WEEK 4

59

JAVASCRIPT REVIEW

1. What is *missing* from the code below to write the text below to the page using Javascript

one
two

```
document.write(“one”);
document.write(<strong>two</strong>”);
```

2. In the Javascript DOM, DOM stands for _____________

60

REVIEW: WHICH CODE HAS A **SYNTAX ERROR**:

```
a. if (a<b)
document.write( "hello" );

b. if a<b
document.write( "hello" );

c. if (a<b) && (b<c)
 {
  document.write( "hello" );
  document.write( "there" );
 }
```

```
d. if (a<b) && (b<c)
  document.write( "hello" );
  document.write( "there" );

e. if (a=4)
document.write( "hello" );
```

61

61

REVIEW: FIGURE IT OUT

What is the value of x at the end?

```
y = 5;  x= 7;
switch(y)
{
  case 1:  x= x*2; break;
  case 5:  x -= y;
  case 10: x++;   break;
  default: x = 17;
}
```

62

62

REVIEW: FIGURE IT OUT

What is displayed?

```
function process( x, z)
{
  x+= 10;
  z =z + x;
  return z;
}
x = 20;
a = 10;  b= 11;
a= process(a, b);
alert(x);
alert(a);
```

63

63

ARROW FUNCTIONS

- Use the arrow as a shortcut to define and then call a function
- Assume you want a simple function as follows:
  function hello() { return “Hey there!”; }
  - Call the function using: hello()
- Using an arrow function:
  hello = () => {return "Hello World!“};
- Simplfy further to:
  hello = () => "Hello World!";
  - Call the function using: hello()
- Add parameters:
  hello = (num) => "Two times " + num + " is " + 2 * num;

64

EXAMPLE: USE AN ARROW FUNCTION AS A PARAMETER

```
add = (a,b) => a+b;
sub = (a,b) => a-b;
function operate (a, b, op)
{
        return op(a,b);
}
alert(operate (4,7, add));    //what is displayed?
```

65

CONCEPT: ANONYMOUS FUNCTIONS

- You don’t need to name a function when you are only using it once – this is considered an anonymous function
- Anonymous functions make sense when using a function as a
  - Parameter
  - Call back
  - Array method
- Arrow functions are frequently used when an anonymous function would be appropriate

66

CALLBACK FUNCTIONS

- A function can be called automatically when something else has completed.
- Example: setTimeout

```
setTimeout(function() {alert("a second just passed!")}, 1000))
```

Or

```
setTimeout( ()=>alert("one second") , 1000 )
```

67

67

JAVASCRIPT OBJECT LIBRARY (DOM)
Document Object Model

- Properties
  - Characteristics, State
- Methods
  - Things the object can do
    These will require parenthesis
    because they area function
- Events
  - The object can respond to events

- Use "new" to instance an object (sometimes there are shortcuts without *new*)
- *new* will call a constructor for the object
- Use the dot notation to call a method or access a property using an object
  arr = new Array()
  count = arr.length

68

68

THE MATH OBJECT: HELPFUL METHODS

- Math.random()
  - Returns a number between 0 and 1. Multiply it to get a larger range
  - Example- random number from 0 to 5
    ```
    n = Math.random() * 5;
    ```
  - Example- random number from 1 to 20
    ```
    n = Math.random() * 19 + 1;
    ```
- `Math.ceil()  Math.floor()`
- `Math.max ()  Math.min()`

69

69

THE STRING OBJECT

- Easily instanced using assignment to a quoted string.
- Helpful properties:
  - length
- Helpful methods:
  - charAt
  - indexOf / lastIndexOf
  - substr / substring
  - toLowerCase / toUpperCase
  - split

Example: display characters of a string with an asterisk (*) between them

```
s = "I am a string";
for (n=0; n<s.length;n++)
  document.write (s.charAt(n) + " * " );
```

70

70

STRING METHODS

length	number of characters in a string
charAt()	returns the character at the specified index
concat()	joins two or more strings, and returns a copy of the joined strings
indexOf()	returns the position of the first occurrence of a specified string
lastIndexOf()	returns the position of the last occurrence of a string
slice()	extracts a part of a string and returns a new string
split()	splits a string into an array of substrings
substr()	gets a substring defined by a start position and a number of characters
substring()	gets a substring defined by a start and end index
toLowerCase()	returns the string in lower case
toUpperCase()	returns the string in uppercase

DATE

Given: d = new Date();

d.getDate()	Returns day of the month (1-31)
d.getDay()	Returns day of the week (0-6)
d.getFullYear()	Returns the year (four digits)
d.getHours()	Returns the hour (0-23)
d.getMinutes()	Returns the minutes (0-59)
d.getMonth()	Returns the month (0-11)
d.getSeconds()	Returns the seconds (0-59)

Try it: display the current date as "June 12, 2025" using the date methods

ARRAYS IN JAVASCRIPT

- Arrays are implemented with the Array object.
- An array is a group or collection of items that are referenced using a group name and an index
- The first index is 0
- Array elements can be differing types!
- The size of the array is dynamic!
- Create an Array:
  - Use the *Array* object

```
    things = new Array(1,2,3);
```

  - Use literal notation [ ]

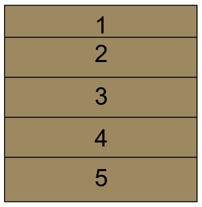
```
    things = [1,2,3];
```

73

73

A CONCEPTUAL VIEW …

- numbers = [1,2,3,4,5];



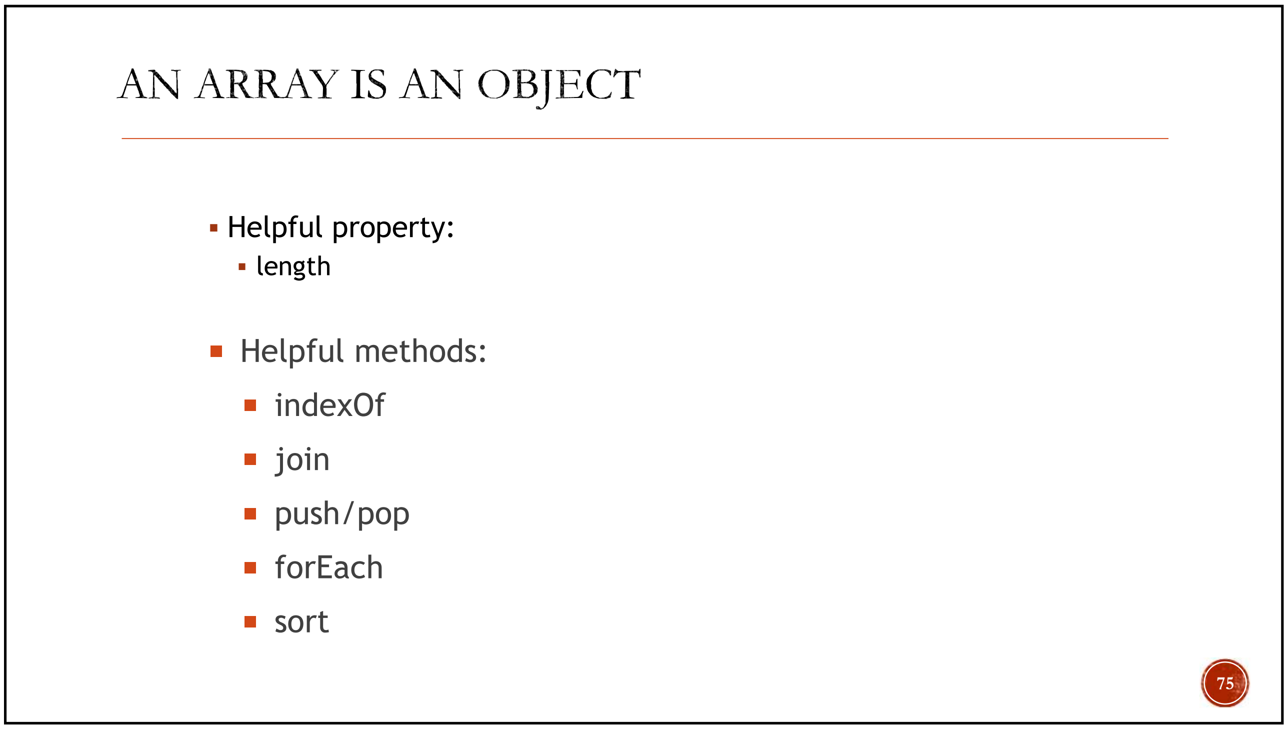
- *Create an empty array with 5 elements*

```
var numbers = new Array(5);
```

- document.write (numbers[2]);
  - Displays the 3rd element in the array – in this case "3"

74

74

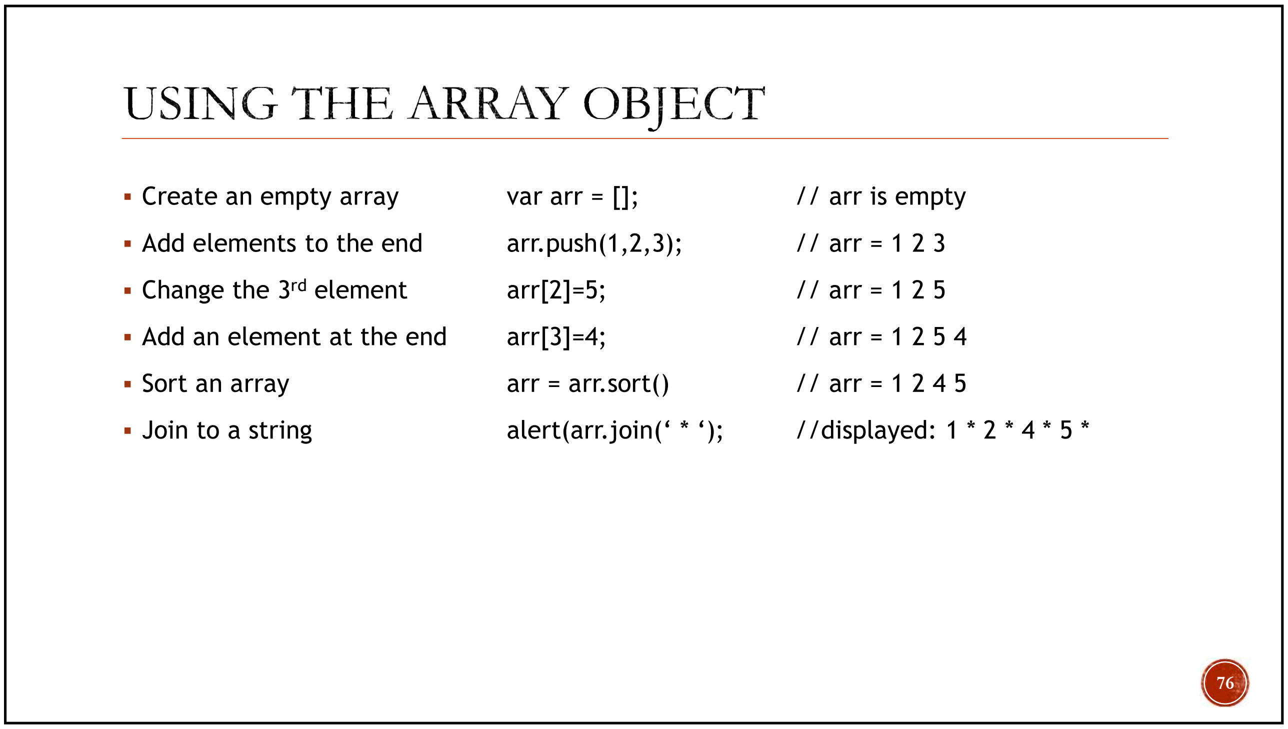


AN ARRAY IS AN OBJECT

- Helpful property:
  - length
- Helpful methods:
  - indexOf
  - join
  - push/pop
  - forEach
  - sort

75

75



USING THE ARRAY OBJECT

▪ Create an empty array	var arr = [];	// arr is empty
▪ Add elements to the end	arr.push(1,2,3);	// arr = 1 2 3
▪ Change the 3rd element	arr[2]=5;	// arr = 1 2 5
▪ Add an element at the end	arr[3]=4;	// arr = 1 2 5 4
▪ Sort an array	arr = arr.sort()	// arr = 1 2 4 5
▪ Join to a string	alert(arr.join(' * ');	//displayed: 1 * 2 * 4 * 5 *

76

76

LOOPS AND ARRAYS

- A loop counter can be used to iterate through an array.
  - To display the array
  - To do something to every element
  - To give a value to every element

```
//add 3 to each item
numbers = [2,4,6,8,10];
for (i=0; i<numbers.length;i++)
   numbers[i] += 3;
```

77

77

EXAMPLE: SEARCH AN ARRAY

```
arr = ["jim", "bill", "sam", "natalie", "sally", "fran"];
match = "sam";
for (i=0; i<arr.length; i++)
{
  if (arr[i] == match)
          break;  //stop when you find it
}
if (i<arr.length)
  document.write("match found at position: " + i);
else
  document.write("no match");
```

78

78

A FEW MIND-BLOWING ARRAY CONCEPTS!

Cool things we can do with arrays!

- split/ join
- forEach
- map
- associative arrays

There's lots more – look up Javascript ES6 if you are curious!

79

SPLIT AND JOIN

- Split breaks up a string into an array.
  - s.split(delimeter)
- Join combines an array into a string.
  - arr.join(str)
- Example:
  - str = "a,b,c,d,e,f";
  - arr= str.split(','); // each element in the array is a letter
  - str= “abcdef”;
  - arr = str.split(‘’); // each element in the array is a letter
  - str = arr.join("-"); //str is now: “a-b-c-d-e-f-”

80

ITERATION USING FOREACH

Several array methods such as forEach, map, and reduce are immutable- they do NOT change the original array. This forms the basis for functional programming.

```
//add all of the items
numbers = [2,4,6,8,10];
sum=0;
for (i=0; i<numbers.length;i++)
  sum+= numbers[i];
```

```
//add all of the items
numbers = [2,4,6,8,10];
sum=0;
numbers.forEach (function(item) {
   sum+= item;
});
```

81

81

EXAMPLE REVISITED

Display characters of a string with an asterisk (*) between them

```
s = "I am a string";

for (n=0; n<s.length;n++)
  document.write (s.charAt(n) + " * " )

// better!

s.split('').join('*')
```

82

ITERATION USING MAP

Map produces a new array from an existing array.

Items in the new array are calculated by applying the given function to the corresponding item from the original array.

```
//add 3 to each item
numbers = [2,4,6,8,10];
numbers = numbers.map
  (function(item) {
       item + 3;
})
```

83

USING ARROW FUNCTIONS WITH FOREACH OR MAP

```
//add all of the items
numbers = [2,4,6,8,10];
sum=0;
numbers.forEach ((item)=> {sum+= item;})

//add 3 to each item
numbers = [2,4,6,8,10];
numbers.map (item=> item + 3);   //why is { } not needed?
```

SEARCH AN ARRAY USING FOREACH

```
// Display the index of the array element that matches

arr = ["jim", "bill", "sam", "natalie", "sally", "fran"];
match = "sam";

index=-1;
arr.foreach((item, i) =>{if (index==-1 && item == match) index = i;})

document.write( index==-1 ? "no match" : "match found at position: " + index)
```

85

SEARCH AN ARRAY USING REDUCE

```
// Display the index of the array element that matches
arr = ["jim", "bill", "sam", "natalie", "sally", "fran"];
match = "sam";

index = arr.reduce((idx, item, i)=> item == match? i : idx, -1)

document.write( index==-1 ? "no match" : "match found at position: " + index)
```

86

ASSOCIATIVE ARRAYS

- An associative array associates a string value with each array element
- Ie: the index is a string.
- This creates a set of key-value pairs

```
flowers = [];
flowers["daisy"] = 12
flowers["rose"] = 15
flowers["carnation"] = 8
document.write(flowers["rose"])  //displays 15

// Iterate through the keys
for (flower in flowers)            //daisy $12 ~ rose $15 ~ carnation $8 ~
   document.write(flower + " $" + flowers[flower] + " ~ ")
```

87

OBJECTS IN JAVASCRIPT

- You can also create an associative array using object notation

```
var flowers = {
        daisy: 12,
        rose: 15,
        carnation: 8
}
document.write (flowers['rose’])  //displays 15
document.write (Object.keys(flowers))
// displays: daisy,rose,carnation
```

88

CLASSES IN JAVASCRIPT

- A class can be used to instance several objects
- Classes can have properties and methods
- Classes can be created using a function which also serves as the constructor
- The keyword "this" refers to the current object
  - function Rectangle(len, wid)
    {
    this.length = len;
    this.width = wid;
    }
- Instance the class using *new*:
  - r = new Rectangle(3,4);
- Access the data members using dot notation
  - area = r.length * r.width;

89

89

ADDING METHODS

- The real utility of an object is when it has methods
- A method is a function that uses the keyword "this" to refer to members of the object.

```
function area()
{
	return this.length * this.width;
}
```

- Attach the method to the class by assigning it in the object constructor

```
function Rectangle(len, wid)
{
     this.length= len;
     this.width= wid;
     this.area= area;
}
```

- Call the method using the dot notation

```
r = new Rectangle(3,4);
rectArea = r.area();
```

90

90

METHODS CREATED WITHIN THE CLASS DEFINITION

```
function Rectangle(len, wid)
{
  this.length = len;
  this.width = wid;
  this.area = function()
  {
        return this.length * this.width;
  }
  this.perimeter= ()=>this.length*2+this.width*2;
}
```

91

EVENT DRIVEN PROGRAMMING

- **Linear programming is sequential**
  - One instruction cannot execute until the last has completed
  - Every instruction in the sequence will execute
- **Event Programming happens on triggers**
  - If the event doesn't happen, the code will not run
  - Event examples are a key click, page loaded, window resized
  - An *Event Handler* is executed in response to an event

Common events:
    Button click
    Text box change
    Select box change
    Enter or exit a form element (focus/blur)
    Page loaded or unloaded

92

HOW TO ASSIGN AN EVENT HANDLER: THREE OPTIONS

1. Within the HTML code:
   - `<input type ="button" value="Press This!"  name="btn1" onclick="doSomething()">`

2. *** Using the Javascript event property
   - `btn1.onclick = doSomething;`
     `or`

   -
     ```
     btn1.onclick = function()        // anonymous function
       { /* do something here */ }
     ```

3. Add an event listener
   - `btn1.addEventListener("click", doSomething);`
   - More than one event handler can be associated with an event this way.

93

93

WHEN TO ASSIGN AN EVENT HANDLER

- Event handlers cannot be assigned *until an element is fully loaded*
  - Prior to that the page elements may not exist
- Best practice
  - window.onload=init;
  - window.onload= function()
    ```
    {
            // do event associations here
    }
    ```

94

94

EXAMPLE: ONLOAD EVENT

```
<script>
      window.onload = function() {
            alert (“Page is loaded!");
            btn1 = document.getElementById('button1');
            btn1.onclick = function() {
                  /* event handler for button click*/
             }
            // additional event handler assignments here
      }
</script>
```

95

95

READING/CHANGING PAGE CONTENT

- Change the text
  - innerHTML vs innerText
  - Example: obj.innerHTML = “hello!<br>”
  - Both access the contents of an element (innerHTML can include HTML)
- Read the class(es)
  - c = obj.className
- Change a class
  - Changing the class/classes of an element is a powerful way to affect the look of the element
  - obj.classList.add(‘n’)
  - obj.classList.remove(‘n’)
- Change or access an attribute
  - obj.getAttribute, obj.setAttribute
  - Change an HTML attribute – i.e., change the src of an <img> will change the file of the image that is displayed
- Change CSS property - value pairs for an element
  - obj.style.text-align = “center”

96

96

REFERENCING AN ITEM ON THE PAGE

- HTML elements on a page can be represented as a JavaScript object so that they can be read and/or modified.
- There are two options for accessing these objects
  - Use the hierarchy as defined by the JavaScript DOM
  - Use methods of the document object

97

ACCESSING AN OBJECT USING THE DOM

- For example, given:

```
<form name='form1'>
        <input type='text' name= 'text1' id='text1id' class='class1' >
</form>
```

Access text1:

```
document.form1.text1
```

Access the contents of text1:

```
document.form1.text1.value
```

98

ACCESSING AN OBJECT PROGRAMMATICALLY

JS Code	What it returns
document.getElementById(n)	an object corresponding to the element with id = n
document.getElementsByName(n)	an array of objects corresponding to the element with name = n
document.getElementsByClassName(n)	an array of objects corresponding to the element that has a class = n
document.querySelector(n)	The first object on the page that matches the selector pattern = n
document.querySelectorAll(n)	An array of objects that match the selector pattern = n

Example, access the value for:`<input type='text' id='txt1' name='mytext'>`

```
val = document.getElementById('txt1').value;
fields = document.getElementsByName('mytext'); val = fields[0].value;
```

99

PROPERTIES OF FORM ELEMENTS

- Text, password, hidden
  - value
  - readOnly
  - size, maxLength
  - disabled
  - select()
- Radio/ Checkbox
  - checked
  - value

- Select
  - selectedIndex
  - length
  - options
    - text
    - value
    - selected

100

100

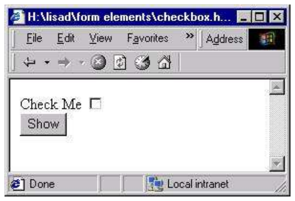
FORM TECHNIQUES
READING FROM A TEXT BOX

Note: This also works for a Password or hidden field

```
<form name="form1">
<input type="text" name="txt1" id="txt1">
<input type="password" name="pw1" id="pw1">
<input type="hidden" value="hidden value" name="h1" id="h1">
</form>
```

```
<script>
function showValue()
{
   val = document.getElementById("txt1").value;
    val = document.form1.txt1.value;
   alert(val);
}
</script>
```

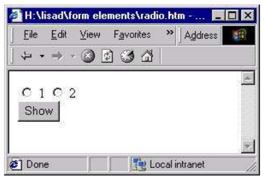
FORM TECHNIQUES
READING FROM A CHECK BOX


H:\lisad\form elements\checkbox.h...
File Edit View Favorites Address
Check Me
Show
Done
Local intranet

Check Me <input type="checkbox" name="chk1" id="chk1">

```
<script>
function showValue()
{
   var val;
   if (document.getElementById("chk1").checked)
            val = "checked";
   else
            val = "not checked";
   alert("The box is" + val);
}
</script>
```

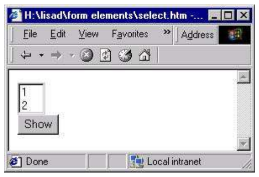
FORM TECHNIQUES
READING FROM A RADIO BUTTON


H:\lisad\form elements\radio.htm - ...
File Edit View Favorites Address
1 2
Show
Done
Local intranet

```
<form name="myForm">
<input type="radio" name="rad1" value="One"> 1
<input type="radio" name="rad1" value="Two"> 2
```

```
<script>
function showValue()
{
   var val;
   if (document.myForm.rad1[0].checked)
            val = document.myForm.rad1[0].value;
   else if (document.myForm.rad1[1].checked)
            val = document.myForm.rad1[1].value;
   else
            val = "none selected“;
   alert(“Selected item is " + val);
}
</script>
```

FORM TECHNIQUES
READING FROM SELECT


H:\lisad\form elements\select.htm - ...
File Edit View Favorites Address
1
2
Show
Done
Local intranet

```
<form name="myForm">
<select name="sel1“ id=“sel1” size=‘2’>
<option value="One"> 1 </option>
<option value="Two"> 2 </option>
</select>
```

Set size attribute to ‘1’ for a drop down list

```
<script>
function showValue()
{
   var val;
    index = document.getElementById(“sel1”).selectedIndex ;
   if (index>=0)
       val =
   document.getElementById(“sel1”).options[index].value;
   else
            val = "none selected“;
   val = document.getElementById(“sel1”).value  // also works
   alert(“Selected item is " + val);
}
</script>
```

VALIDATING FORMS: ERROR MESSAGES

- Despite best efforts for fault-free design, there will still be room for user error
- Users must *always* be informed when they make an error
- Without good error feedback
  - The errors will persist
  - User will go elsewhere

Characteristics of good error messages
1. Clear statement of the problem
2. Avoid humorous error messages
3. Explain how to recover
4. Position the error near the problem
5. Make the message obvious (ex- color in red)

105

105

SOME OPTIONS FOR DISPLAYING ERRORS

- Display first error found in a pop-up message
- Note all errors in a pop-up message
- Display errors only after submit
- Display errors after defocus from each field (onchange or onblur)
- Display errors on screen adjacent to field
- Display indicator (i.e., *) next to field and then list all errors below the form.

106

POSSIBLE VALIDATION ISSUES- TEXT BOX

- Required field not present
- Does it match a pattern (i.e., email or social security)?
- Does it have a minimum or maximum length?
- Is it a number?
- Is it a number within a range?
- Does it match another field (ex: confirming a password or email)

107

VALIDATION ISSUES- CHECKBOX & RADIO BUTTON

- Is the checkbox checked?
- Are a certain number of checkboxes checked?
- Is any radio button selected?
- Is any other than the first radio button selected?
- Is the “other” button selected- if so, may need to also inspect a text box

108

VALIDATION ISSUES- SELECT

- Is any option selected?
- Is any other than the first option selected?
- Is a particular option selected?
- For multiple option- are a minimum or maximum number of options selected?

109

FORM VALIDATION - HOW TO

- Add an *onsubmit* event handler to the <form> tag

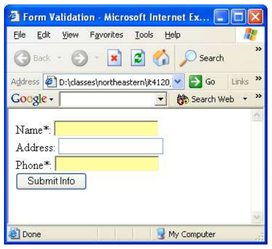
```
onsubmit="return validate()"
```

- The validation function returns “true” to indicate that the form action should occur, false otherwise.
- Do not use a *click* event on the submit button

110

EXAMPLE: FORM VALIDATION

```
<form
    onsubmit="return validate()"
   name = “data“
  method=“post”>
Name*: <input type = "text" name="name"> <br />
Address: <input type = "text” name="address"> <br />
Phone*: <input type = "text" name="phone"> <br />
<input type = "submit" value = "Submit”>
</form>
```


Form Validation - Microsoft Internet Ex...
File Edit View Favorites Tools Help
Back Search
Address D:\classes\northeastern\it4120 Go Links
Google Search Web
Name*:
Address:
Phone*:
Submit Info
Done My Computer

The event can also be assigned using window.onload

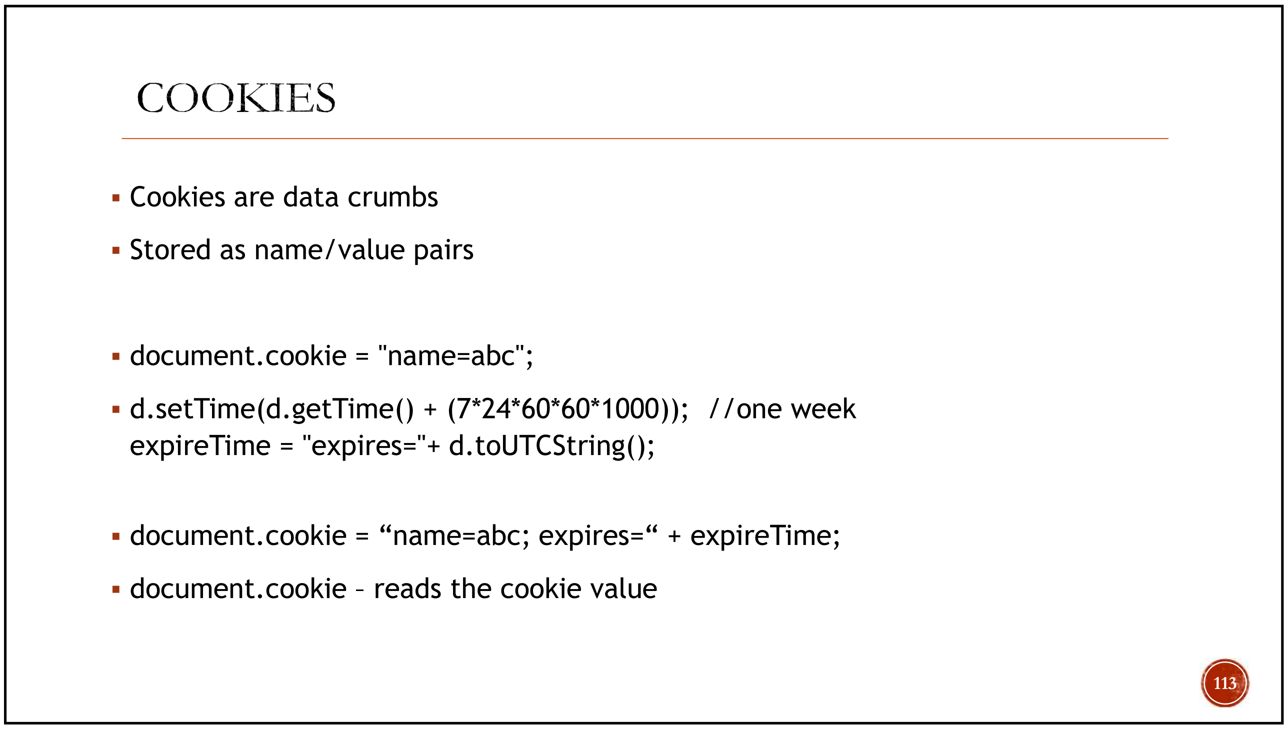
FORM VALIDATION: CHECK FOR REQUIRED ITEM

```
if (document.data.name.value == "")                 // check if field is blank
{
        alert("Must enter a value for the name");   // tell user what is wrong

        document.data.name.focus();                 // put cursor in text box

        document.data.name.select();                // highlight text in the box

        return false;                               // prevent form action
}
```



COOKIES

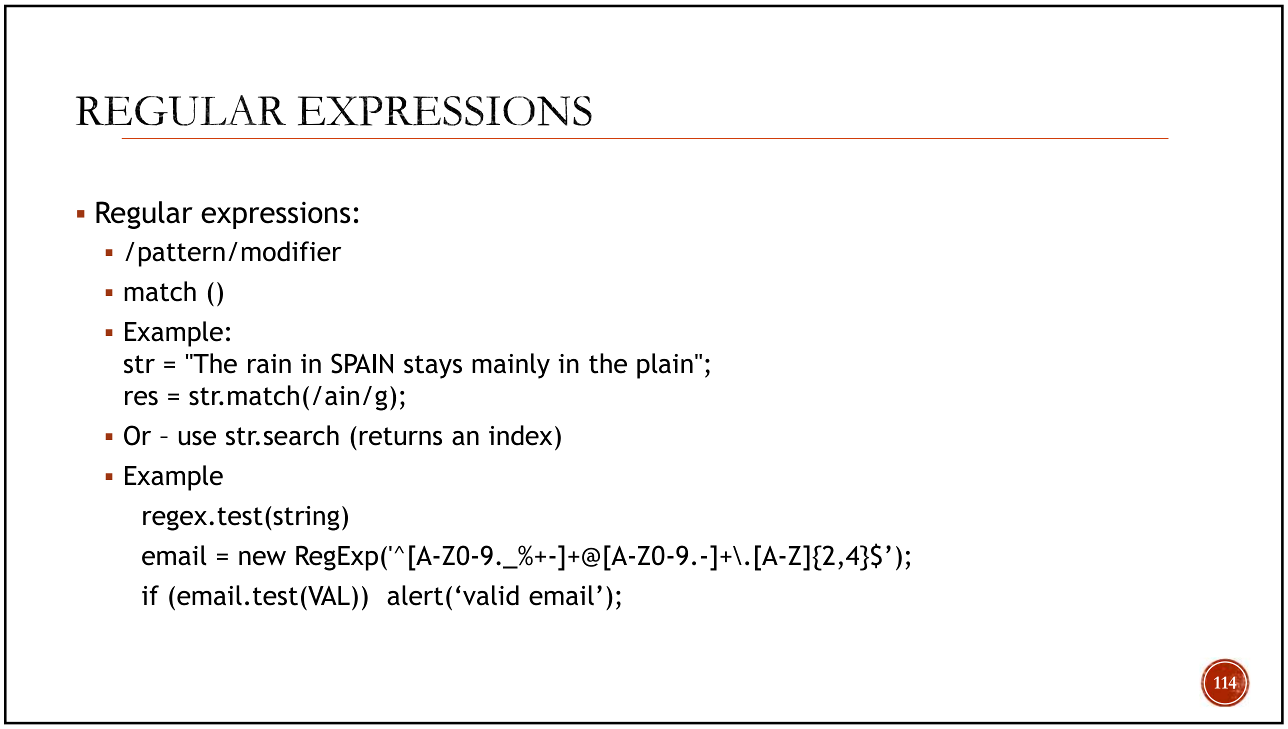
- Cookies are data crumbs
- Stored as name/value pairs

- document.cookie = "name=abc";
- d.setTime(d.getTime() + (7*24*60*60*1000)); //one week
  expireTime = "expires="+ d.toUTCString();

- document.cookie = “name=abc; expires=“ + expireTime;
- document.cookie – reads the cookie value

113

113



REGULAR EXPRESSIONS

- Regular expressions:
  - /pattern/modifier
  - match ()
  - Example:
    str = "The rain in SPAIN stays mainly in the plain";
    res = str.match(/ain/g);
  - Or – use str.search (returns an index)
  - Example
    regex.test(string)
    email = new RegExp('^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$’);
    if (email.test(VAL)) alert(‘valid email’);

114

114

REGULAR EXPRESSION EXAMPLES
BEGINS OR ENDS WITH

^The"	matches any string that starts with "The".
"of despair$"	matches a string that ends in with "of despair".
"^abc$"	a string that starts and ends with "abc" - effectively an exact match comparison.
"notice"	a string that has the text "notice" in it.

115

115

JQUERY

116

JQUERY

- Add-on library written in Javascript
- Go to www.jquery.com to download or use CDN (https://code.jquery.com/)
- Reference the library as an *external Javascript* source file
- JQuery
  - Uses $() notation
  - Object oriented
  - Alternate way to
    - Attach events
    - Update content
    - Special F/X

117

EXAMPLE

Javascript only

```
x = document.getElementById("a1").value;
message = "The answer is " + x * 2;
document.getElementById("result").innerHTML = message;
```

Javascript with jQuery

```
x =$("#a1").val();
message = "The answer is " + x * 2;
$("#result").html(message);
```

118

118

SELECTING PAGE ELEMENTS
(SAME SELECTORS AS CSS)

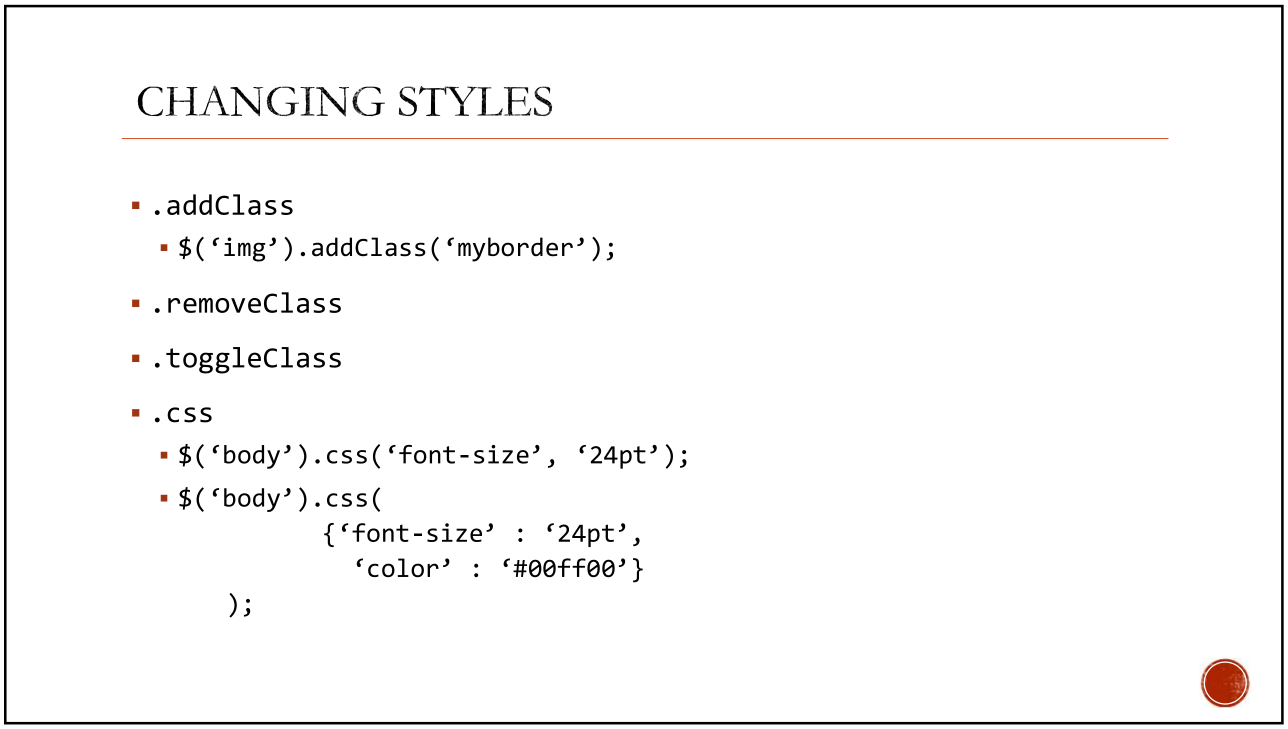
▪ id	$('#name')	
▪ Tag	$('tag')	
	$('tag:even')	also ('tag:odd')
▪ Class name	$('.className')	
▪ Tag with class	$('span.error')	
▪ Child of tag	$('ul>li')	only <li> within <ul>
▪ Child of id	$('#navbar a')	only links within navbar
▪ Adjacent tags	$('h2 + p')	only <p> after <h2>
▪ Attributes	$('p[align]')	
	$('p[align=center]')	match attr value
	$('a[href^=http://]')	starts with http://
	$('a[href.=.pdf]')	ends with .pdf
	$('a[href*=sitex]')	sitex is anywhere in the href string

119

CHANGING PAGE CONTENT

- .html
  - $('#err_name').html ("Must provide your name");
- .append .prepend
- .after .before
- .attr
  - $('#slides img').attr('src', 'slide2.jpg');
- .removeAttr

120

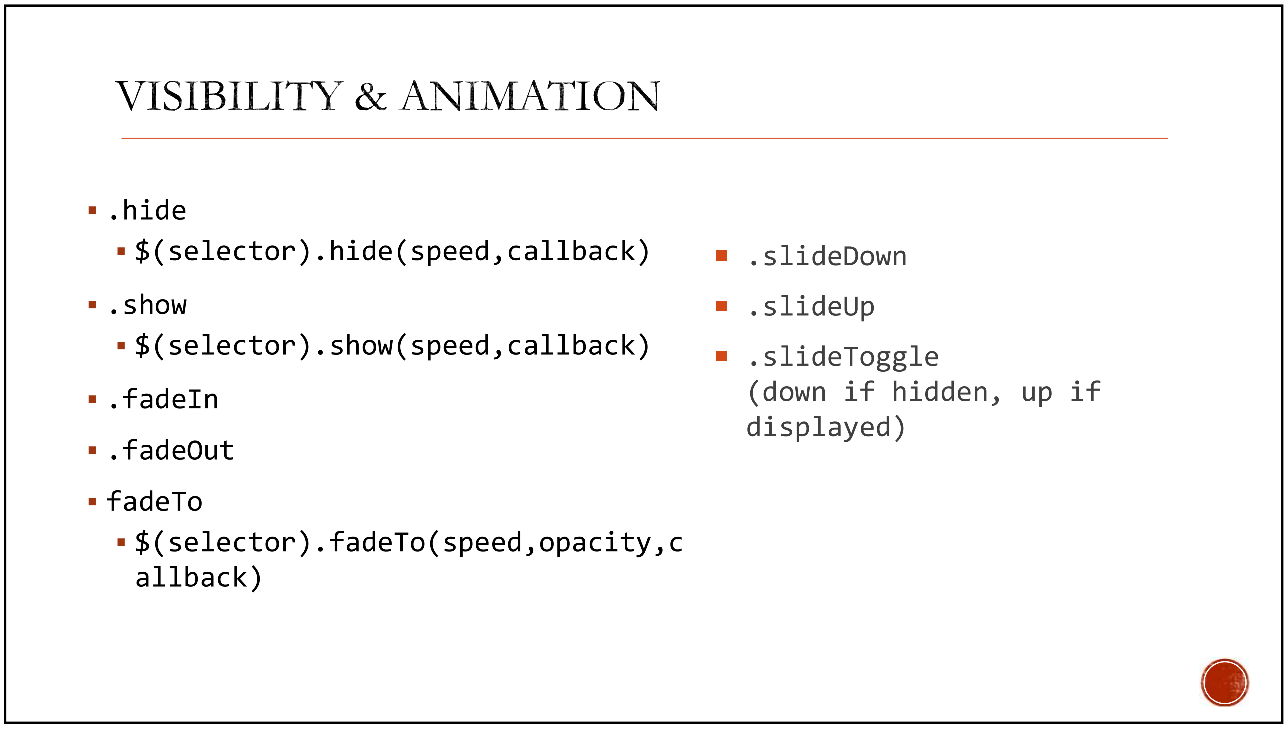


CHANGING STYLES

- .addClass
  - $(‘img’).addClass(‘myborder’);
- .removeClass
- .toggleClass
- .css
  - $(‘body’).css(‘font-size’, ‘24pt’);

  -
    ```
    $(‘body’).css(
              {‘font-size’ : ‘24pt’,
                ‘color’ : ‘#00ff00’}
        );
    ```

121



VISIBILITY & ANIMATION

- .hide
  - $(selector).hide(speed,callback)
- .show
  - $(selector).show(speed,callback)
- .fadeIn
- .fadeOut
- fadeTo
  - $(selector).fadeTo(speed,opacity,callback)

- .slideDown
- .slideUp
- .slideToggle
  (down if hidden, up if displayed)

122

ATTACHING EVENT HANDLERS

- *The event is set by using a method.*
- Assign event handlers using the selector notation along with the event method
  - $('#menu1').mouseover(func);
  - $('#menu1').bind('mouseover', 'func');
  - $('#menu1').mouseover(function() { });
- *Example*:

```
$('#menu1').mouseover(function() {
     $('#submenu').show();
   });
```

Frequently Used Events

click	dblclick
mouseenter	mouseleave
keypress	keyup
focus	blur
hover	toggle
submit	load
resize	

`$(document).ready(function(){})`
replaces window.onload

123

EXAMPLE

Problem: Detect when a button is pressed and display a message on the page

- The HTML

```
<body>
<div id="button1">Press Me</div>
<div id="message1">&nbsp;</div>
</body>
```

124

EXAMPLE, CONTINUED

- The CSS:

```
#button1 {
       font-family: Arial, Helvetica, sans-serif;   font-size: 36px;
       font-weight: bold; text-align: center;  line-height: 220px;
       color: #f7f7f7;     width: 210px; height: 220px;  margin-bottom: 20px;
       background-color: #2A00FF;
       }
#button1:hover { opacity: .5; }
#message1 {font-size: 20px;}
#message1.showme {
    background-color: #ECEA34;  padding: 6px 8px; display:inline-block;
}
```

125

EXAMPLE, CONTINUED

- The jQuery code:

```
 <script language="javascript">
$(document).ready (
    function ()
    {
           $("#button1").click(function(e) {
              $("#message1").html("That tickles!");
              $("#message1").addClass("showme");
           });
    }
</script>
```

126

127 APPENDIX: JAVASCRIPT DOM

127

DOM PAGE ELEMENTS

- window
  - location
  - status
  - document
    - images
    - forms
      - Textbox (includes hidden, password, textarea)
      - Radio button/ Checkbox
      - Select element
        - options
      - Button
    - anchors
- Tables
      - rows
        - cells
- Style

128

128

WINDOW

- document
- location
  - href
  - reload()
- navigator
  - appName
  - appVersion
  - cookieEnabled
- Status
- moveTo()
- open()

129

129

DOCUMENT

- cookie
- title
- write()
- getElementByID()
- getElementsByName()
- getElementsByTagName()

130

130

IMAGE

- src
- height, width
- lowsrc
- border

131

131

FORM

- elements
  - name
  - value
  - type
  - focus()
- length
- action
- method
- submit(), reset()

132

132