8. Programming with JavaScript

Topic 8:
Procedural programming with JavaScript

1. Event-driven software
2. Elements of JavaScript
3. Branches and loops in JavaScript

Reading: Ch. 10 (pp. 500-519); handouts

Overview

- *JavaScript*: A procedural language
- The JavaScript text between the HTML tags `<script language = "Javascript">` and `</script>` will execute when browser displays HTML file
- *Motivation*:
  - Working with IT means thinking abstractly and concretely about data and operations
  - Design, coding, and testing of solutions are part of learning problem solving
Objectives

8a. Explain what a procedural language is
8b. Write simple code in a procedural language to implement a branch and a loop
8c. Explain event-driven programming
8d. Explain the concepts of debugging and infinite loops
8e. Create a simple event-driven web page
8f. Distinguish syntax errors from logic errors

Generations of programming languages

- First generation: Machine languages (binary)
- 2nd generation: Assembler languages, processor specific
- 3rd generation: Procedural, high-level, hardware independent (C, BASIC, JavaScript)
- 4th generation: Nonprocedural query or report-generation languages (SQL, RPG)
- Declarative languages (Prolog)
- Object-oriented languages (C++, Java)
- Functional languages (Lisp)
1. Event-driven software

- *Browsers* and most other apps are *interactive*, alternating input and output
- *Command-line environments*: URL line in browser, Google prompt, DOS or UNIX prompt
- *Features of graphical user interfaces*: windows, icons, menus, dialog boxes, buttons
- *Common feature*: User generates *events*, e.g., clicks, drags, keystrokes, timeouts
- Forms of interaction in browser: hyperlinks; embedding of event-based JavaScript programs in HTML files

Problem specifications and user interfaces

- Designer must consider *assumptions* about
  - Problem domain (e.g., business, education, personal, healthcare)
  - User needs and expectations
- *Interface* refers to how application (e.g., at web site) appears and responds to user
- Most user interfaces today are *graphical*
- Implementation (coding) is partly independent of interface
Simple JavaScript example

```html
<html> <!-- hello.htm -->
<head><title>DK-Hello</title></head>
<body>63.120 says Hello!
    <script language="JavaScript">
        alert("hello");
    </script>
</body>
</html>
```

- Displays “hello” in an alert box (a kind of dialog)
- `alert` is a JavaScript function (a kind of procedure)
- JavaScript may be used after `script` tag

Simple button

```html
<html> <!--button.htm-->
<head><title>63120 Hello</title></head>
<body>
    <input type=button value = "Hello" onClick = 'alert("Hi")'>
</body></html>
```

- This code displays “Hi” when “Hello” button is pressed
- `<input>` tag defines an input button object
- `Event handler`: code that specifies application’s response to a particular event, such as user click on a button
Counting button clicks

```html
<html>  
<head><title>yes-no counter</title></head>  
<body>  
/* count-yes.htm Displays Yes, No buttons for user to click, counts # clicks on each. Four 'event-handlers' specify the program's response to four different input events: Yes, No, Stats, Reset. When user presses 'Yes' button, variable 'num_yes' is incremented. */
<script language="JavaScript">  
var num_yes=0, num_no=0; // Variables
</script>  
<td><input type=button value = "Yes" onClick = 'num_yes = num_yes + 1'></td>
<td><input type=button value = "No" onClick = 'num_no = num_no + 1'></td>
<td><input type=button value = "Stats" onClick = 'alert("Yes: "+num_yes + " No: "+num_no)'></td>
<td><input type=button value = "Reset" onClick = 'num_yes = num_no = 0'></td>
</body>  
</html>
```

Text input/output

```html
<head><title>Input echo</title></head>  
<body>  
<form name="Input"><table>  
<!-- Display prompt and get input:-->  
<td>Enter your user name:  
</td>  
<!-- Generate input-box:-->  
<input type=text name=user_name value="" size = 15>  
<td>  
<!-- Wait for button-press, display message:-->  
<input type=button value="Done" onClick = 'alert("Hello " + user_name.value)'>  
</td>  
<!-- Assigning a value to onClick defines JavaScript response to clicking button-->  
</table></form>  
</body>
```
2. Elements of JavaScript

- Statements
- Variables
- Expressions

Statements

- *Statements* are *executable* (or are variable declarations)
- Statements include
  - *assignment* `a = 4;`
  - compound statements `{ in braces }
  - *if, if ... else*
  - *while*
  - *Simple* statements, e.g., assignment, terminate with “;”
JavaScript variables

- Variables have name, type, value
- Must declare a variable to use it
  ```javascript
  var a;
  ```
- Valid types (interpretation of bits):
  - Number: ```javascript
      var a=2, b=3;
  ```
  - String: ```javascript
      var name = "Bill";
  ```
  - Truth value: ```javascript
      var greater = (a > b);
  ```
- Assignment can change variable’s value:
  ```javascript
  a = b * 4;
  ```

Expressions

- Expressions have values and are used in statements
- Expressions are literals, variables, or function calls, or may be formed with arithmetic operators (+, -, *, /, %)
- Boolean expressions have truth values and may be built with relational operators (==, !=, <, >, <=, >=) or logical operators (!, &&, ||)
3. Branches and loops in JavaScript

- **Branch:**
  
  ```javascript
  if (a > b) {
    max = a;
  } else {
    max = b;
  }
  ```

  See example code from D. Keil, C. Breuning

- **Loop:**
  
  ```javascript
  i = 1;
  while (i < 10) {
    sum = sum + i;
    i = i + 1;
  }
  ```

Testing and debugging

- Software and web sites require testing before deployment
- Testing is often done by quality assurance departments
- All software writing entails error and **debugging**
- JavaScript is easy to test on a browser, but the browser does not supply error locations or other diagnostics
Example with branch control structure

```
<html>  
<head><title>Yes-no with memory</title></head>  
<body>  
<script language="JavaScript">  
/* remember.htm  Displays 2 buttons for user to click,  
tells which was clicked most recently */  
var most_recent="A", message="Last you clicked was ";  
</script>  
<td><input type=button value = "A"  
onClick = 'most_recent="A"'></td>  
<td><input type=button value = "B"  
onClick = 'most_recent="B"'></td>  
<td><input type=button value = "Check"  
onClick = 'if (most_recent == "A") alert(message+"A");  
else alert(message+"B")'>  
</td>  
</body>  
</html>  
[remember.htm]
```

Example with loop

```
<html>  
<! power.htm>  
<head><title>Input exponentiator</title></head>  
<body>  
<script language="JavaScript">  
var i,y;  
</script>  
<form name="Input">  
<table>  
<! Display prompt and get input:>  
<td>Enter a  base:       <! Generate input-box:>  
<input type=text name=x1 value="0" size = 8>  
</td>  
<td>Enter an exponent:  
<input type=text name=x2 value="0" size = 8>  
</td>  
<td><input type=button value="Done"  
onClick = '  
i = parseInt(x2.value);  
y = 1;  
while(i > 0) {  
y = y * parseInt(x1.value);  
i = i - 1;  
}  
alert(x1.value + "^" + x2.value +  
" = " + y);  
'>  
</td>  
</table></form>  
</body>  
</html>
```
8. Programming with JavaScript

References


