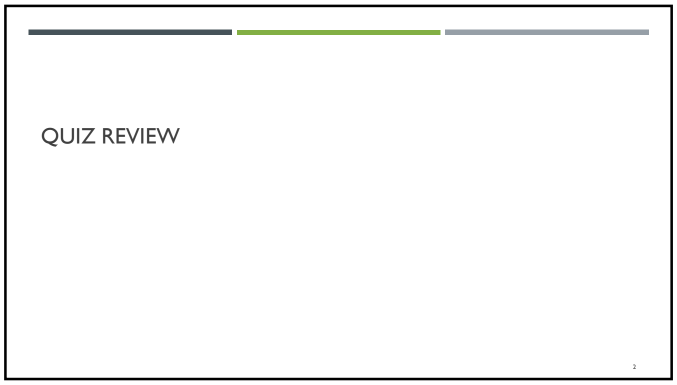
COMP 20 WEB PROGRAMMING

WEEK 7

1

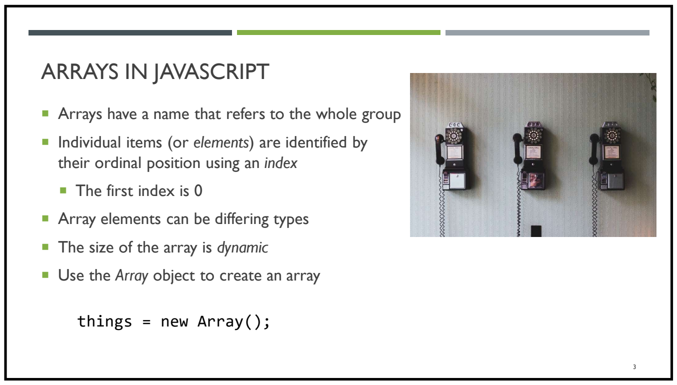
1



QUIZ REVIEW

2

2



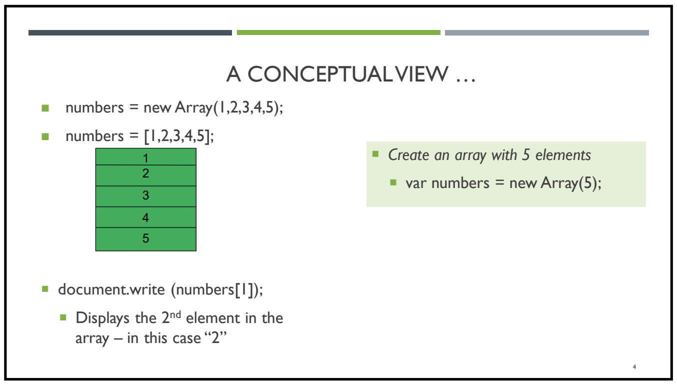
ARRAYS IN JAVASCRIPT

- Arrays have a name that refers to the whole group
- Individual items (or *elements*) are identified by their ordinal position using an *index*
  - The first index is 0
- Array elements can be differing types
- The size of the array is *dynamic*
- Use the *Array* object to create an array

```
things = new Array();
```

3

3



A CONCEPTUAL VIEW …

- numbers = new Array(1,2,3,4,5);
- numbers = [1,2,3,4,5];

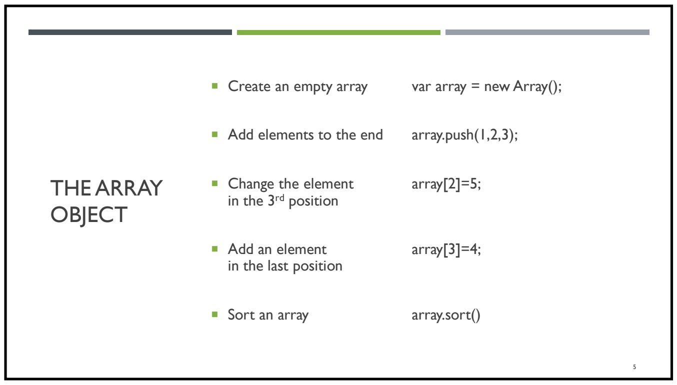
1
2
3
4
5

- *Create an array with 5 elements*
  - var numbers = new Array(5);

- document.write (numbers[1]);
  - Displays the 2nd element in the array – in this case "2"

4

4

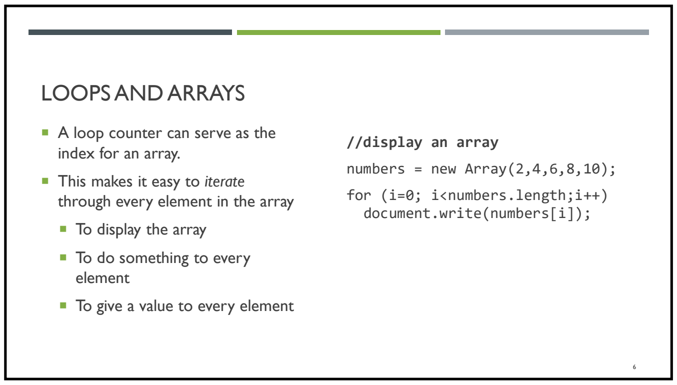


THE ARRAY OBJECT

Create an empty array	var array = new Array();
Add elements to the end	array.push(1,2,3);
Change the element in the 3rd position	array[2]=5;
Add an element in the last position	array[3]=4;
Sort an array	array.sort()

5

5



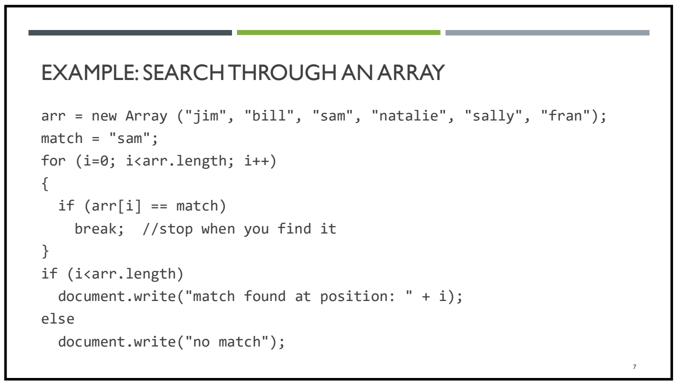
LOOPS AND ARRAYS

- A loop counter can serve as the index for an array.
- This makes it easy to *iterate* through every element in the array
  - To display the array
  - To do something to every element
  - To give a value to every element

```
//display an array
numbers = new Array(2,4,6,8,10);
for (i=0; i<numbers.length;i++)
  document.write(numbers[i]);
```

6

6

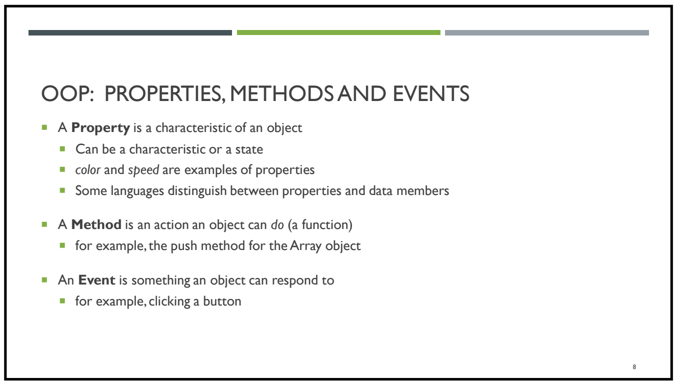


EXAMPLE: SEARCH THROUGH AN ARRAY

```
arr = new Array ("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");
```

7

7

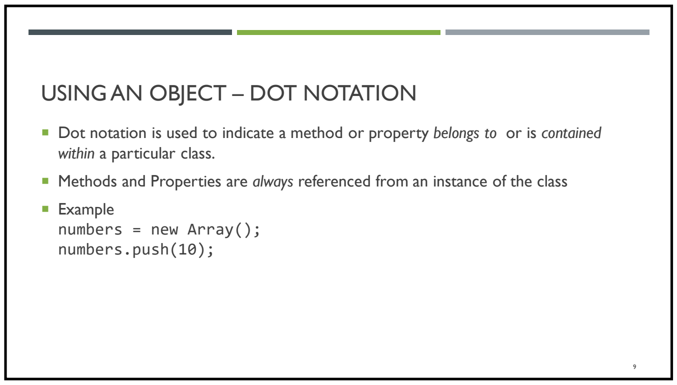


OOP: PROPERTIES, METHODS AND EVENTS

- A **Property** is a characteristic of an object
  - Can be a characteristic or a state
  - *color* and *speed* are examples of properties
  - Some languages distinguish between properties and data members
- A **Method** is an action an object can *do* (a function)
  - for example, the push method for the Array object
- An **Event** is something an object can respond to
  - for example, clicking a button

8

8

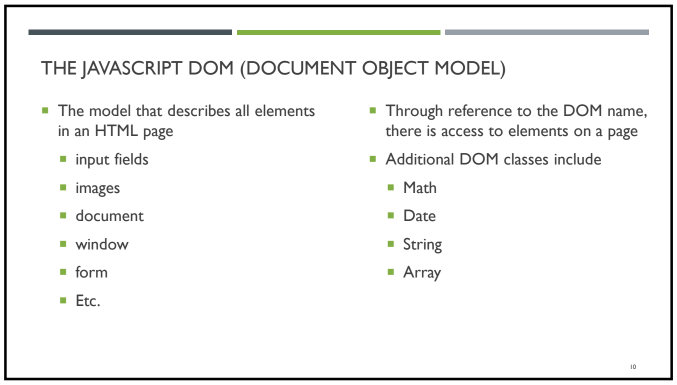


USING AN OBJECT – DOT NOTATION

- Dot notation is used to indicate a method or property *belongs to* or is *contained within* a particular class.
- Methods and Properties are *always* referenced from an instance of the class
- Example
```
numbers = new Array();
numbers.push(10);
```

9

9

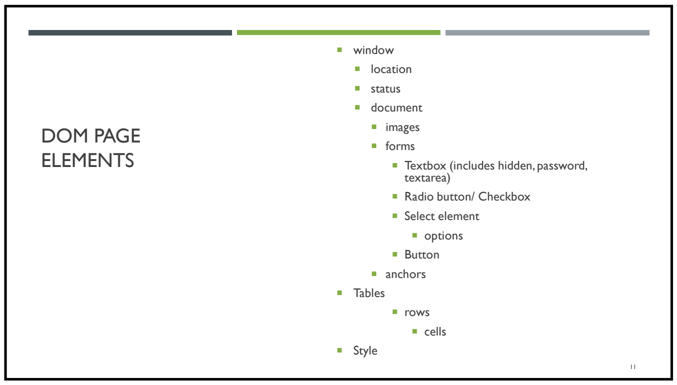


THE JAVASCRIPT DOM (DOCUMENT OBJECT MODEL)

- The model that describes all elements in an HTML page
  - input fields
  - images
  - document
  - window
  - form
  - Etc.
- Through reference to the DOM name, there is access to elements on a page
- Additional DOM classes include
  - Math
  - Date
  - String
  - Array

10

10

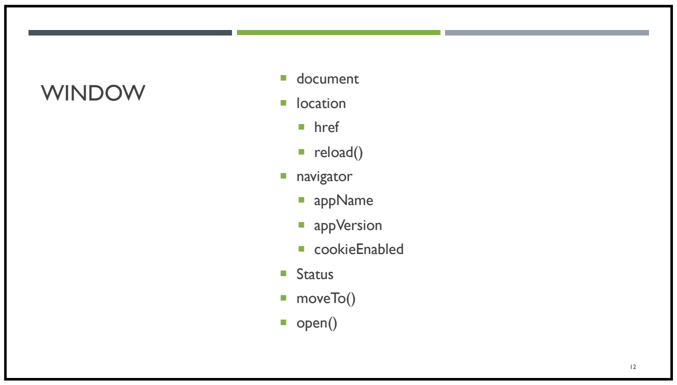


DOM PAGE ELEMENTS

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

11

11

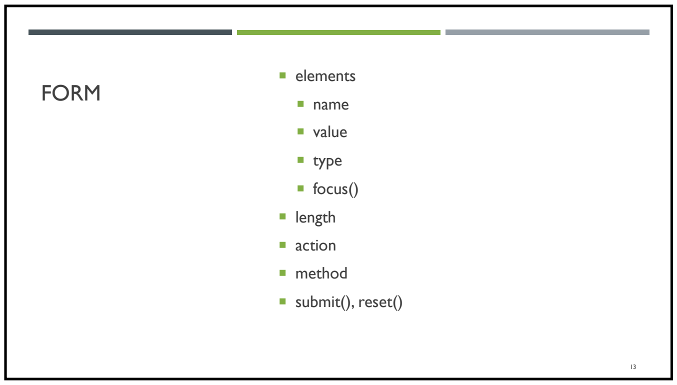


WINDOW

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

12

12

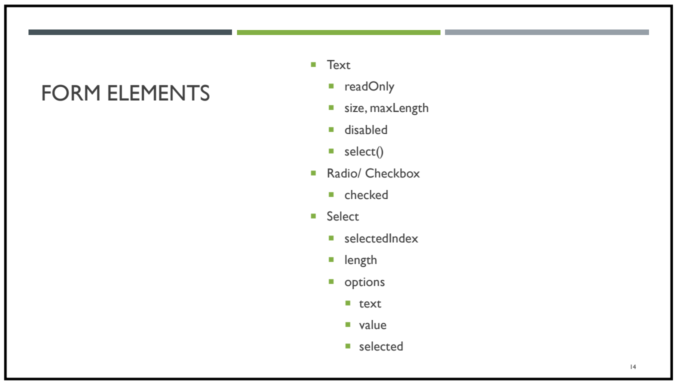


FORM

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

13

13

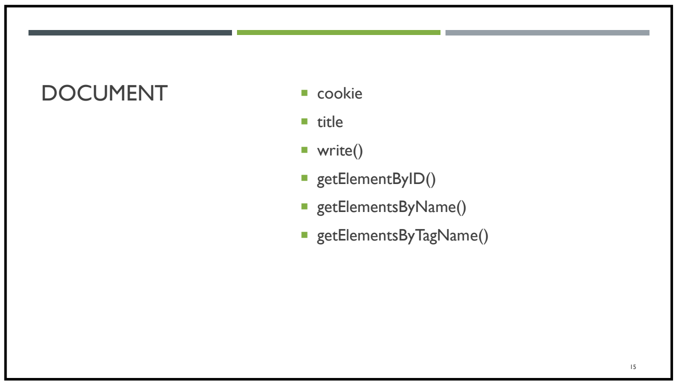


FORM ELEMENTS

- Text
  - readOnly
  - size, maxLength
  - disabled
  - select()
- Radio/ Checkbox
  - checked
- Select
  - selectedIndex
  - length
  - options
    - text
    - value
    - selected

14

14

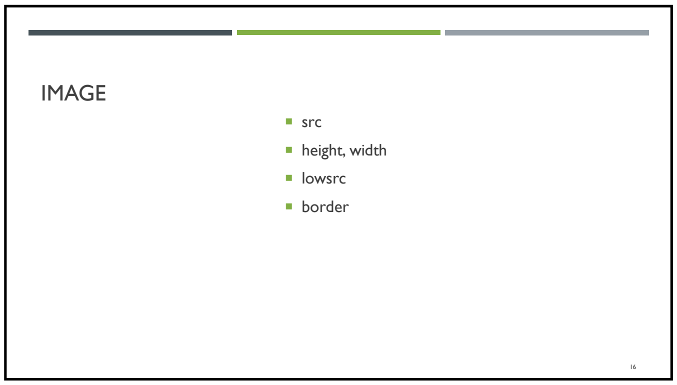


DOCUMENT

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

15

15



IMAGE

- src
- height, width
- lowsrc
- border

16

16

THE STRING OBJECT

- 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

17

17

THE DATE OBJECT

- 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)

18

18

GROWING YOUR OWN: DEFINING OBJECTS IN JAVASCRIPT

- Start by creating a *constructor* method.
- The keyword "this" is used to reference 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 the dot notation
  - area = r.length * r.width;

19

19

ADDING METHODS

- The real utility of objects is when methods are utilized
- Create functions as usual EXCEPT – use the keyword "this" to reference internal members of the object.
  - function area()
    ```
    {
        return this.length * this.width;
    }
    ```
- Attach the method to the class within the original constructor method
  - 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();

20

20

ANONYMOUS FUNCTIONS

- Are useful when you do not need to reference the function by a separate name
  - function Rectangle(len, wid)

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

21

21

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)

22

22

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.

23

FORM VALIDATION

- Add *onsubmit* event handler to the <form> tag
  - onsubmit="return validate()"

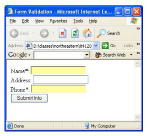
- Add a validation function that returns "true" to indicate that the form action should occur.

- Do not use *onclick*

24

VALIDATING A FORM

```
<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 bound via window.onload

25

25

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

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

26

26

"WITH" CONSTRUCT

- Works with DOM path
- Useful when validating several fields at the same time

-
  ```
  with (document.form1)
  {
   if (txtName.value == "")
   {  // error processing here }

  if (txtPassword.value == "")
   {  // error processing here }
  }
  ```

27

POSSIBLE TEXTBOX ISSUES

- 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)

28

POSSIBLE CHECKBOX & RADIO BUTTON VALIDATION

- 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

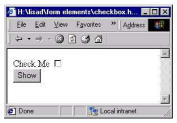
29

POSSIBLE SELECT VALIDATION

- 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?

30

FORM TECHNIQUES ~ READING FROM A CHECK BOX


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

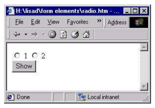
```
Check Me <input type="checkbox" name="chk1" id="chk1">
```

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

31

31

FORM TECHNIQUES ~ READING FROM A RADIO BUTTON

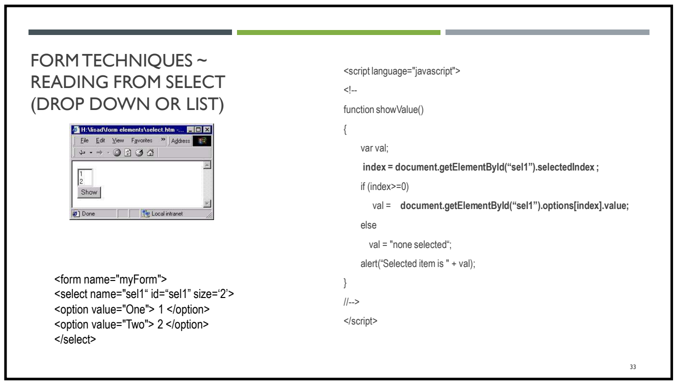

H:\lisad\form elements\radio.htm
File Edit View Favorites Address
1 2
Show
Done
Local intranet

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

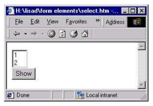
```
<script language="javascript">
<!--
function showValue()
{
    var val;
    if (document.myForm.rad1[0].checked)
      val = document.myForm.rad1[0].value;
    else if (document.myForm.rad1[1].checked)
      val = document.myForm.rad1[1].value;
    else
      val = "none selected";
    alert("Selected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

32

32



FORM TECHNIQUES ~ READING FROM SELECT (DROP DOWN OR LIST)

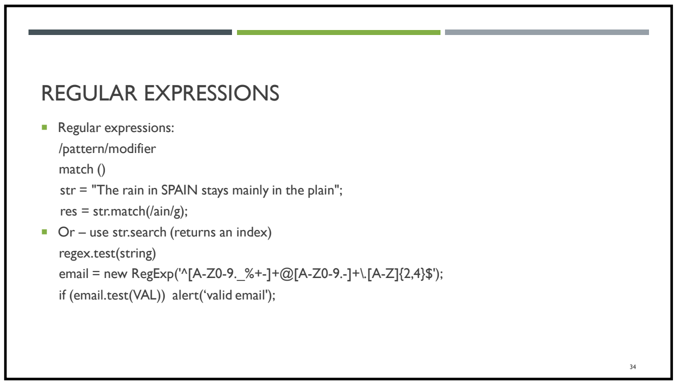

H:\lisad\form elements\select.htm - ...
File Edit View Favorites Address
1
2
Show
Done
Local intranet

```
<form name="myForm">
<select name="sel1" id="sel1" size='2'>
<option value="One"> 1 </option>
<option value="Two"> 2 </option>
</select>
```

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

33

33

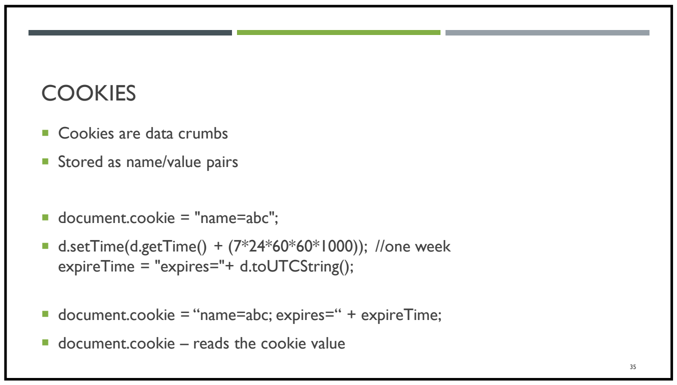


REGULAR EXPRESSIONS

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

34

34



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

35

35