COMP519 Web Programming

Lecture 16: JavaScript (Part 7)

Handouts

Ullrich Hustadt

Department of Computer Science
School of Electrical Engineering, Electronics, and Computer Science
University of Liverpool

Contents

Window and Document Objects

JavaScript provides two objects that are essential to the creation of dynamic web pages and interactive web applications:

window object

- a JavaScript object that represents a browser window or tab
- automatically created whith every instance of a `body` or `frameset` element
- allows properties of a window to be accessed and manipulated
- also allows new window objects to be created and manipulated
  Example: window.open('http://www.csc.liv.ac.uk','Home')
- whenever an object method or property is referenced in a script without an object name and dot prefix it is assumed by JavaScript to be a property of the window object

  Example: We can write `alert()` instead of `window.alert()`

Window Object

- A window object represents an open window in a browser.
- If a document contain frames, then there is
  - one window object, `window`, for the HTML document
  - and one additional window object for each frame, accessible via an array `window.frames`
- A window object has properties including

`document`	document object for the window
`history`	history object for the window
`location`	location object (current URL) for the window
`navigator`	navigator (web browser) object for the window
`opener`	reference to the window that created the window
`innerHeight`	inner height of a window's content area
`innerWidth`	inner width of a window's content area
`closed`	boolean value indicating whether the window is (still) open

Navigator Object

Properties of a navigator object include

`navigator.appName`	the web brower's name
`navigator.appVersion`	the web brower's version

Example: Load different style sheets depending on browser

```
<html><head><title>Navigator example</title>
<script>
if (navigator.appName == 'Netscape') {
  document.writeln('<link rel=stylesheet type="text/css" '+
                           href="Netscape.css">')
} else if (navigator.appName == 'Opera') {
  document.writeln('<link rel=stylesheet type="text/css" '+
                            href="Opera.css">')
} else {
  document.writeln('<link rel=stylesheet type="text/css" '+
                            href="Others.css">')
}
</script></head>
```

Window object

Methods provided by a window object include

- `open(`*`url`*`, `*`name`*` [, `*`features`*`])`
  - opens a new browser window/tab
  - returns a reference to a window object
  - *`url`* is the URL to access in the new window; can be the empty string
  - *`name`* is a name given to the window for later reference
  - *`features`* is a string that determines various window features

The standard sequence for the creation of a new windows is not:

```
// new instance of `Window' class
var newWin = new Window(...)
newWin.document.write('<html>...</html>')
```

instead it is

```
// new window created by using `open' with an existing one
var newWin = window.open(...)
newWin.document.write('<html>...</html>')
```

Window object

Methods provided by a window object include

- `close()`
  - closes a browser window/tab
- `focus()`
  - give focus to a window (bring the window to the front)
- `blur()`
  - removes focus from a window (moves the window behind others)
- `print()`
  - prints (sends to a printer) the contents of the current window

Window Object: Example

```
<html lang="en-GB"><head><title>Window handling</title>
<script>
function Help() {
  var OutputWindow = window.open('','Help','resizable=1');
  with (OutputWindow.document) {
    open()
    writeln("<!DOCTYPE html><html><head><title>Help</title>\
    </head><body>This might be a context-sensitive help\
    message, depending on the application and state of the\
    page.</body></html>");
    close()
  }
}
</script></head><body>
<form name="ButtonForm" id="ButtonForm" action="">
<p>
  <input type="button" value="Click for Help"
         onclick="Help();">
</p>
</form></body></html>
```

Window Object: Dialog Boxes

- Often we only want to open a new window in order to
  - display a message
  - ask for confirmation of an action
  - request an input

- For these purposes, the window object in JavaScript provides pre-defined methods for the handling of dialog boxes (windows for simple dialogs):
  - `null alert(message_string)`
  - `bool confirm(message_string)`
  - `string prompt(message_string, default)`

Window Object: Dialog Boxes

- `null alert(message_string)`
  - creates a dialog box displaying *message_string*
  - *message_string* is not interpreted as HTML markup
  - focus shifts away from the current window to the dialog box
  - the box contains an 'OK' button that the user will have to click (alternatively, the dialog box can be closed)
    for the execution of the remaining code to proceed

Example:

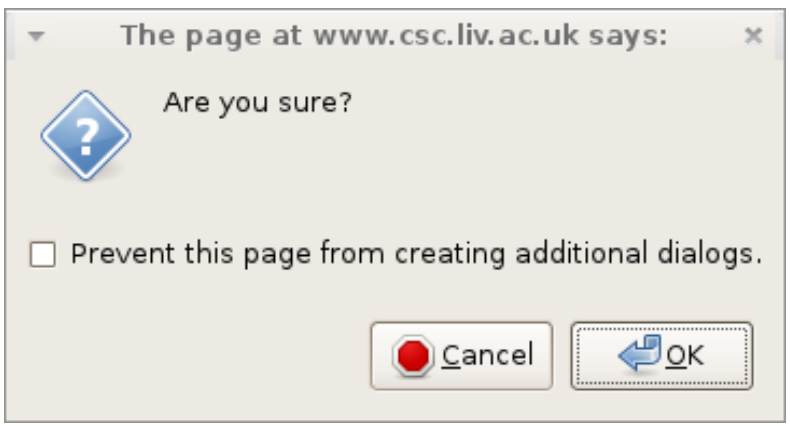
```
alert("Local time: " + (new Date).toString())
```

Window Object: Dialog Boxes

- `bool confirm(`*`message_string`*`)`
  - creates a dialog box displaying *`message_string`*
  - the box contains two buttons 'Cancel' and 'OK'
  - *`message_string`* is not interpreted as HTML markup
  - focus shifts away from the current window to the dialog box
  - the function returns `true` if the user selects 'OK', `false` otherwise

Example:

```
var answer = confirm("Are you sure?")
```

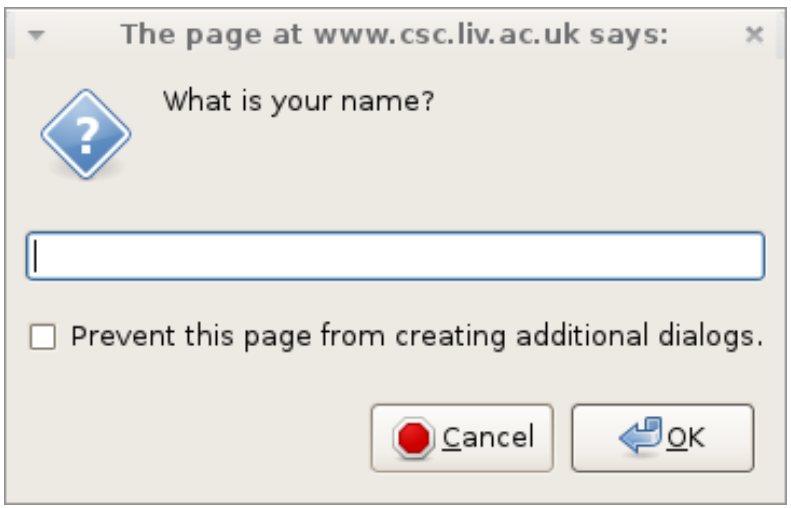


Window Object: Dialog boxes

- `string prompt(message_string, default)`
  - creates a dialog box displaying *message_string* and an input field
  - if a second argument *default* is given, *default* will be shown in the input field
  - the box contains two buttons 'Cancel' and 'OK'
  - if the user selects 'OK' then the current value entered in the input field is returned as a string, otherwise `null` is returned

Example:

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



Window Object: Dialog boxes

- `prompt()` always returns a string, even if the user enters a number
- To convert a string to number the following functions can be used:
  - `number parseInt(string [,base])`
    – converts *string* to an integer number wrt numeral system *base*
    – only converts up to the first invalid character in *string*
    – if the first non-whitespace character in *string* is not a digit, returns `NaN`
  - `number parseFloat(string)`
    – converts *string* to a floating-point number
    – only converts up to the first invalid character in *string*
    – if the first non-whitespace character in *string* is not a digit, returns `NaN`
  - `number Number(string)`
    – returns `NaN` if *string* contains an invalid character

Dialog Boxes: Example

```
<!DOCTYPE html>
<html lang="en-GB">
 <head><title>Interaction example</title></head>
<body>
<script>
do {
  string   = prompt("How many items do you want to buy?")
  quantity = parseInt(string)
} while (isNaN(quantity) || quantity <= 0)
do {
  string = prompt("How much does an item cost?")
  price  = parseFloat(string)
} while (isNaN(price) || price <= 0)
buy = confirm("You will have to pay "+
              (price*quantity).toFixed(2)+
              "\nDo you want to proceed?")
if (buy) alert("Purchase made")
</script>
</body></html>
```

http://cgi.csc.liv.ac.uk/~ullrich/COMP519/examples/jsPrompt.html

User Input Validation

- A common use of JavaScript is the validation of user input in an HTML form before it is processed:
  - check that required fields have not been left empty
  - check that fields only contain allowed characters or comply to a certain grammar
  - check that values are within allowed bounds

```
<form method="post" action="process.php"
      onSubmit="return validate(this)">
  <label>User name:     <input type="text" name="user"></label>
  <label>Email address: <input type="text" name="email"></label>
  <input type="submit" name="submit" />
</form>
<script>
function validate(form) {
  fail  = validateUser(form.user.value)
  fail += validateEmail(form.email.value)
  if (fail == "") return true
  else { alert(fail); return false } }
</script>
```

User Input Validation

```
function validateUser(field) {
  if (field == "") return "No username entered\n"
  else if (field.length < 5)
    return "Username too short\n"
  else if (/[^a-zA-Z0-9_-]/.test(field))
    return "Invalid character in username\n"
  else return ""
}

function validateEmail(field) {
  if (field == "") return "No email entered\n"
  else if (!((field.indexOf(".") > 0) &&
             (field.indexOf("@") > 0)) ||
           /[^a-zA-Z0-9\.\@\_\-]/.test(field))
    return "Invalid character in email\n"
  else return ""
}
```

http://cgi.csc.liv.ac.uk/~ullrich/COMP519/examples/jsValidate.html

Revision and Further Reading

- Read
  - Chapter 21: Introduction to JavaScript: The Browser Object
  - Chapter 22: Using JavaScript: Meet the DOM

  of J. Niederst Robbins: Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics (5th ed). O'Reilly, 2018.
  E-book `https://library.liv.ac.uk/record=b5647021`

- Read
  - Chapter 8: The Browser Object Model
  - Chapter 9: Client Detection

  of N. C. Zakas: Professional JavaScript for Web developers. Wrox Press, 2009.
  Harold Cohen Library 518.59.Z21 or
  E-book `http://library.liv.ac.uk/record=b2238913`