JavaScript Basics

Banu Prakash

Objectives

- List usage of Java script
- Understand basic program constructs in Java script
- Understand how to handle user events in Java script
- Understand how to build dynamic HTML contents using Java script

JavaScript

- JavaScript is the scripting language of the Web.
- JavaScript is a lightweight, interpreted programming language.
- JavaScript are embedded in the web pages.
- It can enhance the dynamics and interactive features of your page by allowing you to perform calculations, check forms, write interactive games, add special effects, customize graphics selections, create security passwords and more.
- All modern HTML pages are using JavaScript

JavaScript

- The merits of using JavaScript are:
    - **Less server interaction.**
    - **Make web pages more interactive**
    - **Build Rich User interfaces**
    - **Immediate feedback to the visitors.**

JavaScript syntax

- A JavaScript consists of statements that are placed within the <script>... </script> HTML tags in a web page.
- You can place the <script> tag containing your JavaScript anywhere within you web page but it is preferred way to keep it within the <head> tags.

JavaScript syntax

The script tags says that anything within that tags is a scripting language.

The Browser knows that it should interpret this as scripting language.

```
<html>
   <head>
    <title>Welcome Page</title>
    <script type="text/JavaScript">

     </script>
   </head>
   <body>
                Body content here
   </body>
</html>
```

JavaScript data types

- JavaScript allows you to work with three primitive data types:
  - Numbers. [123, 120.50 etc.]
  - Strings [“BanuPrakash“, “Bangalore”, etc.]
  - Boolean [true or false.]
  - JavaScript also defines two trivial data types:
    - *null* and *undefined*, each of which defines only a single value.
  - JavaScript supports a composite data type known as *object*.

JavaScript variables

- Variables can be thought of as named containers.
- You can place data into these containers and then refer to the data simply by naming the container

In JavaScript variables are created using “var” keyword

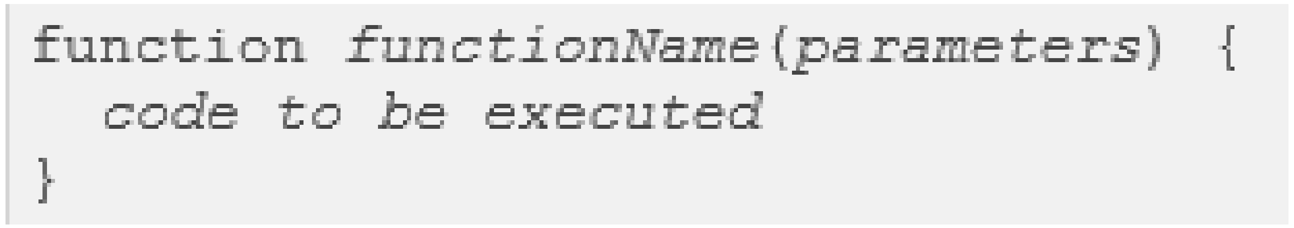
```
<html>
   <head>
    <title>Welcome Page</title>
    <script type="text/javascript">
      var name = “Smith”;
      var age = 24;
      var married = false;
     </script>
   </head>
   <body>
                 Body content here
   </body>
</html>
```

JavaScript Comments

- JavaScript supports both C-style and C++-style comments, Thus:
- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

JavaScript functions

- A JavaScript function is a block of code that may be executed when invoked.
- Syntax:

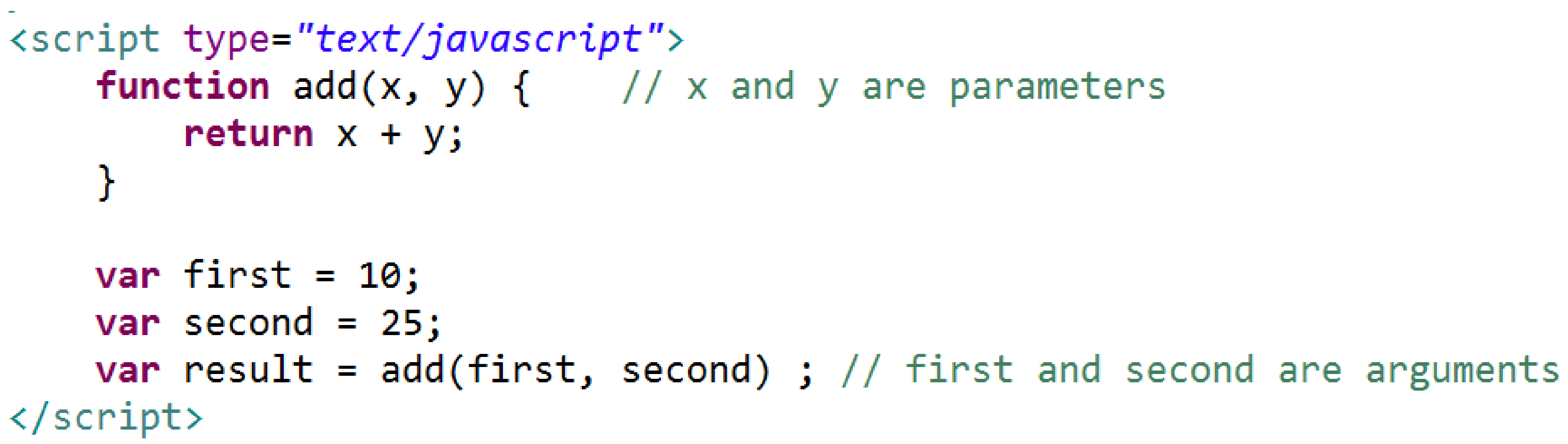


```
function functionName(parameters) {
  code to be executed
}
```

- A JavaScript function is defined with the **function** keyword, followed by a ***functionName***, and single brackets: **()**
- The single brackets may include a list of parameter names: **(*parameter1, parameter2, ...*)**
- The code to be executed by the function is placed inside curly brackets: **{ }**

JavaScript functions

- **Function Parameters and Arguments**
  - Identifiers used, in the function definition, are called **parameters**
  - Values used, for the parameters, when the function is invoked, are called **arguments**



```
<script type="text/javascript">
    function add(x, y) {    // x and y are parameters
        return x + y;
    }

    var first = 10;
    var second = 25;
    var result = add(first, second) ; // first and second are arguments
</script>
```

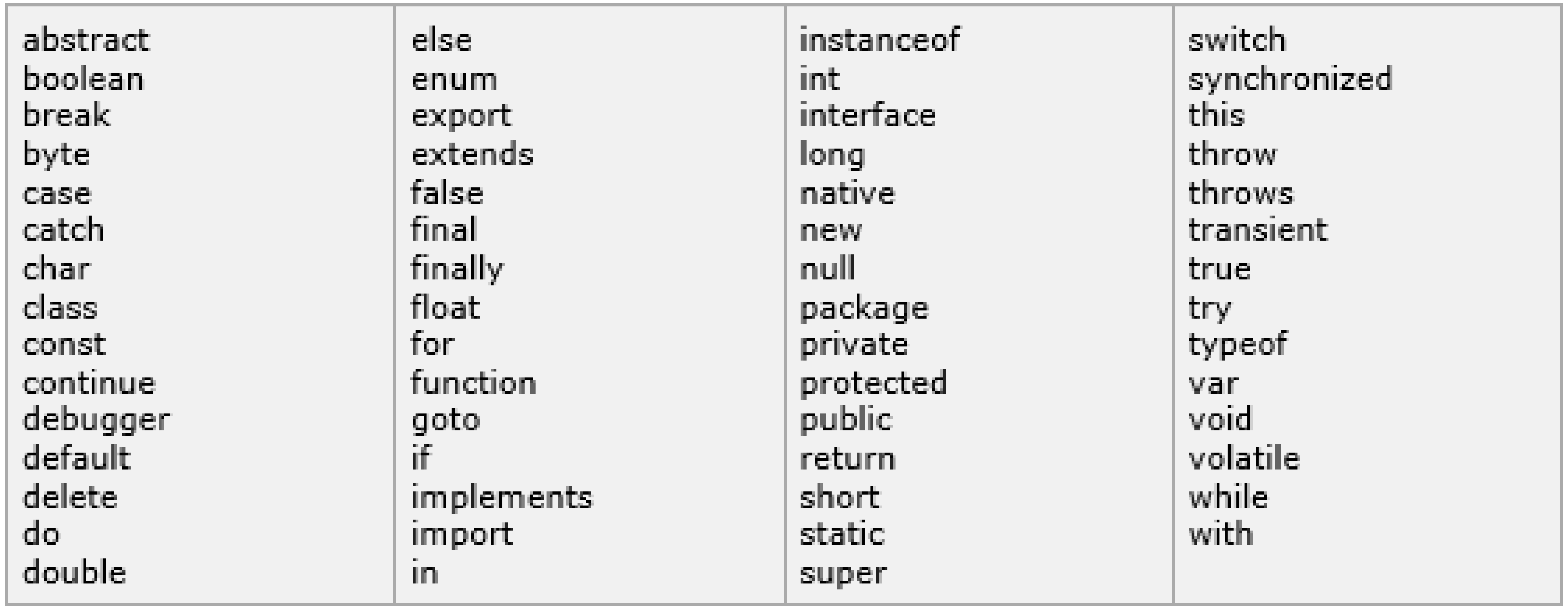
JavaScript variable scope

- The scope of a variable is the region of your program in which it is defined. JavaScript variable will have only two scopes.
  - **Global Variables:** A global variable has global scope which means it is defined everywhere in your JavaScript code.
  - **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

```
<script type="text/javascript">
    var myVar = "global"; // Declare a global variable
    function checkscope() {
        var myVar = "local"; // Declare a local variable
        document.write(myVar); //prints local
    }
</script>
```

JavaScript keywords

- The following are reserved words in JavaScript. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.



abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

JavaScript operators

- Operators denotes a specific operation.
- JavaScript language supports following type of operators.
  - Arithmetic Operators
  - Comparison Operators
  - Logical (or Relational) Operators
  - Assignment Operators
  - Conditional (or ternary) Operators

JavaScript operators

- Arithmetic operators

Operator	Description
+	Adds two operands
-	Subtracts second operand from the first
*	Multiply both operands
/	Divide numerator by denominator
%	Modulus Operator and remainder of after an integer division
++	Increment operator, increases integer value by one
--	Decrement operator, decreases integer value by one

JavaScript operators

- **The Comparison Operators**

Operator	Description
==	Checks if the value of two operands are equal or not.
!=	Checks if the value of two operands are equal or not.
>	Checks if the value of left operand is greater than the value of right operand
<	Checks if the value of left operand is less than the value of right operand
>=	Checks if the value of left operand is greater than or equal to the value of right operand
<=	Checks if the value of left operand is less than or equal to the value of right operand

JavaScript operators

- **The Logical Operators**

Operator	Description
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.
||	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.

JavaScript operators

- **The Assignment Operators**

  = Simple assignment operator, Assigns values from right side operands to left side operand

  += Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand

  -= Subtract AND assignment operator

  ***=** Multiply AND assignment operator

  /= Divide AND assignment operator

  %= Modulus AND assignment operator

JavaScript operators

- **The Conditional Operator (? :)**
  - If Condition is true ? Then value X : Otherwise value Y
  - Example:

```
var a = 10;
var b = 20;
var result = (a > b) ? 100 : 200;
```

JavaScript conditional statements

- In JavaScript we have the following conditional statements:
  - **if statement** - use this statement to execute some code only if a specified condition is true
  - **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false
  - **if...else if....else statement** - use this statement to select one of many blocks of code to be executed
  - **switch statement** - use this statement to select one of many blocks of code to be executed

JavaScript conditional statements

- The if statement: Use the if statement to execute some code only if a specified condition is true
  - Syntax:

```
if(condition) {
    code to be executed if condition is true
}
```

  - Example:

```
var age = 20;
if( age > 18 ){
   document.write("eligible to vote");
}
```

JavaScript conditional statements

- **If...else Statement:** Use the if....else statement to execute some code if a condition is true and another code if the condition is not true
  - Syntax:

```
if( condition ){
    code to be executed if condition is true
} else {
    code to be executed if condition is not true
}
```

  - Example

```
var age = 20;
if( age > 18 ){
   document.write("eligible to vote");
} else {
     document.write("not eligible to vote");
}
```

JavaScript conditional statements

- **If...else if...else Statement:** Use the if....else if...else statement to select one of several blocks of code to be executed
- **The switch statement**
  - Use the switch statement to select one of many blocks of code to be executed

```
switch(n) {
 case 1:
  execute code block 1
   break;
 case 2:
  execute code block 2
   break;
 default:
  code to be executed if n is different from case 1 and 2
 }
```

JavaScript conditional statements

- Switch statement example

```
var day = new Date().getDay();
switch (day) {
case 0:
    x = "Today is Sunday"; break;
case 1:
    x = "Today is Monday"; break;
case 2:
    x = "Today is Tuesday"; break;
case 3:
    x = "Today is Wednesday"; break;
case 4:
    x = "Today is Thursday"; break;
case 5:
    x = "Today is Friday"; break;
case 6:
    x = "Today is Saturday"; break;
}
```

JavaScript loop statements

- The while loop
  - Syntax

```
while (expression){
        code to be executed if expression is true
}
```

  - Example:
    - Prints multiplication table of number “5”

```
var number = 5;
var count = 1
while ( count <=10 ){
       document.write( number + " * " + count + " = " + (number * count));
       document.write("<br />");
       count++;
}
```

JavaScript loop statements

- **The *do...while* Loop**
  - The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is *false.*
  - *Syntax:*

```
do {
   code to be executed;
} while (expression);
```

JavaScript loop statements

- **The *for* Loop**
  - The **for** loop includes the following three parts:
    - The loop initialization where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
    - The test statement which will test if the given condition is true or not. If condition is true then code given inside the loop will be executed otherwise loop will come out.
    - The iteration statement where you can increase or decrease your counter.
  - Syntax:

```
for (initialization; test condition; iteration statement){
     code to be executed if test condition is true
}
```

JavaScript loop control statements

- **The *break* Statement:**
  - The **break** statement, which was briefly introduced with the *switch* statement, is used to exit a loop early, breaking out of the enclosing curly braces
- **The *continue* Statement:**
  - The **continue** statement tells the interpreter to immediately start the next iteration of the loop and skip remaining code block.

```
var count = 0;
document.write("Print Even numbers<br /> ");
while (count <= 10) {
    count = count + 1;
    if (count % 2 != 0) {
        continue; // skip rest of the loop body
    }
    document.write(count + "<br />");
}
```

Dynamic HTML using JavaScript

Banu Prakash

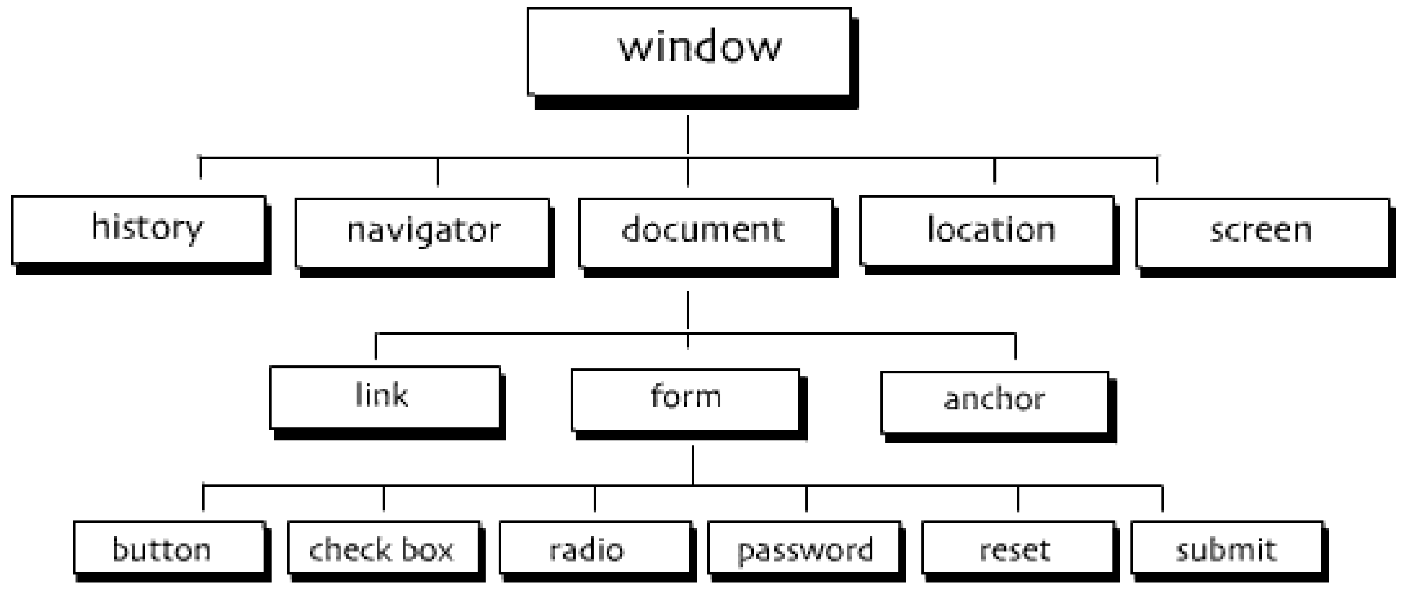
Objectives

- Understand HTML DOM
- Understand how to handle user events in Java script
- Understand how to build dynamic HTML contents using Java script

DOM [Document Object Model]

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page

Tree Structure of the DOM



JavaScript and DOM

- With the object model, JavaScript gets all the power it needs to create dynamic HTML:
  - JavaScript can change all the HTML elements in the page
  - JavaScript can change all the HTML attributes in the page
  - JavaScript can change all the CSS styles in the page
  - JavaScript can remove existing HTML elements and attributes
  - JavaScript can add new HTML elements and attributes
  - JavaScript can react to all existing HTML events in the page
  - JavaScript can create new HTML events in the page

DOM Programming interface

- **Objects**: In the DOM, all HTML elements are defined as **objects**.
  - Examples of browser objects
    - Window
    - Navigator
    - Location
- **Property: property** is a value that you can get or set (like changing the content of an HTML element).
  - Examples of window object properties
    - status : Sets or returns the text in the status bar of a window
- **Method**: A **method** is an action you can do (like add or deleting an HTML element).
  - Example of methods of window object
    - alert( ) : Displays an alert box with a message and an OK button

Example using DOM interface

- **The getElementById Method:** The most common way to access an HTML element is to use the id of the element.
- **The innerHTML Property:** To get the content of an element

```
<div id="txtDiv">JavaScript is fun!!!</div>
<script type="text/javascript">
     var txt = document.getElementById("txtDiv").innerHTML;
     window.alert(txt);
</script>
```

- To change the content of an HTML element, use this syntax:
  - document.getElementById(*id*).innerHTML= *new HTML*

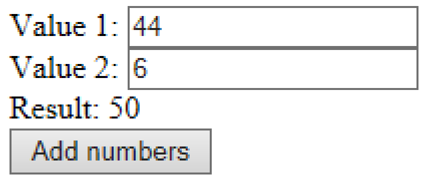
JavaScript events

Event attribute	Description	Applies to
onSubmit	User Submits a form	Submit button
onClick	User clicks a form element	Buttons, Links, Submit button, radio and checkboxes
onChange	User changes value of element	select, text, text area
onLoad	User loads the page in browser	Document body
onFocus	User gives input focus to window or form element	Windows and all form elements.
onBlur	User removes input focus from window or form element	Windows and all form elements.
onUnload	User exists the page	Document body

JavaScript events

- Example of using onclick event and accessing data using “id” attribute.

```
<title>JavaScript examples</title>
    <script type="text/javascript">
        function addNumbers() {
            var x = document.getElementById("t1").value;
            var y = document.getElementById("t2").value;
            var sum = parseInt(x) + parseInt(y);
            document.getElementById("resultSpan").innerHTML = sum;
        }
    </script>
</head>
<body>
     <form>
        Value 1: <input type="text" name="v1" id="t1" /> <br />
        Value 2: <input type="text" name="v2" id="t2" /> <br />
        Result: <span id="resultSpan"></span> <br />
        <input type="button" value="Add numbers" onclick="addNumbers()" />
     </form>
</body>
```



Value 1: 44

Value 2: 6

Result: 50

The name attribute uniquely identifies a field within a form

The id attribute uniquely identifies element in a page

JavaScript events

- Example of using onclick event and accessing data using “name” attribute.

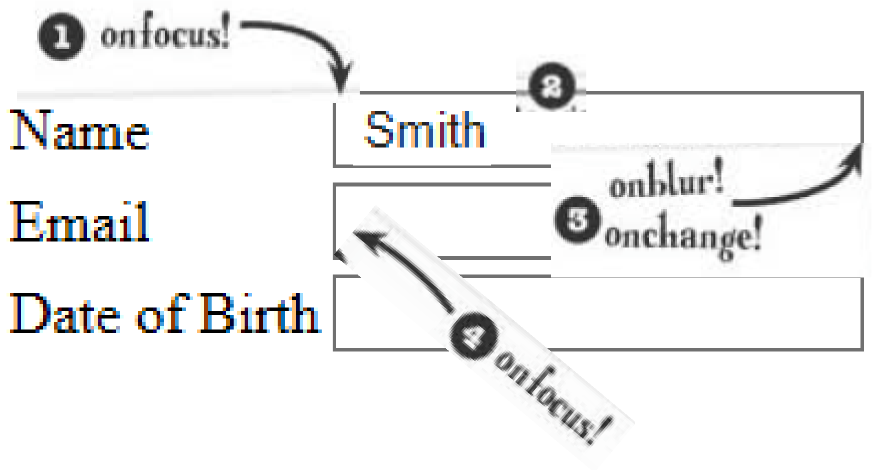
```
    <script type="text/javascript">
        function addNumbers() {
            var x = document.forms[0].v1.value;
            var y = document.forms[0].v2.value;
            var sum = parseInt(x) + parseInt(y);
            document.getElementById("resultSpan").innerHTML = sum;
        }
    </script>
</head>
<body>
     <form>
        Value 1: <input type="text" name="v1" id="t1" /> <br />
        Value 2: <input type="text" name="v2" id="t2" /> <br />
        Result: <span id="resultSpan"></span> <br />
        <input type="button" value="Add numbers" onclick="addNumbers()" />
     </form>
</body>
```

The name attribute uniquely identifies a field within a form

The id attribute uniquely identifies element in a page

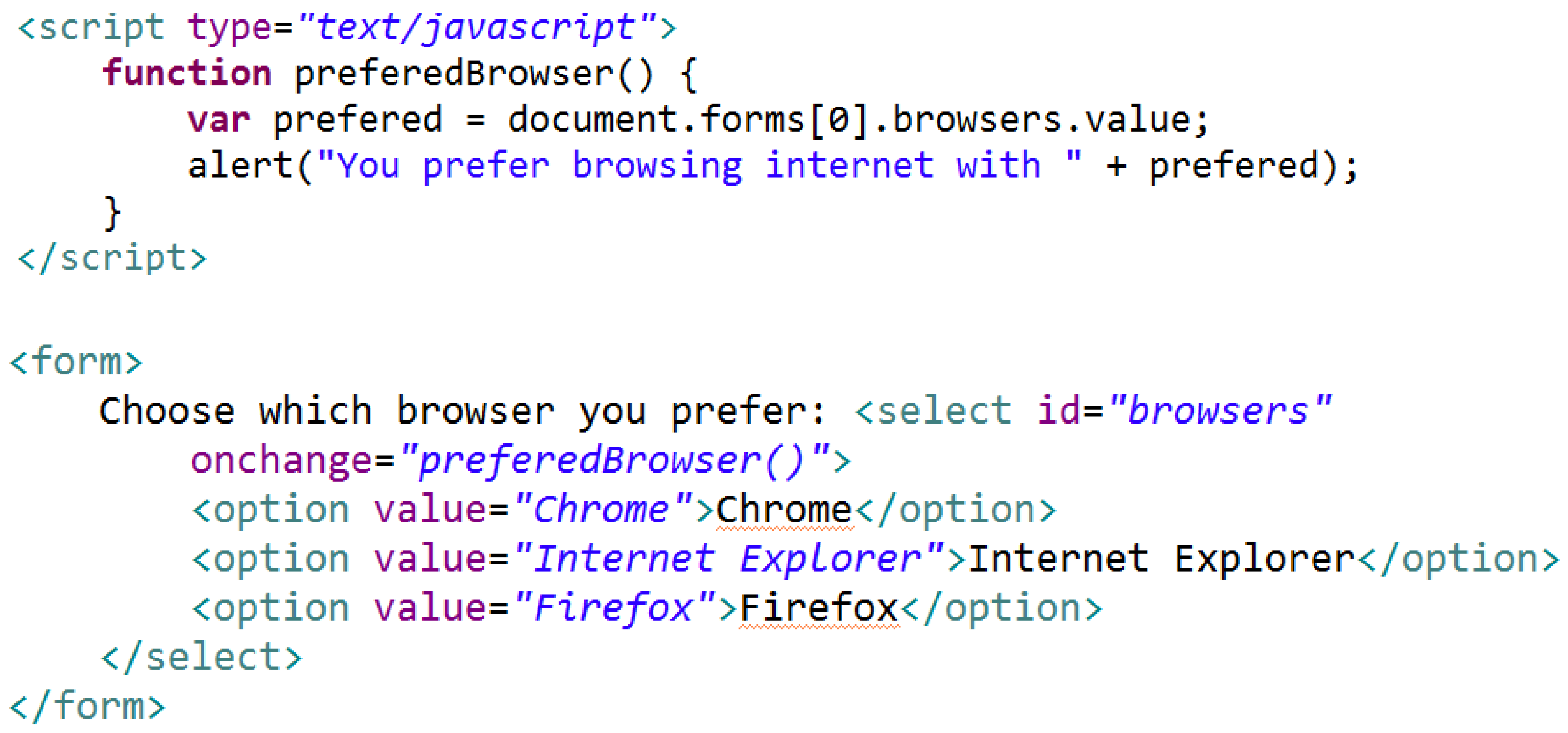
JavaScript events

- Form filed contains a chain of events
  1. Select the input field (onfocus)
  2. Enter data into the field
  3. Leave the input field to move to next one (onblur/onchange)
  4. Select the next input field (onfocus)



JavaScript events

onchange event example: When a user selects a dropdown value



```
<script type="text/javascript">
    function preferedBrowser() {
        var prefered = document.forms[0].browsers.value;
        alert("You prefer browsing internet with " + prefered);
    }
</script>

<form>
    Choose which browser you prefer: <select id="browsers"
        onchange="preferedBrowser()">
        <option value="Chrome">Chrome</option>
        <option value="Internet Explorer">Internet Explorer</option>
        <option value="Firefox">Firefox</option>
    </select>
</form>
```

JavaScript events: form validation

```
<div id="errDiv"> </div>
<form method="POST" action="Register"
        onsubmit="return validate(this);">
        First Name<input type="text" name="firstName" />
        <input type="submit" value="Register" />
</form>
```

1. Pass the form as argument on submit to function validate

```
function validate(frm) {
        var msg = "";
        var valid = true;
        var firstName = frm.firstName.value;
        if( firstName == "") {
                msg += "<br /> First Name is required";
                valid = false;
        }
        if( !valid ) {
                document.getElementById("errDiv").innerHTML = msg;
        }
        return valid;
}
```

2. Read the value of firstName form field

3. If firstName is not entered place error message within tag with id “errDiv”.

4. If value true is returned the form is submitted else form is not submitted.