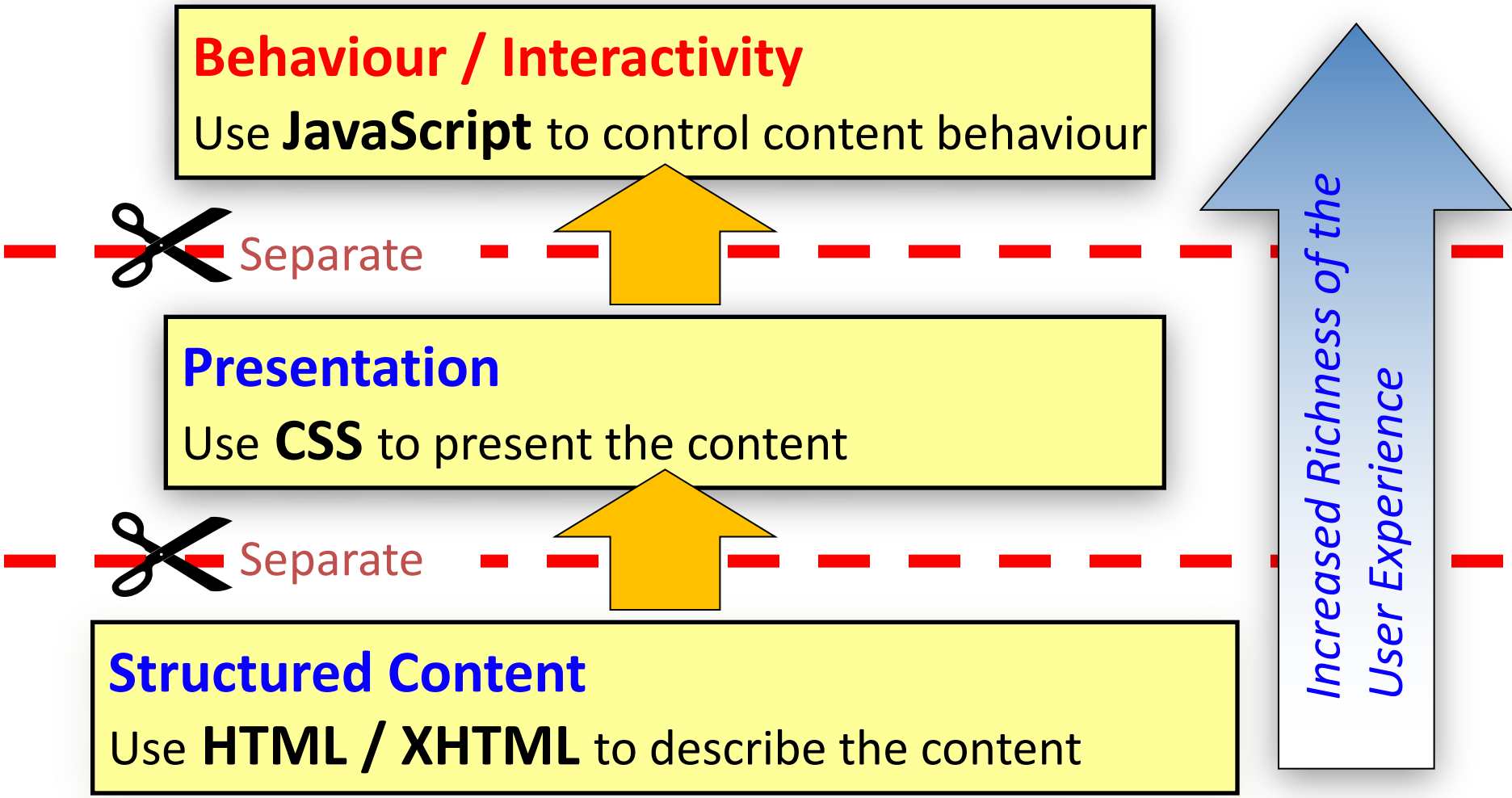
SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10005
Web Development

Module 7 – JavaScript Part 1

Web Development – The Process



Work from the bottom up !

Contents

Scripting

- A “scripting language” is a programming language that is usually
  - used for small tasks
  - an interpreted language (not “compiled”)
  - usually simple and flexible
  - relies on the script “interpreter” and the “interpreter environment” to allow it to do things.

    (For web pages, the “interpreter environment” is the browser!)
- Scripting languages can be used for both Client-side and Server-side situations

JavaScript

- An event-driven language, meaning it only does something when an event occurs, e.g.,
  - user clicks on the mouse button – `onclick`
  - user submits a form – `onsubmit`
  - user clicks on a link – `onclick`
  - User hovers mouse over an image – `onmouseover`

JavaScript: History

- Started in 1995 called LiveScript, then **JavaScript**
- An implementation of **ECMAScript**
- Not related to Java, but influenced by it
- Primarily used within web browsers, also
  - Embedded in PDF
  - Used to create desktop widgets
  - Used as basis of dynamic server-side functionality
  - Web and Windows Widgets
  - Web and Windows Apps …

So learning JavaScript is useful beyond Web Development ☺

JavaScript: During 1990s

- Mainly used to
  - Create alerts, pop-up windows
  - Check HTML form data client-side
  - Play audio files
  - Create so called dynamic HTML 'DHTML'
- Browser support for JavaScript was inconsistent

JavaScript: In 2005

- JavaScript usually only responds to events with client side processing
- The rise of Ajax
  - **A**synchronous **J**avaScript **a**nd **X**ML, according to Jesse James Garrett, who coined the term)
- Ajax makes a request of a server side resource, without reloading the webpage

JavaScript: The Rise of Frameworks

- A library of code whose purpose is to expedite development
- Work successfully regardless of browser
- Some frameworks
  - script.aculo.us (script.aculo.us)
  - Yahoo User Interface (developer.yahoo.com/yui)
  - jQuery (jquery.com)
  - ExtJS (www.sencha.com)
  - Dojo Toolkit (dojotoolkit.prg)
  - Prototype (prototypejs.org)

Contents

- Introducing Scripting
- Common JavaScript Uses
- Client-Server Interaction
- JavaScript Programming
- JavaScript Data Types
- JavaScript Variable and Constant
- JavaScript Expressions

Common JavaScript Uses

- **Check form data entered by users**

  On the client-side, check the form data entered by users, e.g., validity of username.

- **Control browser features**

  Display simple messages, pop open new windows, generate HTML code 'on the fly' etc.

- **Modify web page content**

  Change page content including text and graphics 'on the fly'.

- **Modify web page presentation**

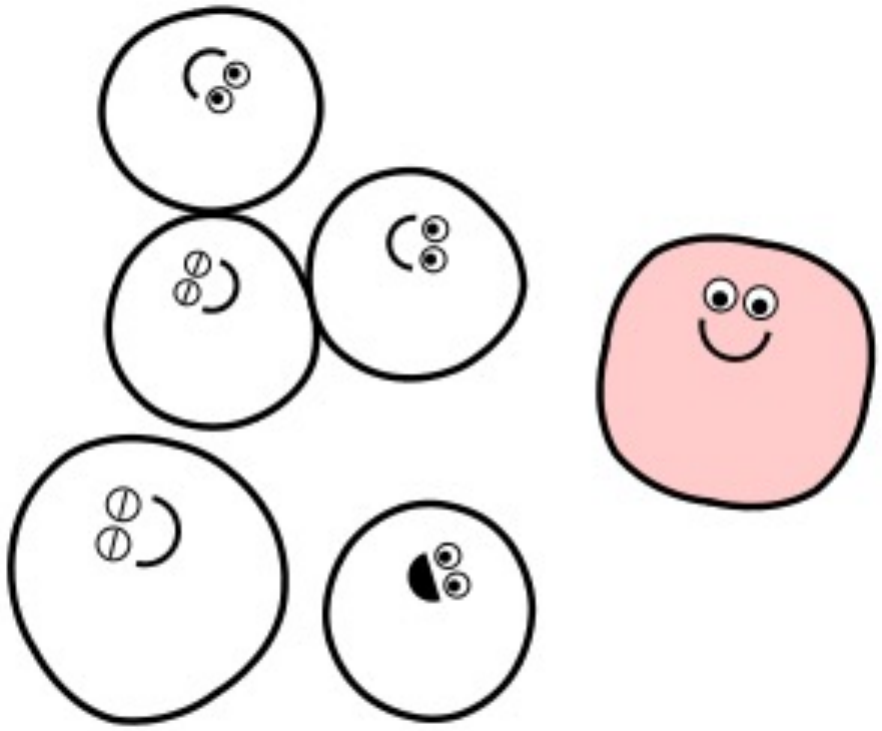
  Change CSS, background colour, text colour, hide and show info.

- **Manipulate images**

  Allows images to be changed (swapped), displayed randomly or in a sequence, "slide shows", "roll-over" effects etc. eg
  http://www.bom.gov.au/products/IDR024.loop.shtml

JavaScript Uses - Games etc …

- eg. JavaScript XHMTL +SVG
  an interactive animated SVG blob
  http://www.blobsallad.se/

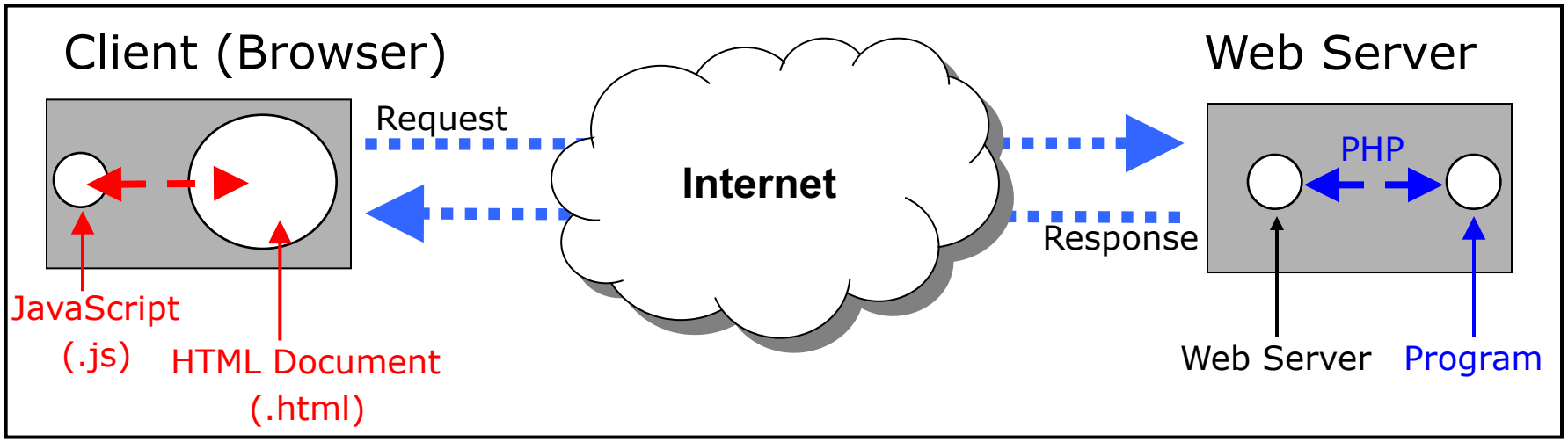


Contents

- Introducing Scripting
- Common JavaScript Uses
- Client-Server Interaction
- JavaScript Programming
- JavaScript Data Types
- JavaScript Variable and Constant
- JavaScript Expressions

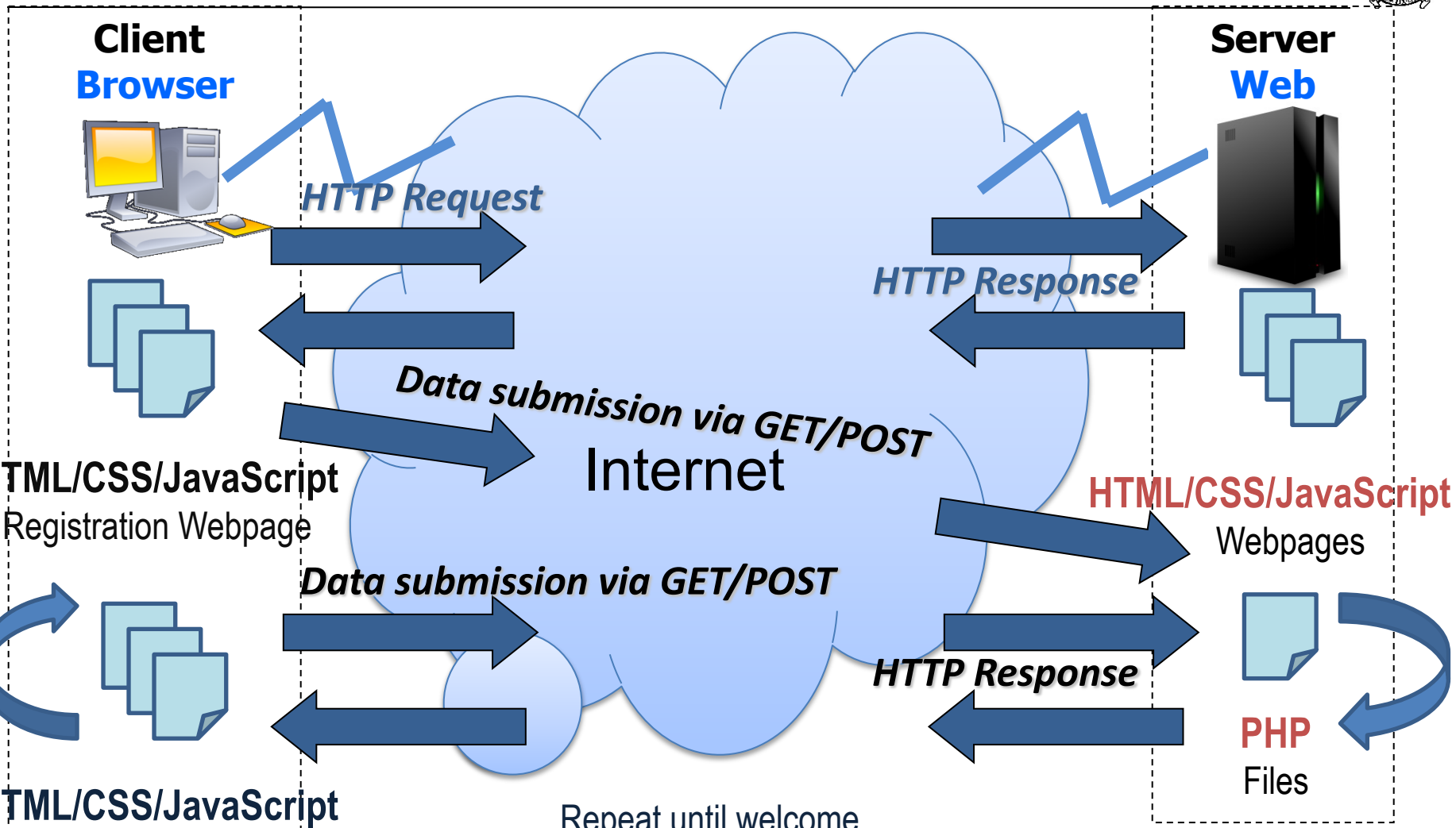
Client and Server Interaction

- Client-side scripting allows simple programs to be developed for the client.

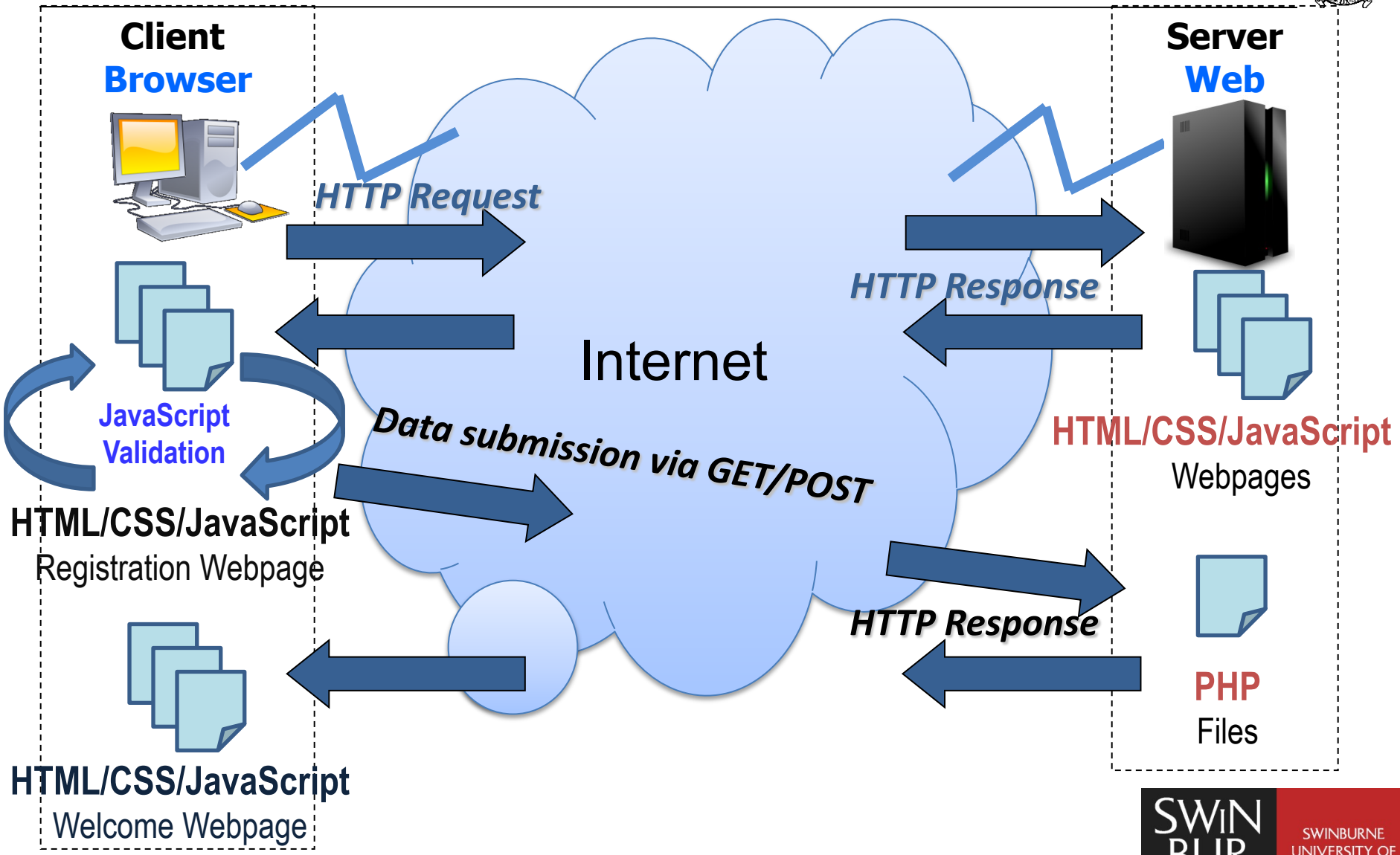


- **JavaScripts** are either **linked** or **embedded** into HTML files – when loaded, the script is executed (interpreted) **client-side** by the **browser**
- In contrast with **Server-side embedded scripting** like **PHP** which require interaction with the **server**.

Server Side Data Checking (PHP)



Client-Side Data Checking (JavaScript)



Contents

- Introducing Scripting
- Common JavaScript Uses
- Client-Server Interaction
- JavaScript Programming
- JavaScript Data Types
- JavaScript Variable and Constant
- JavaScript Expressions

JavaScript

- *JavaScript is **case sensitive**.*
- Statements are terminated in semicolons ;
  ***Always* terminate a statement with a semicolon ; ; ;**
- Has *Keywords* (reserved words) that have special meanings within the language syntax, such as

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

JavaScript

- For the purpose of demonstration and discussion of JavaScript, two inbuilt functions or methods can be used
  - **alert** displays a message through a pop-up window

```
alert("Welcome!");
```

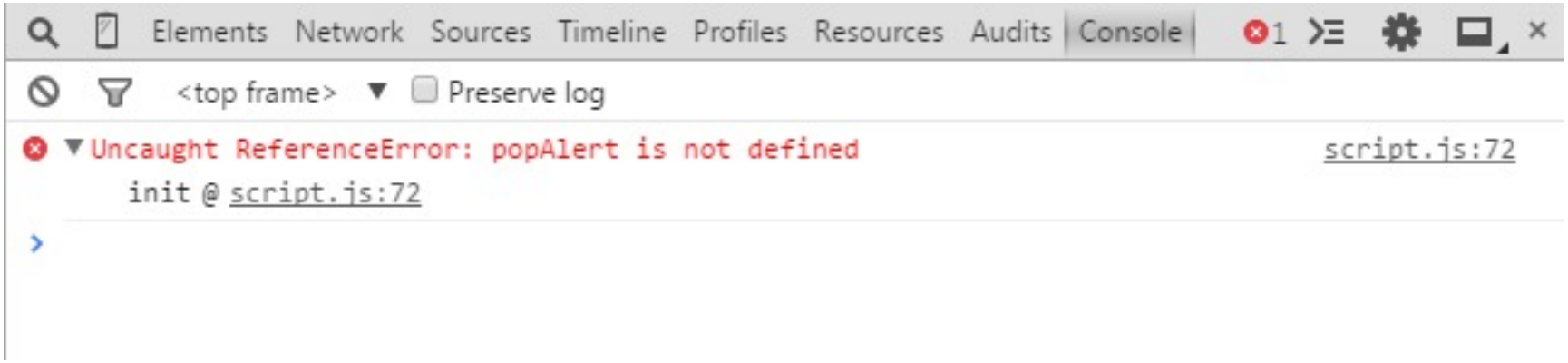
  - **prompt** displays a dialog box, returns a keyboard input from the user

```
variable =
        prompt("Your age?", 18);
```

Another way to retrieve user input. What was the first one?

JavaScript: Writing and Debugging

- **Chrome**
  - **From Menu: More Tools / Developer Tools**
  - **Keyboard Shortcut: *Ctrl-Shift-J (Windows)***
    Cmd + Opt + J (Mac OS)



JavaScript: Step 1 - Linking to HTML

- Embedded within HTML

```
<script>…</script>
```

- Included as an external file

```
<script
    type="text/javascript"
        src="folder/file.js">
</script>
```

JavaScript: Step 2

- In HTML, ensure HTML elements are given an ID as necessary.
- For example,

```
<a id="linkRun" href="#" >
    Run
</a>
```

JavaScript: Step 3

- In the .js file, create JavaScript functions to handle various events

```
function functionName () {
    /*JavaScript codes;*/
}
```

- For example,

```
function run() {
    alert("All good!");
    return true;
}
```

JavaScript: Step 4

- Create an initialisation function to assign each HTML element with an *event listener* and execute it once the HTML page completes loading
- For example,

```
function init() {
    /* obtain HTML elements
    link functions to the HTML elements' events */
}
window.onload = init;
```

IMPORTANT!

JavaScript: Step 4a

- Obtain a reference to a HTML element object in the following format

```
var linkRun =
    document.getElementById("linkRun");
```

- Alternatively,

```
var linkRun;
linkRun =
    document.getElementById("linkRun");
```

JavaScript: Step 4b

- Link a function to an HTML element's event using the following format
  - Example

```
linkRun.onclick = run;
```

JavaScript Application: Step 4b

- *Event* is an action that can occur, e.g.,
  - user clicks on the mouse button - onclick
  - user submits a form - onsubmit
- The following tables show *some* useful common events that can be used.
  More are being developed as devices change.

JavaScript Application: Step 4b

Mouse Event	The event occurs when the user
onclick	clicks on an element
ondblclick	double-clicks on an element
onmousedown	presses a mouse button over an element
onmousemove	Moves pointer is moving while it is over an element
onmouseover	Moves the pointer is moved onto an element
onmouseout	moves the mouse pointer out of an element
onmouseup	releases a mouse button over an element

JavaScript Application: Step 4b

Keyboard event	The event occurs when the user
onkeydown	is pressing a key
onkeypress	presses a key
onkeyup	releases a key

JavaScript Application: Step 4b

Form events	The event occurs when
onblur	a form element loses focus
onchange	the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)
onfocus	an element gets focus (for <label>, <input>, <select>, <textarea>, and <button>)
onreset	a form is reset
onselect	a user selects some text (for <input> and <textarea>)
onsubmit	a form is submitted

JavaScript: Putting All Pieces Together

```
<!DOCTYPE html>
<html lang="en">
<head>
    …
    <script src="my_jsfile.js"></script>
</head>
<body>
    …

    <button type="button" id="clickme">
…
</body>
</html>
```

```
/* Filename: my_jsfile.js
   …
*/

function doSomething()
{

    /* do something here in response to user's click on
       the button
    */

}

function init() {
    var clickme = document.getElementById("clickme");
    clickme.onclick = doSomething;
}

window.onload = init;
```

Contents

- Introducing Scripting
- Common JavaScript Uses
- Client-Server Interaction
- JavaScript Programming
- JavaScript Data Types
- JavaScript Variable and Constant
- JavaScript Expressions

Primitive Data Types

- String `var name = "John Doe";`
- Number `var age = 18;`
- Boolean `var isHuman = true;`

These are collectively referred to primitive data types.

String

- Is a sequence of characters
- created directly by placing the series of characters between double or single quotes, for example
  - "This is a string"
  - 'This is also a string'

String

- Uses embedded control characters
- For example,
  - "There will be a new line here\n and this will be on the next line"

Seq	Usage	Seq	Usage
\b	backspace	\\	backslash
\f	formfeed	\"	double quote
\n	newline	\'	single quote
\r	carriage return	\###	Octal encoded character
\t	horizontal tab	\uHHHH	Unicode encoded character

Number

- Integers can be positive, 0, or negative
- Integers can be expressed
  - in decimal (base 10), e.g., `30.00`
  - hexadecimal (base 16), e.g., `0xFF (=255)`
  - and octal (base 8), e.g., `0200 (=128)`
- A decimal integer literal consists of a sequence of digits without a leading 0 (zero)

  example: `255`

Number

- A leading 0 (zero) on an integer literal indicates it is in octal

  example: `0377 = 255`

- Octal integers can include only the digits 0-7.
- A leading 0x (or 0X) indicates hexadecimal.

  example: `0xFF = 255`

- Hexadecimal integers can include digits (0-9) and the letters a-f and A-F.

```
255 = 0377 = 0xFF
```

Number

- A floating-point number can contain either
  - a decimal point
  - an "e" (uppercase or lowercase) which is used to represent "ten to the power of" in scientific notation
  - or both
- exponent part is an "e" or "E" followed by an integer, which can be signed (preceded by "+" or "-")
  - `1.025e3 = 1025`
  - `130e-3 = 0.130`

Boolean

- Boolean values are ***true*** and ***false***
  - `var isSelected = true;`
  - `var isSelected = false;`

Contents

- Introducing Scripting
- Common JavaScript Uses
- Client-Server Interaction
- JavaScript Programming
- JavaScript Data Types
- JavaScript Variable and Constant
- JavaScript Expressions

Variable

- A container that contains
  - a value (can be any of the primitive data type)

```
var fName="Michael";
var age;
age=10;
```

- The name you assign to a variable is used as an **identifier** or a **reference.**

Variable Name

- Must start any letter of the alphabet or underscore
  - `abc, AB_C, _abc` √ `1abc, %abc, +abc` X
- Can include any letter of the alphabet, digits 0-9, and underscore
  - `Abc10, abc_10` √ `abc-10, abc%10` X
- **Cannot** include spaces, or punctuation characters such as comma, full stop
  - `Abc 123, abc.123` X
- **Is case-sensitive**.

Variable Name

- You should follow a consistent variable naming style
  - `votingAge`
  - `voting_age`
  - `votingage`
  - `VotingAge`
  - `VOTING_AGE`
- Are the two variable names below referring to the same variable (identifier)?
  - `firstName`
  - `FirstName`

Variable Declaration

- Specifying and creating a variable name is called **declaring** the variable
  - `var abc;`
- Assigning a first value to a variable is called **initialising** the variable
  - `var abc = 10;`

Variable Declaration

- Variables are declared using the **var** keyword
  - declaring one variable

```
var firstName;
```

  - declaring multiple variables

```
var firstName, lastName;
```

  - declaring and assigning one variable

```
var firstName = "Java";
```

  - declaring and assigning multiple variables

```
var firstName = "Java", lastName = "Script";
```

Variable Declaration (Global)

- You must declare and initialise a global variable in the same statement

```
script.js:
var vGlobal=100;

function testGlobal() {
     alert(vGlobal);
}
```

- You can change the variable's value in any functions in that `.js` file

```
vGlobal = 60;
```

Variable Declaration (Local)

- You declare a local variable in a function

```
script.js:

function testLocal() {
        var vLocal=100;
        alert(vLocal);
}
```

You can use a local variable only in the function where it is declared.

```
function testLocal1() {
        var vLocal=100;
        alert(vLocal);
}

function testLocal2() {
        alert(vLocal);
}
```

√

X

Constants

- Contains information that does not change during the course of program execution
- Declared with the **const** keyword.
- Must start with a letter or underscore and can contain alphabetic, numeric, or underscore characters
- Name usually are all in uppercase.

```
const PI = 3.14;
const GST = '10%';
```

Contents

- Introducing Scripting
- Common JavaScript Uses
- Client-Server Interaction
- JavaScript Programming
- JavaScript Data Types
- JavaScript Variable and Constant
- JavaScript Expressions

Expressions

- An **expression** is a ~~literal value or variable~~ combination of operands and operations that can be evaluated to produce a result
- **Operands** are variables and literals contained in an expression
- A **literal** is a value such as a literal string or a number
- **Operators** are symbols (e.g. +, *) that are used in expressions to manipulate operands

Expressions

Operator Type	Description
String	Performs operations on strings
Arithmetic	Performs mathematical calculations
Assignment	Assigns values to variables
Comparisons	Compares operands and returns a Boolean value
Logical	Performs Boolean operations on Boolean values

String Operator

- **String operator** is used to concatenate two string

Operator	Name	Description
+	Concatenation	Joins two operands

"This text" **+** "That Text"

- Results

"This TextThat Text"

Arithmetic Operators

Operator	Name	Description
+	Addition	Adds two operands
-	Subtraction	Subtracts one operand from another operand
*	Multiplication	Multiplies one operand from another operand
/	Division	Divides one operand by another
%	Modulus	Divides one operand by another and returns the remainder

Modulus:

7%5=2 8%5=3 8%3=2

Arithmetic Operators

- The increment (++) and decrement (--) operators can be used to increase or decrease an operand by 1

Operator	Name	Description
++	Increment	Increases an operand by a value of one
--	Decrement	Decreases an operand by a value of one

```
var v = 4;
++v;          (v: 5)
--v;          (v: 4)
```

Assignment Operators

- **Assignment operators** are used for assigning a value to a variable:

```
myFavoriteSuperHero = "Batman";
```

- **Compound assignment operators** perform mathematical calculations on variables and literal values in an expression, and then assign a new value to the left operand

Assignment Operators

Operator	Name	description
=	Assignment	Assigns the value of the right operand to the left operand
+=	Compound addition assignment	Adds the value of the right operand to the value of the left operand and assigns the sum to the left operand
-=	Compound subtraction assignment	Subtracts the value of the right operand to the value of the left operand and assigns the difference to the left operand
*=	Compound multiplication assignment	Multiplies the value of the right operand to the value of the left operand and assigns the product to the left operand
/=	Compound division assignment	Divides the value of the right operand to the value of the left operand and assigns the quotient to the left operand
%=	Compound modulus assignment	Divides the value of the right operand to the value of the left operand and assigns the remainder (modulus) to the left operand

Assignment Operators

- Some operators are created to allow the use of fewer characters of code

```
var x = 100;
var y = 200;
x += y;   same as      x = x + y;     x=? y=?

var x = 2;
var y = 6;
x *= y;   same as      x = x * y;     x=? y=?

var myName = "Andreea";
myName += "M";      (myName: "AndreeaM")
same as
myName = myName + "M";
```

Comparison Operators

- **Comparison operators** are used to compare two operands and determine how one operand compares to another
- A Boolean value of `true` or `false` is returned after two operands are compared
- The comparison operator compares values, whereas the assignment operator assigns values
- Comparison operators are used with **conditional statements** and **looping statements**

Comparison Operators

Operator	Name	Description
==	Equal	Returns true if the operands are equal
!=	Not equal	Returns true if the operands are not equal
>	Greater than	Returns true if the left operand is greater than the right operand
<	Less than	Returns true if the left operand is less than the right operand
>=	Greater than or equal	Returns true if the left operand is greater than or equal to the right operand
<=	Less than or equal	Returns true if the left operand is less than or equal to the right operand

Logical Operators

- **Logical operators** are used for comparing two Boolean operands for equality
- A Boolean value of **true** or **false** is returned after two operands are compared

Operator	Name	Description
&&	AND	Returns true if both the left operand and right operand return a value of true; otherwise, it returns a value of false
||	OR	Returns true if either the left operand or right operand returns a value of true; if neither operand returns a value of true, it returns a value of false
!	NOT	Returns true if an expression is false and returns false if an expression is true

Operator Precedence

- **Operator precedence** determines the order in which operators are evaluated.
- Starting from the highest precedence with the operators presented, we have
  - Arithmetic operators (unary)
  - Arithmetic operators (binary - *, /, % then +, -)
  - Comparison operators
  - Logical operators
  - Assignment operators

Evaluation of Expression

Consider the following example:

- `25 + 100 * 4;`

Is it 425 or 500?

- `4*2+4;`

Is it 24 or 12?

- `(60 * 5) + ((40 / 8) - 9) - ((4 * 6) / 2);`

Is it 284 or 275?

- `7 % 5`

How about this?

Evaluation of Expression

Given that `x = 6` and `y = 3`

What is the value of x in the following statements?

- `x = x + y;`
- `x = x % y;`
- `x++;`

What is the result returned after evaluating the following expression?

- `(x < 10 && y > 1)`
- `(x==5 || y==5)`
- `!(x==y)`
- `x===5`

NEXT LECTURE:

JAVASCRIPT – PART 2

SWINBURNE UNIVERSITY OF TECHNOLOGY