COSC 122
Computer Fluency

Programming Basics

Dr. Ramon Lawrence
University of British Columbia Okanagan
ramon.lawrence@ubc.ca

Key Points

1) We will learn JavaScript to write instructions for the computer.
◆ The fundamental programming concepts apply to all languages.

2) The key programming concepts covered:
◆ variables, values, and locations
◆ initialization and assignment
◆ expressions
◆ decisions and Boolean conditions

History: The First Programmers

Did you know that the first programmers were almost all women?
◆ Women worked on the first computer - the ENIAC (Electronic Numerical Integrator and Calculator) developed for the US Army in 1946 by J. Eckert and John Mauchley.

◆ These women were recruited from the ranks of "computers", humans that used mechanical calculators to solve complex math problems before the invention of computers.

◆ These pioneer programmers laid the foundation of many of the original ideas including compilers and programming languages.

Introduction to Programming

Remember that an **algorithm** is a precise sequence of steps to produce a result. A **program** is an encoding of an algorithm in a **language** to solve a particular problem.

There are numerous languages that programmers can use to specify instructions. Each language has its different features, benefits, and usefulness.

The language we will use is called JavaScript. However, our focus will be understanding the primary programming concepts that apply to all languages.

Introduction to JavaScript

JavaScript is a *scripting* language used primarily for web pages.
◆ JavaScript was developed in 1995 and released in the Netscape web browser (since renamed to Mozilla Firefox).
◆ JavaScript is standardized and supported by most browsers.

Despite the name, JavaScript is not related to Java, although its syntax is similar to other languages like C, C++, and Java.
◆ There are some major differences between JavaScript and Java that will not concern us here.
◆ Aside: The term **scripting** means the language is interpreted (processed when needed) instead of compiled (converted to machine language directly). The difference is irrelevant to us.

Some Quotes

If you can't write it down in English, you can't code it.
-- Peter Halpern

If you lie to the computer, it will get you.
-- Peter Farrar

JavaScript: Basic Rules

To program in JavaScript you must follow a set of rules for specifying your commands. This set of rules is called a *syntax*.

◆ Just like any other language, there are rules that you must follow if you are to communicate correctly and precisely.

Important general rules of JavaScript syntax:

◆ JavaScript is *case-sensitive*.
  ⇨ Main() is not the same as main() or MAIN()
◆ JavaScript accepts *free-form layout*.
  ⇨ Spaces and line breaks are not important except to separate words.
  ⇨ You can have as many words as you want on each line or spread them across multiple lines.
  ⇨ However, you should be consistent and make your code easy to read.

Our Running Example
Do you want fries with that?

We will use an example program for our discussion that calculates the total cost of a fast food order.
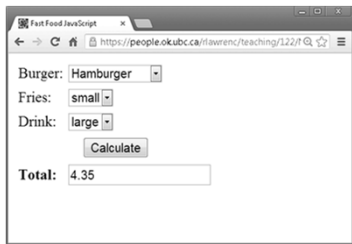
Inputs:
◆ burger – may be "none", "hamburger", or "cheeseburger"
◆ fries – may be "none", "small", or "large"
◆ drink – may be "none", "small", or "large"

Output:
◆ the total in dollars of the order including tax (7%)

Fast Food Example

Fast Food Example Code

```
var total;
var taxRate = 0.07;            } Declare and initialize variables

total = 0;

if (burger == "hamburger" || burger == "cheeseburger")
   total = 0.99;   ← Assignment
                                    Decision using IF
if (fries == "small")
   total = total + 1.19;
if (fries == "large")
   total = total + 1.79;

if (drink == "small")
   total = total + 1.49;
else if (drink == "large")
   total = total + 1.89;

total = total + total * taxRate;
```

Flow of Execution
-Start at first statement at top and proceed down executing each statement.
-For if statement only execute one of the possibilities if condition is true otherwise go to next statement after if.

Expression

Values, Variables, and Locations

A *value* is a data item that is manipulated by the computer.

A *variable* is the name that the programmer uses to refer to a location in memory.

A *location* has an address in memory and stores a value.
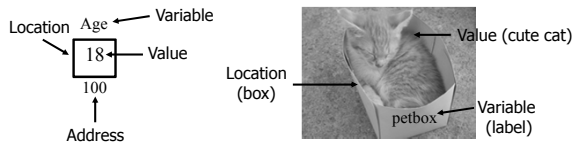
IMPORTANT: The *value* at a given location in memory (named using a variable name) can change using initialization or assignment.

Values, Variables, and Locations Example

We want to store a number that represents the total order value.

Step #1: Declare the variable by giving it a name

```
var total;
```

◆ The computer allocates space for the variable in memory (at some memory address). Every time we give the name total, the computer knows what data item we mean.

Variable Name Lookup Table

Name	Location	Type
total	16	number

Memory

16	????????
20	
24	
28	

Values, Variables, and Locations Example (2)

Step #2: Initialize the variable to have a starting value

◆ If you do not initialize your variable to a starting value when you first declare it, the value of the variable is **undefined**.

Example:
```
total = 1;
```

Variable Name Lookup Table

Name	Location	Type
total	16	number

Memory

16	1
20	
24	
28	

Values, Variables, and Locations Example (3)

Step #3: Value stored in location can be changed throughout the program to whatever we want using **assignment** ("**=**" symbol).

```
total = total * 5 + 20;
```

Variable Name Lookup Table

Name	Location	Type
total	16	number

Memory

16	25
20	
24	
28	

Variable Rules

Variables are also called identifiers. An **identifier** is a name that **must begin with a letter** and cannot contain spaces.

The keyword **var** is used to declare to the computer that you want a variable created. This declaration is a type of **statement**.

Rules:

◆ Every variable used in a program must be declared.

◆ Variables can be declared anywhere in the program, but usually should be declared right at the start.

◆ Variable names **ARE** case-sensitive. Numbers are allowed (but not at the start). Only other symbol allowed is underscore ('_');

◆ Beware of declaring two variables with the same name.

◆ The syntax of the language allows you to declare and initialize multiple variables in the same statement:
```
var total = 0, taxRate = 0.07;
```

Aside: Good Variable Names

As a programmer you have flexibility on the names that you assign to your variables.

◆ However, names should be meaningful and explain how the variable is actually used in your program.

Example:
```
var t = 0;
var total = 0;
```

Avoid naming variables as reserved words. A **reserved word** is a string that has special meaning in the language.

◆ e.g. `if`, `var`, `else`

Variables – Basic Terminology

Question: Of the following three terms, what is most like a **box**?

A) value

B) variable

C) location

Variables - Definitions

Question: Which of the following statements is correct?

A) The location of a variable may change during the program.

B) The name of a variable may change during the program.

C) The value of a variable may change during the program.

Variables – Correct Variable Name

Question: Which of the following is a valid JavaScript variable?

A) `aBCde123`

B) `123test`

C) `t_e_s_t!`

General Syntax Rules

A program is a list of statements (instructions).

PRIMARY RULE: Every statement must be terminated by a semi-colon ";".
◆Note the statement terminator character varies by language.

Other rules:
◆You may have multiple statements on a line as long as each ends with a semi-colon.
◆A statement may cross multiple lines.

General Syntax Rules: Comments

Comments are used by the programmer to document and explain the code. Comments are ignored by the computer.

There are two choices for commenting:
◆1) One line comment: put "`//`" before the comment and any characters to the end of line are ignored by the computer.
◆2) Multiple line comment: put "`/*`" at the start of the comment and "`*/`" at the end of the comment. The computer ignores everything between the start and end comment indicators.

Example:
```
/* This is a multiple line
      comment.
With many lines. */

// Single line comment
// Single line comment again
d = 5.0; // Comment after code
```

Variable Types

A variable has a **name** for a data item and a **type**.
◆JavaScript is different than most languages because you do not have to tell the computer what type the variable is when you declare it. The variable can store any type (although it is not recommended to change types).

The data types that we will use are:
◆numbers – both integers and float/doubles
◆strings – sequences of characters
◆Boolean – true or false

Strings

Strings are sequences of characters that are surrounded by either single or double quotes.

Example:
```
var personName = "Ramon Lawrence";
personName = "Joe Smith";
```

Question: What is the difference between these two statements?

Rules for Strings in JavaScript

String rules:
◆Must be surrounded by single or double quotes.
◆Can contain most characters except enter, backspace, tab, and backslash.
  ⇨These special characters must be escaped by using an initial "`\`".
  ⇨e.g. `\n` – new line, `\'` – single quote, `\\` - backslash, `\"` – double quote
◆Double quoted strings can contain single quoted strings and vice versa.
◆Any number of characters is allowed.
◆The minimum number of characters is zero `""`, which is called the *empty string*.
◆String *literals* (values) have the quotation marks removed when displayed.

Practice Questions

1) Write the statements to create two variables: one called `hourlyRate` and the other called `hoursWorked`.

2) Are the following variable names valid or invalid:
```
var A;
var A123;
var 123A;
var aReallyLongName;
```

3) Using your statements from question #1, write the code to calculate and store a person's salary by multiplying their `hoursWorked` times their `hourlyRate`.

4) Create a string variable that has an initial value of `'Joe's Place'`.

The Assignment Statement

An **assignment statement** changes the value of a variable.
- ⇨ The variable on the left-hand side of the = is assigned the value from the right-hand side.
- ⇨ The value may be changed to a constant, to the result of an expression, or to be the same as another variable.
- ⇨ The values of any variables used in the expression are always their values before the start of the execution of the assignment.

Examples:
```
var A, B;

A = 5;
B = 10;
A = 10 + 6 / 2;
B = A;
A = 2*B + A - 5;
```

Question: What are the values of A and B?

Expressions

An **expression** is a sequence of operands and operators that yield a result. An expression contains:
- ◆ **operands** - the data items being manipulated in the calculation
  - ⇨ e.g. 5, "Hello, World", myDouble
- ◆ **operators** - the operations performed on the operands
  - ⇨ e.g. +, -, /, *, % (modulus - remainder after integer division)

An operator can be:
- ◆ **unary** - applies to only one operand
  - ⇨ e.g. d = - 3.5;   // "-" is a unary operator, 3.5 is the operand
- ◆ **binary** - applies to two operands
  - ⇨ e.g d = e * 5.0;   // "*" is binary operator, e and 5.0 are operands

---

Expressions - Operator Precedence

Each operator has its own priority similar to their priority in regular math expressions:
- ◆ 1) Any expression in parentheses is evaluated first starting with the inner most nesting of parentheses.
- ◆ 2) Unary + and unary - have the next highest priorities.
- ◆ 3) Multiplication and division (*, /, %) are next.
- ◆ 4) Addition and subtraction (+,-) are then evaluated.

String Operators: Concatenation

The **concatenation operator** is used to combine two strings into a single string. The notation is a plus sign '+'.

Example:
```
var string1 = "Hello";
var string2 = " World!";
var result = string1 + string2; //result = "Hello World!"
```

The plus sign is used for addition, but it makes sense as the symbol for string concatenation as well.

Using the same symbol as a operator in multiple different ways is called **operator overloading**.

Assignment

Question: What are the values of A and B after this code?
```
var A, B;

A = 2;
B = 4;
A = B + B / A;
B = A * 5 + 3 * 2;
```

A) A = 6, B = 36

B) A = 4, B = 26

C) A = 6, B = 66

String Concatentation

Question: What is the value of result after this code?

```
var st1="Joe", st2="Smith";

var result = st1 + st2;
```

A) `"Joe Smith"`

B) `"JoeSmith"`

String Concatentation (2)

Question: What is the result after this code?

```
var st1="123", st2="456";

var result = st1 + st2;
```

A) `579`

B) `"579"`

C) `"123456"`

Running a JavaScript Program

We run JavaScript programs within a web browser.
This means several things:

◆1) The file that stores the program will be an HTML document. It should have a name like `myProgram.html`.

◆2) The JavaScript program is part of the HTML file.

◆3) Edit the document using a text editor. Test the document by opening it in Internet Explorer, FireFox, Chrome, or Safari.

Hello World Example
JavaScript Code

```
<html>
<head>
<title>HelloWorld using JavaScript</title>
</head>

<body>
<h1>

   <script type="text/javascript">
     document.write("Hello, world!");
   </script>
</h1>

</body>
</html>
```

`<script>` tag indicating code

JavaScript code

document is HTML document
document.write() puts that text into the document at this location

Getting Input into a JavaScript Program

There are two ways to get input from the user into your program:

◆1) Make the user fill in form fields and get the value of those fields when the user clicks a button.
  ⇒We will see how to do this later.

◆2) Prompt the user with a separate window asking them for a value.

Getting Input Using JavaScript Code

```
<html>
<head>
<title>Prompt for a Value using JavaScript</title>
</head>

<body>
<h1>
   <script type="text/javascript">
     var val = window.prompt("Enter a value: ");
     document.write(val);
   </script>
</h1>
</body>
</html>
```

Prompt for value from user

write out value retrieved

Outputting from a JavaScript Program

There are three ways to output information to the user:
◆1) Have your code set the value of a form field.

◆2) Have your code write out text directly into the HTML document.

◆3) Open an alert output window to the user with a message.

Outputting Data from JavaScript Code

```
<html>
<head>
<title>Display a Value using an Alert Window</title>
</head>

<body>
    <script type="text/javascript">
        var val = window.prompt("Enter a value: ");
        window.alert("You said: "+val);
    </script>
</body>
</html>
```

Prompt for value from user

Open up new window with message and value that the user just entered.

Prompt and Output Example

Prompt window:

Alert (output) window:

Input/Output Question

Question: Assume the user typed in **10** when prompted. What is shown in the HTML document after this code?

```
var val = window.prompt("Enter a value: ");
window.alert("You said: "+val);
```

A) Nothing

B) You said: 10

C) Error

Input/Output Question (2)

Question: Assume the user typed in **10** when prompted. What is shown in the HTML document after this code?

```
var val;
window.prompt("Enter a value: ");
document.write("You said: "+val);
```

A) Nothing

B) You said: 10

C) You said: undefined

D) Error

Practice Questions

For these questions, use slide #33 as an example. Do not copy the HTML code, just write the JavaScript statements.
1) Write the JavaScript code to print:
   Hello, World!
   Goodbye, World!
2) Write the JavaScript code to print:
   Testing…
   1..2..3..
   1+2+3 = 6
   1*2*3 = 6
Note: You must calculate 6 in both cases not just print it!
3) Write a program to calculate and print: (a=5, b=10)
   c = 25*a+b-32

Making Decisions

Decisions allow the program to perform different actions in certain conditions.

◆ For example, if a person applies for a driver's license and is not 16, then the computer should not give them a license.

To make a decision in a program we must:

◆ 1) Determine the *condition* in which to make the decision.

⇨ In the license example, we will not give a license if the person is under 16.

◆ 2) Tell the computer what actions to take if the condition is true or false.

⇨ A decision always has a *Boolean* or true/false answer.

The syntax for a decision uses the *if* statement.

Making Decisions
Performing Comparisons

A *comparison operator* compares two values. Examples:

◆ 5 < 10

◆ N > 5   // N is a variable. Answer depends on what is N.

Comparison operators in JavaScript:

◆ >   - Greater than
◆ >=   - Greater than or equal
◆ <   - Less than
◆ <=   - Less than or equal
◆ ==   - Equal (Note: Not "=" which is used for assignment!)
◆ !=   - Not equal

The result of a comparison is a *Boolean value* which is either *true* or *false*.

Making Decisions
Example Comparisons

```
var j=25, k = 45;
var d = 2.5, e=2.51;

// Determine if these comparisons are true or false

(j == k)          // false
(j <= k);         // true
(d == e);         // ??
(d != e);         // ??
(k >= 25);        // ??
(25 == j);        // ??
(j > k);          // ??
(e < d);          // ??

j = k;
(j == k);         // ??
```

Valid Comparison Operators Question

Question: Select the operator that is invalid (not allowed).

A) !=

B) ==

C) <=

D) ≥

Making Decisions
If Statement

To make decisions with conditions, we use the *if* statement.

◆ If the condition is true, the statement(s) after `if` are executed otherwise they are skipped.

◆ If there is an `else` clause, statements after `else` are executed if the condition is false.

Syntax:

```
if (condition)        OR        if (condition)
   statement;                      statement;
                                else
                                   statement;
```

Example:

```
if (age > 17)         OR        if (age > 17)
   alert("Adult!");                alert("Adult!");
                                else
                                   alert("Kid!");
```

Making Decisions
Block Syntax

Currently, using our if statement we are only allowed to execute one line of code (one statement).

◆ What happens if we want to have more than one statement?

We use the *block syntax* for denoting a multiple statement block. A block is started with a "**{**" and ended with a "**}**".

◆ All statements inside the brackets are grouped together.

Example:

```
if (age > 17)
{  window.alert("You are an adult");
   window.alert("You can vote!");
   ...
}
```

We will use block statements in many other situations as well.

Making Decisions
If Statement Example

```
var age;
var teenager, hasLicense;
age = window.prompt("Enter your age: ");

if (age > 19)
{  teenager = false;
   hasLicense = true;
}
else if (age < 13)
{  teenager = false;
   hasLicense = false;
}
else
{  teenager = true;   // Do not know if have license
   hasLicense = false;
}
document.write("Is teenager: "+teenager);
document.write("Has license? "+hasLicense);
```

Making Decisions

Question: What is the output of this code?

```
var num=10;

if (num > 10)
    document.write("big");
else
    document.write("small");
```

A) big

B) small

C) bigsmall

Making Decisions (2)

Question: What is the output of this code?

```
var num=10;

if (num != 10)
    document.write("big");
document.write("small");
```

A) big

B) small

C) bigsmall

Making Decisions (3)

Question: What is the output of this code?

```
var num=10;

if (num == 10)
{  document.write("big");
   document.write("small");
}
```

A) big

B) small

C) bigsmall

Decision Practice Questions

1) Write a program that reads an integer *N*.
◆If *N* < 0, print "Negative number", if *N* = 0, print "Zero", If *N* > 0, print "Positive Number".

2) Write a program that reads in a number for 1 to 5 and prints the English word for the number. For example, 1 is "one".

3) Write a program to read in your name and age and print them.
◆a) Modify your program to also print "Not a teenager" if your age is greater than 19 otherwise print "Still a teenager".

Nested Conditions and Decisions
Nested If Statement

We ***nest*** if statements for more complicated decisions.
◆Verify that you use blocks appropriately to group your code!

Example:

```
if (age > 16)
{  if (gender == "male")
   {  document.write("Fast driver!");
   }
   else
   {  document.write("Great driver!");
   }
}
else
{  document.write("Sorry! Too young to drive.");
}
```

Making Decisions
Nested If Statement Example

```
var salary, tax;
var married;

married = window.prompt("Enter M=married, S=single: ");
salary = window.prompt("Enter your salary: ");

if (married == "S")
{  // Single person
   if (salary > 50000)
      tax = salary*0.5;
   else if (salary > 35000)
      tax = salary*0.45;
   else
      tax = salary*0.30;
}  // End if single person
```

Making Decisions
Nested If Statement Example (2)

```
else if (married == "M")
{  // Married person
   if (salary > 50000)
      tax = salary*0.4;
   else if (salary > 35000)
      tax = salary*0.35;
   else
      tax = salary*0.20;
}  // End if married person
else  // Invalid input
   tax = -1;

if (tax != -1)
{  document.write("Salary: "+salary+"<br/>");
   document.write("Tax: "+tax+"<br/>");
}
else
   document.write("Invalid input!");
```

Nested Conditions and Decisions
Dangling Else Problem

The *dangling else problem* occurs when a programmer mistakes an else clause to belong to a different if statement than it really does.

◆Brackets determine which statements are grouped together, not indentation! By default, an else with no brackets matches the closest if statement regardless of indentation.

Example:

Incorrect	**Correct**

```
if (country == "US"))        if (country == "US")
  if (state == "HI"))        {  if (state == "HI")
     shipping = 10.00;             shipping = 10.00;
else  // Belongs to 2nd if!   }
   shipping = 20.00; // Wrong! else
                                 shipping = 20.00;
```

Nested Conditions and Decisions
Boolean Expressions

A *Boolean expression* is a sequence of conditions combined using AND (**&&**), OR (**||**), and NOT (**!**).

◆Allows you to test more complex conditions

◆Group subexpressions using parentheses

Syntax: *(expr1)* **&&** *(expr2)*   - expr1 AND expr2

        (expr1) **||** *(expr2)*   - expr1 OR expr2

        !(expr1)   - NOT expr1

Examples:
```
var b;

1) b = (x > 10) && !(x < 50);
2) b = (month == 1) || (month == 2) || (month == 3);
3) if (day == 28 && month == 2)
4) if !(num1== 1 && num2 == 3)
5) b = ((10 > 5 || 5 > 10) && ((10>5 && 5>10));// False
```

Boolean Expressions

Question: Is `result` `true` or `false`?

```
var x = 10, y = 20;
var result = (x > 10) || (y < 20);
document.write(result);
```

A) `true`

B) `false`

Boolean Expressions (2)

Question: Is `result` `true` or `false`?

```
var x = 10, y = 20;
var result = !(x != 10) && (y == 20);
document.write(result);
```

A) `true`

B) `false`

Boolean Expressions (3)

Question: Is result true or false?

```
var x = 10, y = 20;
var result = (x >= y) || (y <= x);
document.write(result);
```

A) true

B) false

Making Decisions (4)

Question: What is the output of this code?

```
var num=12;

if (num >= 8)
    document.write("big");
    if (num == 10)
        document.write("ten");
else
    document.write("small");
```

A) big

B) small

C) bigsmall

D) ten

E) bigten

Making Decisions (5)
Boolean Expressions

Question: What is the output of this code?

```
var x = 10, y = 20;

if (x >= 5)
{  document.write("bigx");
    if (y >= 10)
        document.write("bigy");
}
else if (x == 10 || y == 15)
    if (x < y && x != y)
        document.write("not equal");
```

A) bigx

B) bigy

C) bigxnot equal

D) bigxbigynot equal

E) bigxbigy

Practice Questions

1) Create the Boolean expressions in JavaScript for:
- a) x does not equal y OR y is greater than z
- b) x is greater than 0 AND less than 100
- c) x is not less than 0 OR greater than 100

2) Write a program that reads two numbers and prints them in sorted, descending order. Challenge: Do it for three numbers.

3) Challenge: Write a program that translates a letter grade into a number grade.
- Letter grades are A,B,C,D,F possibly followed by + or - with values 4,3,2,1, and 0. There is no F+ or F-. A + increases the value by 0.3, a - decreases it by 0.3. An A+ equals 4.0.
- You need to use two functions:
  - ⇨ <variableName>.length – length of string given by variableName
  - ⇨ <variableName>.charAt(0) – character at position 0 in string

Review: Key Programming Concepts

Some key concepts in programming:
- ***variables*** – names for data items to be manipulated
- ***locations*** – addresses of data items in memory
- ***values*** – the value stored at a particular location and referenced using a given variable name
- ***initialization*** – setting beginning values for variables
- ***assignment*** – general form of initialization where the value of a variable is set to another value
- ***decisions*** – performing different actions based on testing a condition
- ***expressions*** – consist of operands and operators and yield a result

Conclusion

We learned the basics of the JavaScript language to communicate instructions to the computer including:
- declaring and using variables
- initialization and assignment of values to variables
- expressions
- decisions and Boolean conditions

Objectives

◆Compare and contrast: algorithm and program

◆List and define the key programming concepts covered.

◆Explain the difference between variables, values, and locations.

◆Remember the rules for variables, comments, and statements.

◆Remember the rules for declaring and using strings.

◆Understand and explain assignment operator.

◆Define: operator, operand, unary, binary

◆Remember operator precedence for expressions.

◆Recall the string concatenation operator.

◆Be able to write and execute JavaScript code in HTML files.

◆Define: operator overloading

Objectives (2)

◆Know how to get input and send output to and from the user.

◆Write decisions using the if statement.

◆Define: Boolean, condition

◆List and use the comparison operators.

◆Explain the dangling else problem.

◆Construct and evaluate Boolean expressions using AND, OR, and NOT.