# BTI220 - Internet Architecture and Development

## Week 1: Course Introduction and Internet Basics

# Agenda

- Course introduction
- Internet architecture
- Web client-side programming
- Basic introduction to JavaScript

# Course Overview

- Course outline

  https://ict.senecacollege.ca/course/bti220
- Course standards

  https://scs.senecac.on.ca/~wei.song/bti220/bti220.html#standards
- Academic Honesty Policy

  https://scs.senecac.on.ca/page/academic-honesty-policy
- In this course, you will learn web client side (front-end) development concepts and techniques, including JavaScript, HTML, CSS, and DOM

# Course Overview

- No text book.
- Reference material:
  - Mozilla Developer Network (MDN) start page
    by the Mozilla Developer Network and individual contributors
    http://developer.mozilla.org
  - Web Education Community Group Wiki
    by the W3C Web Education Community Group
    http://www.w3.org/community/webed/wiki/Main_Page
  - Your Web, Documented
    by the W3C and the Web Platform stewards
    http://www.webplatform.org/

# Course Overview

- Lecture notes, labs and assignments will on my website:

  https://scs.senecac.on.ca/~wei.song/

- Tests & Quizzes
  - 2 term tests are scheduled in week 6 and week 11.
  - 3 quizzes are not pre-announced. No make-up for quizzes.
- Labs and assignments
  - Late penalties: 10% for first day, and 5% for each subsequent day to a maximum of 14 days.
- Grades will be posted on Blackboard

# Evaluation

| | | |
|---|---|---|
| All labs | (1 % or 2% each) | 15% |
| 4 assignments | (6% + 6% + 6% + 7%) | 25% |
| term tests | (15% + 15%) | 30% |
| Final exam | | 30% |
| Total | | 100% |

# Promotion Policy

To obtain a credit in this subject, a student must:

- Achieve a grade of 50% or better on the final exam
- Satisfactorily complete all 3 assignments
- Achieve a weighted average of 50% or better for the tests and final exam
- Achieve a grade of 50% or better on the overall course

# How to Get an A in this Course

- Attend all lectures and labs.
- Take your own notes
- Ask question if you don’t understand.
- Complete labs, assignments on time.

# Expectation

- Mine:
  - Be present. Be organized/don't fall behind.
  - Be active in class/labs.
  - Read not only the lecture notes.
  - Learning strategy:
    - Getting the big picture (concepts, framework, architecture)
    - paying attention to the details (coding, syntax, hands-on)
    - Thinking, practicing and memorizing!
- Yours?

# Communication

- Blackboard / mySeneca
  - Announcements, marks, …
- My web site:

  https://scs.senecac.on.ca/~wei.song/
- Email:

  wei.song@senecacollege.ca
- Student help hours:

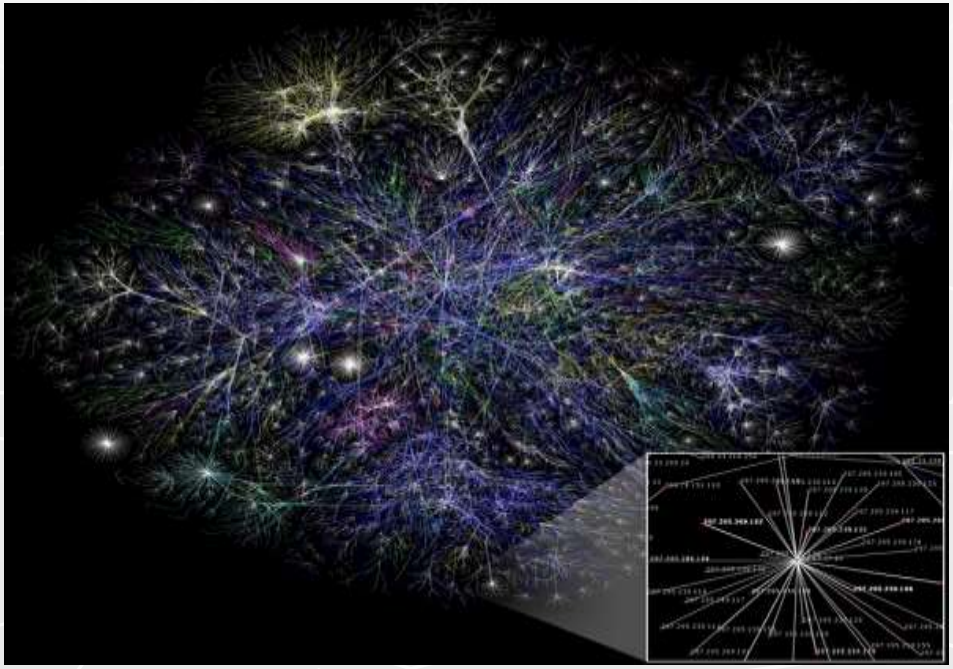
  http://scs.senecac.on.ca/~wei.song/#timetable
- Office room: Tel -2099

# BTI220 Zenit Account

- Your instructor will issue each student registered in the BTI220 course an account/password on zenit.senecac.on.ca for submitting your labs and assignments.
- zenit.senecac.on.ca is a secure Linux server running an Apache web server.
- Your account on zenit has sftp and SSH access.
- The account and password can be found in the Grade section of the Blackboard.
- You must abide by all the rules and regulations set forth in the "Seneca College Acceptable Account Use Policy".
- Sharing your username/password with others will be considered an act of cheating.
- Secure your account and don’t change the password.

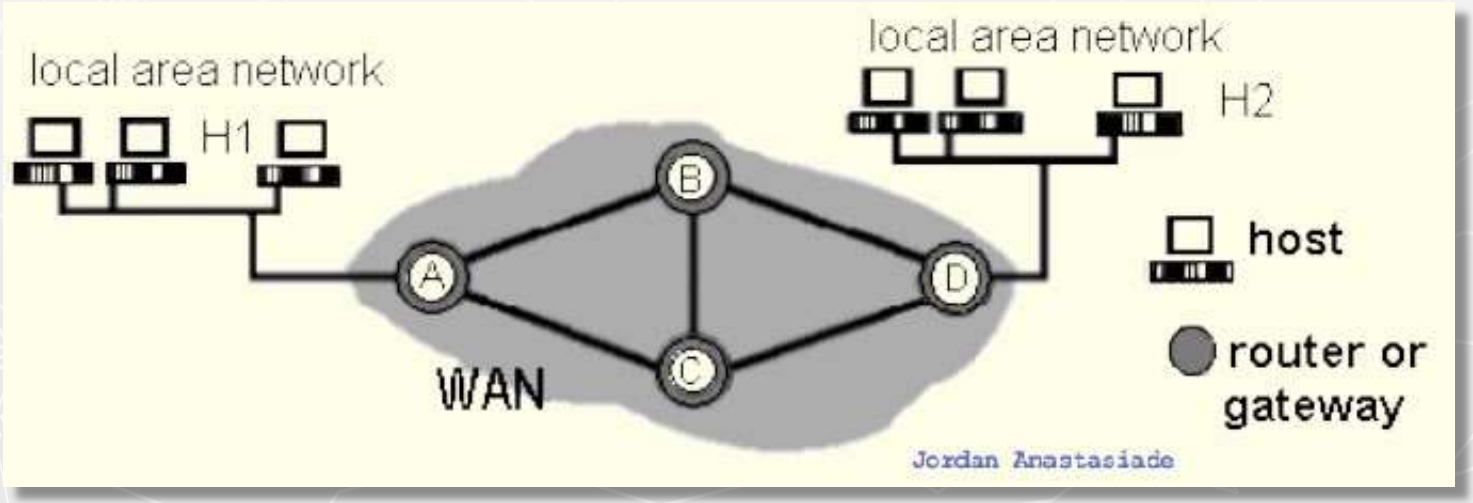
# Internet Architecture

- The internet is a collection of networks connected to each other. The networks are connected together to form the single entity that we know as the Internet.
- The Internet architecture is described in its name, a short form of the two words - interconnected networks.



# Elements of Networks

- Communication links:
  - point-to-point (e.g., A-to-B)
  - broadcast (e.g.,: Ethernet LAN)
- Host: computer running applications which use network (e.g. H1)
- Router: computer routing packet from input line to output line. (e.g. C)
- Gateway : a router directly connects networks (e.g. A)



# Internet Protocol Suite

- Communications protocol
  - a formal description of message formats
  - the rules for exchanging those messages
- Internet Protocol Layers
  - Application layer
    (e.g. HTTP, FTP, Telnet, SMTP, DNS)
  - Transport layer
    - TCP (Transmission Control Protocol),
    - UDP (User Datagram Protocol)
      - Applications that do not require the reliability of a connection
  - Network layer : IP (Internet Protocol)
  - Physical and data link layers
    e.g. Ethernet, token ring

# Internet Application Protocols

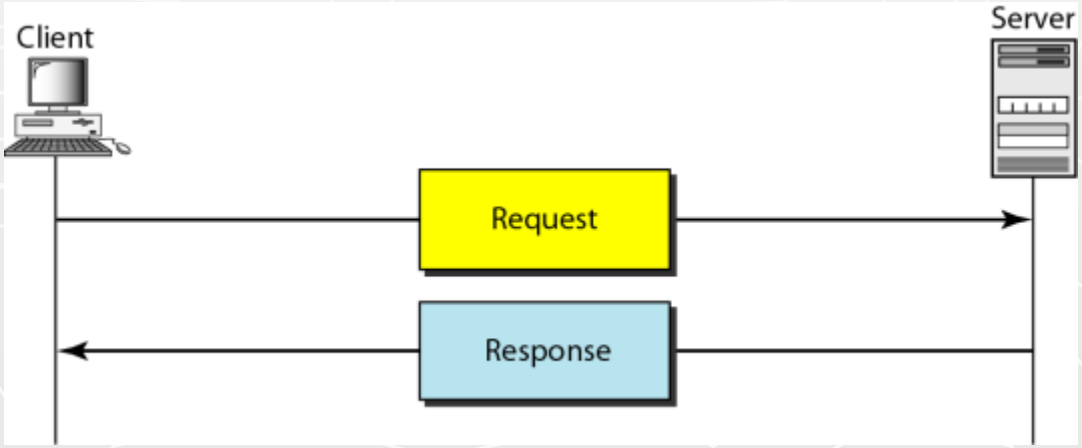
- Remote login category
  - Telnet
  - SSH, **Secure Shell**
- File transfer category
  - FTP, File Transfer Protocol
  - SFTP, Secure File Transfer Protocol
- Support services category
  - DNS, Domain Name System
  - SNMP, Simple Network Management Protocol
  - CMOT, Common Management Information Protocol
- Electronic mail category
  - SMTP, Simple Mail Transfer Protocol
  - IMAP, Internet Message Access Protocol
  - POP, Post Office Protocol
- Other protocols
  - HTTP, HyperText Transfer Protocol
  - HL7, Health Level Seven
  - LDAP, Lightweight Directory Access Protocol
  - NFS, Network File System
  - DHCP, Dynamic Host Configuration Protocol
  - IRC, Internet Relay Chat
  - …

# Services Provided by the Internet

- World Wide Web
  - Abbreviation: WWW or W3,
  - Commonly known as the Web.
  - It is a collection of web pages connected through **hyperlinks** and **URL**s.
  - With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks.
  - It is governed by the Hyper Text Transfer Protocol (**HTTP**) that deals with the linking of files, documents and other resources of the web.
- Other services:
  - Electronic Mail (email), Mailing List, File Transfer Protocol (FTP), Instant Messaging, News Groups, Chat Rooms

# Client Server Model

- The www uses a client-server model
  - client software (web browser) requests an html page
  - the request is sent as a message to the particular web server
  - requested page is sent as a message from the server to the client
  - the client software interprets and displays the html page



# Uniform Resource Locators (URL)

- Also known as a web address. It is a reference (an address) to a resource on the Internet.

**https://scs.senecac.on.ca:443/~wei.song/index.html#timetable**

  - protocol = https://
  - domain / host = scs.senecac.on.ca
  - port = 443, default for HTTPS
  - file / resource ID = ~wei.song/index.html
  - reference = #timetable

- URLs are typically transmitted via HTTP

# DNS (Domain Name System/Server)

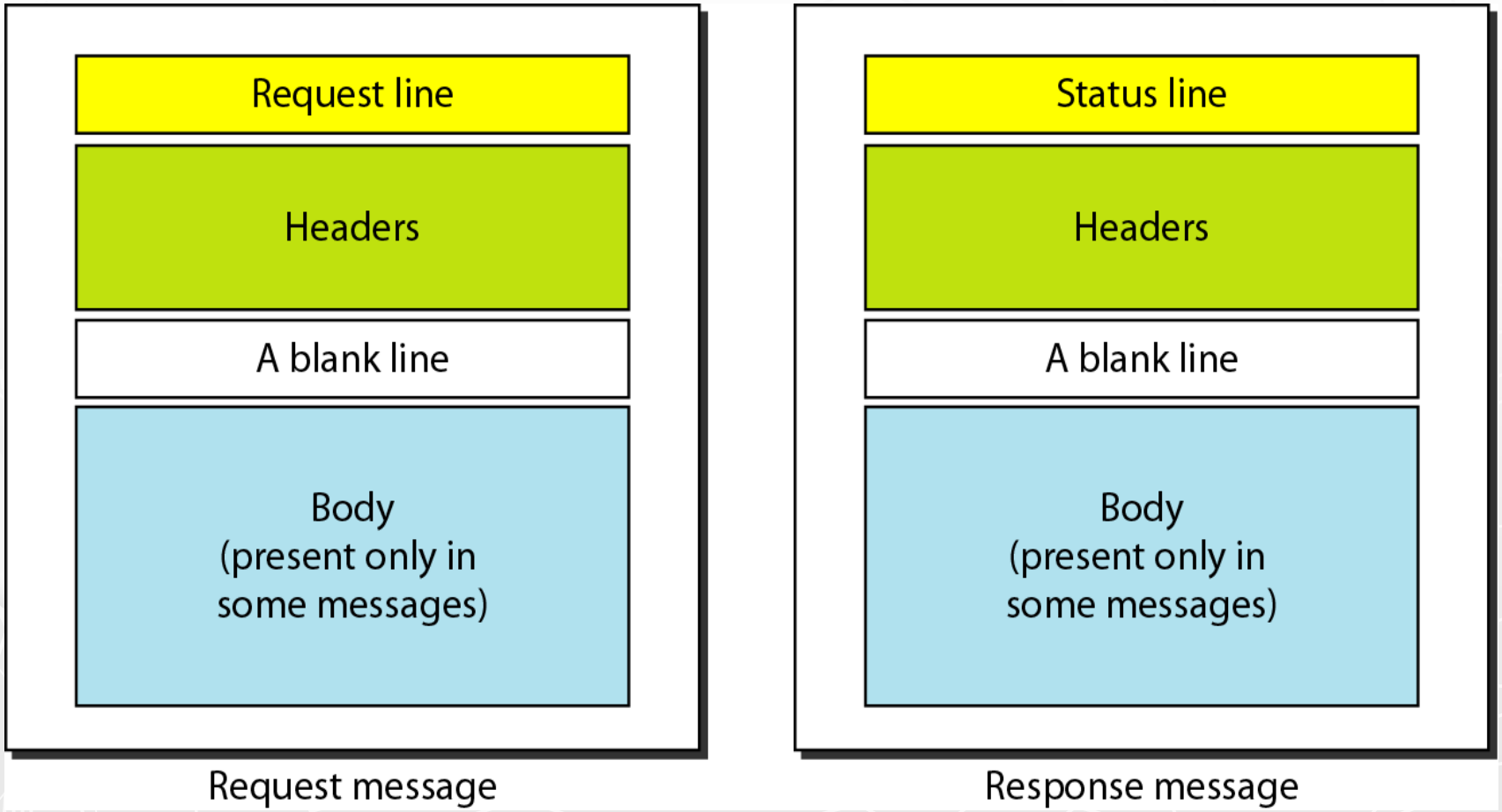
- DNS is used to give names to IP addresses. DNS servers (name servers) associate the domain names with the IP address
- e.g.

  zenit.senecac.on.ca is used to identify IP address 142.204.140.203.
- In addition to ".ca", other common domains include
  - .com - commercial
  - .edu - educational
  - .gov - governmental
  - .net - isp
  - .org - non-profit
  - and many more
- ICANN - Internet Corporation for Assigned Names and Numbers- oversees assignment of names and IP addresses and certifies domain name registrars to manage the process.

# Hypertext Transfer Protocol

- HTTP, the Hypertext Transfer Protocol, is the application-layer protocol that is used to transfer data on the (World Wide) Web.
- HTTP comprises the rules governing the format and content of the conversation between a web client and server.
- HTTP functions as a request-response protocol in the client-server computing model.
  - It follows a classical client-server model, with a client opening a connection, making a request then waiting for a response until it receives it.
- HTTP is stateless.
  - The server doesn't keep any data (state) between two requests.

# HTTP Request and Response Messages



# HTTP Request

- HTTP request example: sending the form result to a server:

```
POST /contact_form.php HTTP/1.1
Host: developer.mozilla.org
Content-Length: 64
Content-Type: application/x-www-form-urlencoded

name=Joe%20User&request=Send%20me%20one%20of%20your%20catalogue
```

- HTTP request methods
  - GET, POST
  - PUT, DELETE, HEAD, TRACE

# HTTP Response

- HTTP response example: successful reception of a web page:

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)
```

- HTTP Response Status Codes
  - 1xx - information
  - 2xx – success. e.g. 200 = request succeeded.
  - 3xx - redirection
  - 4xx – client error. e.g. 403 = forbidden page
  - 5xx – server error. e.g. 500 = internal server error.

# HTTP Secure


- Hypertext Transfer Protocol Secure (HTTPS) is a communications protocol for secure communication over a computer network, with especially wide deployment on the Internet.
- Technically, it is not a protocol in and of itself; rather, it is the result of simply layering the HTTP on top of the SSL(Secure Sockets Layer) / TLS(Transport Layer Security) protocol, thus adding the security capabilities of SSL/TLS to standard HTTP communications.

# Standards & the World Wide Web Consortium (W3C)

- The World Wide Web Consortium (W3C):
  - main international standards organization for the World Wide Web.
  - in order to continue the development of the web, and its languages
  - Founded: October 1994
- Founder: Tim Berners-Lee

# Web Application

- A web application or web app:
  - A distributed application that uses Web-based technologies (and generally relies on Web browsers for the presentation of user-interfaces) is typically considered a Web application.
  - Update and maintain web applications without distributing and installing software on potentially thousands of client computers
  - inherent support for cross-platform compatibility.
- Common web applications include:
  - Webmail, online retail sales, online auctions, wikis and more…

# Front-end Web Application

- The rising popularity of so-called "modern" web apps means that web developers are focusing on writing more and more front-end, or client-side code.
- Although back-end, or server-side code still plays an important factor, the fact is that web developers are working more directly with HTML5, CSS3, JavaScript and the DOM.

# Front-end Web Application

- The four pillars of front-end web development:
  - HyperText Markup Language (HTML5)
    - The main language for creating web pages
    - Content and structure of the Document
  - Cascading Style Sheets (CSS)
    - Used for describing the appreance and formatting of a web page
    - Presentation or style web pages
  - JavaScript (JS)
    - Allows client-side scripts to interact with the user
    - Behavior and State of the frontend
  - Document Object Model (DOM)
    - 2-way programming API for JavaScript

# Front-end Web Apps Demos

- The Planetarium

  Beautiful Web Introduction to the solar system
- BananBread

  Web FPS shooter
- CSS Tricks

  Web Animation (using data from 3d body tracking)
- Responsive Web Application

  For multi-media types:
  - Screen (mobile, desktop), print, projection, tty, …

# Let’s start with JavaScript

# Firefox Developer Tool: Scratchpad

- Why use Scratchpad?
  - JavaScript always runs inside a host environment (mostly the browser).
- Starting Scratchpad
  - Go to the "Web Developer" menu. Then select "Scratchpad" from that menu, and you'll get a text editor window.
  - Hint for Mac users: look for the "Web Developer" menu under "Tools".
  - Shortcut: Shift+F4
- More on Developer tools:
  - Firefox developer tools
  - Chrome developer tools
  - JSLint – JavaScript code analysis tool
  - JSFiddle - an online playground for web developers

# Scratchpad

- How to run a script:

1. Enter some code
2. Select a portion of the code
3. Choose one of the three commands from the Execute

# Input and output

- JavaScript does not include input and output facilities. Its host environment (e.g. a browser) provides 3 popup Boxes (Modal windows) :
  - alert()
  - prompt()
  - confirm()

  as well as
  - console.log()

# alert()

- Alert box, popups a modal window, and user need to click on “OK” to continue.
- Outputs info.
- Example code:

```
window.alert(“Welcome to BTI220!”);
```

- Same as:

```
alert(“Welcome to BTI220!”);
```

# confirm()

- Confirm box, needs user to verify or accept something.
- Returns a boolean value: true or false
- Example code:

```
var result = confirm("Are you OK?");
if (result) {
    alert("You pressed OK!");
} else {
    alert("You pressed Cancel!");
}
```

# prompt()

- Prompt box, displays a dialog box (Modal window) that prompts the user for input.
- Returns a string entered by the user input; or return the value null.
- Example code:

```
var cl = prompt("Enter your favorite color", "green");
if (cl != "") {
    alert(cl);
}

if(!cl) { // JavaScript Falsy
    alert("No color entered.")
}

1+2; // Inspect/display executes the selected code,
     //and prt the result right into your editor.
```

# console.log()

- Shows a message in web console.
- Outputs info.
- Example code:

```
var anObject = { str: "Some text", id: 5 };

console.log(anObject);
```

- View log:
  - Ctrl + Shift + J to open browser console.

# Introduction to JavaScript

- JavaScript (sometimes shortened to JS) is a lightweight, interpreted, high-level language used along with html code.
- The language syntax is somewhat similar but not the same as the C language. Today, JavaScript is the scripting language for Web pages.
- JavaScript is not Java
- An interpreted language interprets and executes each statement - one-by-one - in the order they appear.
- JavaScript always runs inside a  host environment (mostly the browser).

# Introduction to JavaScript

- Netscape originally developed JavaScript under the name "Mocha" then "LiveScript" and finally renamed to JavaScript.
- The JavaScript standard is based on the European Computer Manufacturers Association (ECMAScript). As of 2012, all modern browsers fully support ECMAScript 5.1.
- JavaScript is useful for making dynamic web pages, validating user input and changing the way the web page responds to events on the web page.
- JavaScript statements can be stored in an external file with a .js file extension or embedded within HTML code.

# About JavaScript

- JavaScript is one of the world's most popular programming languages .
  - The role as the scripting language of the WWW.
  - simple and easy to learn
- JavaScript is the world's most misunderstood programming language.
  - The name, typecasting, used by amateurs, object-oriented,…
- JavaScript is, in the future, the most important language you will learn(? Dart?)

# Basic JavaScript Rules

1. JavaScript is case-sensitive
   - When writing a JavaScript script, be aware of upper and lower case characters. CustomerCount is not the same as Customercount nor is it the same as customerCount
2. JavaScript statement
   - A JavaScript typically consists of a series of statements. A statement is a single line of instruction to the computer made up of objects, expressions, variables, and events/eventhandlers.
   - Every statement has the same structure:
     - A statement starts on its own line (recommended but not required)
     - The first item in a statement is a command
     - The second part following the command is some information about that command
     - The last part of a statement is a terminating semi-colon;
3. Command block
   - A Command block is a group of statements that is treated as a single entity and are grouped within braces - the curly brackets - {  }

# Basic JavaScript Rules

- Matching Pairs
  - Opening and closing symbols need to work in pairs. For example, if you use the left brace { to indicate the start of a command block, then you must use the right brace } to end the command block. The same matching pairs applies to single '.....' and double "....." quotes to designate text strings.
- The use of white Space
  - JavaScript ignores extras spaces however it is recommended that you use them to make your scripts easier to read.
- The use of comments
  - While comments are not required as part of the JavaScript language, it is recommended that you include comments in your scripts.
  - JavaScript uses the symbols /* to designate a comment */
  - JavaScript also uses the // for short comments.

# JavaScript data types

- There are 3 main (primitive) data types:
  - string
    - must be enclosed in single or double quotes
  - number
    - can be integers or floating point
    - Special number: Infinity, NaN
  - boolean
    - values are binary, with the values (1) "true" and (0) "false" (without the quotes)
- Other types:
  - undefined, null, object, function

# JavaScript data types

- JavaScript is a loosely typed language.
  - You do not have to specify the data type of a variable when you declare it.
  - Data types are converted automatically as needed during script execution.

# JavaScript Variable

- Variable naming rules are: Must start with a letter, underscore (_), or dollar sign ($)
- Cannot be a reserved (key) word
- Subsequent characters can be letters
  - upper case (A...Z) or lower case (a...z),
  - numbers
  - underscores
- JavaScript reserved words
  - Similar to other programming languages, JavaScript has a list of words that are considered "reserved".

# Declare and Refer Variables

- You must use the "var" keyword to precede a variable name.
- Unlike the C language, you do not need a type specifier.
  - The variable's initial value will set its initial type.
- Declaration syntax:

```
var variableName;
```

Or:

```
var variableName = "Summer";

// Referring to and using syntax:
variableName = 2015;
alert(variableName);
```

- Dynamic typing
  - a JavaScript variable can have a different type in different parts of a program

# Variables Example

| Declaration | Type | Value |
|---|---|---|
| var identOne = "some text"; | String | some text |
| var identOne = 'some text'; | String | some text |
| var IdentOne = '172'; | String | 172 |
| var _identOne = 25; | Number (Integer) | 25 |
| var _identTwo = 56.2564; | Number (float) | 56.2564 |
| var ident_A = true; | Boolean | true (1) |
| var ident_B = false; | Boolean | false (0) |
| var ident_C; | undefined | undefined |
| var ident_D="Yes",<br>ident_E="No"; | String / String | Yes / No |

# Special values

- Infinity
  - Number Data Type
  - e.g. alert(12/0);
- NaN
  - Number Data Type
- null
  - both a value and a data type
- undefined
  - both a value and a data type
  - e.g. var x;
    alert(x);

# Expressions

- An expression is any valid set of literals, variables, operators, and expressions that evaluates to a single value.
- The value may be a number, a string, or a logical value.
- Two types of expressions:

  1) those that assign a value to a variable, e.g. *x = 7* .

  2) those that simply have a value, e.g., *3 + 4* simply evaluates to 7; it does not perform an assignment.
- JavaScript has the following kinds of expressions:

  1) Arithmetic - evaluates to a *number*
  2) String - evaluates to a character *string*
  3) Logical - evaluates to *true or false*

# Conditional Expression

- A conditional expression can have one of two values based on a condition. The syntax:

(condition) ? val1 : val2;

- If the condition is true, the expression has the value of val1, Otherwise it has the value of val2.

| condition | When True | When False |
|---|---|---|

```
var status = (age >= 18) ? "adult" : "minor";
```

# Arithmetic Operators

| Operator | Operation | Example |
|---|---|---|
| + | addition of numbers<br>Concatenation of strings | y + x;<br>"INT" + "222" |
| - | subtraction | x - y; |
| * | multiplication | x * y; |
| / | division | x / y; |
| % | modulo | x % y; // remainder of x divided by y |
| ++ | post/pre -increment | x = y++; // assign y to x, then increment y (y+=1)<br>x = ++y; //increment y < (y+=1), then assign y to x |
| -- | post/pre decrement | x = y--; // assign y to x, then decrement y (y-=1)<br>x = --y; // decrement y < (y-=1), then assign y to x |

# Assigning Values

| Operator | Example | Equivalent | For |
|---|---|---|---|
| = | a = b; | a = b; | Numbers, strings, … |
| += | a += b; | a = ( a + b); | numbers or strings |
| -= | a -= b; | a = ( a - b); | numbers |
| *= | a *= b; | a = ( a * b); | numbers |
| /= | a /= b; | a = ( a / b); | numbers |
| %= | a %= b; | a = ( a % b); // divide a by b, // assign remainder to a | numbers |

# Logical Operators

- Given x = 2;

| Operator | Operation | Example |
|---|---|---|
| && | Logical AND | (x > 3 && x == 2) is false |
| \|\| | Logical OR | (true \|\| x > 10 ) is true |
| ! | Logical NOT | !(x ==2) is false |

# Other Operators

- The typeof operator (for variable or values):
  - possible return values:

```
typeof "John"                 // Returns string
typeof 3.14                   // Returns number
typeof false                  // Returns boolean
typeof [1,2,3,4]              // Returns object
typeof {name:'John', age:34}  // Returns object
```

- The instanceof operator
  - Used for objects

# Comparison Operators

| Operator | Description | Example |
|---|---|---|
| == | Equal<br>(The operands are converted to the same type before being compared.) | 1 == 1 is true<br>1 == "1" is true<br>1 == true is true<br>0 == false is true |
| === | strictly equal<br>(There is no type conversion.) | 1 === 1 is true<br>1 === "1" is false<br>1 === true is false<br>0 === false is false |
| != | not equal<br>(with type conversion) | 1 != 1 is false<br>1 != '1' is false |
| !== | not equal<br>(without type conversion) | 1 !== 1 is false<br>1 !== '1' is true |
| > | greater than | expr1 > expr2 |
| >= | greater than or equal to | expr1 >= expr2 |
| < | less than | expr1 < expr2 |
| <= | less than or equal to | expr1 <= expr2 |

# Strings and Quotation Marks

- Literal strings can be denoted by either single or double quotes, which gives you some flexibility about how to handle awkward situations such as quotation marks inside a string:

| Expression | Values |
|---|---|
| "Let's start with JavaScript" | Let's start with JavaScript |
| 'Not "it"!' | Not "it"! |

# Concatenation of Strings

- The main operation on strings is the concatenation operator, +:

| Expression | Value |
|---|---|
| “BTI" + "220" | BTI220 |
| "Stephen" + " Harper" | Stephen Harper |

# Adding Strings and Numbers

- x =5+5;
  alert(x);
- x="5"+"5";
  alert(x);
- x=5+"5";
  alert(x);
- x="5"+5;
  alert(x);

# Example - Evaluating Expressions

```
var x = prompt("Enter a number.");
x = x + 2;
alert ("The value of x is " + x);

x = 2 * x;
alert("The new value of x is now " + x);

x = x + 1;
alert("x is now " + x);

alert("x divided by 3 is: " + x/3);
```

evaluating-arithmetic-expressions.js

# Programming Constructs

- JavaScript execution flow is determined using the following four (4) basic control structures:
  1. Sequential:
     an instruction is executed when the previous one is finished.
  2. Conditional
     a logical condition is used to determine which instruction will be executed next - similar to the "if" and "switch" statements in C.
  3. Looping
     a series of instructions are repeatedly executed until some condition is satisfied - similar to the "for" and "while" statements in C.
  4. Transfer
     jump to a different part of the code - similar to calling a function in C.
- Example: programming-constructs.js

# Programming Constructs (1) – Sequence

```
var a = 3;
var b = 6;
var c = a + b;
alert(c);
```

# Programming Constructs (2) – Selection

- Make decisions and perform single or multiple tasks based on the outcome of the decision (true or false).
- Types of conditional statements :
  - if
  - if / else
  - switch / case

# Conditional Statements

- Conditional Statements give JavaScript scripts the ability to make decisions and perform single or multiple tasks based on the outcome of the decision (true or false).
- **The if-else**
  - expression / condition is in parentheses (expression)
  - relational operators include: == != > < >= <=
  - && (and) and || (or) can be used to create compound conditions
  - ! (not) can be used to invert a condition
  - the else clause is optional
  - if statements may be nested
  - multiple action statements must be enclosed in brace brackets { }

# The General Format for an If Statement

The general format for an if / else statement is as follows:

```
if (expression) {
  statement;
}

else {
  statement;
}
```

```
if (expression) {
  statement;
  statement;
  statement;
}
else {
  statement;
  statement;
}
```

# The general format for an **if / else / if** statement:

```
if (exp-a) {
   statement1;  // if exp-a is True
}
else {
   if (exp-b) {
      statement2;   // if exp-b is True
   }
   else {
      statement3;   // if exp-b is False
   }
}
```

```
// equivalent to following:

if (exp-a) {
   statement1; // if exp-a is True
}
  else if (exp-b) {
   statement2; // if exp-b is True
}
else {
   statement3; // if exp-b is False
}
```

# Example

```
var grade, mark=prompt("Enter your mark:");

if (mark >= 90)
    grade='A+';
else if (mark >= 80)
    grade='A';
else if (mark >= 70)
    grade='B';
else if (mark >= 60)
    grade='C';
else if (mark >= 50)
    grade='D';
else
    grade="F";

alert("Your grade: " + grade);
```

# Conditional Statement

- The switch / case statement:
  - select one of many blocks of code to be executed.

```
switch (expression)
{
   case  401:
            statement1;
            break;
   case  403:
            statement2;
            break;
   case  407:
            statement3;
            break;
   default:
            statement4;
}
```

# Example

```
var semester = prompt("Enter CPD semester number (1 to 4)");

switch (semester) {
        case '1':   alert("IPC144, ULI101");
                    break;
        case '2':   alert("OOP244, INT222");
                    break;
        case '3':   alert("OOP344, INT322");
                    break;
        case '4':   alert("JAC444, INT422");
                    break;
        default:
                    alert("You may have graduated from CPD");
}
```

# Programming Constructs (3) – Iteration

- loop - an action that occurs again and again until a certain condition is met.
- Continuously check a condition and based on the outcome, either terminate the loop or repeat a set of statements.
- Three basic types of loop structures:
  - The for loop
  - The for / in loop
  - The while loop
  - The do-while loop

# Loop Structures

1. The for loop
   - e.g.

```
var days = "The days in september: \n";
for (var ident = 1 ; ident <= 30 ; ident++) {
        days += ident + "\n";
}
alert(days);
```

# Loop Structures

2. The for in loop
   - Iterates over the enumerable properties of an object, in arbitrary order. For each distinct property, statements can be executed.

```
var student = {name:"John", program:"CPD", semester:2};
var str = "Student info:\n\n";
for (var x in student) {
    str += x + ": " + student[x] + "\n";
}
alert(str);
```

# Loop Structures

3. The example of while loop

```
var text = "";
var i = 0;
while (i < 10) {
    text += "\nThe number is " + i++;
}
console.log(text);
```

4. The example of do...while loop

```
var i=10;
do {
    alert("week " + i++);
} while (i<15)
```

# Break Statements

- Break the loop and continue executing the code that follows after the loop (if any).

```
var i=1;
while (i<5)    {
  alert("week "+i);
  if (i==3)
    break;
  else
    i++;
}
```

- Break the current loop and continue with the next value.  (nested loops).

```
var i=1, j=1;
while (i<5) {
   alert('week: '+i );
   for (j=1; j<=7; j++){
      alert('day:'+ j +'of week:'+ i);
      if (j==3)  break;
   } // for
   i++;
} // while
```

# Resourceful Links

- Mozilla Developer Network (MDN) start page
  by the Mozilla Developer Network and individual contributors
  http://developer.mozilla.org
- Web Education Community Group Wiki
  by the W3C Web Education Community Group
  http://www.w3.org/community/webed/wiki/Main_Page
- Your Web, Documented
  by the W3C and the Web Platform stewards
  http://www.webplatform.org/

# Thank You!