SENG1050
Web Technologies

Lecture Week 8: JavaScript

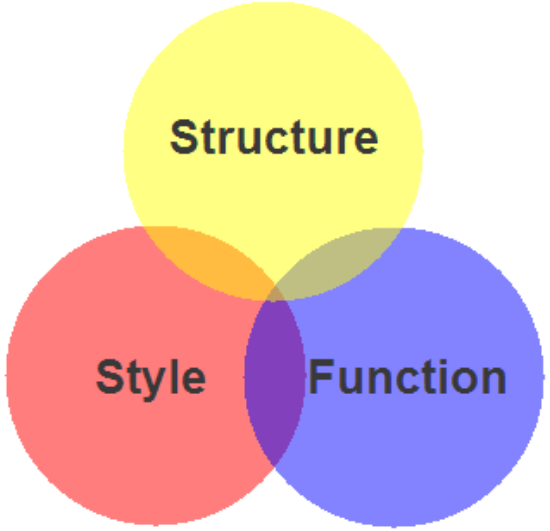
THE UNIVERSITY OF NEWCASTLE AUSTRALIA
FACULTY OF ENGINEERING AND BUILT ENVIRONMENT
www.newcastle.edu.au

Lecture Overview

- Introduction
- JavaScript Basic
  - Variable
  - Types
  - Operators
  - Popup Box
  - Controls
  - Functions
  - DOM
- Form Validation
- Comments
- Methods

Dr Mira Park

Background



Triad of technologies for web developers:

1. **HTML:** Specifies the content of webpages
2. **CSS:** Specifies the presentation of webpages
3. **JavaScript:** Specifies the behaviour of web pages

Introduction

- JavaScript is “the” **programming language** of the web.
- Powerful feature: Any code written in JavaScript can **run** as-is in **any modern web browser**.
- It is the most **ubiquitous** programming language in history because it is used in all modern web browsers on
  - Desktops
  - Game consoles
  - Tablets
  - Smart phones
- Easy to learn, Developed by Netscape

What Can JavaScript Do?

- adding dynamic features to Web pages
- Script is executed when it is encountered as the HTML is loaded
  - Script can perform calculations
  - Script can define functions for use by other scripts on the page
  - Script can dynamically generate HTML

```
write("<h1>My Home Page</h1>")
```

What Can JavaScript Do?

- JavaScript enables shopping carts, form validation, special graphic and text effects, image swapping, image mapping, clocks, handling cookies and more.
  - http://www.creativebloq.com/web-design/examples-of-javascript-1233964
  - https://www.firebase.com/docs/web/examples.html -- source code

- **validation of form data** **(probably the most commonly used application)**

JavaScript

- The name “JavaScript” is somewhat misleading
- JavaScript ≠ Java
  - It “looks” like Java, but it is not Java
  - It is *object-oriented*
  - It is much simpler than Java
  - JavaScript is *(usually) interpreted*
- JavaScript ≠ HTML
  - But it was designed to work inside HTML

Object-Oriented

- An analogy
  - A car is a physical object
  - A car has certain properties – make, model, colour, number of doors, distance driven, age
  - There are certain actions (methods) you can perform on the car – start, stop, turn, spray paint; they often affect the car's properties
  - Cars can respond to certain events – e.g., colliding with another car

Object-Oriented

- Important Terminology
  - **Object**: A black-box that stores information
  - **Property**: Directly accessible pieces of information
  - **Method**: Something an object can DO – often used to access or change information
  - **Event**: An actions or occurrence that affects an object – **triggers** a method

Object-Oriented and JavaScript

- In JavaScript…
  - A HTML document is an *object* (**`document`**)
  - It has *properties*
    - **`title="A new Title"`**
  - It has *methods*
    - **`write("<h1>My Home Page</h1>")`**
  - You can link methods with certain *events*
    - **`onLoad`** and **`onClick`**

Structure


```
<!DOCTYPE html>
<html>
   <head>
      <script type="text/javascript">
            javascript code
                  :
      </script>
   </head>
   <body>
      :
   </body>
</html>
```

<script type="text/javascript">

Container tag for JavaScript

Can be in **<head>** or **<body>**

Can have more than one **<script>**

Dr Mira Park

Variables

- Variables are used to store data.
- A variable is a "**container**" for information you want to store.
- A variable's value can change during the script.
- You can refer to a variable by **name** to see its value or to change its value.

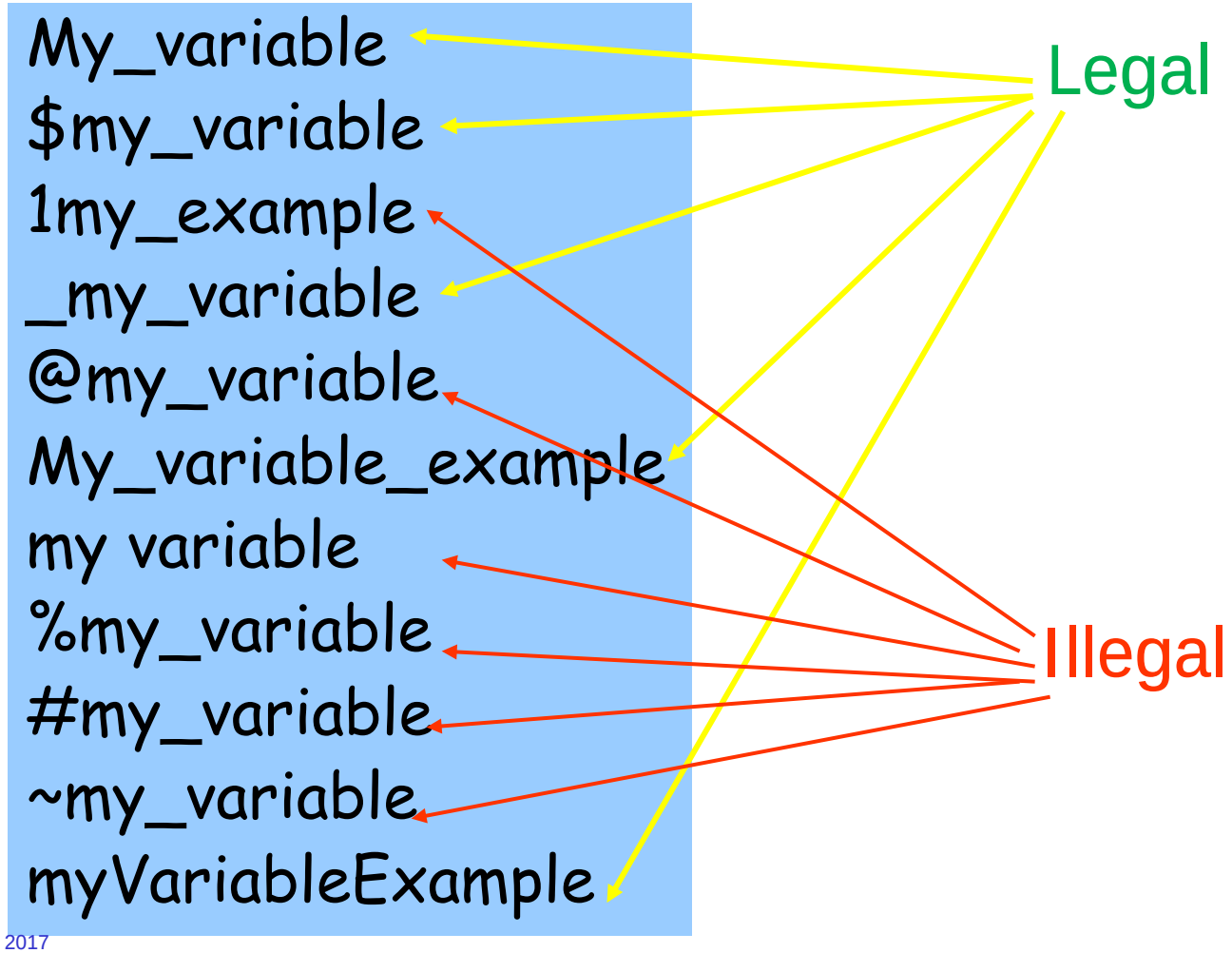
Variables - Rules for variable names

- Variable names are case-SENSITIVE
  - **`index != INDEX != Index`**
- An *identifier* is a unique name (for a variable, an object, or a function)
  - Must start with **a letter** or the **underscore character or $**
  - Can contain **letters** (both upper/lowercase) and **digits**
- Contains only 'A' – 'Z', 'a' – 'z', '0' – '9', '_', '$'
- First character cannot be a digit
- Cannot be reserved words or keywords

Variables

- Variables contain values and use the equal sign (=) to specify their value.
- Variables are created by declaration using the **`var`** command with or without an initial value state.
  - e.g. **`var valueA;`**
  - e.g. **`var valueA = 10;`**
  - **`var valueB = 100;`**

Variables


My_variable
$my_variable
1my_example
_my_variable
@my_variable
My_variable_example
my variable
%my_variable
#my_variable
~my_variable
myVariableExample
Legal
Illegal

Variables and Scope

- *Scope* = places in the program where a particular identifier is associated with a particular data store
- **`var`** ***identifier*****`;`**
  - The scope of ***identifier*** is …
    - The function in which **`var`** ***identifier*** appears = *local*
    - Otherwise, the entire Web page (including frames?) = *global*
- **`function`** ***identifier*****`(`*****parameters*****`) { … }`**
  - The scope of the ***parameters*** is **`{ … }`**
  - The scope of ***identifier*** is the entire Web page

```
// code here can not use carName

function myFunction() {
    var carName = "Volvo";  //local

    // code here can use carName

}
```

```
var carName = " Volvo";  //global

// code here can use carName

function myFunction() {

    // code here can use carName

}
```

Primitive Types

- Five primitive types
  - Number
  - String
  - Boolean
  - Undefined
  - Null

Primitive Types - number

- integers – whole-number values

```
var myint = 10; var myint2 = 20
```

- floating-point – real numbers; with a fraction

```
var myfloat = 10.6;
var mybigfloat = 6.023e23;
```

  - **e** = *scientific notation*
  - **3e5** = 3×10^5 = $3 \times (10 \times 10 \times 10 \times 10 \times 10)$

Primitive Types – boolean, null, undefined

- boolean – can be **`true`** or **`false`**

```
var myboolean = true;
var myotherboolean = false;
```

- null – a null value, a reserved word
- undefined – declared but not assigned a value

Primitive Types – boolean, null, undefined


```
<html>
    <head>
        <title> Null and Undefined example </title>
        <script type="text/JavaScript">
                var test1, test2 = null;
                alert("No value assigned to the variable" + test1);
                alert("A null value was assigned" + test2);
        </script>
    </head>
<body> … </body>
</html>
```

Primitive Types - Strings

- A string variable can store a sequence of alphanumeric characters, spaces and special characters.
- A string can be enclosed by a pair of single quotes (') or double quote (").
- Use escaped character sequence to represent special character (e.g.: `\", \n, \t`)

```
var a_string="This is an example";
var b_string="So is this"; var c_string="";
```

`var myString = "punk" + "rock"` assigns the string `"punkrock"` to the variable `myString`

`var myString = "punk" + " " + "rock"` assigns the string `"punk rock"` to the variable `myString`

Primitive Types

- JavaScript does not stop you changing the type of a variable – this is *weak (loose) typing*

```
var myint=20;
myint="Ok, so now it is a string";
myint=20e10; myint=true;
```

DO NOT DO THIS!

  - Leads to very messy, buggy and hard-to-follow code
- Treat JavaScript as if it were *strongly typed*
  - After you say `var myint=20`, you should only use `myint` as an integer

Operators

- Operators are used to handle variables.
- Types of operators with examples:
  - Arithmetic operators, such as plus (+).
  - Comparisons operators, such as equals (==).
  - Logical operators, such as and (&&).
  - Control operators, such as if.
  - Assignment and String operators.

JavaScript Operators - 1

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2,5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

JavaScript Operators - 3

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

JavaScript Operators - 4

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

JavaScript – Arithmetic Operators

- Example

```
var x = 20; var y = 30; var z = 10;
x = y/(z-20);
y = (z-9) * x;
z--; z++; z--;
```

- What are the values of **x**, **y** and **z** after each statement?

JavaScript – Relational Operators

- Examples

`20 < 30` is `true`,
`30 == 29` is `false`,
`20 <= 20` is `true`,
`8.34 != 1.01` is `true`,
`(-100 > -190)` is `true`,
`(10-20)*10 >= 10-(20*10)` is `true`

JavaScript – Logical Operators

- Examples

```
var a = true; var b = false; var c = true;
```

`a && c` is true

`a || b` is true

`!b` is true

`a && (b || b)` is false

`!(!a || b) && !b` is true

JavaScript – Assignment Operators

- Examples:

```
var x = 1; var y = 2; var z = 3;
```

`x = 10;` `x` is assigned the value `10`

`y += x;` `y` is assigned the value `12` (`2+10`)

`z -= x;` `z` is assigned the value `-7` (`3-10`)

`x *= y;` `x` is assigned the value `120` (`10*12`)

`x /= 12;` `x` is assigned the value `10` (`120/12`)

Popup box - alert method

- **`alert`** is a built-in function – it creates a pop-up dialog displaying a piece of text and an OK button (which closes the dialog)

```
alert(“Form Completed");
```

Popup box - Confirm method

- The confirm methods displays a message provided as a parameter
  - The confirm dialog has two buttons: OK and Cancel
- If the user presses OK, true is returned by the method
- If the user presses Cancel, false is returned

```
var question =
    confirm("Do you want to continue this download?");
```

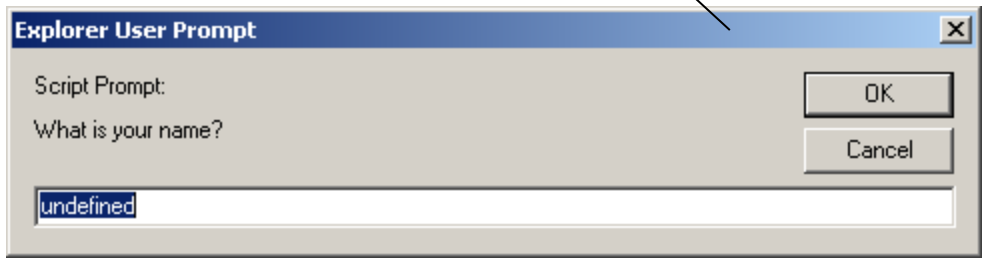
Popup box - Prompt method

- This method displays its string argument in a dialog box
  - A second argument provides a default content for the user entry area
- The dialog box has an area for the user to enter text
- The method returns a String with the text entered by the user

```
name = prompt("What is your name?", "");
```

Three methods


```
<script type="text/javascript">
alert("This is an Alert method");
confirm("Are you OK?");
prompt("What is your name?");
prompt("How old are you?","20");
</script>
```



The first script


```
<!DOCTYPE html>
<html>
    <head>
        <script type="text/javascript">
            alert("Hello world");
        </script>
    </head>
    <body>
        :
        :
    </body>
</html>
```

Control Statement – Decisions

- `if (`*condition*`)`
  `{`*statements*`}`
  - The *condition* between the `( )` is evaluated
  - If *condition* is true, then the *statements* are executed
  - If *condition* is false, then the *statements* are skipped

- `if (`*condition*`)`
  `{`*statementsA*`}`
  `else`
  `{`*statementsB*`}`
  - If *condition* is true, then the *statementsA* are executed
  - If *condition* is false, then the *statementsB* are executed

Control Statement – Decisions

- Example

```
if (warning_level <= 50)
  { document.body.style.color = 'black'; }
else
  { document.body.style.color = 'red'; }
```

Control Statement – Looping

- `for (`*initial*`;` *condition*`;` *update*`)`
  `{`*statements*`}`

1. The first time this `for` loop is encountered, the *initial* expression is evaluated
2. Then, the *condition* is evaluated
   - If *condition* is true, then the *statements* are executed, the *update* expression is evaluated, and repeat from 2
   - If *condition* is false, then the for loop ends – the *statements* and the *update* expression are skipped
   - The *condition* describes the stopping point for the loop

Control Statement – Looping

- Example

```
for (var i=1; i<=8; i++)
   { alert(i); }
```

Control Statement – Looping

- **`while (`*condition*`)`**
  `{` ***statements*** **`}`**

1. The ***condition*** between the **`( )`** is evaluated
   - If ***condition*** is true, then the ***statements*** are executed, and we repeat from 1
   - If ***condition*** is false, then the ***statements*** are skipped and the while loop ends

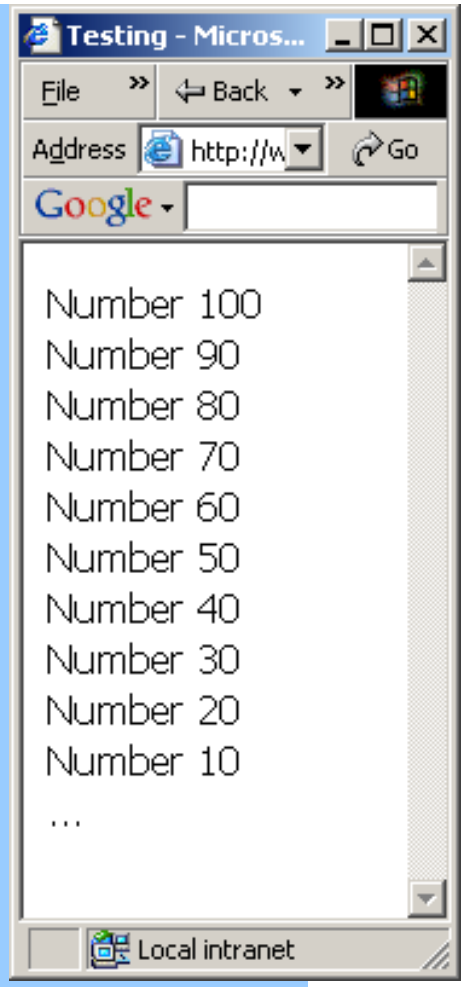
Control Statement – Looping

- Example

```
var i=1;
while (confirm("At " + i + " - keep going?"))
   { i++; }
```

Control Statement – Looping


```
<html>
<head>
<title>While loop example</title>
<script type="text/JavaScript">
      var counter = 100;
      var numberlist = "";
      while (counter > 0) {
            numberlist += "Number " + counter + "<br>";
            counter -= 10;
      }
      document.write(numberlist);
</script>
</head>
<body>
      :
      :
```



Functions (Return Values)

- You can define functions
- Code structure – splitting code into parts (functions)
- Function like mini-program

```
function <function_name> (parameters)
{
        // code segments;
}
```

Functions (Return Values)

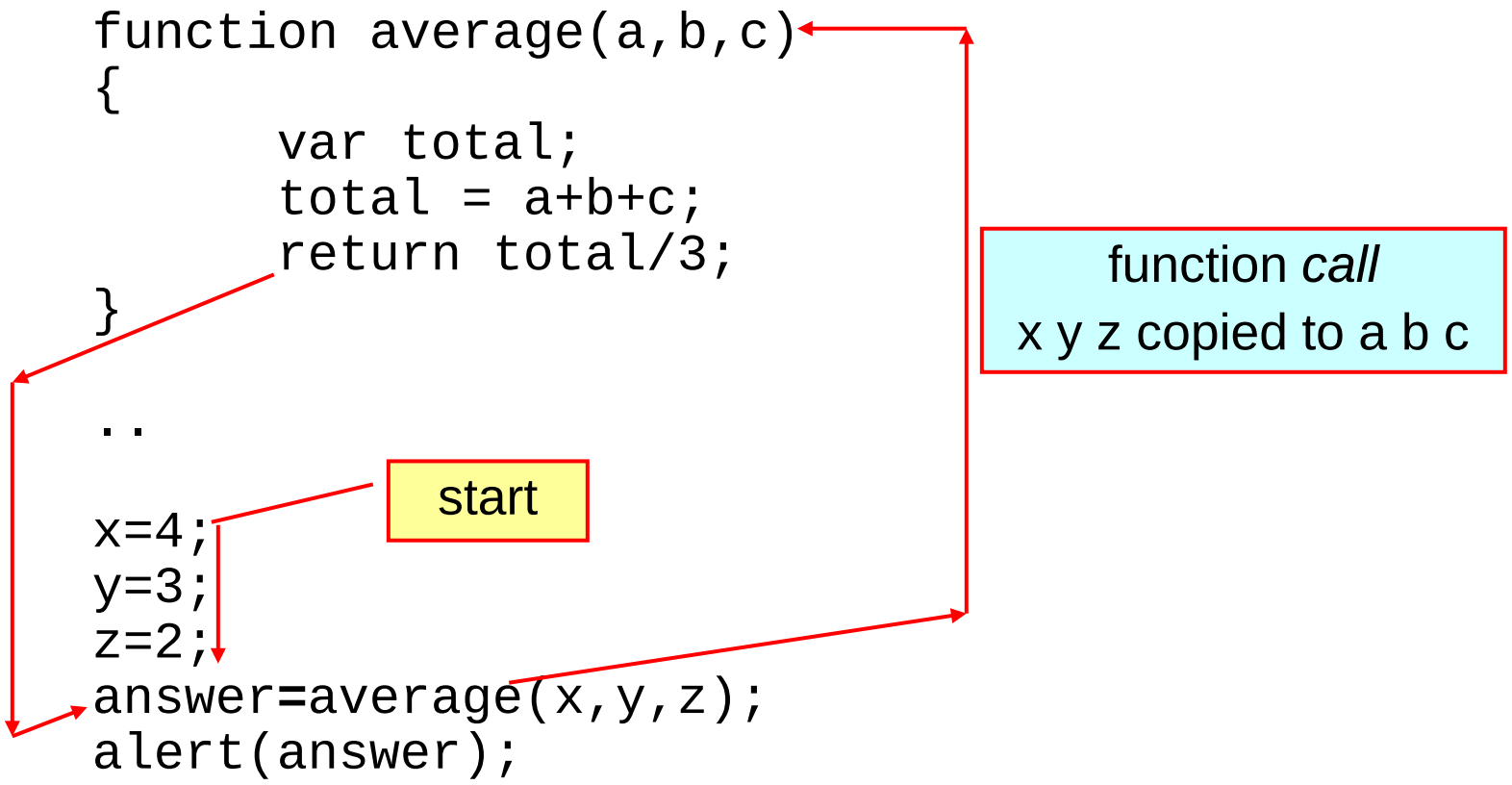
- Data comes in, processed, result returned

```
function average(a,b,c)
{
      var total;
      total = a+b+c;
      return total/3;
}
```

Values come in here

Value returned here

Functions (Return Values)



Functions (Return Values)


```
// A function can return value of any type using the
// keyword "return".
// The same function can possibly return values
// of different types
function foo (p1) {
      if (typeof(p1) == "number")
         return 0;       // Return a number
      else if (typeof(p1) == "string")
               return "zero"; // Return a string

      // If no value being explicitly returned
      // "undefined" is returned.
}

foo(1);              // returns 0
foo("abc");          // returns "zero"
foo();               // returns undefined
```

JavaScript – onclick


```
onclick="javascript:function_name(parameters)"
```

- **onclick** is an event
  - The JavaScript code in the attribute value is executed when the user clicks on the button
  - Note the use of '…' in the JavaScript code so as not to clash with the "…" around the HTML attribute value

```
<input type="button" value="red"
       onclick="javascript:change_color('red')" />
<input type="button" value="green"
       onclick="javascript:change_color('green')" />
```

JavaScript – Example 1


```
<html>
   <head>
   <script type="text/javascript">
        function change_color(new_color)
        {
           document.body.style.backgroundColor = new_color;
        }
   </script>
   </head>
   <body>
    <form>
      <input type="button" value="red"
                onclick="javascript:change_color('red')” />
      <input type="button" value="green"
                onclick="javascript:change_color('green')” />
       <input type="button" value=“purple"
                onclick="javascript:change_color(‘purple’)” />

    </form>
   </body>
</html>
```

Lecture Overview

- Introduction
- JavaScript Basic
  - Variable
  - Types
  - Operators
  - Popup Box
  - Controls
  - Functions
  - DOM: Document Object Model
- Form Validation
- Comments
- Methods

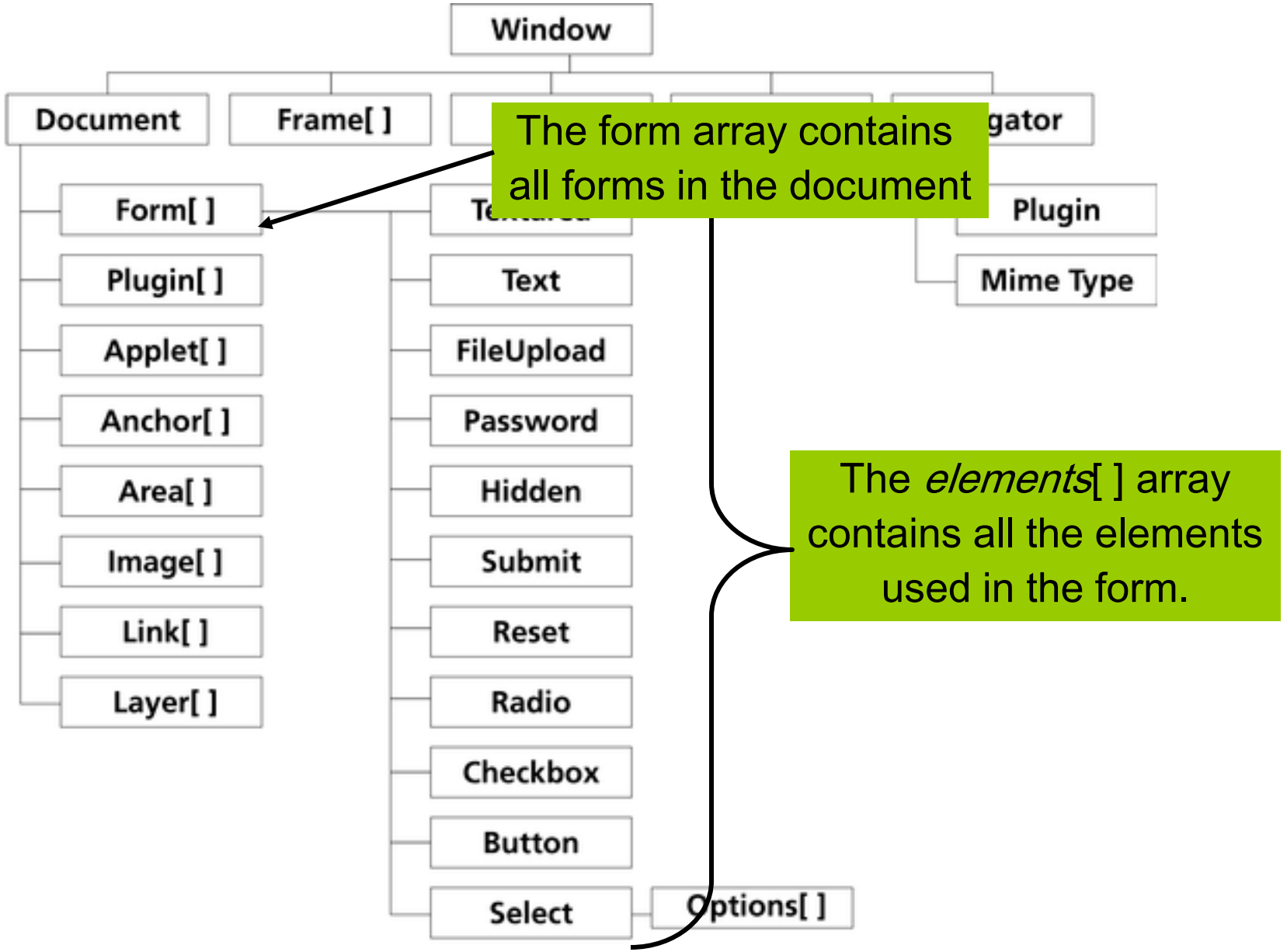


Figure 5-2: Browser object model

Document object model

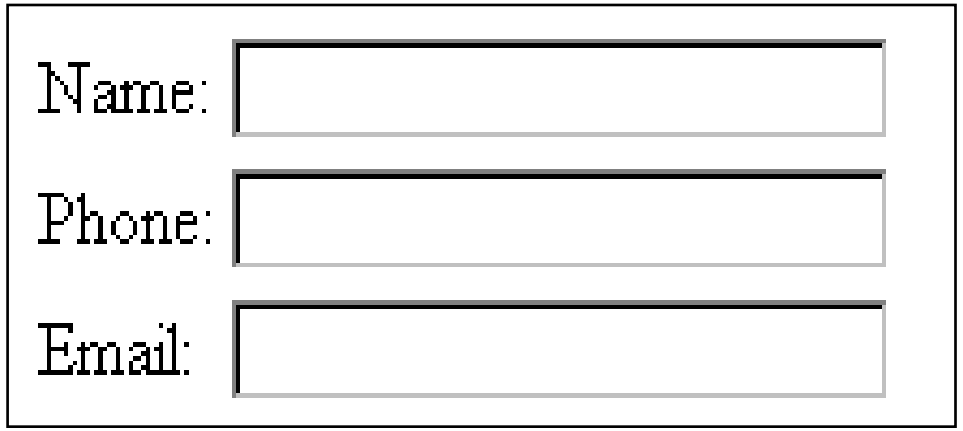
- The window object
  - The highest-level object
  - It is the default object
  - `window.document.write("a test message");`
  - We may omit writing `window` explicitly
    - `document.write("a test message");`
- The document object
  - A web document or a page in a browser window
- The Form object
  - The browser creates a unique "form" object for each form in a document
  - **`document.formname`** or **`document.forms[1]`**

Document object model

*document.**formname.elementname**.value*

Thus:

```
document.addressform.yourname.value
document.addressform.phone.value
document.addressform.email.value
```



Document object model

- OR
- The first text box has **`id=“yourname“`**– so it is accessed by **`document.getElementById(“yourname")`**
- Text boxes have properties – e.g.:
  - **`value`** (the string of text that has been typed)

⇒ **`document.getElementById(“yourname").value`** is the string that is typed in the text box

Lecture Overview

- Introduction
- JavaScript Basic
  - Variable
  - Types
  - Operators
  - Popup Box
  - Controls
  - Functions
  - DOM: Document Object Model
- Form Validation
- Comments
- Methods

Form validation

- Before an HTML form is submitted, JavaScript can be used to provide client-side data validation.
- If the form is not valid, the form is not submitted until the errors are fixed
- For example,
  - were required fields left empty?
  - was a valid e-mail address entered?
  - was a valid date entered?
  - was text entered in a numeric field?
  - were numbers entered in an text field?
  - did the number entered have a correct range?

Form validation

Server-side validation

1) The user submits the form to the Web server.

2) The Web server validates the user's responses and, if necessary, returns the form to the user for correction.

3) After correcting any errors, the user resubmits the form to the Web server for another validation.

Client-side validation

1) The user submits the form, and validation is performed on the user's computer.

2) After correcting any errors, the user submits the form to the Web server.

Form validation – text box

- Let's validate that the name field in our form is not blank.

```
<form name="easyform" ... >
   Please enter your name <input type="text" name="yourname">
   Please enter your birth year <input type="text" size="5" name="birthyear">
   ... other inputs here
</form>
```

- The contents of these two fields are

  document.easyform.yourname.value
  document.easyform.birthyear.value

Form validation – text box

- To validate the name field

```
If  (the name text box is blank) {
        Show an alert with a reminder message
        Form is not done yet
}  else  {
        Show a personalized thank you message
        Form is done
```

English	JavaScript
The name text box is blank	document.easyform.yourname.value == “”
Show an alert with a reminder message	alert(“argh! Please provide your name.”);
Show a personalized thank you message	Alert(“Thank you,” + document.easyform.yourname.value+”.”);
Form is done	Return true;
Form is not done yet	Return false;

Form validation – text b

checkName() to be
a boolean function that will
- Return true if the name field is filled out, and
- Return false if it is blank

- Javascript

```
<head>
<script type="text/JavaScript">
    function checkName()
    {
        if (document.easyform.yourname.value == "") {
            alert("argh! Please provide your name.");
            return false;
        } else {
            alert("Thank you, "+ document.easyform.yourname.value + ".");
            return true;
        }
} /* checkName() */
</script>
</head>
```

Form validation – text b


```
<input type="submit"
       value="Click Here to Submit"
       onClick="return checkName();"
>
```

- Javascript

```
<head>
<script type="text/JavaScript">
   function checkName()
   {
      if (document.easyform.yourname.value == "") {
         alert("argh! Please provide your name.");
         return false;
      } else {
         alert("Thank you, "+ document.easyform.yourname.value + ".");
         return true;
      }
} /* checkName() */
</script>
</head>
```

Form validation – text I


```
<form   name      =“easyform”
        method    =“post”
        action    =“erfrom.cgi”
        onSubmit =“return checkName();”>
```

- Javascript

```
<head>
<script type="text/JavaScript"
   function checkName()
   {
      if (document.easyform.yourname.value == "") {
         alert("argh! Please provide your name.");
         return false;
      } else {
         alert("Thank you, "+ document.easyform.yourname.value + ".");
         return true;
      }
} /* checkName() */
</script>
</head>
```

Form validation – text box

- To call checkName(), you need either

in the submit button's `onClick` handler, or

in the form's `onSubmit` handler.

Not in both!

Form validation - checkboxes

How do we distinguish which Is which?

- How are the checkboxes named?

```
Were you a ghost for halloween?
Yes <input  type="checkbox" name="ghost" value="YES" >
 No <input  type="checkbox"  name="ghost" value="NO" >
It's a secret <input  type="checkbox" name="ghost"  value="Secret" >
```

By counting them.

document.easyform.ghost[0] is the "yes" box,

document.easyform.ghost[1] is the "no" box, and

document.easyform.ghost[2] is the "it's a secret" box

Form validation - checkboxes

- The checked property is true when the box has a check in it, and it's false otherwise.

```
If (the first check box is not checked AND
    the second check box is not checked AND
    the third check box is not checked)   {
            Show alert message
            Not done yet
}
```

```
If (  !document.easyform.ghost[0].checked &&
      !document.easyform.ghost[1] .checked &&
      !document.easyform.ghost[2] .checked ) {
            altert("Please answer the ghost question.");
            return false;
            }
```

Form validation - checkboxes

- The checked property is true when the box has a check in it, and it's false otherwise.

```
If (the first check box is not checked AND
    the second check box is not checked AND
    the third check box is not checked)   {
            Show alert message
            Not done yet
}
```

```
If (  !document.easyform.ghost[0].checked &&
      !document.easyform.ghost[1] .checked &&
      !document.easyform.ghost[2] .checked ) {
            altert("Please answer the ghost question.");
            return false;
            }
```

Form validation - checkboxes

Note !

- the inputs are numbered starting with **0**
- Use the boolean operators above to create expressions

Form validation – radio buttons

How can we refer to radio button?

```
document.easyform.graduton[1]
```

```
if ( !document.easyform.graduation[0].checked &&
     !document.easyform.graduation[1].checked &&
     !document.easyform.graduation[2].checked &&
     !document.easyform.graduation[3].checked)
      {
            alert("Please select your graduation year.");
            return false;
      } else  {
            alert ("Thanks for telling me your graduation date.");
            return true;
      }
```

Form validation – pulldown menu

- selectedIndex: it's value is the number (starting, of caurse, from 0) of the menu item that was selected.

Document.easyform.cartoon.selectedIndex	Menu choice
0	Select one
1	Bugs Bunny
2	Mickey Mouse
3	Simpsons
4	Futurama

```
if  (document.easyform.cartoon.selectedIndex == 0) {
        alert("                                          ");
        return _________;
} else {
        alert(“                                          ");
        return _________;
}
```

Lecture Overview

- Introduction
- JavaScript Basic
  - Variable
  - Types
  - Operators
  - Popup Box
  - Controls
  - Functions
  - DOM: Document Object Model
- Form Validation
- Comments
- Methods

Comments


```
<script language="JavaScript">
<!--
Here you put your JS code.
Old browsers will treat it
as an HTML comment.
//-->
</script>
```

- ***// a comment line***
  - Everything to the en
- ***/* a comment block***
  - Everything inside th multiple lines
- ***<!-- a comment line***
  - The start of an HTML comment also acts as a JavaScript line comment
  - Placing **<!--** at the start of a script and **// -->** at the end hides the script from old browsers that don't understand it

Comments

- You MUST comment your JavaScript
  - Good coding practice
  - It makes it easier to understand
  - Useful if you want to reuse your code later
  - Useful if someone else needs to understand or reuse your code
  - Needed to get full marks in assignments

JavaScript – others

- **`location.href = url;`**
  - **`location`** is a different object to **`document`**
  - It has a property **`href`**
  - In this case, **`url`** (which is either **`"http://www.yahoo.com/"`** or **`"http://www.google.com/"`**) is assigned to **`href`**
  - This causes the browser to load this new location

```
<script type="text/javascript">
  <!-- This commenting hides the script from older browsers.
              function jump() {
                 var select = document.
                   getElementById("jumpto");
                 var index = select.selectedIndex;
                 var url = select.options[index].value;
                 location.href = url;
               } //-->
   </script>
</head>

<body>
  <form method="get" action="this_page">
    <p>
    <select id="jumpto" size="1">
                 <option value="http://www.yahoo.com/">
                  Yahoo</option>
                 <option value="http://www.google.com/">
                  Google</option>
     </select>
     <input type="button" value="Go!"
          onclick="javascript:jump()">
     </p>
   </form>
</body>
```

Lecture Overview

- Introduction
- JavaScript Basic
  - Variable
  - Types
  - Operators
  - Popup Box
  - Controls
  - Functions
  - DOM: Document Object Model
- Form Validation
- Comments
- Methods

Methods

- **Date Methods** - set variables to the clock time i
- **Window Methods** - used to open and close nev
- **Document Methods** - Generate new document
- **Form Methods** - select form items, send the cu submit forms.
- **History Methods** - Press the reade
- **Math Methods** - sin, cos, round, ra
- **MessageBox Methods** - Alert, Pro

```
var x=new Date();
x.setFullYear(2100,0,14);
var today = new Date();

if (x>today)
  {
  alert("Today is before 14th January 2100");
  }
else
  {
  alert("Today is after 14th January 2100");
  }
```

```
myWindow=window.open('','','width=200,height=100')
myWindow.document.write("<p>This is 'myWindow'</p>")
myWindow.focus()
```

```
document.write("<h1>Hello World!</h1><p>Have a nice d
```

```
var x = Math.PI; // Returns PI
var y = Math.sqrt(16); // Returns the square root of 16
```

```
<html>
<head>
<script>
function goBack()
  {
  window.history.back()
  }
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()">

</body>
```

References

- Web Development and Design Foundations with XHTML Terry Felke Morris, AW 2009, Chapter 14
- Doing Web Development: Client-Side Techniques by Deborah Kurata, APress, 2002 – Chapters 7-10
- David Flanagan: JavaScript: The Definite Guide. O'Reilly, 6th edition, 2011.
- Michael Bolin: Closure: The Definite Guide. O'Reilly, 2010.
- JavaScript Tutorials
  - http://www.webteacher.com/javascript/index.html
  - http://www.pageresource.com/jscript/index2.htm
  - http://javascript.internet.com/
  - http://www.w3schools.com/js/default.asp