SOEN 287: WEB PROGRAMMING

Chapter 4
Basics of JavaScript (Part 1)

Javascript

We will cover in this first part:

- **History & uses**
- Syntactic characteristics
- Primitives, operations & Expressions
- I/O
- Arrays
- Control statements

Quick History

- Created in 10 days in 1995 by Brendan Eich while he worked at Netscape
- Names: Mocha → LiveScript → JavaScript (when received a trademark license from Sun – joint venture Sun and Microsoft)
- 1996-97 was send to ECMA (European Computer Manufacturers Association) to establish a standard which other web browsers could use → official release of ECMA-262:ECMAScript
- *JavaScript* is one implementation of this standard and most common name
- *Jscript* is Microsoft's version

JavaScript …

- … and Java are only related through syntax
- … and Java have similar programming concepts
- … is **not** an object-oriented programming language but an object-based language.
- … is dynamically typed
- … is **not** a subset or version of Java
- … is a scripting language meaning it adds functionality to a web page.

Client vs. Server side

http://blogs.nuxeo.com/industry-insight/2011/11/requirements-and-challenges-for-a-modern-content-platform-3-supporting-more-than-just-the-server-side/

- JS has three parts:
  1. The **core** of the language
  2. **Client-side** supports control of browser and interaction with user
  3. **Server-side** to control interaction with Web server (Ex.database)

- Client side programming with JavaScript much more popular

- In this chapter will cover **core** components of JavaScript

Some JavaScript uses on client side

- Monitor user events & specify reactions
- Make computations based on user input and display results
- Change style and position of displayed elements
- Pop up new windows or menus
- Detect browser type, version, and features
- Modify/transform page content
- Validating user input
- Perform and control CSS transitions and animations
- Handling dates and time
- ……

NOTE: Most actions are event driven

Placement of JavaScript code

1. JavaScript code referred to as a ***script***
2. Scripts can be *explicitly* or *implicitly* imbedded in HTML document
3. Explicit imbedding in the HTML code not always ideal
   - Can be in the page's `<head>` element
     - if is a script that reacts to user action
     - Or only when requested (functions)
   - Can be in the page's `<body>` element
     - When script that is interpreted only once (when interpreter finds it)

Placement of JavaScript code …

4. Implicit imbedding in a separate file (.js)

   - Hides the Script(s) from browser

   - Use when JavaScript code is meant for more than one page

Javascript

We will cover in this first part:

- History & uses
- **Syntactic characteristics**
- Primitives, operations & Expressions
- I/O
- Arrays
- Control statements

JavaScript: General Syntax

- Import a JavaScript file

```
 <script type = "text/javaScript"
         src = "myScript.js">
</script>
```

- Embed JavaScript code

```
 <script type = "text/javaScript">
  <!--
        JavaScript script
    //-->
</script>
```

- JavaScript comments: both `//` and `/* … */`

Hello World: Example of JS in body

```
<!DOCTYPE html>
<html lang = "en">
  <head>
    <meta charset="utf-8">
    <title> Hello world </title>
  </head>
  <body>
    <script type = "text/javascript">
    <!--
      document.write("Hello, World!");
    // -->
    </script>
  </body>
</html>
```

helloWorld.html

Javascript

We will cover in this first part:

- History & uses
- Syntactic characteristics
- **Primitives, operations & expressions**
- I/O
- Arrays
- Control statements

Primitive Data Types

- **Numbers**:
  - Example of numbers:

```
123, 1.23, -123, 1E2, 1e2, 1.23E-2
```

- **String**:
  - Can be between ' or "
  - Example of strings:

```
"Tuesday"    'Tuesday\n'
'Sam\'s work'  "C:\\root "    ""  ''
```

- **Boolean:**
  - values are `true` and `false`

Special Data Types

null:

- The `null` data type has only one value in JavaScript: `null`.
- The `null` keyword cannot be used as the name of a function or variable.
- A variable is `null` when not declared or not explicitly assigned a value.
- You can erase the contents of a variable (without deleting the variable) by assigning it the `null` value.

undefined:

- The undefined data type has only one value: `undefined`
- A variable is `undefined`, when declared but not assigned a value

Declaring variables

- Can be *explicitly* or *implicitly* declared

- *Explicit* declaration:

```
var num1, num2 = 10;
```

- *Implicit* declaration

```
num3 = 10;
```

Just checking:
Which of these variables has the value `undefined`?

Numeric Operations

- Numeric operators: `+, -, *, /, %`
- Shortcut Operators: `++, --, +=, %=,` …
- The **`Math`** object provides methods:

`Math.max(x,y,z,...,n)`	returns largest value
`Math.ceil(x)`	returns x, rounded upwards to the nearest integer
trig functions	e.g., `Math.cos(x)`

For more methods see
http://www.w3schools.com/jsref/jsref_obj_math.asp

The `Number` Object

- Wrapper for primitive numeric values
- To create one, call constructor: `var n = new Number();`

❖ But with JavaScript, methods and properties are also available to primitive values, because JavaScript treats primitive values as objects when executing methods and properties.

- Number Properties: `MAX_VALUE`, `MIN_VALUE`, `NaN`, …
  - e.g., `Number. MAX_VALUE`
- An arithmetic operation that creates overflow returns `NaN` (**N**ot **a** **N**umber)
- `NaN` is not `==` to any number, not even itself
- Number Methods:
- Number.isNan(), Number.toString(),……
- Test for it with `isNaN(x)`

  Example: Number.isNaN(0 / 0) //true

- Number properties and methods:
- http://www.w3schools.com/jsref/jsref_obj_number.asp

String Operations

- Operator: + (Concatenation)

  <u>Rule</u>: When both operands are numbers + is addition, otherwise string concatenation

- What is outcome of each expression?
  - `1 + 6`
  - `"6" + 1`
  - `1 + "6" + 2`
  - `"1" + "6"`
  - `"Jan " + 2010`
  - ` 1 + " " + 6`
  - `1 + 6 + "JS"`

String Operations

- What happens in these examples?

```
➢ 7 * '3'
➢ "6" * 2
➢ 2 * "Jan "
➢ 6 / "2"
➢ 1 - '6'
➢ 2 - "JS"
```

When use a non-string operator with strings, will try to convert string to a number. Two possible outcomes

1. A number
2. NaN (Not a number)

Other operators in this case?

- If one operand is number, and the other can be converted to a number, < is a number comparison,
- If one operand is number, and the other <u>cannot be</u> converted to a number, *false* all the time.
- If two operands are string, < is a string comparison

- What happens in these examples?
  - ➢ `11 < 2`
  - ➢ `"11" < 2`
  - ➢ `11 < "2 "`
  - ➢ `"11" < "2"`
  - ➢ `11 < "bird"`
  - ➢ `11 < 2 + "birds"`

String Operations

- Explicit conversions
  - Use the `String` and `Number` constructors
  - Use `toString` method of numbers
  - Use `parseInt` and `parseFloat` on string

```
var num = 6;
var str = String(num);
var str2 = num.toString();
var n1 = Number("6");
var n2 = parseInt("6");
```

String Operations

In JavaScript, strings are objects and have many useful fields and methods,

- `str.length` the length of the string *str*
- `str.charAt(i)` char at position *I*
- `str.substr(3)` or `str.substr(3,6)` or `str.substr(-3)`
- `str.substring(3)` or `str.substring(3,7)`

http://www.w3schools.com/jsref/jsref_obj_string.asp

String Operations

- **substring():** extracts the characters from a string, between two specified indices, and returns the new sub string. This method extracts the characters in a string between "start" and "end", not including "end" itself.

- If "start" is greater than "end", this method will swap the two arguments, meaning str.substring(1,4) == str.substring(4,1).

/
/substring does not accept negative values

When it is negative it starts from last

- **substr():**The difference with substring() is that the second parameter specifies the **length** of the extracted part

http://www.w3schools.com/jsref/jsref_obj_string.asp

String Operations …

- `str.indexOf(substr)` or -1 if not found
- `str.lastIndexOf(substr)` or -1 if not found

- **indexOf()** : Returns the position of the first found occurrence of a specified value in a string
- **indexOf()** :Returns the position of the last found occurrence of a specified value in a string

- `str.toLowerCase()` or `str.toUpperCase()`
- `str.concat(str2)`
- `str.replace(str1, str2)`

http://www.w3schools.com/jsref/jsref_obj_string.asp

The `Date` Object

- Some methods ….

Example : var d = new Date();
d.getTime();

`toLocaleString`	– returns a string of the date
`getDate`	– returns the day of the month
`getMonth`	– returns the month of the year (0 – 11)
`getDay`	– returns the day of the week (0 – 6)
`getFullYear`	– returns the year
`getTime`	– returns the number of milliseconds since January 1, 1970
`getHours`	– returns the hour (0 – 23)
`getMinutes`	– returns the minutes (0 – 59)
`getMilliseconds`	– returns the millisecond (0 – 999)

http://www.w3schools.com/jsref/jsref_obj_date.asp

Javascript

We will cover in this first part:

- History & uses
- Syntactic characteristics
- Primitives, operations & Expressions
- **I/O**
- Arrays
- Control statements

Output

- The `Document` object has a method, `write`, which dynamically creates content in the browser window

- The parameter is a string, often concatenated from parts, some of which are variables

Example:

```
document.write("Answer: " + result + "<br />");
```

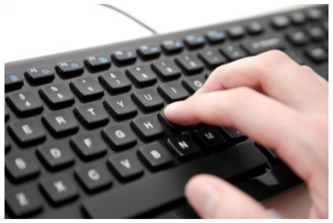
- The parameter is sent to the browser, so it can be anything that can appear in an HTML document (`<br />`, but not `\n`)

Just a note about `document.write`

- JavaScript treats the browser as a console
- The console is accessed via the document object
- Writing to the browser is done via the `write` or `writeln` method
    - html can be output and processed by the browser

```
document.write("<h1>My Header</h1>");
```

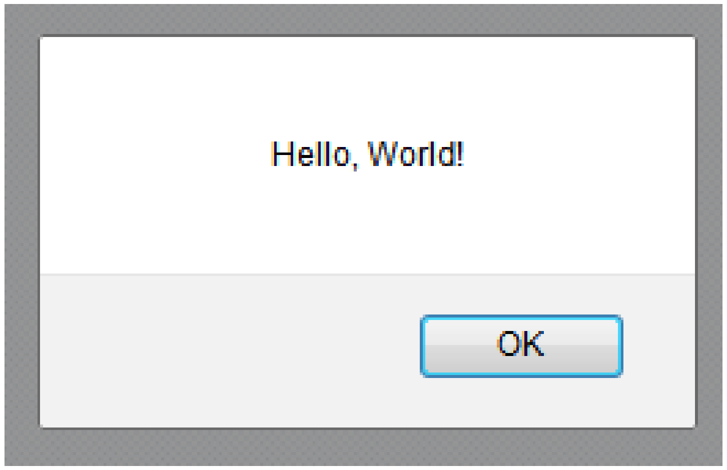
Input/Interacting with user



- The `Window` object has three methods for creating dialog boxes, `alert`, `confirm`, and `prompt`

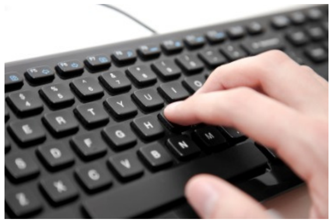
1. **`Alert`**: opens a dialog box, displays its parameter and displays an OK button.

```
alert("Hello, World!");
```



Hello, World!

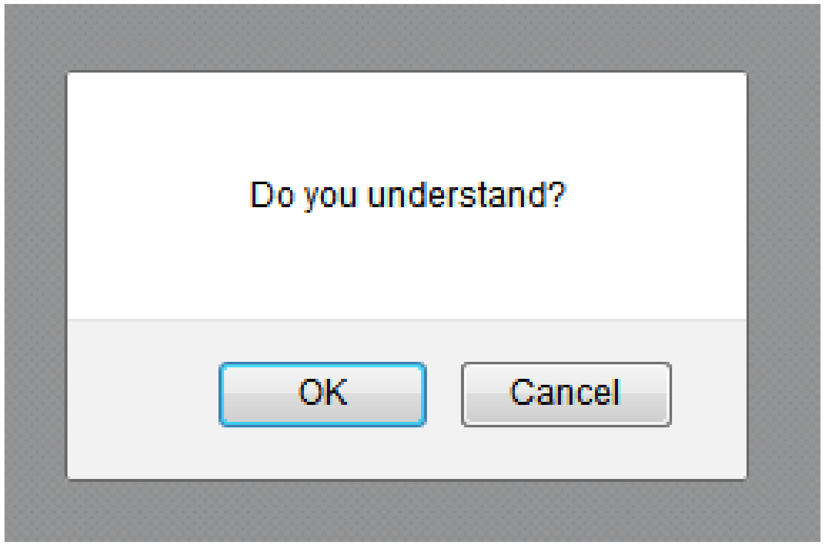
Input/Interacting with user …



2. **Confirm:** opens a dialog box, displays its parameter and displays `OK` and `Cancel` buttons.

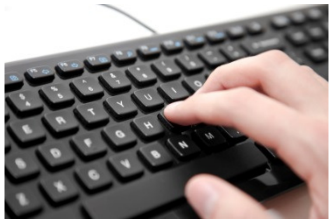
Ex:

```
confirm("Do you understand?");
```



Do you understand?

Input/Interacting with user …

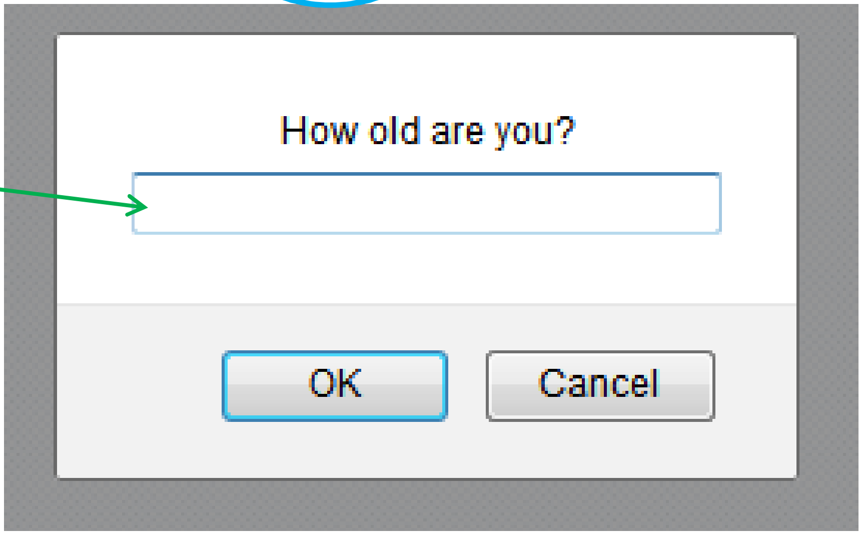


3. **Prompt**: opens a dialog box, displays its parameter and displays `OK` and `Cancel` buttons.

Ex:

```
prompt("How old are you?", "21");
```

Text Box



- Input returned as a string
- if nothing entered,
  value returned will be 21

roots.html

Javascript

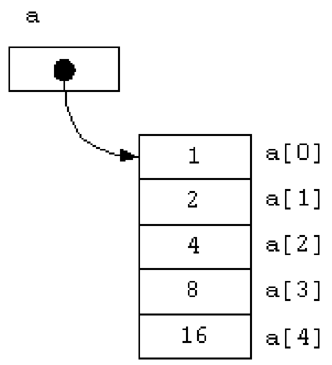
We will cover in this first part:

- History & uses
- Syntactic characteristics
- Primitives, operations & Expressions
- I/O
- **Arrays**
- Control statements

Composite (Reference) Data Types …

Array Object

- can have variables of different types in a same array



- Creating arrays:

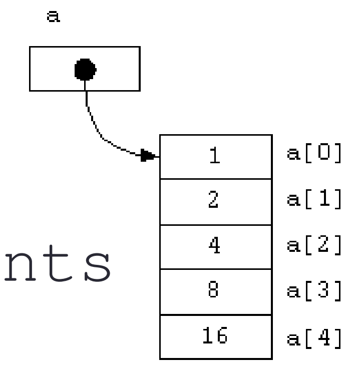
```
1. var b = new Array(entry, ...);
   var things = new Array( “Anna", "SOEN", 287);

2. var c = [entry, ...];
   var things = [“Anna", "SOEN", 287];
```

- Both 1 & 2 do exactly the same thing. For simplicity and readability best to use 2nd format. (based on W3C)

Just a note about `new Array()`

- The `new` keyword complicates your code and produces nasty side effects:



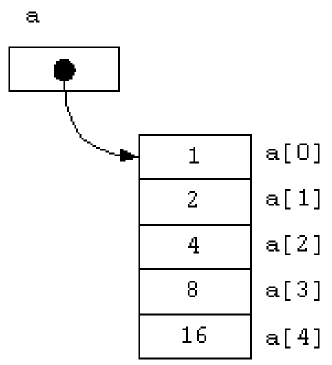
```
var points = new Array(40, 100);
// Creates an array with two elements
// 40 and 100)
```

```
var points = new Array(40);
// Creates an array with 40 undefined
// elements !!!!!
```

http://www.w3schools.com/js/js_arrays.asp

JavaScript Arrays

- Array elements can be set and retrieved with
  - `a[0] = "first";`
  - `a[1] = "second";`
  - `var value = b[6];`



- Accessing an undefined array entry gives the value `undefined`.

- Assigning to an element beyond the end of the array increases its length.

Just checking

Given:

```
var anArray = [1,2,3,4];
```

what is stored in array? What is size of array?

1. `anArray[4]= 4;`

   what is stored in array? What is size of array?

2. `anArray[6] = 10;`

   what is stored in array? What is size of array?

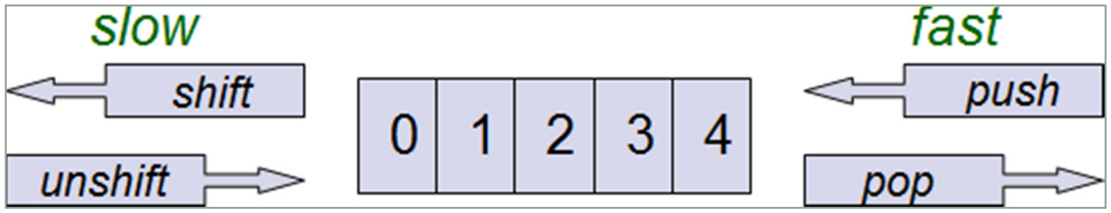
3. `anArray[3] = “Anna";`

   what is stored in array? What is size of array?

JavaScript Array Methods

- `pop():` removes and returns **last** element
- `shift():` removes and returns **first** element

  ex: `e = a.shift()`



http://javascript.info/tutorial/array

- `unshift(e1,e2, ...):`

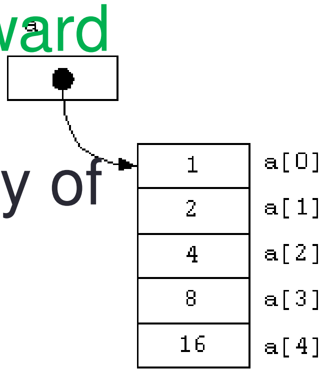
  inserts elements in **front** and returns new length
- `push(e1,e2, ...):`

  inserts elements at **end** and returns new length

http://www.w3schools.com/jsref/jsref_obj_array.asp

JavaScript Array Methods

- `concat(arr2, ...)`: returns a new array by joining the array with the given array(s)
- `reverse()`: changes the array itself to go backward
- `split(delimeter)`: split a string into an array of substrings



- `splice(index, howmany, item1, item2, …)`: adds/removes items to/from an array, and returns the removed item(s) (It changes the original array)

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
 fruits.splice(2, 1, "Lemon", "Kiwi");
Result: Banana,Orange,Lemon,Kiwi,Mango
```

http://www.w3schools.com/jsref/jsref_obj_array.asp

Just checking

Given:

```
var anArray = [1,2,3,4];
```

1. `var n = anArray.pop();`

   what is stored in array and n? What is size of array?

2. `var m = anArray.unshift(5,6);`

   what is stored in array and m? What is size of array?

3. `var p = anArray.shift();`

   what is stored in array? What is size of array?

Just checking

Given:

```
var anArray = [1,2,3,4];
```

```
4. var r = anArray.push(1,2);
```

what is stored in array and r? What is size of array?

```
5. var s = anArray.concat(anArray);
```

what is stored in array and s? What is size of array?

Just checking

Given:

```
var anArray = [1,2,3,4];
```

```
6. anArray.splice(1,1);
```

what is stored in array? What is size of array?

```
7. anArray.splice(1,0,5,7);
```

what is stored in array? What is size of array?

```
8. anArray.splice(2,2,6,8);
```

what is stored in array? What is size of array?

Javascript

We will cover in this first part:

- History & uses
- Syntactic characteristics
- Primitives, operations & Expressions
- I/O
- Arrays
- **Control statements**

Control Statements

- Similar to C, Java, and C++
- Compound statements are delimited by braces, but compound statements are not blocks

Control expressions – three kinds

1. *Primitive values* (operands must be identical)

   - If it is a string, it is true unless it is empty or "0"

   - If it is a number, it is true unless it is zero

Control Statements

2. *Relational Expressions*
   - *The usual six*: ==, !=, <, >, <=, >=
   - Operands are coerced if necessary
   - If one is a string and one is a number, it attempts to convert the string to a number
   - If one is Boolean and the other is not, the Boolean operand is coerced to a number (1 or 0)
   - T*he unusual two*: === and !==
     Same as == and !=, except that no coercions are done (operands must be identical)

Given that x = 5 ....

x == 8	true or false?
x == 5	true or false?
x == "5"	true or false?
x === 5	true or false?
x === "5"	true or false?
x != 8	true or false?
x !== "5"	true or false?
x !== 5	true or false?

Control Statements

3. Compound Expressions

- The usual operators: `&&`, `||`, and `!`
- The Boolean object has a method, `toString`, to allow Boolean values to be printed (`true` or `false`)

```
var x=false;
document.write("last test:",x.toString())
```

Control Statements

Selection Statements

1. The usual `if-then-else` (clauses can be either single statements or compound statements)

2. `Switch`

```
switch (expression) {
   case value_1:
      // value_1 statements
   case value_2:
      // value_2 statements
   …
   [default:
      // default statements]
}
```

http://alishagordon.com/2011/05/09/if-then/

Control Statements

Selection Statements

The statements can be either statement sequences or compound statements

The control expression can be a number, a string, or a Boolean

Different cases can have values of different types

Control Statements

Repetition Statements

1. `while (control_expression)`
   `statement` or `compound`

2. `for (init; control; increment)`
   `statement` or `compound`
   init can have declarations, but the scope of such variables is the whole script

```
3. do
     statement or compound
   while (control_expression);
```

The *foreach* loop

- Syntax:

```
for (var key in arr)
 { /* do something with arr[key] */ }
```

- Example:

```
ans=0;
grades=[7,8,9];
for(var k in grades) { ans += grades[k]; }
```

What is stored in `ans?`

Examples

What is output?

```
var bid = "35";
if (bid <= 50){
    document.write(bid +
        "does not meet minimum bid.<br />");
}
else {
    document.write("your bid of " + bid +
       " will be considered.<br />");
}
```