

WEB222 - Web Programming Principles

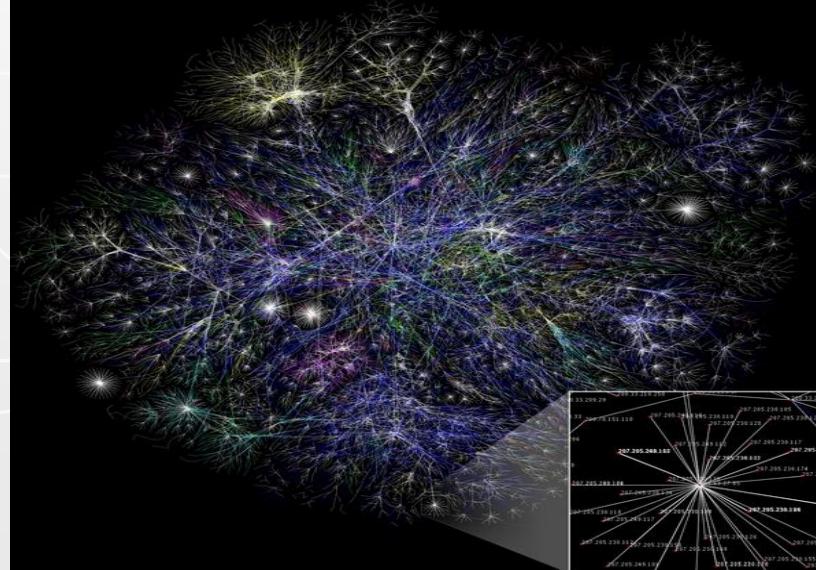
**Week 1: Internet Architecture and
Introduction to JavaScript**

Agenda

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

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.



Internet Protocol Suite

- Communications protocol
 - A communication protocol is a system of rules for exchanging message over the Internet.
- Internet Protocol Layers
 - Application layer
 - ▶ **HTTP** - HyperText Transfer Protocol
 - ▶ **SSH** - Secure Shell
 - ▶ **SFTP** - Secure File Transfer
 - ▶ **DNS** - Domain Name System
 - ▶ **SMTP** - Simple Mail Transfer Protocol
 - ▶
 - Transport layer
 - ▶ **TCP** (Transmission Control Protocol),
 - Network layer : **IP** (Internet Protocol)
 - Physical and data link layers: Ethernet, token ring

Services Provided by the Internet

➤ World Wide Web

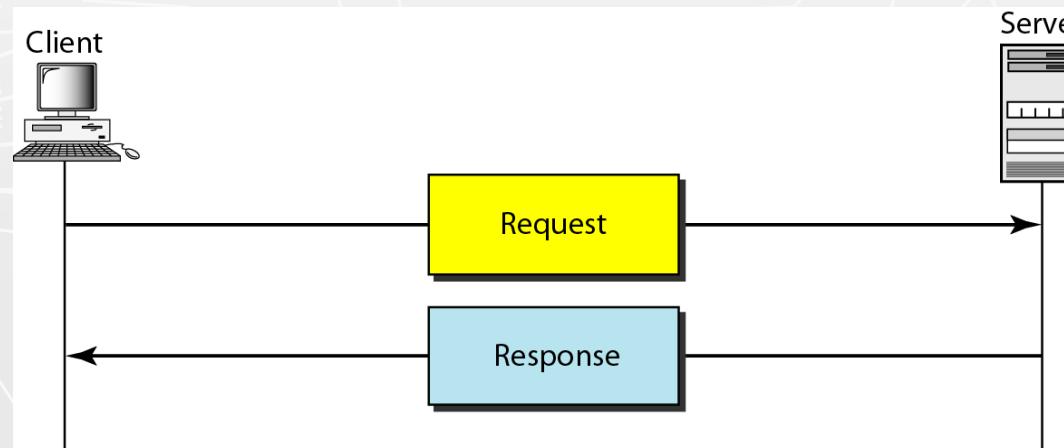
- Abbreviation: **WWW** or **W3**,
- Commonly known as **the Web**.
- It is a collection of web pages connected through **hyperlinks** and **URLs**.
- With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks.
- It 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:

- 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

HTML URL Encoding

- URL encoding converts characters into a format that can be transmitted over the Internet.
 - URLs can only be sent over the Internet using the ASCII character-set.
 - URLs often contain characters outside the ASCII set
 - ▶ has to be converted into a valid ASCII format.
 - URLs cannot contain spaces.
- Example: https://www.youtube.com/results?search_query=web222&feature=related
- URL encoding replaces unsafe ASCII characters with a "%" followed by two hexadecimal digits. e.g.

'&' → %26

' ' → %20

'<' → %3C

'>' → %3E

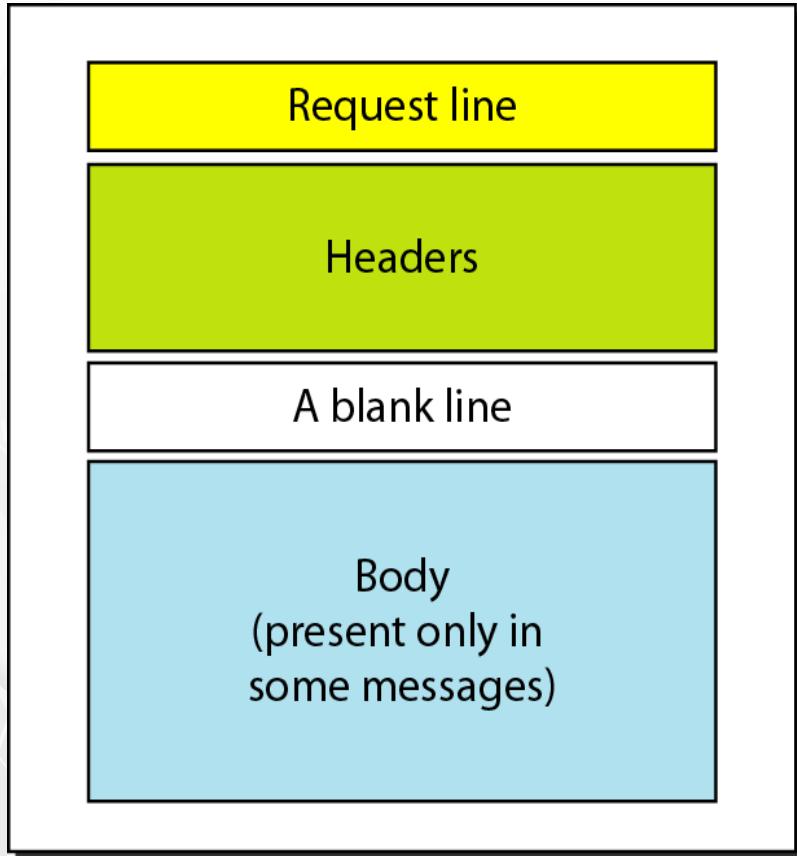
DNS (Domain Name System/Server)

- 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.

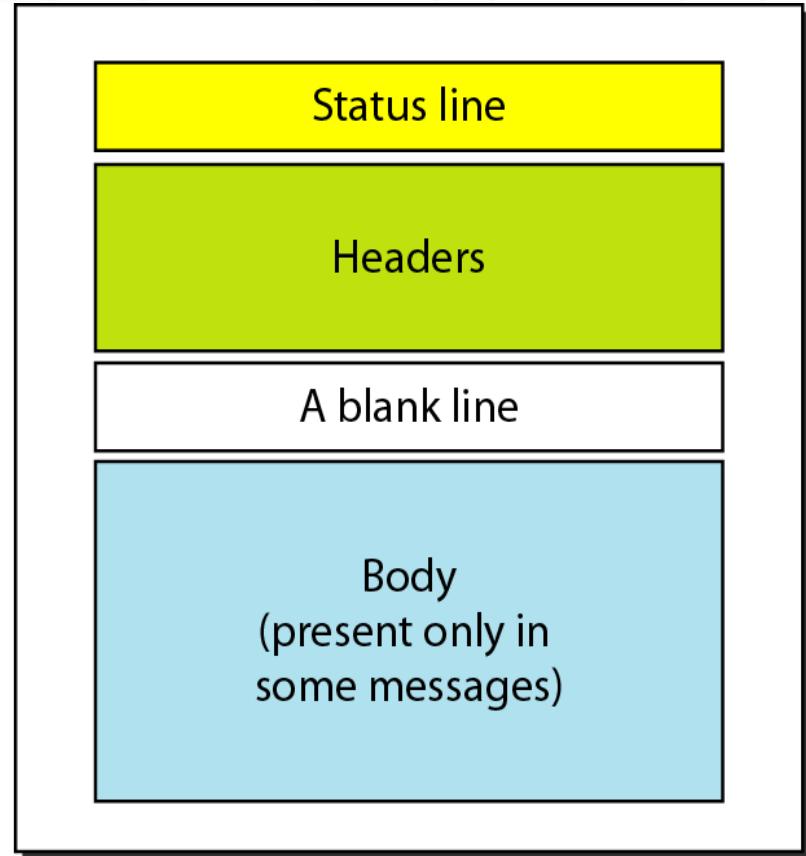
Hypertext Transfer Protocol

- **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.
- HTTP is **stateless**.
 - The server doesn't keep any data (state) between two requests.

HTTP Request and Response Messages



Request message



Response message

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.

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 "**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 appearance 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

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.
Keyboard shortcut: **Shift+F4**
 - For Mac users: look for the "Web Developer" menu under "Tools".
Shortcut: **fn+Shift+F4** (The "fn" key is just to the right of the main set of QWERTY keys)
- More on Developer tools:
 - [Firefox developer tools](#)
 - [Chrome developer tools](#)
 - [JSLint – validator & error checker](#)

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

console.log()

- The console.log function to show a messages in a web console.

- Example code:

```
var anObject = { str: "Some text", id: 5 };  
console.log(anObject);
```

- open web console in Firefox:

- **Ctrl + Shift + k**
- For Mac users: **Cmd + Opt + k / Cmd + Opt + I**

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) { // if cl is not null  
    console.log (cl);  
} else { // cl is null  
    console.log ("No color entered.");  
}
```

- Note: you should not use this function in a web app.

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).
- The JavaScript standard is based on the European Computer Manufacturers Association (**ECMAScript**). As of 2012, all modern browsers fully support ECMAScript 5.1.

Introduction to JavaScript (cont'd)

- 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.
- 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 may be, in the future, the most important language you will learn.

Basic JavaScript Rules

- **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
- **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.
- **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 comments

- Block/Multi-line comment: `/* */`
- Single line comments: `//`

➤ The use of white Space

- JavaScript ignores extras spaces however it is recommended that you use them to make your scripts easier to read.

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;  
console.log(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", ident_E="No";	String / String	Yes / No

Special values

- Infinity
 - Number data type
 - e.g. `console.log(12/0);`
- NaN
 - means "**Not a Number**"; Number data type
- null
 - both a value and a data type
- undefined
 - both a value and a data type
 - e.g. `var x;`
 - `console.log(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

Expressions - Ternary Operator

- 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 Concatenation of strings	$y + x;$ "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 ($y+=1$) $x = ++y;$ //increment y < ($y+=1$), then assign y to x
--	post/pre decrement	$x = y--;$ // assign y to x, then decrement y ($y-=1$) $x = --y;$ // decrement y < ($y-=1$), then assign y to x

Arithmetic Operators - 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
<code>&&</code>	Logical AND	$(x > 3 \&& x == 2)$ is false
<code> </code>	Logical OR	$(\text{true} x > 10)$ is true
<code>!</code>	Logical NOT	$!(x == 2)$ is false

Comparison Operators

Operator	Description	Example
<code>==</code>	Equal (The operands are converted to the same type before being compared.)	<code>1 == 1</code> is true <code>1 == "1"</code> is true <code>1 == true</code> is true <code>0 == false</code> is true
<code>===</code>	Strictly equal (There is no type conversion.)	<code>1 === 1</code> is true <code>1 === "1"</code> is false <code>1 === true</code> is false <code>0 === false</code> is false
<code>!=</code>	Not equal (with type conversion)	<code>1 != 1</code> is false <code>1 != '1'</code> is false
<code>!==</code>	Not equal (without type conversion)	<code>1 !== 1</code> is false <code>1 !== '1'</code> is true
<code>></code>	Greater than	<code>expr1 > expr2</code>
<code>>=</code>	greater than or equal to	<code>expr1 >= expr2</code>
<code><</code>	Less than	<code>expr1 < expr2</code>
<code><=</code>	less than or equal to	<code>expr1 <= expr2</code>

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

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
"WEB" + "222"	WEB222
"John" + " Smith"	John Smith

Adding Strings and Numbers

- `x =5+5;
console.log(x); // Output: 10`
- `x="5"+"5";
console.log(x); // Output: 55`
- `x=5+"5";
console.log(x); // Output: 55`
- `x="5"+5;
console.log(x); // Output: 55`

Example - Evaluating Expressions

```
var x = 5;  
x = x + "2";  
console.log("The value of x is " + x);
```

```
x = 2 * x;  
console.log("The new value of x is now " + x);
```

```
x = x + 1;  
console.log("x is now " + x);  
console.log("x divided by 3 is: " + x/3);
```

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.

Construct (1) - Sequence

- Tasks are executed one after another in “sequence” – e.g.

```
var a = 3;  
var b = 6;  
var c = a + b;
```

```
console.log(c);
```

Construct (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

if-else Example

```
var grade, mark = 86;
```

```
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";
```

```
console.log( "Your grade: " + grade);
```

Switch-case Example

```
var semester = 3;
```

```
switch (semester) {
    case '1': console.log("IPC144, ULI101");
        break;
    case '2': console.log("OOP244, WEB222");
        break;
    case '3': console.log("OOP344, INT322");
        break;
    case '4': console.log("JAC444, INT422");
        break;
    default:
        console.log("You may have graduated from CPD");
}
```

Construct (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

for loop Example

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

for in loop Example

- 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) { // x is the current property ('key') – ie:  
  name, program, or semester  
  str += x + ": " + student[x] + "\n";  
}  
console.log(str);
```

while & do...while loop Examples

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

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

break and continue Statements

- **break**: breaks the loop and continue executing the code that follows after the loop (if any).
- **continue**: breaks one iteration (in the loop), and continues with the next iteration in the loop.

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

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

Thank you!

Any Questions?