}
```

49

CSS3 grid-columns Property

- The grid-columns property specifies the width of each column in the grid.

Default value:	**none**
Inherited:	no
Version:	CSS3
JavaScript syntax:	object.style.gridColumns="50% * * 200px"

- CSS Syntax: `grid-columns: length|%|none|inherit;`

Value	Description
length	Refers to the grid of the containing block
%	Refers to the width of the containing block
none	
inherit	

Example:

```
div
{
grid-columns:50% * * 200px;
}
```

50

© Firebrand Training Ltd

CSS3 grid-rows Property

- The grid-rows property specifies the height of each row in the grid.

Default value:	none
Inherited:	no
Version:	CSS3
JavaScript syntax:	object.style.gridRows="100px (30px 60px)"

- CSS Syntax: `grid-rows: length|%|none|inherit;`

Value	Description
length	Refers to the grid of the containing block
%	Refers to the height of the containing block
none	
inherit	

Example:

```
div
{
grid-rows:100px (30px 60px);
}
```

51

Style left Property

- Sets or returns the left position of a positioned element.
- Specifies the left position of the element including padding, scrollbar, border and margin.
- Example: `document.getElementById("myBtn").style.left = "100px";`
- Syntax:
  - Return: `object.style.left`
  - Set: `object.style.left = "auto|length|%|initial|inherit"`
- Property Values

Value	Description
auto	Lets the browser set the left position. This is default
length	Defines the left position in length units. Negative values are allowed
%	Sets the left position in % of the width of the parent element
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

© Firebrand Training Ltd

CSS3 opacity Property

- Sets the opacity level for an element.
- Describes the transparency-level, where 1 is not transparent at all, 0.5 is 50% see-through, and 0 is completely transparent.

opacity 0.2

opacity 0.5

opacity 1 (default)

Default value:	1
Inherited:	no
Animatable:	yes, *see individual properties*.
Version:	CSS3
JavaScript syntax:	*object*.style.opacity="0.5"

CSS3 opacity Property (continued)

- Syntax: `opacity: number|initial|inherit;`
- Property Values:

Value	Description
number	Specifies the opacity. From 0.0 (fully transparent) to 1.0 (fully opaque)
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

- Example:

```
div {
    opacity: 0.5;
}
```


© Firebrand Training Ltd

CSS3 Colors

- CSS supports color names, hexadecimal and RGB colors.

- CSS3 also introduces:
  - RGBA colors
  - HSL colors
  - HSLA colors
  - opacity

55

CSS3 Colors: RGBA Colors

- RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
- An RGBA color value is specified with: rgba(red, green, blue, alpha).
- The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).
- Example:

```
#p1 {background-color: rgba(255, 0, 0, 0.3);}  /* red with opacity */
#p2 {background-color: rgba(0, 255, 0, 0.3);}  /* green with opacity */
#p3 {background-color: rgba(0, 0, 255, 0.3);}  /* blue with opacity */
```

rgba(255, 0, 0, 0.2);

rgba(255, 0, 0, 0.4);

rgba(255, 0, 0, 0.6);

rgba(255, 0, 0, 0.8);

56

© Firebrand Training Ltd

CSS3 Colors: HSL Colors

- HSL stands for Hue, Saturation and Lightness.
- An HSL color value is specified with: hsl(hue, saturation, lightness).
- Hue is a degree on the color wheel (from 0 to 360):
  - 0 (or 360) is red
  - 120 is green
  - 240 is blue
- Saturation is a percentage value: 100% is the full color.
- Lightness is also a percentage; 0% is dark (black) and 100% is white.
- Examples:

```
#p1 {background-color: hsl(120, 100%, 50%);}  /* green */
#p2 {background-color: hsl(120, 100%, 75%);}  /* light green */
#p3 {background-color: hsl(120, 100%, 25%);}  /* dark green */
#p4 {background-color: hsl(120, 60%, 70%);}   /* pastel green */
```

hsl(0, 100%, 30%);

hsl(0, 100%, 50%);

hsl(0, 100%, 70%);

hsl(0, 100%, 90%);

57

CSS3 Colors: HSLA Colors

- HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.
- Specified with: hsla(hue, saturation, lightness, alpha), where the alpha parameter defines the opacity.
- The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).
- Example:

```
#p1 {background-color: hsla(120, 100%, 50%, 0.3);}  /* green with opacity */
#p2 {background-color: hsla(120, 100%, 75%, 0.3);}  /* light green with opacity */
#p3 {background-color: hsla(120, 100%, 25%, 0.3);}  /* dark green with opacity */
#p4 {background-color: hsla(120, 60%, 70%, 0.3);}   /* pastel green with opacity */
```

hsla(0, 100%, 30%, 0.3);

hsla(0, 100%, 50%, 0.3);

hsla(0, 100%, 70%, 0.3);

hsla(0, 100%, 90%, 0.3);

58

© Firebrand Training Ltd

CSS3 Colors: Opacity

- Sets the opacity for the whole element (both background color and text will be opaque/transparent).
- Must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).
- Example:

```
#p1 {background-color:rgb(255,0,0);opacity:0.6;}  /* red with opacity */
#p2 {background-color:rgb(0,255,0);opacity:0.6;}  /* green with opacity */
#p3 {background-color:rgb(0,0,255);opacity:0.6;}  /* blue with opacity */
```

rgb(255, 0, 0);opacity:0.2;

rgb(255, 0, 0);opacity:0.4;

rgb(255, 0, 0);opacity:0.6;

rgb(255, 0, 0);opacity:0.8;

CSS text-transform Property

- The text-transform property controls the capitalization of text.

Default value:	none
Inherited:	yes
Version:	CSS1
JavaScript syntax:	object.style.textTransform="uppercase"

Example:

```
p.uppercase {
    text-transform: uppercase;
}
```

- CSS Syntax: `text-transform: none|capitalize|uppercase|lowercase|initial|inherit;`
- Property Values

Value	Description
none	No capitalization. The text renders as it is. This is default
capitalize	Transforms the first character of each word to uppercase
uppercase	Transforms all characters to uppercase
lowercase	Transforms all characters to lowercase
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

© Firebrand Training Ltd

CSS3 word-wrap Property

- Allows long words to be able to be broken and wrap onto the next line.

Example:

```
p.test {
    word-wrap: break-word;
}
```

Default value:	normal
Inherited:	yes
Animatable:	no
Version:	CSS3
JavaScript syntax:	*object*.style.wordWrap="break-word"

- CSS Syntax: `word-wrap: normal|break-word|initial|inherit;`
- Property Values

Value	Description
normal	Break words only at allowed break points
break-word	Allows unbreakable words to be broken
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

61

HTML5 SVG

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web
- SVG is a W3C recommendation
- The HTML <svg> element is a container for SVG graphics.
- SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

SVG Circle

```
<!DOCTYPE html>
<html>
<body>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
</svg>

</body>
</html>
```

© Firebrand Training Ltd

HTML5 SVG (continued)

SVG Rectangle

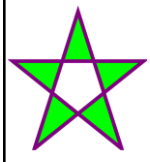
```
<svg width="400" height="100">
  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-
width:10;stroke:rgb(0,0,0)" />
</svg>
```

SVG Rounded Rectangle

```
<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
</svg>
```

HTML5 SVG (continued)

SVG Star



```
<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
```

SVG Logo


```
<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-family="Verdana" x="50" y="86">SVG</text>
  Sorry, your browser does not support inline SVG.
</svg>
```

© Firebrand Training Ltd

HTML5: Differences Between SVG and Canvas

- SVG is a language for describing 2D graphics in XML.
  - SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.
  - In SVG, each drawn shape is remembered as an object.
  - If attributes of an SVG object are changed, the browser can automatically re-render the shape.
- Canvas draws 2D graphics, on the fly (with a JavaScript).
  - Canvas is rendered pixel by pixel.
  - In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

65

HTML5: Differences Between SVG and Canvas (continued)

Canvas	SVG
• Resolution dependent • No support for event handlers • Poor text rendering capabilities • You can save the resulting image as .png or .jpg • Well suited for graphic-intensive games	• Resolution independent • Support for event handlers • Best suited for applications with large rendering areas (Google Maps) • Slow rendering if complex (anything that uses the DOM a lot will be slow) • Not suited for game applications

66

© Firebrand Training Ltd

HTML5 <aside> Tag

- The <aside> tag defines some content aside from the content it is placed in.
- The aside content should be related to the surrounding content.
- Example:

```
<p>My family and I visited The Epcot center this summer.</p>

<aside>
  <h4>Epcot Center</h4>
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

67

HTML5 <footer> Tag

- Defines a footer for a document or section.
- Should contain information about its containing element.
- A <footer> element typically contains:
  - authorship information
  - copyright information
  - contact information
  - sitemap
  - back to top links
  - related documents
- You can have several <footer> elements in one document.
- Example:

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

68

© Firebrand Training Ltd

HTML5 <article> Tag

- Specifies independent, self-contained content.
- An article should make sense on its own and it should be possible to distribute it independently from the rest of the site.
- Potential sources for the <article> element:
  - Forum post
  - Blog post
  - News story
  - Comment
- Example

```
<article>
  <h1>Google Chrome</h1>
  <p>Google Chrome is a free, open-source web browser developed by Google, released in
2008.</p>
</article>
```

69

HTML5 <section> Tag

- Defines sections in a document, such as chapters, headers, footers, or any other sections of the document.
- Example:

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>
```

70

© Firebrand Training Ltd

HTML5 <input> pattern Attribute

- Specifies a regular expression that the <input> element's value is checked against.
- Works with the following input types: text, date, search, url, tel, email, and password.
- Syntax: `<input pattern="regexp">`
- Attribute Values:

Value	Description
regexp	Specifies a regular expression that the <input> element's value is checked against

- Example:

```
<form action="/action_page.php">
Country code: <input type="text" name="country_code"
pattern="[A-Za-z]{3}" title="Three letter country code">
<input type="submit">
</form>
```

71

HTML5 Video

- The HTML5 <video> element specifies a standard way to embed a video in a web page.
- How it works
  - The **controls** attribute adds video controls, like play, pause, and volume.
  - It is a good idea to always include **width** and **height** attributes. If height and width are not set, the page might flicker while the video loads.
  - The **<source>** element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.
  - The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video> element.
- Example:

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

72

© Firebrand Training Ltd

HTML5 Video (continued)

- HTML5 defines DOM methods, properties, and events for the <video> element.
- Allows you to load, play, and pause videos, as well as setting duration and volume.
- There are also DOM events that can notify you when a video begins to play, is paused, etc.

- HTML Video - Media Types

File Format	Media Type
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

- HTML5 Video Tags

Tag	Description
<video>	Defines a video or movie
<source>	Defines multiple media resources for media elements, such as <video> and <audio>
<track>	Defines text tracks in media players

HTML5 Attributes

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple
- pattern (regexp)
- placeholder
- required
- step
- and the following attributes for <form>:
- autocomplete
- novalidate

© Firebrand Training Ltd

HTML5 Input Attributes

- The **value** Attribute: specifies the initial value for an input field:

```
<form action="">
First name:<br>
<input type="text" name="firstname" value="John">
</form>
```

- The **readonly** Attribute: specifies that the input field is read only (cannot be changed):

```
<form action="">
First name:<br>
<input type="text" name="firstname" value="John" readonly>
</form>
```

- The **disabled** Attribute: specifies that the input field is disabled.

```
<form action="">
First name:<br>
<input type="text" name="firstname" value="John" disabled>
</form>
```

- The **size** Attribute: specifies the size (in characters) for the input field:

```
<form action="">
First name:<br>
<input type="text" name="firstname" value="John" size="40">
</form>
```

75

HTML5 Input Attributes (continued)

- The **maxlength** Attribute: specifies the maximum allowed length for the input field:

```
<form action="">
First name:<br>
<input type="text" name="firstname" maxlength="10">
</form>
```

- The **autocomplete** Attribute: specifies whether a form or input field should have autocomplete on or off.

- The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.
- When autocomplete is on, the browser automatically complete the input values based on values that the user has entered before.

```
<form action="/action_page.php" autocomplete="on">
  First name:<input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  E-mail: <input type="email" name="email" autocomplete="off"><br>
  <input type="submit">
</form>
```

76

© Firebrand Training Ltd

HTML5 Input Attributes (continued)

- The **novalidate** Attribute: is a <form> attribute. When present, novalidate specifies that the form data should not be validated when submitted.

```
<form action="/action_page.php" novalidate>
  E-mail: <input type="email" name="user_email">
  <input type="submit">
</form>
```

- The **autofocus** Attribute: specifies that the input field should automatically get focus when the page loads.

```
First name:<input type="text" name="fname" autofocus>
```

- The **form** Attribute: specifies one or more forms an <input> element belongs to.

```
<form action="/action_page.php" id="form1">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
</form>

Last name: <input type="text" name="lname" form="form1">
```

HTML5 Input Attributes (continued)

- The **formaction** Attribute: specifies the URL of a file that will process the input control when the form is submitted.
  - The formaction attribute overrides the action attribute of the <form> element.
  - The formaction attribute is used with type="submit" and type="image".

```
<form action="/action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formaction="/action_page2.php"
  value="Submit as admin">
</form>
```

- The **formenctype**: specifies how the form data should be encoded when submitted (only for forms with method="post").
  - The formenctype attribute overrides the enctype attribute of the <form> element.
  - The formenctype attribute is used with type="submit" and type="image".

```
<form action="/action_page_binary.asp" method="post">
  First name: <input type="text" name="fname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
  value="Submit as Multipart/form-data">
</form>
```

78

© Firebrand Training Ltd

HTML5 Input Attributes (continued)

- The **formmethod** Attribute: defines the HTTP method for sending form-data to the action URL.

- The formmethod attribute overrides the method attribute of the <form> element.
- The formmethod attribute can be used with type="submit" and type="image".

```
<form action="/action_page.php" method="get">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit">
  <input type="submit" formmethod="post" formaction="action_page_post.asp"
  value="Submit using POST">
</form>
```

- The **formnovalidate** Attribute: overrides the novalidate attribute of the <form> element.

- The formnovalidate attribute can be used with type="submit".

```
<form action="/action_page.php">
  E-mail: <input type="email" name="userid"><br>
  <input type="submit" value="Submit"><br>
  <input type="submit" formnovalidate value="Submit without validation">
</form>
```

79

HTML5 Input Attributes (continued)

- The **formtarget** Attribute: specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

- The formtarget attribute overrides the target attribute of the <form> element.
- The formtarget attribute can be used with type="submit" and type="image".

```
<form action="/action_page.php">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  <input type="submit" value="Submit as normal">
  <input type="submit" formtarget="_blank"
  value="Submit to a new window">
</form>
```

- The **height** and **width** Attribute: specify the height and width of an <input type="image"> element.

```
<input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
```

- Always specify the size of images. If the browser does not know the size, the page will flicker while images load.

80

© Firebrand Training Ltd

HTML5 Input Attributes (continued)

- The **list** Attribute: refers to a <datalist> element that contains pre-defined options for an <input> element

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

- The **min** and **max** Attributes: specify the minimum and maximum values for an <input> element.
  - The min and max attributes work with the following input types: number, range, date, datetime-local, month, time and week.

```
Enter a date before 1980-01-01:
<input type="date" name="bday" max="1979-12-31">

Enter a date after 2000-01-01:
<input type="date" name="bday" min="2000-01-02">

Quantity (between 1 and 5):
<input type="number" name="quantity" min="1" max="5">
```

81

HTML5 Input Attributes (continued)

- The **multiple** Attribute: specifies that the user is allowed to enter more than one value in the <input> element. Uses the following input types: email, and file.

```
Select images: <input type="file" name="img" multiple>
```

- The **pattern** Attribute: specifies a regular expression that the <input> element's value is checked against.Uses the following input types: text, search, url, tel, email, and password.

```
Country code: <input type="text" name="country_code" pattern="[A-Za-z]{3}"
title="Three letter country code">
```

- The **placeholder** attribute: specifies a hint that describes the expected value of an input field. It is displayed before the user enters a value. Uses the following input types: text, search, url, tel, email, and password.

```
<input type="text" name="fname" placeholder="First name">
```

© Firebrand Training Ltd

HTML5 Input Attributes (continued)

- The **required** Attribute: specifies that an input field must be filled out before submitting the form. Uses the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

```
Username: <input type="text" name="usrname" required>
```

- The **step** Attribute: specifies the legal number intervals for an <input> element. Uses with the following input types: number, range, date, datetime-local, month, time and week.

```
<input type="number" name="points" step="3">
```

83

CSS3 column-width Property

- Specifies a suggested, optimal width for the columns.

Default value:	auto
Inherited:	no
Animatable:	yes.
Version:	CSS3
JavaScript syntax:	object.style.columnWidth="100px"

- CSS Syntax: `column-width: auto|length|initial|inherit;`
- Property Values

Value	Description
auto	Default value. The column width will be determined by the browser
length	A length that specifies the width of the columns
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

84

© Firebrand Training Ltd

HTML5 <nav> Tag

- Defines a set of navigation links.
- The <nav> element is intended only for major block of **navigation links.**
- Example

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

85

HTML5 Input Types

- Text : **<input type="text">** defines a **one-line text input field:**

```
<form>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

- Password: **<input type="password">** defines a **password field:**

```
<form>
  User name:<br>
  <input type="text" name="username"><br>
  User password:<br>
  <input type="password" name="psw">
</form>
```

86

© Firebrand Training Ltd

HTML5 Input Types (continued)

- Submit : **<input type="submit">** defines a button for **submitting** form data to a **form-handler**.

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

- Reset: **<input type="reset">** defines a **reset button** that will reset all form values to their default values:

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```

87

HTML5 Input Types (continued)

- Radio : **<input type="radio">** defines a **radio button.**

```
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```

- Checkbox: **<input type="checkbox">** defines a **checkbox**.

```
<form>
  <input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
  <input type="checkbox" name="vehicle2" value="Car"> I have a car
</form>
```

- Button: **<input type="button">** defines a **button:**

```
<input type="button" onclick="alert('Hello World!')" value="Click Me!">
```

- Color: The **<input type="color">** is used for input fields that should contain a color.

```
<form>
  Select your favorite color:
  <input type="color" name="favcolor">
</form>
```

88

© Firebrand Training Ltd

HTML5 Input Types (continued)

- Date: The **<input type="date">** is used for input fields that should contain a date.

```
<form>
  Birthday:
  <input type="date" name="bday">
</form>
```

- Datetime-local: The **<input type="datetime-local">** specifies a date and time input field, with no time zone.

```
<form>
  Birthday (date and time):
  <input type="datetime-local" name="bdaytime">
</form>
```

- Email: The **<input type="email">** is used for input fields that should contain an e-mail address.

```
<form>
  E-mail:
  <input type="email" name="email">
</form>
```

- Month: The **<input type="month">** allows the user to select a month and year.

```
<form>
  Birthday (month and year):
  <input type="month" name="bdaymonth">
</form>
```

89

HTML5 Input Types (continued)

- Number: The **<input type="number">** defines a **numeric** input field. You can also set restrictions on what numbers are accepted.

```
<form>
  Quantity (between 1 and 5):
  <input type="number" name="quantity" min="1" max="5">
</form>
```

Attribute	Description
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

© Firebrand Training Ltd

HTML5 Input Types (continued)

- Range: The **<input type="range">** defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. Able to set restrictions on what numbers are accepted with the min, max, and step attributes:

```
<form>
  <input type="range" name="points" min="0" max="10">
</form>
```

- Search: The **<input type="search">** is used for search fields (a search field behaves like a regular text field).

```
<form>
  Search Google:
  <input type="search" name="googlesearch">
</form>
```

- Tel: The **<input type="tel">** is used for input fields that should contain a telephone number.

```
<form>
  Telephone:
  <input type="tel" name="usrtel">
</form>
```

91

HTML5 Input Types (continued)

- Time: The **<input type="time">** allows the user to select a time (no time zone).

```
<form>
  Select a time:
  <input type="time" name="usr_time">
</form>
```

- Url: The **<input type="url">** is used for input fields that should contain a URL address.

```
<form>
  Add your homepage:
  <input type="url" name="homepage">
</form>
```

- Week: The **<input type="week">** allows the user to select a week and year.

```
<form>
  Select a week:
  <input type="week" name="week_year">
</form>
```

92

© Firebrand Training Ltd

HTML5 onblur Event Attribute

- The **onblur** attribute fires the moment that the element loses focus.
- Onblur is most often used with form validation code (e.g. when the user leaves a form field).
- Syntax: `<element onblur="script">`
- Attribute Values:

Value	Description
script	The script to be run on onblur

- Example:

```
<input type="text" name="fname" id="fname" onblur="myFunction()">
```

93

HTML5 onfocus Event Attribute

- The **onfocus** attribute fires the moment that the element gets focus.
- Onfocus is most often used with <input>, <select>, and <a>.
- Syntax: `<element onfocus="script">`
- Attribute Values

Value	Description
script	The script to be run on onfocus

- Example:

```
<input type="text" id="fname" onfocus="myFunction(this.id)">
```

94

© Firebrand Training Ltd

JavaScript: Using FormData Objects

- The FormData object lets you compile a set of key/value pairs to send using XMLHttpRequest.
- Primarily intended for use in sending form data.
- Can be used independently from forms in order to transmit keyed data.
- The transmitted data is in the same format that the form's submit() method would use to send the data if the form's encoding type were set to multipart/form-data.

```
var formData = new FormData();

formData.append("username", "Groucho");
formData.append("accountnum", 123456); // number 123456 is immediately converted to a string "123456"

// HTML file input, chosen by user
formData.append("userfile", fileInputElement.files[0]);

// JavaScript file-like object
var content = '<a id="a"><b id="b">hey!</b></a>'; // the body of the new file...
var blob = new Blob([content], { type: "text/xml"});

formData.append("webmasterfile", blob);

var request = new XMLHttpRequest();
request.open("POST", "http://foo.com/submitform.php");
request.send(formData);
```

FileSystem Api

- The File and Directory Entries API interface FileSystem is used to represent a file system.
- These objects can be obtained from the filesystem property on any file system entry.
- There are two ways to get access to a FileSystem object:

- Directly ask for one representing a sandboxed file system created just for your web app directly by calling window.requestFileSystem(). If that call is successful, it executes a callback handler, which receives as a parameter a FileSystem object describing the file system.
- You can get it from a file system entry object, through its filesystem property.

- Properties

FileSystem.name Read only

A USVString representing the file system's name. This name is unique among the entire list of exposed file systems.

FileSystem.root Read only

A FileSystemDirectoryEntry object which represents the file system's root directory. Through this object, you can gain access to all files and directories in the file system.

96

© Firebrand Training Ltd

FileSystem Api (continued)

- The File and Directory Entries API includes both asynchronous and synchronous versions of the interfaces.
- The asynchronous API can be used in cases where you don't want an outstanding operation to block the UI.
- The synchronous API, on the other hand, allows for simpler programming model, but it must be used with WebWorkers.

- Usefulness of the API
  - It lets apps have offline and storage features that involve large binary blobs.
  - It can improve performance by letting an app pre-fetch assets in the background and cache locally.
  - It lets users of your web app directly edit a binary file that's in their local file directory.
  - It provides a storage API that is already familiar to your users, who are used to working with file systems.

97

HTML <input> required Attribute

- A **boolean** attribute.
- Specifies that an input field must be filled out before submitting the form.
- Works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.
- Syntax: `<input required>`
- Example:

```
<form action="/action_page.php">
  Username: <input type="text" name="usrname" required>
  <input type="submit">
</form>
```

98

© Firebrand Training Ltd

JavaScript: Window alert() Method

- Displays an alert box with a specified message and an OK button.
- Syntax: `alert(message)`
- Parameter Values:

Parameter	Type	Description
message	String	Optional. Specifies the text to display in the alert box, or an object converted into a string and displayed

- Example:

```
alert("Hello! I am an alert box!!");
```

99

Cascade in CSS

- The *cascade* is a fundamental feature of CSS.
- It is an algorithm defining how to combine properties values originating from different sources.
- Cascading Order: Determines how to find the value to apply for each property for each document element.

1. It first filters all the rules from the different sources to keep only the rules that apply to a given element.

2. Then it sorts these rules according to their importance, that is, whether or not they are followed by !important, and by their origin. The cascade is in ascending order: (see image on right)

3. In case of equality, the specificity of a value is considered to choose one or the other.

	Origin	Importance
1	user agent	normal
2	user	normal
3	author	normal
4	CSS Animations	*see below*
5	author	!important
6	user	!important
7	user agent	!important

100

© Firebrand Training Ltd

JavaScript Array sort() Method

- Sorts the items of an array.
- Can be either alphabetic or numeric, and either ascending (up) or descending (down).
- By default, the sort() method sorts the values as strings in alphabetical and ascending order.
- Use the "compare function" when sorting numbers to have a correct result.
- Syntax: `array.sort(compareFunction)`

101

JavaScript Array sort() Method (continued)

- Parameter Values:

Parameter	Description
compareFunction	•Optional. A function that defines an alternative sort order. The function should return a negative, zero, or positive value, depending on the arguments, like:function(a, b){return a-b} When the sort() method compares two values, it sends the values to the compare function, and sorts the values according to the returned (negative, zero, positive) value. **Example:** When comparing 40 and 100, the sort() method calls the compare function(40,100). The function calculates 40-100, and returns -60 (a negative value). The sort function will sort 40 as a value lower than 100.

- Example:

Sort an array:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();
```

The result of *fruits* will be:

```
Apple,Banana,Mango,Orange
```

102

© Firebrand Training Ltd

JavaScript Array push() Method

- Adds new items to the end of an array, and returns the new length.
- The new item(s) will be added at the end of the array.
- **Tip:** To add items at the beginning of an array, use the unshift() method.
- Syntax: `array.push(item1, item2, ..., itemX)`
- Parameter Values:

Parameter	Description
item1, item2, ..., itemX	Required. The item(s) to add to the array

- Example:

Add a new item to an array:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Kiwi");
```

The result of *fruits* will be:

```
Banana,Orange,Apple,Mango,Kiwi
```

103

HTML DOM innerHTML Property

- The innerHTML property sets or returns the HTML content (inner HTML) of an element.
- Syntax:
- Return: `HTMLElementObject.innerHTML`
- Set: `HTMLElementObject.innerHTML = text`
- Property Values

Value	Description
text	Specifies the HTML content of an element

- Example:

```
document.getElementById("demo").innerHTML = "Paragraph changed!";
```

104

© Firebrand Training Ltd

JavaScript For Loop

- Loops can execute a block of code a number of times.
- Different Kinds of Loops
  - **for** - loops through a block of code a number of times
  - **for/in** - loops through the properties of an object
  - **while** - loops through a block of code while a specified condition is true
  - **do/while** - also loops through a block of code while a specified condition is true
- The For Loop
- Often the tool you will use when you want to create a loop.

105

JavaScript For Loop (continued)

- Syntax:

```
for (statement 1; statement 2; statement 3) {
    code block to be executed
}
```

- **Statement 1** is executed before the loop (the code block) starts.
- **Statement 2** defines the condition for running the loop (the code block).
- **Statement 3** is executed each time after the loop (the code block) has been executed.

```
for (i = 0; i < 5; i++) {
    text += "The number is " + i + "<br>";
}
```

- Statement 1 sets a variable before the loop starts (var i = 0).
- Statement 2 defines the condition for the loop to run (i must be less than 5).
- Statement 3 increases a value (i++) each time the code block in the loop has been executed.

106

© Firebrand Training Ltd

JavaScript For Loop (continued)

- The For/In Loop: Loops through the properties of an object:

- Example For/In Loop:

```
var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
    text += person[x];
}
```

107

JavaScript While Loop

- Loops through a block of code as long as a specified condition is true.

- Syntax:

```
while (condition) {
    code block to be executed
}
```

- Example: The code in the loop will run, over and over again, as long as a variable (i) is less than 10:

```
while (i < 10) {
    text += "The number is " + i;
    i++;
}
```

- Note: If you forget to increase the variable used in the condition, the loop will never end. This will crash your browser.

108

© Firebrand Training Ltd

JavaScript While Loop (continued)

- A variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.
- Syntax:

```
do {
    code block to be executed
}
while (condition);
```

- Example: The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

```
do {
    text += "The number is " + i;
    i++;
}
while (i < 10);
```

109

Comparing For and While Loop

- A while loop is much the same as a for loop, with statement 1 and statement 3 omitted.
- The loop in this example uses a **for loop** to collect the car names from the cars array:

```
var cars = ["BMW", "Volvo", "Saab", "Ford"];
var i = 0;
var text = "";

for (;cars[i];) {
    text += cars[i] + "<br>";
    i++;
}
```

- The loop in this example uses a **while loop** to collect the car names from the cars array:

```
var cars = ["BMW", "Volvo", "Saab", "Ford"];
var i = 0;
var text = "";

while (cars[i]) {
    text += cars[i] + "<br>";
    i++;
}
```

110

© Firebrand Training Ltd

jQuery Misc each() Method

- The each() method specifies a function to run for each matched element.
- Syntax: `$(selector).each(function(index,element))`
- Parameter Values:

Parameter	Description
function(index,element)	•Required. A function to run for each matched element.index - The index position of the selector •element - The current element (the "this" selector can also be used)

- Example Alert the text of each <li> element:

```
$("button").click(function(){
    $("li").each(function(){
        alert($(this).text())
    });
});
```

111

jQuery appendTo() Method

- Inserts HTML elements at the end of the selected elements.
- **Tip:** To insert HTML elements at the beginning of the selected elements, use the prependTo() method.
- Syntax: `$(content).appendTo(selector)`

Parameter	Description
content	Required. Specifies the content to insert (must contain HTML tags). Note: If content is an existing element, it will be moved from its current position, and inserted at the end of the selected elements.
selector	Required. Specifies on which elements to append the content to

- Example Insert a element at the end of each <p> element:

```
$("button").click(function(){
    $("<span>Hello World!</span>").appendTo("p");
});
```

112

© Firebrand Training Ltd

HTML DOM createElement() Method

- Creates an Element Node with the specified name.
  - **Tip:** Use the createTextNode() method to create a text node.
  - **Tip:** After the element is created, use the *element*.appendChild() or *element*.insertBefore()method to insert it to the document.
- Syntax: `document.createElement(nodename)`
- Parameter Values:

Parameter	Type	Description
nodename	String	Required. The name of the element you want to create

- Example:

```
var btn = document.createElement("BUTTON");
```

113

Image alt Property

- Sets or returns the value of the alt attribute of an image.
- The required alt attribute specifies an alternate text for an image, if the image for some reason cannot be displayed.
- Example: `var x = document.getElementById("myImg").alt;`
- Syntax:
  - Return: *imageObject*.alt
  - Set: *imageObject*.alt = *text*
- Property Values

Value	Description
text	•Specifies an alternate text for an image. Guidelines for the alt text:The text should describe the image if the image contains information •The text should explain where the link goes if the image is inside an <a> element •Use alt="" if the image is only for decoration

© Firebrand Training Ltd

Image src Property

- Sets or returns the value of the src attribute of an image.
- Specifies the URL of an image.
- Syntax
  - Return: `imageObject.src`
  - Set: `imageObject.src = URL`

Example:
```
document.getElementById("myImg").src = "hackanm.gif";
```

- Property Values

Value	Description
URL	•Specifies the URL of the image. Possible values:An absolute URL - points to another web site (like src="http://www.example.com/default.htm") •A relative URL - points to a file within a web site (like src="default.htm")

115

HTML5 Geolocation

- Used to get the geographical position of a user.
- The position is not available unless the user approves it.
- The getCurrentPosition() method is used to return the user's position.
- getCurrentPosition() - Return Data:

Property	Returns
coords.latitude	The latitude as a decimal number (always returned)
coords.longitude	The longitude as a decimal number (always returned)
coords.accuracy	The accuracy of position (always returned)
coords.altitude	The altitude in meters above the mean sea level (returned if available)
coords.altitudeAccuracy	The altitude accuracy of position (returned if available)
coords.heading	The heading as degrees clockwise from North (returned if available)
coords.speed	The speed in meters per second (returned if available)
timestamp	The date/time of the response (returned if available)

11

© Firebrand Training Ltd

HTML5 Geolocation

- watchPosition() - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).
- clearWatch() - Stops the watchPosition() method.
- Example shows the watchPosition() method:

```
<script>
var x = document.getElementById("demo");
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.watchPosition(showPosition);
    } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
    }
}
function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>
```

117

jQuery ajax() Method

- Used to perform an AJAX (asynchronous HTTP) request.
- Syntax: `$.ajax({name:value, name:value, ... })`

Name	Value/Description
async	A Boolean value indicating whether the request should be handled asynchronous or not. Default is true
beforeSend(*xhr*)	A function to run before the request is sent
cache	A Boolean value indicating whether the browser should cache the requested pages. Default is true
complete(*xhr,status*)	A function to run when the request is finished (after success and error functions)
contentType	The content type used when sending data to the server. Default is: "application/x-www-form-urlencoded"
context	Specifies the "this" value for all AJAX related callback functions
data	Specifies data to be sent to the server
dataFilter(*data,type*)	A function used to handle the raw response data of the XMLHttpRequest

118

© Firebrand Training Ltd

jQuery ajax() Method (continued)

Name	Value/Description
dataType	The data type expected of the server response.
error(*xhr,status,error*)	A function to run if the request fails.
global	A Boolean value specifying whether or not to trigger global AJAX event handles for the request. Default is true
ifModified	A Boolean value specifying whether a request is only successful if the response has changed since the last request. Default is: false.
jsonp	A string overriding the callback function in a jsonp request
jsonpCallback	Specifies a name for the callback function in a jsonp request
password	Specifies a password to be used in an HTTP access authentication request.
processData	A Boolean value specifying whether or not data sent with the request should be transformed into a query string. Default is true

jQuery ajax() Method (continued)

Name	Value/Description
scriptCharset	Specifies the charset for the request
success(*result,status,xhr*)	A function to be run when the request succeeds
timeout	The local timeout (in milliseconds) for the request
traditional	A Boolean value specifying whether or not to use the traditional style of param serialization
type	Specifies the type of request. (GET or POST)
url	Specifies the URL to send the request to. Default is the current page
username	Specifies a username to be used in an HTTP access authentication request
xhr	A function used for creating the XMLHttpRequest object

© Firebrand Training Ltd

jQuery ajax() Method (continued)

- Example

```
$.ajax({
    url: webServiceURL,
    type: "POST",
    dataType: "xml",
    data: soapMessage,
    processData: false,
    contentType: "text/xml; charset=\"utf-8\"",
    success: OnSuccess,
    error: OnError
});
```

121

XML SOAP

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- An application communication protocol
- A format for sending and receiving messages
- Platform independent
- Based on XML
- SOAP Building Blocks: An ordinary XML document containing:
  - An Envelope element that identifies the XML document as a SOAP message
  - A Header element that contains header information
  - A Body element that contains call and response information
  - A Fault element containing errors and status information

122

© Firebrand Training Ltd

XML SOAP (continued)

- SOAP Message Syntax Rules:
  - MUST be encoded using XML
  - MUST use the SOAP Envelope namespace
  - MUST use the SOAP Encoding namespace
  - Must NOT contain a DTD reference
  - Must NOT contain XML Processing Instructions
- Example:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

123

ActiveXObject Object (JavaScript)

- Enables and returns a reference to an Automation object.
- Only used to instantiate Automation objects, and has no members.
- Syntax: `newObj = new ActiveXObject(servername.typename[, location])`
- Parameters:

`newObj`
Required. The variable name to which the `ActiveXObject` is assigned.

`servername`
Required. The name of the application providing the object.

`typename`
Required. The type or class of the object to create.

`location`
Optional. The name of the network server where the object is to be created.

- Example:

```
var ExcelApp = new ActiveXObject("Excel.Application");
var ExcelSheet = new ActiveXObject("Excel.Sheet");
```

124

© Firebrand Training Ltd

jQuery submit() Method

- Occurs when a form is submitted.
- Can only be be used on <form> elements.
- Triggers the submit event, or attaches a function to run when a submit event occurs.
- Syntax:
  - Trigger: `$(selector).submit()`
  - Attach Function: `$(selector).submit(function)`

Parameter	Description
function	Optional. Specifies the function to run when the submit event is triggered

- Example:

```
$("form").submit(function(){
    alert("Submitted");
});
```

125

:first Selector jQuery

- *Selects the first matched DOM element.*
- Selected elements are in the order of their appearance in the document.
- Example:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>first demo</title>
  <style>
  td {
    color: blue;
    font-weight: bold;
  }
  </style>
  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>

<table>
  <tr><td>Row 1</td></tr>
  <tr><td>Row 2</td></tr>
  <tr><td>Row 3</td></tr>
</table>

<script>
$( "tr:first" ).css( "font-style", "italic" );
</script>

</body>
</html>
```

© Firebrand Training Ltd

jQuery Misc data() Method

- Attaches data to, or gets data from, selected elements.
- To remove data, use the removeData() method.
- Return Data Syntax: `$(selector).data(name)`

Parameter	Description
name	Optional. Specifies the name of data to retrieve. If no name is specified, this method will return all stored data for the element as an object

- Attach Data Syntax: `$(selector).data(name,value)`

Parameter	Description
name	Required. Specifies the name of data to set
value	Required. Specifies the value of data to set

127

jQuery Misc data() Method

- Attach Data using an Object Syntax: `$(selector).data(object)`

Parameter	Description
object	Required. Specifies an object containing name/value pairs

- Example:

```
$("#btn1").click(function(){
    $("div").data("greeting", "Hello World");
});
$("#btn2").click(function(){
    alert($("div").data("greeting"));
});
```

128

© Firebrand Training Ltd

JavaScript RegExp Reference

- A regular expression is an object that describes a pattern of characters.
- Used to perform pattern-matching and "search-and-replace" functions on text.
- Syntax: `/pattern/modifiers;`
- Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

129

JavaScript RegExp Reference (continued)

- Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
(x|y)	Find any of the alternatives specified

- Metacharacters are characters with a special meaning:

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word

© Firebrand Training Ltd

JavaScript RegExp Reference (continued)

- Metacharacters (continued):

Metacharacter	Description
\B	Find a match not at the beginning/end of a word
\0	Find a NUL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\uxxxx	Find the Unicode character specified by a hexadecimal number xxxx

JavaScript RegExp Reference (continued)

- Quantifiers:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{X}	Matches any string that contains a sequence of *X* *n*'s
n{X,Y}	Matches any string that contains a sequence of X to Y *n*'s
n{X,}	Matches any string that contains a sequence of at least X *n*'s
n$	Matches any string with *n* at the end of it
^n	Matches any string with *n* at the beginning of it
?=n	Matches any string that is followed by a specific string *n*
?!n	Matches any string that is not followed by a specific string *n*

© Firebrand Training Ltd

JavaScript RegExp Reference (continued)

- RegExp Object Properties:

Property	Description
constructor	Returns the function that created the RegExp object's prototype
global	Checks whether the "g" modifier is set
ignoreCase	Checks whether the "i" modifier is set
lastIndex	Specifies the index at which to start the next match
multiline	Checks whether the "m" modifier is set
source	Returns the text of the RegExp pattern

- RegExp Object Methods:

Method	Description
compile()	Deprecated in version 1.5. Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false
toString()	Returns the string value of the regular expression

jQuery serialize() Method

- Creates a URL encoded text string by serializing form values.
- You can select one or more form elements, or the form element itself.
- Can be used in the URL query string when making an AJAX request.
- Syntax: `$(selector).serialize()`

© Firebrand Training Ltd

JavaScript decodeURIComponent() Function

- Decodes a URI component.
- Use the encodeURIComponent() function to encode a URI component.
- Syntax: `decodeURIComponent(uri)`
- Parameter Values:

Parameter	Description
uri	Required. The URI to be decoded

- Example:

Decode a URI after encoding it:

```
var uri = "https://w3schools.com/my test.asp?name=ståle&car=saab";
var uri_enc = encodeURIComponent(uri);
var uri_dec = decodeURIComponent(uri_enc);
var res = uri_enc + "<br>" + uri_dec;
```

The result of *res* will be:

```
// Encoded URI
https%3A%2F%2Fw3schools.com%2Fmy%20test.asp%3Fname%3Dst%C3%A5le%26car%3Dsaab

// Decoded URI
https://w3schools.com/my test.asp?name=ståle&car=saab
```

135

JavaScript Errors - Throw and Try to Catch

- The **try** statement lets you test a block of code for errors.
- The **catch** statement lets you handle the error.
- The **throw** statement lets you create custom errors.
- The **finally** statement lets you execute code, after try and catch, regardless of the result.
- The JavaScript statements **try** and **catch** come in pairs:

- The **try** statement allows you to define a block of code to be tested for errors while it is being executed.
- The **catch** statement allows you to define a block of code to be executed, if an error occurs in the try block.

```
try {
    Block of code to try
}
catch(err) {
    Block of code to handle errors
}
```

136

© Firebrand Training Ltd

JavaScript Errors - Throw and Try to Catch (continued)

- JavaScript Throws Errors
- JavaScript will **throw an exception (throw an error).**
- The **throw** statement allows you to create a custom error.
- The exception can be a JavaScript String, a Number, a Boolean or an Object:

```
throw "Too big";    // throw a text
throw 500;          // throw a number
```

- The finally Statement
- The **finally** statement lets you execute code, after try and catch, regardless of the result:

```
try {
    Block of code to try
}
catch(err) {
    Block of code to handle errors
}
finally {
    Block of code to be executed regardless of the try / catch result
}
```

137

jQuery.each()

- *A generic iterator function, which can be used to seamlessly iterate over both objects and arrays.*
- *Arrays and array-like objects with a length property (such as a function's arguments object) are iterated by numeric index, from 0 to length-1.*
- *Other objects are iterated via their named properties.*

jQuery.each(array, callback) **jQuery.each(object, callback)**

- *Example:*

```
1 $.each([ 52, 97 ], function( index, value ) {
2   alert( index + ": " + value );
3 });
```

138

© Firebrand Training Ltd

jQuery parent() Method

- Returns the direct parent element of the selected element.
- **The DOM tree:** This method only traverse a single level up the DOM tree. To traverse all the way up to the document's root element, use the parents() or the parentsUntil() method.
- **Tip:** To traverse a single level down the DOM tree, or all the way down to the last descendant (to return children or other descendants), use the children() or find() method.
- Syntax: `$(selector).parent(filter)`

Parameter	Description
filter	Optional. Specifies a selector expression to narrow down the parent search

- Example:

```
$(document).ready(function(){
    $("span").parent().css({"color": "red", "border": "2px solid red"});
});
```

139

jQuery attr() Method

- Sets or returns attributes and values of the selected elements.
- When this method returns it returns the value of the FIRST matched element.
- When this method sets it sets one or more attribute/value pairs for the set of matched elements.
- Syntax

Return: `$(selector).attr(attribute)`
Set: `$(selector).attr(attribute,value)`
Set using a function: `$(selector).attr(attribute,function(index,currentvalue))`
Set using multiple attributes and values:
`$(selector).attr({attribute:value, attribute:value,...})`

Parameter	Description
attribute	Specifies the name of the attribute
value	Specifies the value of the attribute
function(*index*,*currentvalue*)	•Specifies a function that returns the attribute value to set*index* - Receives the index position of the element in the set •*currentvalue* - Receives the current attribute value of selected elements

© Firebrand Training Ltd

CSS3 @media Rule

- Used to define different style rules for different media types/devices.
- Syntax:

```
@media not|only mediatype and (media feature) {
    CSS-Code;
}
```

- You can also have different *stylesheets* for different media:

```
<link rel="stylesheet" media="mediatype and|not|only (media feature)"
href="mystylesheet.css">
```

- Media Types:

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

141

CSS3 @media Rule (continued)

- Media Features:

Value	Description
any-hover	Does any available input mechanism allow the user to hover over elements? (added in Media Queries Level 4)
any-pointer	Is any available input mechanism a pointing device, and if so, how accurate is it? (added in Media Queries Level 4)
aspect-ratio	The ratio between the width and the height of the viewport
color	The number of bits per color component for the output device
color-index	The number of colors the device can display
grid	Whether the device is a grid or bitmap
height	The viewport height
hover	Does the primary input mechanism allow the user to hover over elements? (added in Media Queries Level 4)
inverted-colors	Is the browser or underlying OS inverting colors? (added in Media Queries Level 4)

142

© Firebrand Training Ltd

CSS3 @media Rule (continued)

Value	Description
light-level	Current ambient light level (added in Media Queries Level 4)
max-aspect-ratio	The maximum ratio between the width and the height of the display area
max-color	The maximum number of bits per color component for the output device
max-color-index	The maximum number of colors the device can display
max-device-aspect-ratio	The maximum ratio between the width and the height of the device
max-device-height	The maximum height of the device, such as a computer screen
max-device-width	The maximum width of the device, such as a computer screen
max-height	The maximum height of the display area, such as a browser window
max-monochrome	The maximum number of bits per "color" on a monochrome (greyscale) device
max-resolution	The maximum resolution of the device, using dpi or dpcm
max-width	The maximum width of the display area, such as a browser window

CSS3 @media Rule (continued)

Value	Description
min-aspect-ratio	The minimum ratio between the width and the height of the display area
min-color	The minimum number of bits per color component for the output device
min-color-index	The minimum number of colors the device can display
min-device-aspect-ratio	The minimum ratio between the width and the height of the device
min-device-width	The minimum width of the device, such as a computer screen
min-device-height	The minimum height of the device, such as a computer screen
min-height	The minimum height of the display area, such as a browser window
min-monochrome	The minimum number of bits per "color" on a monochrome (greyscale) device
min-resolution	The minimum resolution of the device, using dpi or dpcm
min-width	The minimum width of the display area, such as a browser window
monochrome	The number of bits per "color" on a monochrome (greyscale) device
orientation	The orientation of the viewport (landscape or portrait mode)
overflow-block	How does the output device handle content that overflows the viewport along the block axis (added in Media Queries Level 4)
overflow-inline	Can content that overflows the viewport along the inline axis be scrolled (added in Media Queries Level 4)
pointer	Is the primary input mechanism a pointing device, and if so, how accurate is it? (added in Media Queries Level 4)
resolution	The resolution of the output device, using dpi or dpcm
scan	The scanning process of the output device
scripting	Is scripting (e.g. JavaScript) available? (added in Media Queries Level 4)
update-frequency	How quickly can the output device modify the appearance of the content (added in Media Queries Level 4)
width	The viewport width

© Firebrand Training Ltd

CSS3 @media Rule (continued)

- Example:

```
@media only screen and (max-width: 500px) {
    .gridmenu {
        width:100%;
    }

    .gridmain {
        width:100%;
    }

    .gridright {
        width:100%;
    }
}
```

145

JavaScript Switch Statement

- Used to perform different actions based on different conditions.
- Syntax:

```
switch(expression) {
    case n:
        code block
        break;
    case n:
        code block
        break;
    default:
        code block
}
```

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

146

© Firebrand Training Ltd

JavaScript Switch Statement (continued)

- **Break**: When JavaScript reaches a **break** keyword, it breaks out of the switch block.
- **Default**: The **default** keyword specifies the code to run if there is no case match.
- Example:

```
switch (new Date().getDay()) {
    case 4:
    case 5:
        text = "Soon it is Weekend";
        break;
    case 0:
    case 6:
        text = "It is Weekend";
        break;
    default:
        text = "Looking forward to the Weekend";
}
```

147

Create a Web Worker File

- Create a web worker in an external JavaScript.
- Here, we create a script that counts. The script is stored in the "demo_workers.js" file:

```
var i = 0;

function timedCount() {
    i = i + 1;
    postMessage(i);
    setTimeout("timedCount()",500);
}

timedCount();
```

- The important part of the code above is the **postMessage()** method - which is used to post a message back to the HTML page.
- **Note**: Normally web workers are not used for such simple scripts, but for more CPU intensive tasks.

148

© Firebrand Training Ltd

Window close() Method

- The close() method closes the current window.
- Syntax: `window.close()`
- Example

```
function openWin() {
    myWindow = window.open("", "myWindow", "width=200, height=100");   //
Opens a new window
}

function closeWin() {
    myWindow.close();   // Closes the new window
}
```

149

Window open() Method

- The open() method opens a new browser window.
- Syntax: `window.open(URL, name, specs, replace)`
- Example

```
var myWindow = window.open("", "", "width=200,height=100");
```

150

© Firebrand Training Ltd

HTML5 Local Storage

- Web applications can store data locally within the user's browser.
- More secure
- Large amounts of data can be stored locally(at least 5MB), without affecting website performance.
- Information is never transferred to the server.
- Local storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.
- Example: Stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

```
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML =
localStorage.getItem("lastname");
```

151

HTML5 Drag and Drop

- When you "grab" an object and drag it to a different location.
- Make an Element Draggable:
  - **First of all**: Set the draggable attribute to true: `<img draggable="true">`
- What to Drag - ondragstart and setData()
  - **Then**, specify what should happen when the element is dragged.
  - The dataTransfer.setData() method sets the data type and the value of the dragged data:

```
function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}
```

- In this case, the data type is "text" and the value is the id of the draggable element ("drag1").

152

© Firebrand Training Ltd

HTML5 Drag and Drop (continued)

Where to Drop - ondragover

- The ondragover event specifies where the dragged data can be dropped.
- By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.
- This is done by calling the event.preventDefault() method for the ondragover event: `event.preventDefault()`

Do the Drop - ondrop

- When the dragged data is dropped, a drop event occurs.
- In the example above, the ondrop attribute calls a function, drop(event):

```
function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
```

153

HTML5 Drag and Drop (continued)

Code Explained:

```
function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
```

1. Call preventDefault() to prevent the browser default handling of the data (default is open as link on drop)
2. Get the dragged data with the dataTransfer.getData() method. This method will return any data that was set to the same type in the setData() method
3. The dragged data is the id of the dragged element ("drag1")
4. Append the dragged element into the drop element

154

© Firebrand Training Ltd

CSS3 Flexible Box

- Ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.
- Flexbox consists of flex containers and flex items.
- A flex container is declared by setting the display property property of an element to either flex (rendered as a block) or inline-flex (rendered as inline).
- Inside a flex container there is one or more flex items.
- Flex items are positioned inside a flex container along a flex line. By default there is only one flex line per flex container.

155

CSS3 Flexible Box (continued)

- Example:

```
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-direction: row-reverse
    flex-direction: row-reverse;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

Property	Description
display	Specifies the type of box used for an HTML element
flex-direction	Specifies the direction of the flexible items inside a flex container
justify-content	Horizontally aligns the flex items when the items do not use all available space on the main-axis
align-items	Vertically aligns the flex items when the items do not use all available space on the cross-axis
flex-wrap	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
align-content	Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
flex-flow	A shorthand property for flex-direction and flex-wrap
order	Specifies the order of a flexible item relative to the rest of the flex items inside the same container
align-self	Used on flex items. Overrides the container's align-items property
flex	Specifies the length of a flex item, relative to the rest of the flex items inside the same container

156

© Firebrand Training Ltd

JavaScript Operators Reference

JavaScript Arithmetic Operators

Operator	Description	Example	Result in y	Result in x
+	Addition	x = y + 2	y = 5	x = 7
-	Subtraction	x = y - 2	y = 5	x = 3
*	Multiplication	x = y * 2	y = 5	x = 10
/	Division	x = y / 2	y = 5	x = 2.5
%	Modulus (division remainder)	x = y % 2	y = 5	x = 1
++	Increment	x = ++y	y = 6	x = 6
		x = y++	y = 6	x = 5
--	Decrement	x = --y	y = 4	x = 4
		x = y--	y = 4	x = 5

JavaScript String Operators

Operator	Example	text1	text2	text3
+	text3 = text1 + text2	"Good "	"Morning"	"Good Morning"
+=	text1 += text2	"Good Morning"	"Morning"	""

JavaScript Operators Reference (continued)

JavaScript Assignment Operators

Operator	Example	Same As	Result in x
=	x = y	x = y	x = 5
+=	x += y	x = x + y	x = 15
-=	x -= y	x = x - y	x = 5
*=	x *= y	x = x * y	x = 50
/=	x /= y	x = x / y	x = 2
%=	x %= y	x = x % y	x = 0

Conditional (Ternary) Operator

Syntax	Example
variablename = (*condition*) ? *value1*:*value2*	license = (age < 18) ? "Too young":"Old enough";

- **Example explained:** If the variable "age" is a value below 18, the value of the variable "license" will be "Too young", otherwise the value of license will be "Old enough".

© Firebrand Training Ltd

JavaScript Operators Reference (continued)

Comparison Operators

Operator	Description	Comparing	Returns
==	equal to	x == 8	false
		x == 5	true
===	equal value and equal type	x === "5"	false
		x === 5	true
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== "5"	true
		x !== 5	false
>	greater than	x > 8	false
<	less than	x < 8	true
>=	greater than or equal to	x >= 8	false
<=	less than or equal to	x <= 8	*true*

JavaScript Operators Reference (continued)

Logical Operators

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x === 5 || y === 5) is false
!	not	!(x === y) is true

JavaScript Bitwise Operators

Operator	Description	Example	Same as	Result	Decimal
&	AND	x = 5 & 1	0101 & 0001	0001	1
|	OR	x = 5 | 1	0101 | 0001	0101	5
~	NOT	x = ~ 5	~0101	1010	10
^	XOR	x = 5 ^ 1	0101 ^ 0001	0100	4
<<	Left shift	x = 5 << 1	0101 << 1	1010	10
>>	Right shift	x = 5 >> 1	0101 >> 1	0010	2

© Firebrand Training Ltd

JavaScript Operators Reference (continued)

- The typeof Operator : returns the type of a variable, object, function or expression:

```
typeof "John"                 // Returns string
typeof 3.14                   // Returns number
typeof NaN                    // Returns number
typeof false                  // Returns boolean
```

You cannot use **typeof** to define if a JavaScript object is an array (or a date).

- The data type of NaN is number
- The data type of an array is object
- The data type of a date is object
- The data type of null is object
- The data type of an undefined variable is undefined

- The delete Operator : deletes a property from an object:

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
delete person.age;   // or delete person["age"];
```

JavaScript Operators Reference (continued)

- The in Operator : returns true if the specified property is in the specified object, otherwise false:

```
// Arrays
var cars = ["Saab", "Volvo", "BMW"];
"Saab" in cars          // Returns false (specify the index number instead of
value)
0 in cars               // Returns true
1 in cars               // Returns true
4 in cars               // Returns false (does not exist)
"length" in cars        // Returns true  (length is an Array property)
```

- The instanceof Operator: returns true if the specified object is an instance of the specified object:

```
var cars = ["Saab", "Volvo", "BMW"];

cars instanceof Array;          // Returns true
cars instanceof Object;         // Returns true
cars instanceof String;         // Returns false
cars instanceof Number;         // Returns false
```

- The void Operator: evaluates an expression and returns **undefined**.

```
<a href="javascript:void(0);">
  Useless link
</a>

<a href="javascript:void(document.body.style.backgroundColor='red');">
  Click me to change the background color of body to red
</a>
```

© Firebrand Training Ltd

Window setInterval() Method

- Calls a function or evaluates an expression at specified intervals (in milliseconds).
- Syntax: `setInterval(function, milliseconds, param1, param2, ...)`
- Parameter Values:

Parameter	Description
function	Required. The function that will be executed
milliseconds	Required. The intervals (in milliseconds) on how often to execute the code. If the value is less than 10, the value 10 is used
param1, *param2*, ...	Optional. Additional parameters to pass to the *function* (Not supported in IE9 and earlier)

- Example: `setInterval(function(){ alert("Hello"); }, 3000);`

163

SVG Code Example: Rotating Ellipses

```
<!DOCTYPE html>
<html>
<body>

<p><strong>Note:</strong> This example does not work in Internet
Explorer and Safari.</p>

<svg width="100%" height="300px">
<g id="R1" transform="translate(250 250)">
  <ellipse rx="100" ry="0" opacity=".3">
  <animateTransform attributeName="transform" type="rotate"
dur="7s" from="0" to="360" repeatCount="indefinite" />
  <animate attributeName="cx" dur="8s" values="-20; 220; -20"
repeatCount="indefinite" />
  <animate attributeName="ry" dur="3s" values="10; 60; 10"
repeatCount="indefinite" />
  </ellipse>
</g>
<use xlink:href="#R1" transform="rotate(72 390 150)" />
<use xlink:href="#R1" transform="rotate(144 390 150)" />
<use xlink:href="#R1" transform="rotate(216 390 150)" />
<use xlink:href="#R1" transform="rotate(288 390 150)" />
</svg>

</body>
</html>
```

164

© Firebrand Training Ltd

CSS3 border-radius Property

- Used to add rounded corners to an element.
- If you specify only one value for the border-radius property, this radius will be applied to all 4 corners.

Specify each corner separately:
- **Four values:** first value applies to top-left, second value applies to top-right, third value applies to bottom-right, and fourth value applies to bottom-left corner
- **Three values:** first value applies to top-left, second value applies to top-right and bottom-left, and third value applies to bottom-right
- **Two values:** first value applies to top-left and bottom-right corner, and the second value applies to top-right and bottom-left corner
- **One value:** all four corners are rounded equally

1. Four values - border-radius: 15px 50px 30px 5px:

2. Three values - border-radius: 15px 50px 30px:

3. Two values - border-radius: 15px 50px:

CSS3 border-radius Property

- Syntax: `border-radius: 1-4 length|% / 1-4 length|%|initial|inherit;`

- Property Values:

Value	Description
length	Defines the shape of the corners. Default value is 0
%	Defines the shape of the corners in %
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element

- Example:

```
div {
    border: 2px solid;
    border-radius: 25px;
}
```

166

© Firebrand Training Ltd

jQuery.ajax accepts property

- A set of key/value pairs that map a given dataType to its MIME type, which gets sent into the Accept request header.
- This header tells the server what kind of response it will accept in return.
- For example, the following defines a custom type myCustomType to be sent with the request:

```
$.ajax({
  accepts: {
    mycustomtype: 'application/x-some-custom-type'
  },

  // Instructions for how to deserialize a `mycustomtype`
  converters: {
    'text mycustomtype': function(result) {
      // Do Stuff
      return newresult;
    }
  },

  // Expect a `mycustomtype` back from server
  dataType: 'mycustomtype'
});
```

Note: You will need to specify a complementary entry for this type in `converters` for this to work properly.

167

HTML <figure> Tag

- Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
- It's position is independent of the main flow, and if removed it should not affect the flow of the document.
- Example:

```
<figure>
  <img src="img_pulpit.jpg" alt="The Pulpit Rock" width="304" height="228">
</figure>
```

- Result:

168

© Firebrand Training Ltd

HTML <figcaption> Tag

- Defines a caption for a <figure> element.
- Can be placed as the first or last child of the <figure> element.
- Example:

```
<figure>
  <img src="img_pulpit.jpg" alt="The Pulpit Rock" width="304" height="228">
  <figcaption>Fig1. - A view of the pulpit rock in Norway.</figcaption>
</figure>
```

- Result:

Fig.1 - A view of the pulpit rock in Norway.

169

HTML canvas fillStyle Property

- Sets or returns the color, gradient, or pattern used to fill the drawing.

Default value:	#000000
JavaScript syntax:	*context*.fillStyle=*color*|*gradient*|*pattern*;

Value	Description
color	A CSS color value that indicates the fill color of the drawing. Default value is #000000
gradient	A gradient object (linear or radial) used to fill the drawing
pattern	A pattern object to use to fill the drawing

- Example:

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="#FF0000";
ctx.fillRect(20,20,150,100);
```

Define a red fill-color for the rectangle:

© Firebrand Training Ltd

HTTP Status Messages

- When a browser requests a service from a web server, an error might occur.
- This is a list of HTTP status messages that might be returned:
- 1xx: Information

Message:	Description:
100 Continue	The server has received the request headers, and the client should proceed to send the request body
101 Switching Protocols	The requester has asked the server to switch protocols
103 Checkpoint	Used in the resumable requests proposal to resume aborted PUT or POST requests

HTTP Status Messages (continued)

- 2xx: Successful

Message:	Description:
200 OK	The request is OK (this is the standard response for successful HTTP requests)
201 Created	The request has been fulfilled, and a new resource is created
202 Accepted	The request has been accepted for processing, but the processing has not been completed
203 Non-Authoritative Information	The request has been successfully processed, but is returning information that may be from another source
204 No Content	The request has been successfully processed, but is not returning any content
205 Reset Content	The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view
206 Partial Content	The server is delivering only part of the resource due to a range header sent by the client

© Firebrand Training Ltd

HTTP Status Messages (continued)

3xx: Redirection

Message:	Description:
300 Multiple Choices	A link list. The user can select a link and go to that location. Maximum five addresses
301 Moved Permanently	The requested page has moved to a new URL
302 Found	The requested page has moved temporarily to a new URL
303 See Other	The requested page can be found under a different URL
304 Not Modified	Indicates the requested page has not been modified since last requested
306 Switch Proxy	*No longer used*
307 Temporary Redirect	The requested page has moved temporarily to a new URL
308 Resume Incomplete	Used in the resumable requests proposal to resume aborted PUT or POST requests

HTTP Status Messages (continued)

4xx: Client Error

Message:	Description:
400 Bad Request	The request cannot be fulfilled due to bad syntax
401 Unauthorized	The request was a legal request, but the server is refusing to respond to it. For use when authentication is possible but has failed or not yet been provided
402 Payment Required	*Reserved for future use*
403 Forbidden	The request was a legal request, but the server is refusing to respond to it
404 Not Found	The requested page could not be found but may be available again in the future
405 Method Not Allowed	A request was made of a page using a request method not supported by that page
406 Not Acceptable	The server can only generate a response that is not accepted by the client
407 Proxy Authentication Required	The client must first authenticate itself with the proxy
408 Request Timeout	The server timed out waiting for the request
409 Conflict	The request could not be completed because of a conflict in the request
410 Gone	The requested page is no longer available
411 Length Required	The "Content-Length" is not defined. The server will not accept the request without it
412 Precondition Failed	The precondition given in the request evaluated to false by the server
413 Request Entity Too Large	The server will not accept the request, because the request entity is too large
414 Request-URI Too Long	The server will not accept the request, because the URL is too long. Occurs when you convert a POST request to a GET request with a long query information
415 Unsupported Media Type	The server will not accept the request, because the media type is not supported
416 Requested Range Not Satisfiable	The client has asked for a portion of the file, but the server cannot supply that portion
417 Expectation Failed	The server cannot meet the requirements of the Expect request-header field

© Firebrand Training Ltd

HTTP Status Messages (continued)

- 5xx: Server Error

Message:	Description:
500 Internal Server Error	A generic error message, given when no more specific message is suitable
501 Not Implemented	The server either does not recognize the request method, or it lacks the ability to fulfill the request
502 Bad Gateway	The server was acting as a gateway or proxy and received an invalid response from the upstream server
503 Service Unavailable	The server is currently unavailable (overloaded or down)
504 Gateway Timeout	The server was acting as a gateway or proxy and did not receive a timely response from the upstream server
505 HTTP Version Not Supported	The server does not support the HTTP protocol version used in the request
511 Network Authentication Required	The client needs to authenticate to gain network access

175

JavaScript test() Method

- Tests for a match in a string.
- Syntax: `RegExpObject.test(string)`
- Parameter Value:

Parameter	Description
string	Required. The string to be searched

- Return Value:

Type	Description
Boolean	Returns true if it finds a match, otherwise it returns false

- Example:

Search a string for the character "e":

```
var str = "The best things in life are free";
var patt = new RegExp("e");
var res = patt.test(str);
```

Since there is an "e" in the string, the result of *res* will be:

```
true
```

176

© Firebrand Training Ltd