JavaScript

You will learn advanced programming tools in JavaScript that allow your programs to automatically repeat and to run alternate courses of execution.

Pictures courtesy of James Tam

Backup Your Work Frequently!

- This is always a good idea but imperative when writing JavaScript programs.
- JavaScript will NOT give you helpful error messages!
  - Usually you get no error message at all.
  - Determining where the problem lies can be a great challenge.
- There is an assignment requirement that you demonstrate some sort of versioning system (along with backups).
  - It's up to you to make sure that you actually create multiple versions in different backup files.

What Is Wrong With This Program

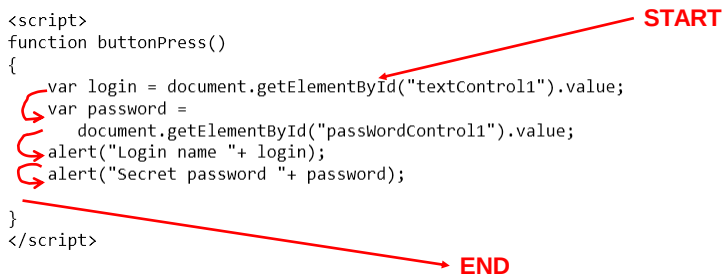
- **Name of example**: `error1.html`
- JavaScript wont' 'tell' you

```
<script>
function main()
    var name = "";
    name = prompt("Name: ","Smith");
    alert(name);
window.onload=main;
</script>
```

- (More on this later).

Recap: Programs You've Seen So Far Is Sequential Execution

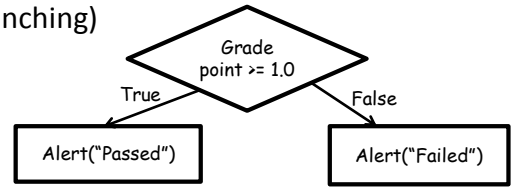
- Each instruction executes from beginning to end, one after the other



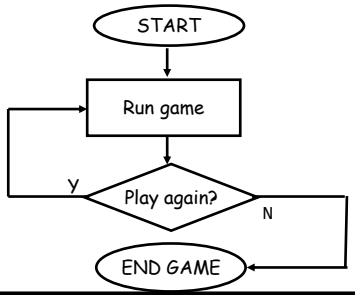
- When the last instruction is reached then the program ends

New Program Writing Concepts (Non-Sequential)

- Making decisions (branching)



- Looping (repetition)



New Terminology

- **Boolean expression**: An expression that must work out (evaluate to) to either a true or false value.
  - e.g., it is over 45 Celsius today
  - e.g., the user correctly entered the password

- **Body**: A block of program instructions that will execute under a specified condition.

```
function() {
    Alert("Fake virus!");
};
```

When the webpage includes instructions to run the function (the 'body' of the function executes).

  - Style requirement
    - The 'body' is indented (4 spaces)
    - A "sub-body" is indented by an additional 4 spaces (8 or more spaces)

Branching: IF

- JavaScript
  - if
  - if, else

    Similar to Excel if-then
  - if, else if, else
  - Similar to Excel nested ifs
  - Multiple-ifs

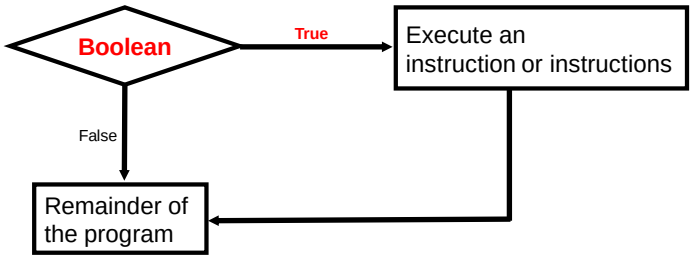
- When to use
  - React when true
  - React true & false
  - At most only one if-case can be true ("select one of the following")
  - A series of conditions must all be independently checked ("select all that apply")

Allowable **Operators** For Boolean Expressions

```
if (value operator value)
```

JavaScript operator	Mathematical equivalent	Meaning	Example
<	<	Less than	5 < 3
>	>	Greater than	5 > 3
==	=	Equal to	5 == 3
<=	≤	Less than or equal to	5 <= 5
>=	≥	Greater than or equal to	5 >= 4
!=	≠	Not equal to	value != password

Decision Making With '`If`'



`If:` General Format

```
if (<Boolean expression>)
{
    <body>;  // Indent the body by an additional 4 spaces
}
```

- Unlike the Excel '`IF-function`', the programming language '`IF`' is not is a function.
  - Excel `IF-function`: returns a value for true or false cases
  - Programming language `IF`: Executes one or more instructions in the body (any programming language instruction).
    - More powerful

`If:` An Example

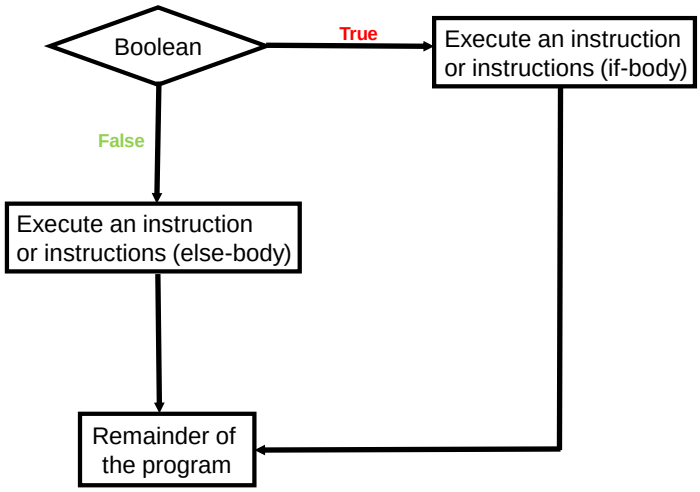
- **Name of example**: `1if.html`

```
<script>
function main()
{
    var age = -1;
    age = prompt("Enter your age (zero or greater):","37");
    if (age < 0)
    {
        alert("Age cannot be negative");
    }
};
window.onload=main;
</script>
```

Decision Making With An '`If, Else`'

- Used when different Actions (separate bodies) are required for the true result (IF-case) vs. the false result (ELSE-case)

Decision Making With An 'If, Else'



If-Else: General Format

Similar to the Excel If-Then

Format:

```
if (<Boolean expression>)
{
    <body if>;
}
else
{
    <body else>;
}
```

If-Else: Example Usage

Example:

```
if (age < 0)
{
    alert("Age cannot be negative");
}
else
{
    alert("Age verified as OK");
}
```

If-Else: An Example

- **Name of example**: 2ifElse.html

```
<script>
function main()
{
    var age = -1;
    age = prompt("Enter your age (zero or greater):","37");
    if (age < 0)
    {
        alert("Age cannot be negative");
    }
    else
    {
        alert("Age verified as OK");
    }
};
window.onload=main;
</script>
```

Logic Can Be Used In Conjunction With Branching

- Typically the logical operators AND, OR are used with multiple conditions/Boolean expressions:
  - If multiple conditions *must all be met* before the body will execute. (And)
  - If *at least one condition* must be met before the body will execute. (Or)
- The logical NOT operator can be used to check if something has 'not' occurred yet
  - E.g., If it's true that the user *did not* enter the correct password then the program will end.
  - Typically used in the form of a 'not equals' check

```
if (response != password)
```

Logical Operators

Logical operation	JavaScript	Example
AND	&&	if (x > 0 && y > 0)
OR	||	if (x > 0 || y > 0)

Logic: The "OR" Operator

- **Format:**

```
if ((Boolean expression) || (Boolean expression))
{
    body;
}
```

- **Name of complete example**: `3ifOr.html`

```
gpa = prompt("GPA: ","4.0");
xp = prompt("Years of job experience: ","6");
if ((gpa > 3.7) || (xp > 5))
{
    result = "Hire applicant";
}
else
{
    result = "Insufficient qualifications";
}
alert(result);
```

Hiring Example: Example Inputs & Results

```
if ((gpa > 3.7) || (experience > 5))
```

GPA	Years job experience	Result
2	*0*	*Insufficient qualifications*
1	10	Hire
4	1	Hire
4	7	Hire

Logic: The "AND" Operator

- **Format:**

```
if ((Boolean expression) && (Boolean expression))
{
    body;
}
```

- **Name of complete example**: `4ifAnd.html`

```
salary = prompt("Salary: ","100000");
years = prompt("Years of employment: ","0");
if ((salary >= 100000) && (years < 2))
{
    result = "Fired!";
}
else
{
    result = "Retained";
}
alert(result);
```

Firing Example: Example Inputs & Results

```
if ((salary >= 100000) && (years < 2))
```

Salary	Years on job	Result
1	100	Retained
50000	1	Retained
123456	20	Retained
1000000	*0*	*Fired!*

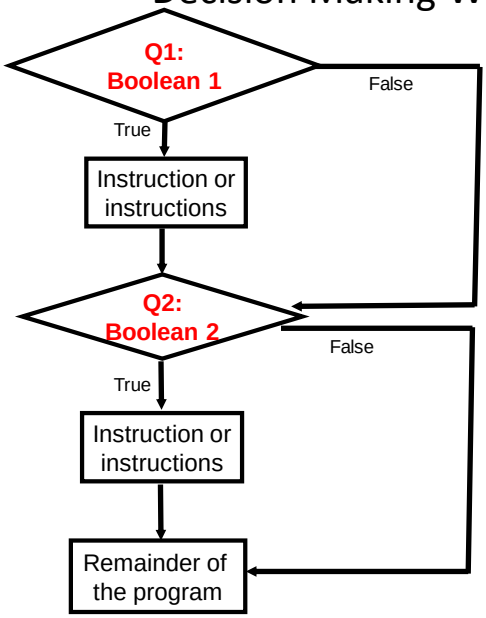
What To Do When Multiple Conditions Must Be Checked

- **Case 1**: If each condition is independent of other questions
  - Multiple `if` expressions can be used
  - Example (each question must always be asked regardless of the answer to the previous question)
  - Q1: Are you an adult?
  - Q2: Are you a Canadian citizen?
  - Q3: Are you currently employed?

What To Do When Multiple Conditions Must Be Checked (2)

- **Case 2 (mutually exclusive)**: If the result of one condition affects other conditions (when one condition is true then the other conditions cannot be true)
  - `If`, `else if`, `else` should be used
  - Which of the following is your place of birth? (Answering true to one option makes the options false)
    a) Calgary
    b) Edmonton
    c) Lethbridge
    d) Red Deer
    e) None of the above

Decision Making With **Multiple `Ifs`**



Each question is independent (previous answers have no effect on later questions because all questions will be asked).
Q1: Are you an adult?
Q2: Are you a Canadian citizen?

Multiple `Ifs`

- Any, all or none of the conditions may be true
- Employ when a series of independent questions will be asked
- **Format:**

```
if (Boolean expression 1)
{
    body 1;
}
if (Boolean expression 2)
{
    body 2;
}
...
statements after the conditions
```

Multiple `if`s: Example

- All questions must all be asked
- **Name of complete example**: `5multipleIfs.html`

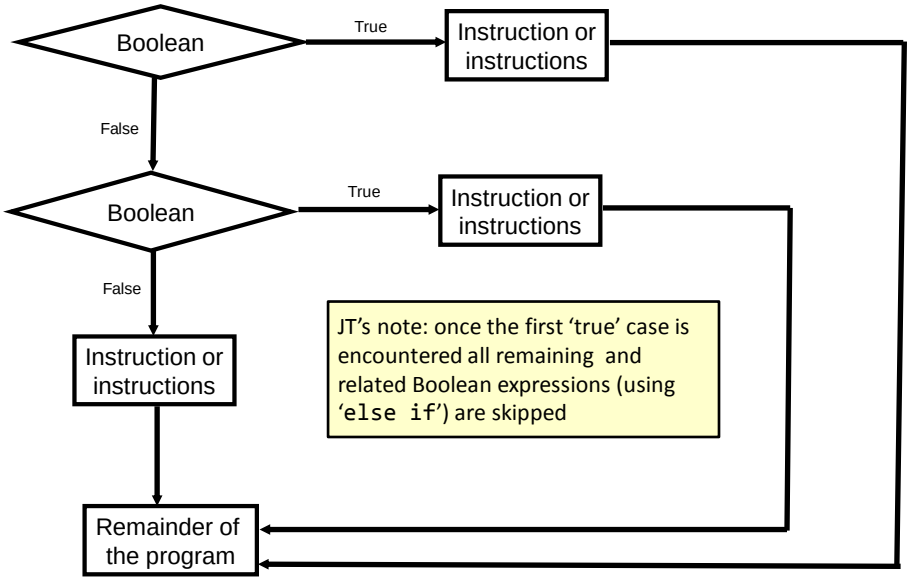
```
age = prompt("Age: ","17");
city = prompt("Current city: ","Calgary");
if ((age >= 6) && (age <= 17)) {
    response = "Grade schooler ";
}
else {
    response = "Other ";
}

if (city == "Calgary") {
    response = response + " in the most important city";
}
else {
    response = response + " some other place..";
}
```

Multiple `if`s: Mutually Exclusive Conditions

- At most *only one* of many conditions can be true (zero can be true as well).
- Can be implemented through multiple `if`s (poor approach)
- **Name of complete example**: `6multipleIfsInefficient.html`

```
if (letter == "A") {
   gpa = 4;
}
if (letter == "B") {
   gpa = 3;
}
if (letter == "C") {
   gpa = 2;
}
if (letter == "D") {
   gpa = 1;
}
if (letter == "F") {
   gpa = 0;
}
```

Decision Making With `if, else if, else`



Multiple `if, else if, else`: Use With Mutually Exclusive Conditions

- Similar to the Excel `nested-If` (only one case is true)
- **Format:**

```
if (Boolean expression 1) {
    body 1;
}
else if (Boolean expression 2) {
    body 2;
}
...
else {
    body n;
}
Statements after the conditions
```

Mutually exclusive

- One condition evaluating to true excludes other conditions from being true
- Example: having your current location as 'Calgary' excludes the possibility of the current location as 'Edmonton', 'Toronto', 'Medicine Hat'

If, Else If, Else: Example

- All questions must all be asked
- **Name of complete example**: `7IfElseIfElse.html`

```
if (letter == "A") {
   gpa = 4;
}
else if (letter == "B") {
   gpa = 3;
}
else if (letter == "C") {
   gpa = 2;
}
else if (letter == "D") {
   gpa = 1;
}
else if (letter == "F") {
   gpa = 0;
}
else {
   letter = "'" + letter
    + "'"
    + " not standard grade";
}
```

If, Else If, Else: Analysis

```
if (letter == "A") {
   gpa = 4;
}
else if (letter == "B") {
   gpa = 3;
}
else if (letter == "C") {
   gpa = 2;
}
else if (letter == "D") {
   gpa = 1;
}
else if (letter == "F") {
   gpa = 0;
}
else {
   letter = "'" + letter + "'" + " not standard grade";
}
alert("Letter: "+letter+" GPA: "+gpa);
```

This approach is more efficient when at most only one condition can be true.

Extra benefit:

The body of the `else` executes only when all the Boolean expressions are false. (Useful for error checking/handling).

JavaScript: `If, Else-If` And **Excel**: `Nested-Ifs`

- These two concepts are comparable:

JavaScript:

```
if (grade >= 4) {
    letter = "A";
}
else if (grade >= 3) {
    letter = "B";
}
else if (grade >= 2) {
    letter = "C";
}
else if (grade >= 1) {
    letter = "D";
}
else {
    letter = "F";
}
```

Excel (display different messages for different grade points e.g. Display “Perfect” if grade point is 4.0 or greater):

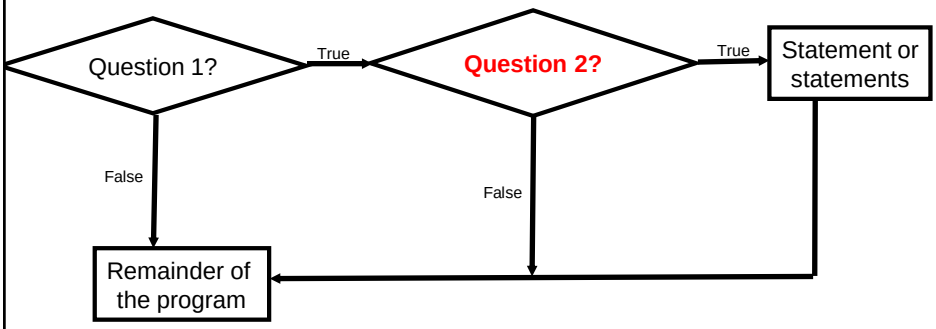
```
=IF(D2>=4, "A",
   IF(D2>=3, "B",
     IF(D2>=2, "C",
       IF(D2>=1, "D",
         "F"))))
```

Recall Excel (Spreadsheet) Nesting

- Conditions that are dependent upon or are affected by previous conditions.
- ‘Nesting’ refers to conditions that are ‘inside of other conditions’

JavaScript (Programming Language): Nested-IFs

- Similar to the IF, ELSE IF: Decision making is dependent.
  - One branch is 'nested' inside of another branch
  - The first decision must evaluate to true ("gate keeper") before successive decisions are even considered for evaluation.
- *Unlike the IF, ELSE IF* **more than one case can be true**:



Nested Decision Making

- One decision is made inside another.
- Outer decisions must evaluate to true before inner decisions are even considered for evaluation.
- **Format:**

```
if (Boolean expression) {
    if (Boolean expression) {
        body;
    }
}
```

Outer body

Inner body

- Both (or all if 2+ IFs) must evaluate to true before the inner-most body will execute.

Nested Decision Making: Indenting

```
if (Boolean expression)
{

   if (Boolean expression)
   {

        body;

   }
}
```

4 more spaces (4+4 = 8)

4 spaces

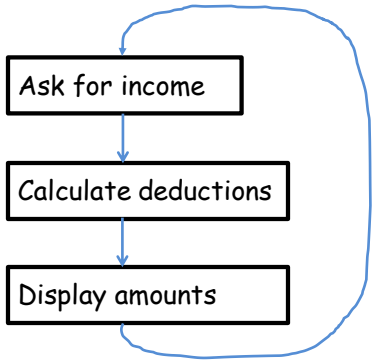
Nested IFs: Simple 'Toy' Example

- **Name of complete example**: 8IfNested.html

```
num1 = prompt("First number","1");
num2 = prompt("Second number","1");
if (num1 > 0)
{
    result = "Num1 positive - ";
    if (num2 > 0)
    {
        result = result + "Num2 positive";
    }
}
else
{
    result = "Num1 not positive, didn't check num2";
}
alert(result);
```

Looping/Repetition

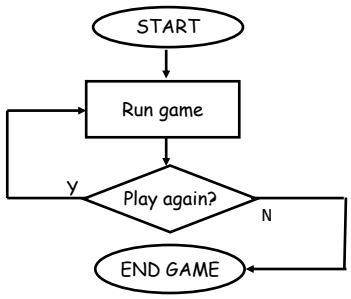
- How to get the program or portions of the program to automatically re-run
  - Without duplicating the instructions
  - Example: you need to calculate tax for multiple people



Loop: allows you to repeat the same tasks over and over again

Looping/Repetition (2)

- The entire program repeats



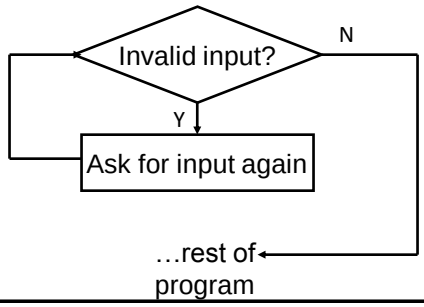
Looping/Repetition (3)

- Only a specific part of the program repeats

```
Enter your age (must be non-negative):  -1
Enter your age (must be non-negative):  27
Enter your gender (m/f):
```

Re-running specific parts of the program

Flowchart



Types Of Loops

- **Fixed repetition** loops: runs some integer 'n' times e.g., generates taxes for 10 clients
  - **`For-loop`**
- **Variable repetition** loops: runs as long as some condition holds true
  - e.g., while the user doesn't quit the program re-run the program
    - How many times does the program run? It depends upon the whims of the user (variable).
  - e.g., while the user enters an erroneous value ask the user for input.
  - **`While-loop`**

For Loops: General Format

```
for (<Set counter to initial value>;
     <Boolean Expression>;
     <Update counter>[1])
{
    <Body>; // Indent 4 spaces
}
```

Note: the three statements in the form loop do not have to reside on three separate lines.

1 Although the update is usually an increase or decrease by one any valid mathematical expression can be employed.

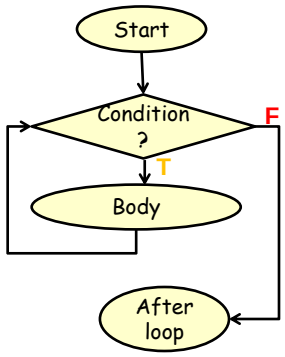
For Loops: An Example

Name of example: 9forLoop.html

```
<script>
function main()
{
    var i = -1;
    var last = -1;
    last = prompt("Enter last value in number series: ", "");
    for (i = 0; i <= last; i = i + 1)
    {
        alert("i=" + i);
    }

};
window.onload=main;
</script>
```

`While` Loops: General Format

```
while (Boolean Expression)
{
    <Body>;  // Indent 4 spaces
}
```



Programming Style: Variable Names

- In general variable names should be self-descriptive e.g., 'age', 'height' etc.
- Loop control variables are an exception e.g., 'i' is an acceptable variable name
  - It's sometimes difficult to come up with a decent loop control name
  - Also loop control variables are given shorter names so the line length of a loop isn't excessive

```
var loopControl = -1;
loopControl = 1;
while (loopControl <= 4)
    ...
```

While Loops: An Example

- **Name of example**: 10whileLoop.html

```
<script>
function main()
{
    var i = -1;
    var last = -1;
    last = prompt("Enter last value in number series: ", "");
    i = 0;
    while (i <= last)
    {
        alert("i=" + i);
        i = i + 1;
    }
};
window.onload=main;
</script>
```

Loops That Never Execute

- **Word document containing the complete program**: 11nonExecutingLoops.html

```
var i = -1;
i = 5;
while (i <= 4)
{
    alert("i=" + i);
    i = i + 1;
}

for (i = 5; i <= 4; i = i + 1)
{
    alert("i=" + i);
}
```

Exercise #1: Loops

- The following program that prompts for and displays the user's age.
  - loopExercise.html

```
function main()
{
    var age = -1;
    age = prompt("Enter age (zero or greater): ", "0");
    alert("Age: " + age);
}
```

Modifications:

- As long as the user enters a negative age the program will continue prompting for age.
- After a valid age has been entered then stop the prompts and display the age.
- Hint: Use a do-while loop (not a for-loop)

Nesting

- Nesting: one construct is contained within another
  - What you have seen: nested branches:

```
if (Boolean) {
    if (Boolean) {
        body;
    }
}
```

- Branches and loops can be nested within each other

```
while (Boolean) {
    if (Boolean) {
        body;
    }
}
```

Indenting: Any Form Of Nesting

General format (loops or branches):

```
Outer (Boolean)
{

    Inner (Boolean)
    {

        body;

    }
}
```

4 more spaces (4+4 = 8)

4 spaces

Example: Nesting

1. Write a program that will count out all the numbers from one to six.
2. For each of the numbers in this sequence the program will determine if the current count (1 – 6) is odd or even.
   a) The program display the value of the current count as well an indication whether it is odd or even.

- Which Step (#1 or #2) should be completed first?

Step #1 Solution

Full program name: `12nestingPart1.html`

```
function main()
{
    var i = -1;
    i = 1;
    while (i <= 6)
    {
        i = i + 1;
    }
};
window.onload=main;
</script>
```

Step #1 Completed: Now What?

- For each number in the sequence determine if it is odd or even.
- This can be done with the modulo (remainder) operator: %
  - An even number modulo 2 equals zero (2, 4, 6 etc. even divide into 2 and yield a remainder or modulo of zero).
  - `If (counter % 2 = 0)` then `'Even`
  - An odd number modulo 2 does not equal zero (1, 3, 5, etc.)
- Pseudo code visualization of the problem

  Loop to count from 1 to 6

  Determine if number is odd/even and display message

  End Loop
  - Determining whether a number is odd/even is a part of counting through the sequence from 1 – 6, checking odd/even is nested within the loop

Exercise #2: Loops & Error Messages

- Exercise #1:
  - The following program that prompts for and displays the user's age.
    - loopExercise.html

```
function main()
{
    var age = -1;
    age = prompt("Enter age (zero or greater): ", "0");
    alert("Age: " + age);
}
```

Modifications:

- As long as the user enters a negative age the program will continue prompting for age.
- After a valid age has been entered then stop the prompts and display the age.

- Modify your previous solution so that whenever a negative is entered the program will display an appropriate error message.

Loops: A More Complex Example

- Learning concepts:
  - How JavaScript can change a document contents via document.`write()`
  - Using a loop to produce 'special effects'
    - Change the image to be added

Loops: A More Complex Example (HTML Portion)

- **Name of complete example**: 13documentWrite.html

```
</script>
<img src="pics/noClick.png">
```


```
<input type="image"
  src="pics/clickable.png"
  onclick="main()"
/><br>
```

Loops: A More Complex Example (JavaScript Portion)

```
// Nesting again, writing to a webpage via document.write()
<script>
function main()
{
    var i = -1;
    funImage = ("<img src='pics/funGuy.jpg'>");
    fungiImage = ("<img src='pics/fungi.jpg'>");
    for (i = 1; i <= 4; i = i + 1)
    {
        prompt("Hit ok to continue", i);
        if (i % 2 == 0)  // % Modulo operator
        {
            document.write(funImage); // even
        }
        else
        {
            document.write(fungiImage); // odd
        }
     }
}
```

Documentation Requirements

- As specified in your assignment if there are special data requirements for your webpage and/or JavaScript program then this must be documented.
- The previous program requires two images to be located in a specific location.
- Program documentation:

```
Requirements for program to run.
 * There is a subfolder called 'pics' in the folder where this
   program resides.
 * The pics folder must contain images with the exact names (you
   can substitute your own images if you wish (content doesn't
   matter just the name): 'fungi.jpg', 'funGuy.jpg', 'noClick.png',
   'clickable.png'.
   Etc.
```

Writing HTML Tags To A Document

- Most any html tag can be written to a document via "`document.write()`"
- The tags are treated as strings and like tags added to page without `write()` they must be enclosed in angled brackets.
- **Forma**t:

```
document.write(<"tag">);
```

- **Example**:

```
document.write(<"br">);
```

Writing Multiple Lines To A Document

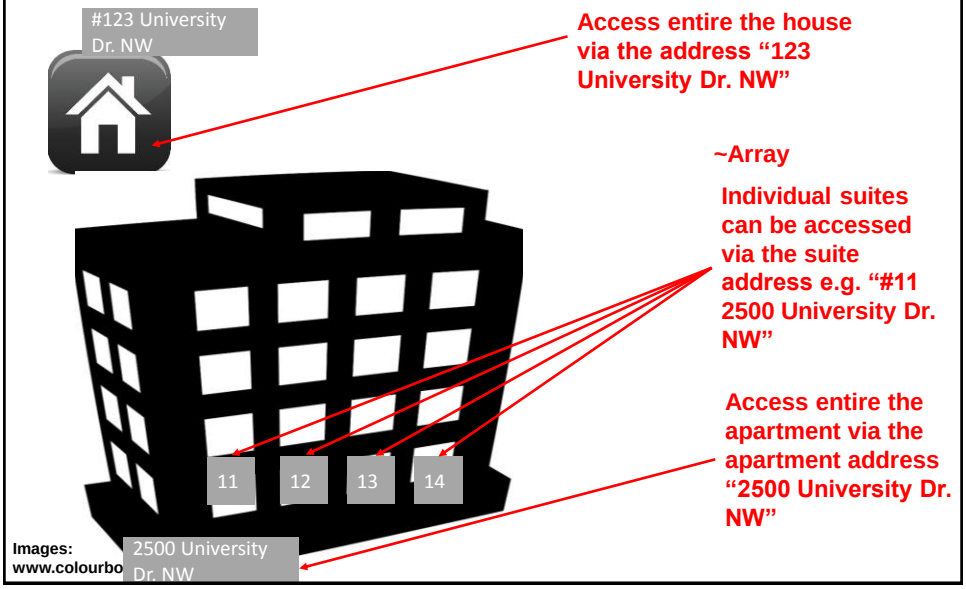
- **Name of complete example**: `14multilineWrite.html`

```
function main()
{
    document.write("<b>Line1</b>: Part 1");
    document.write("<b>Line1</b>: Part 2");
    document.write("<br><b>Line2</b>:");
};
```

Arrays (If There Is Time)

- Unlike 'simple' variables such as integers and real numbers, an array can hold multiple values e.g., a number variable can store *one student's grades* while *an array can store the grades for an entire class*.

Accessing Simple Variables Arrays



Declaring Array Variables

- **Format** (creating an array):

```
var <array name> = new Array(<Number of elements>);
```

- **Example** (creating an array):

```
var grades = new Array(4);
// Specify just the array name affects the whole array
// (creates whole array)
// (~apartment address)
```

Accessing Array Elements

- Use array name and an 'index' (~apartment suite address)
  - The first index begin at zero
  - The last index ends at (size of the array minus 1).
  - Example: an array of size 10 will have index values from 0 to (10-1 or 9)
- **Format** (assigning value to an element):

```
<array name>[<index>] = value;
```

- **Example** (assigning value to elements):

```
grades[0] = 4;
grades[1] = 3;
grades[2] = 2;
grades[3] = 3;
```

A More Realistic Example Program (HTML Portion)

- **Name of example**: 15array.html

```
<img src="pics/noClick.png">

<input type="image" src="pics/clickable.png" onclick="main()"
/><br>
 Requirements for program to run.
  * There is a subfolder called 'pics' in the folder where this
    program resides.
  * This folder contains 4 images with the following names: 0.jpg,
    1.jpg, 2.jpg, 3,jpg
```

A More Realistic Example Program (JavaScript Portion)

```
function main()
{
    var SIZE = 4;
    var pics = new Array(SIZE);
    var i = 1;
    for (i = 0; i < SIZE; i = i + 1)
    {
        pics[i] = "<img src='pics/" + i + ".jpg'>"
    }

    for (i = 0; i < SIZE; i = i + 1)
    {
        prompt("Hit ok to continue","");
        document.write(pics[i]);

    }
}
```

i = 0

i = 1

i = 2

i = 3

Reminder: Backup Your Work Frequently!

- This is always a good idea but imperative when writing JavaScript programs.
- JavaScript will NOT give you helpful error messages!
  - Usually you get no error message at all.
  - Determining where the problem lies can be a great challenge!

Example Program With An Error (Or Errors)

Program name: `error2.html`

```
function main()
{
    const SIZE = 4;
    var pics = new Array(SIZE);
    var i = 1;
    for (i = 0; i < SIZE; i = i + 1)
    {
        pics[i] = "<img src='pics/" + i + ".jpg'>"
    }

    for (i = 0; i < SIZE; i = i + 1)
    {
        prompt("Hit ok to continue","");
        document.write(pics[i]);
    }
}
```

Backups: Benefit

- If you accidentally introduce an error to your program and you cannot find the error.
- You can always:
  - Compare the backup version vs. the current version in order to more easily find the difference
  - Simply revert to the backup (if you cannot find the error)
    - Starting working from the last working backup file.
- When you make backups always ensure that your backup is working prior to making a backup.
- You have been forewarned!
  - Failure to make a proper set of backups won't allow you to get an extension.

If All Else Fails

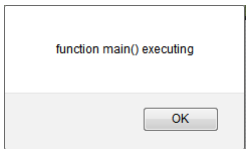
- What to do if you still didn't take this advice and you are desperately trying to find the error.
- First: there are no guarantees with this technique.
- Second: make generous use of comments
  - "Comment out" the entire body of the function save for alert(s) (to tell you which parts of the function is running properly).

"Commenting Out Code" Previous Example

```
function main()
{
    alert("function main() executing");
/*
    const SIZE = 4;
    var pics = new Array(SIZE);
    var i = 1;
    for (i = 0; i < SIZE; i = i + 1)
    {
        pics[i] = "<img src='pics/" + i + ".jpg'>"
    }

    for (i = 0; i < SIZE; i = i + 1)
    {
        prompt("Hit ok to continue","");
        document.write(pics[i]);
    }
*/
}
```

"Commenting Out Code": Gradually Move The Comments To Expose More Code

```
function main()
{
    alert("function main() executing");
/*
    const SIZE = 4;
    var pics = new Array(SIZE);
    var i = 1;
    for (i = 0; i < SIZE; i = i + 1)
    {
        pics[i] = "<img src='pics/" + i + ".jpg'>"
    }

    for (i = 0; i < SIZE; i = i + 1)
    {
        prompt("Hit ok to continue","");
        document.write(pics[i]);

    }
*/
}
```



As you 'uncomment' code take care that you don't introduce new errors e.g., mismatch braces

Finding Errors

- This technique of "commenting out" JavaScript code **will not work if the error is in your html tags**.
- (Recall: documentation is used only conjunction with programming instructions and html does not include programming instructions).
- Example: documentation will not help, what you were taught was commenting out JavaScript code (so BACKUP YOUR WORK)

```
 <script>
function main()
{
    alert("function main() executing");
}
</script>
<img src="pics/noClick.png">
<input type="image" src="pics/clickable.png"
onclick="main" /><br>
```

Using documentation for "commenting out" only works here

Strings

- A series of characters: alpha, numeric and other values that can be entered via the keyboard.
- A string has a length
- Examples:
  - `var string1 = "a12!";  // 4 characters`
  - `var string2 = "";  // 0 characters`
- The length method will return the length
- **Format**:

```
<string>.length;
```

- **Example**:

```
userInput = alert("Name: ", "Bruce Lee");
userInput.length == 0 // userInput must be a String
```

Example Using The Length Method

- **Program name:** `16stringLength.html`

```
function main()
{
   var userInput = document.getElementById("input1").value;
   if (userInput.length == 0)
   {
      alert("Input field empty!");
   }
   else
   {
       alert(userInput);
   }
}

</script>
<input type="button" value="Press me" onclick="main()"/><br>
Enter some text and press the button <input type="input"
id="input1"/>
```

Changing Capitalization

- This can be done with the methods:
  - <*string*>.toUpperCase();
  - <*string*>.toLowerCase();

- **Format**:

```
<string variable> = <string>.toUpperCase();
<string variable> = <string>.toLowerCase();
```

- **Example:**

```
userInput = alert("Name: ", "Bruce Lee");
userInput = userInput.toUpperCase(); //userInput must be
                                     //a string
```

Example Of Changing Case

- **Program name:** 17changingCase.html

```
function main()
{
   var userInput = document.getElementById("input1").value;
   if (userInput.length == 0)
   {
      alert("Input field empty!");
   }
   else
   {
       userInput = userInput.toUpperCase();
       alert(userInput); // All caps
       userInput = userInput.toLowerCase();
       alert(userInput);  // All lower now
   }
}
</script>
<input type="button" value="Press me" onclick="main()"/><br>
Enter some text and press the button <input type="input"
id="input1"/>
```

Strings Are Indexed

- Similar to arrays each character in a string is indexed from: 0 to (length – 1).
- Example:
  - var aString = "ab1";

```
Elements:  a b 1
Index:     0 1 2
```

- The `<string>.indexOf()` method can be used to find the index of the first occurrence of a character.
  - If the character is not found then the method returns -1 (minus one).
- **Format**:

```
<variable> = <string>.indexOf(<character>);
```

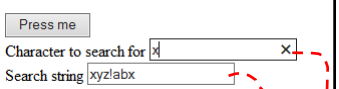
- **Example**:

```
index = aString.indexOf("!");
```

Example: Finding Location

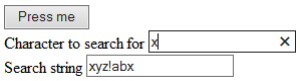
- **Program name:** `18findingIndex.html`

```
<input type="button" value="Press me" onclick="main()"/><br>
Character to search for <input type="input" id="data"/><br>
Search string <input type="input" id="search"/>
```

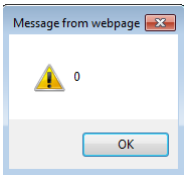



```
function main()
{
   var searchCharacter =
     document.getElementById("data").value;
   var stringSearched =
     document.getElementById("search").value;
   if ((searchCharacter.length == 0) ||
       (stringSearched.length == 0))
   {
       alert("Unable to search. Either search string or
              character empty");
   }
```

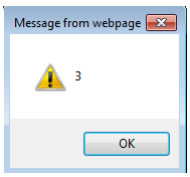
Example: Finding Location (2)




```
    else
    {
      // Index of character in input field
       var index = stringSearched.indexOf(searchCharacter);
       alert(index);
```




```
       // Index of exclamation mark '!'
       var index = stringSearched.indexOf("!");
       alert(index);
    }
}
```



Additional Online Resources

- http://www.w3schools.com/js/
- https://msdn.microsoft.com/en-us/library/ie/ms535262(v=vs.85).aspx
- http://trainingtools.com/online/javascript/index.htm

After This Section You Should Now Know

- Branching/decision making mechanisms:
  - `if`
  - `if-else`
  - `if, else if, else`
- Valid operators for Boolean expressions
- Valid logical operators

After This Section You Should Now Know (2)

- Getting JavaScript instructions to repeat using looping mechanisms:
  - `For`
  - `While`
- Changing the contents of a webpage via `document.write()`
- How to create an array and how to access the elements
- Common string operations
  - Determining length
  - Converting case
  - Finding the location of the first instance of a character
- The importance of backing up and versioning your webpage and JavaScript program
  - Using documentation to find JavaScript errors