

## JavaScript

You will learn more advanced html tags for creating graphical 'GUI' controls and the basics of JavaScript programming.

Pictures courtesy of James Tam

## Advanced HTML

- In the last section you learned how to use html to: format text, embed content and link to other pages.
- In this section you will learn how to use HTML to create graphical controls such as buttons and input fields.

## Creating GUI Controls

- **Format:**

```
<input type="<control type>" value="<Text description>"/>
```

- **Name of example:** 1emptyControl.htm

```
<input type="button" value="Press me"/>
```



- **Pressing the button does nothing! Why???**
- **You need to write the JavaScript instructions to indicate what happens when the button press occurs (details come later)**

## Types Of GUI Controls


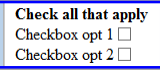

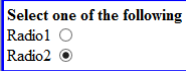

GUI control	HTML	Appearance
Button	<code>&lt;input type="button" value="Press"/&gt;</code>	<input type="button" value="Press me"/>
Checkbox (select 1+)	<code>&lt;input type="checkbox" value="Checkbox one of many"/&gt;</code>	Checkbox opt 2 <input type="checkbox"/>
Password (input hidden)	<code>&lt;input type="password" value="def"/&gt;</code>	Password <input type="password" value="*****"/>
Radio (select exactly 1)	<code>&lt;input type="radio"/&gt;</code>	Radio1 <input checked="" type="radio"/>
Text (single line input)	<code>&lt;input type="Text" value="Type a line of text here"/&gt;</code>	<input type="text" value="Type a line of text here"/>

There's numerous places online where you can find information about how to use these controls

- Example:
- [https://msdn.microsoft.com/en-us/library/ms535260\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms535260(v=vs.85).aspx)

## Graphical Controls: More Complete Example

- Name of example: 2manyEmptyControls.htm

<pre>Button &lt;input type="button" value="Press me" /&gt;&lt;br&gt; &lt;br&gt;</pre>	
<pre>&lt;b&gt;Check all that apply&lt;/b&gt;&lt;br&gt; Checkbox opt 1&lt;input type="checkbox" /&gt;&lt;br&gt; Checkbox opt 2&lt;input type="checkbox" /&gt;&lt;br&gt; &lt;br&gt;</pre>	
<pre>Password &lt;input type="password" value="default" /&gt;&lt;br&gt; &lt;br&gt;</pre>	
<pre>&lt;b&gt;Select one of the following&lt;/b&gt;&lt;br&gt; Radio1 &lt;input type="radio" /&gt;&lt;br&gt; Radio2 &lt;input type="radio" checked="checked" /&gt;&lt;br&gt;</pre>	
<pre>&lt;br&gt; Text &lt;input type="Text" value="Type a line of text here" /&gt;</pre>	

## Adding JavaScript To A Webpage

- This is where you can augment a web page to make it interactive.
  - JavaScript is separate from the webpage content (text, images, formatting tags)
  - Enclose them with a `<script>` tag
  - The JavaScript instructions must be defined within a function.

## Where/How To Include Programs

index.html

```
<script>
```

```
JavaScript  
function goes  
here
```

```
</script>
```

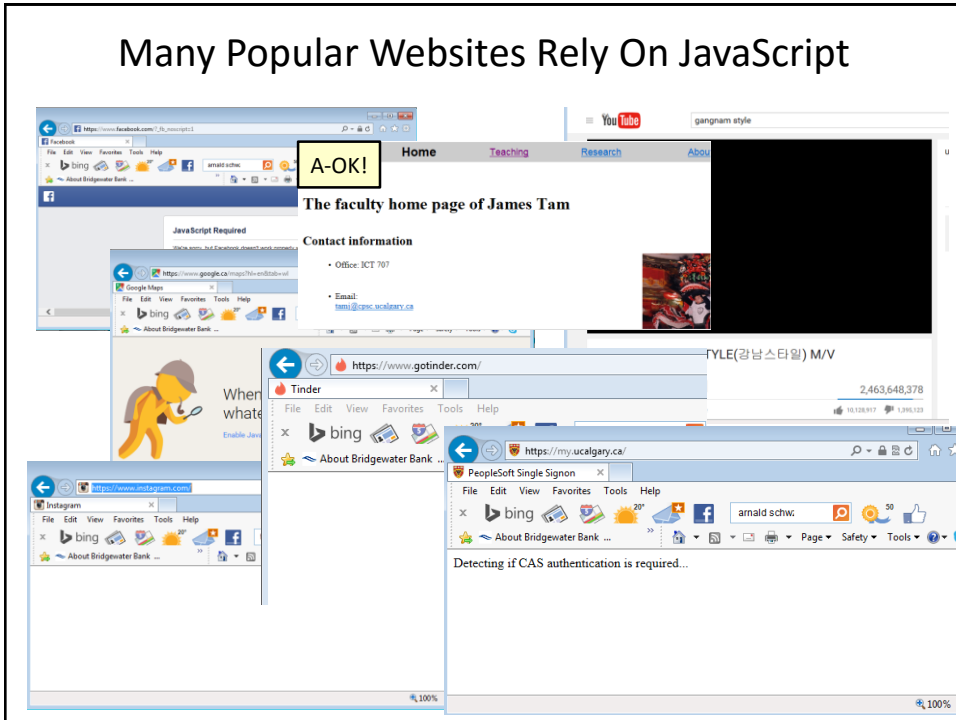
```
This is the <i>normal</i> <b>webpage content</b>
```

- A function is a series of instructions that run when an event occurs e.g., the user clicks on a button
- JavaScript functions for this course are similar to functions you have seen in Excel except that *you will be learning how write your own functions* rather than using a pre-created one.

## Running Programs Through A Webpage

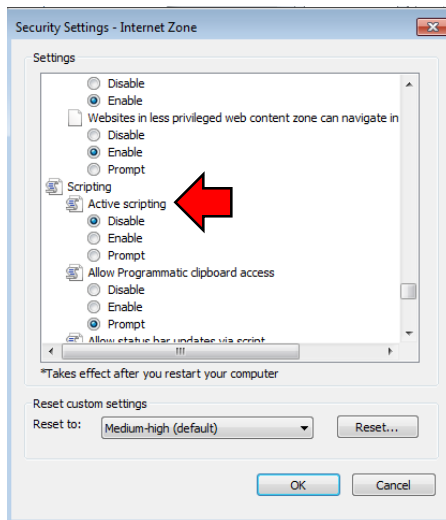
- Why bother?
- Among other things adding a computer program to a webpage can make it more interactive.
  - Error checking user input e.g., email format in a web form
  - Allow visitors to play games
  - Etc.
  - With JavaScript you can write programs that can execute when a webpage is downloaded and viewed through a web browser.

## Many Popular Websites Rely On JavaScript



## Enable/Disable Scripting (JavaScript)

- IE: Tools->Internet Options->Security->Custom Level



## Defining A JavaScript Function

- All the instructions in your JavaScript program must be enclosed in a function.

- **Format:**

```
function <function name>()
{
    Instructions in body (indent 4 spaces);
}
```

- **Example:**

```
function saySmartResponse()
{
    alert("Don't press that button");
}
```

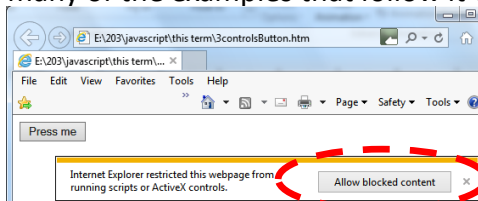
index.html

```
<script>
Function start()
{
    Instructions;
}
</script>

Webpage content
```

## Online Security: Webpages That Include Programs (Example: JavaScript)

- To prevent malicious software from running on your computer (JavaScript programs like any other software can be beneficial or malicious) most configurations may warn website visitors before these programs run.
- Program #3 will run before the user enters a response but many of the examples that follow it will not.



- Also note: although the example programs for this class are benign you should not make that assumption about other programs you encounter online.

## Getting The GUI To **React To User Input**

- **Name of example:** 3controlsButton.htm

```
<script>
function saySmartResponse()
{
    alert("Don't press that button");
}
</script>
```

```
<input type="button" value="Press me"
onclick="saySmartResponse()"/>
```

### **onclick**

- When user clicks on the button

## Common Mistake #1

- Forgetting the semi-colon after each instruction.
  - The **semi-colon** separates one instruction from another.

```
function saySmartResponse()
{
    alert("Instruction 1");
    alert("Instruction 2");
}
```

## Defining Multiple Functions

- Normally a program consists of multiple functions:

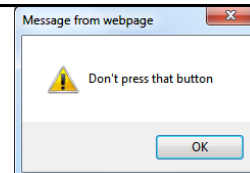
```
<script>
function anotherFunction()
{

}

function saySmartResponse()
{
    alert("Don't press that button");
    anotherFunction();
}
</script>
```

- For this class you need only define a single function (as shown in all/most of my examples)

## Displaying Output: Alert() Box



- (Details of the previous example)
- Creates a popup window that 'outputs' information to the webpage visitor
- Useful for testing
  - Is my program working?
  - Which part is running?
- Also useful for displaying status messages about the current state of the program
- **Format** (basic version: displays a constant string, fixed message):
 

```
alert("<i>Message to display</i>");
```
- **Example:**

```
alert("Don't press that button");
```

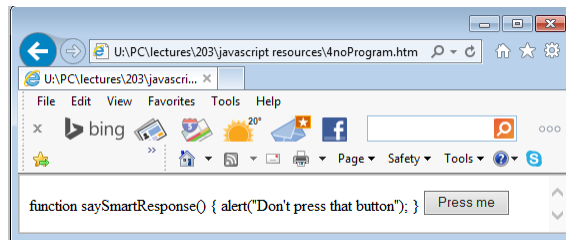


## Common Mistake #2

- Forgetting the opening and closing script tags
- **Name of example:** 4noProgram.htm

```
function saySmartResponse()
{
    alert("Don't press that button");
}
<input type="button" value="Press me"
    onclick="saySmartResponse()"/>
```

### Result



## Variables

- Used to temporarily store information at location in memory
- Variables must be declared (created) before they can be used.

- **Format:**

```
var <variable name> = <value>;
```

- **Example:**

```
var num = -1;
```

## A4: HTML Tags & JavaScript Programming

- Important reminder: it is crucial that you test your submission by:
  - Viewing the effect of the html tags in a web browser.
  - Running the JavaScript program through the web browser to ensure that it properly fulfills assignment requirements.
- Your submission must work on the 203 lab computers in order to be awarded credit.

## Web Page Design And JavaScript Programming

- It is not assumed that you have any prior experience writing computer programs (JavaScript or something else).
- Consequently early examples and concepts will be quite rudimentary i.e., “we will teach programming gradually”
  - This section of notes will only cover very rudimentary programming concepts.
  - The next section covers more advanced but common and useful programming concepts.

## Some Types Of Variables

Type of information stored	Example variable declaration
Whole numbers	<code>var wholeNum = 2;</code>
Real numbers	<code>var realNum = 2.0;</code>
Characters <sup>1</sup>	<code>var aString = "xyz";</code>

1) Any visible character you can type and more e.g., 'Enter' key

## A Program With **Variables**

- **Name of example:** 5variables.htm

```
<script>
function buttonPress()
{
    var num = 2;
    var aString = "abc";
    alert("Num = " + num);
    alert("aString = " + aString);
}
</script>
<input type="button" value="Press"
onclick="buttonPress()"/><br>
```

## Alert(): Displaying Multiple Pieces Of Information

- Mixed output (strings and variables) must use the **concatenation operator '+'** to connect the different types of information.

- Format** (Advanced: displays mixed output):

```
alert("<string> + <contents of variable> +
      <string> + <contents of variable>...");
```

- Examples:**

```
alert("num = " + num);
alert("aString = " + aString);
```

```
"aNum="      : A literal string
num:         : contents of a variable (slot in memory)
```

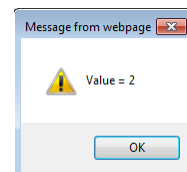
## Common Mistake #3

- A variable is just that – it can change as a program runs.

```
<script>
function buttonPress()
{
    var num = 2;
    alert("Value = " + num);
}
</script>
```

Vs.

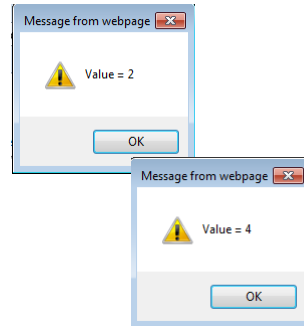
```
<script>
function buttonPress()
{
    alert("Value = 2");
}
</script>
```



## Common Mistake #3: Not Recognizing Variables Can Change

```
<script>
function buttonPress()
{
    var num = 2;
    alert("Value = " + num);

    num = num * 2;
    alert("Value = " + num);
}
</script>
```



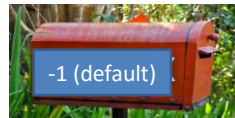
## Variables: Metaphor To Use



- Think of VBA variables like a “mailslot in memory”
- Unlike an actual mailslot computer variables can only hold one piece of information
  - Adding new information results in the old information being replaced by the new information

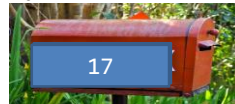
```
var num = -1;
```

num



```
num = 1;
```

num



```
num = 17;
```

## Programming Exercise #1: Variables

- Also each computer variable is separate location in memory.
- Each location can hold information independently of other locations.
- Note: This works differently than mathematical variables!

```
var num1 = -1;  
var num2 = -1;  
num1 = 1;  
num2 = num1;  
num1 = 2
```

What is the result?

## Variable Naming Conventions

- Language requirements:
  - Rules built into the JavaScript programming language.
  - Somewhat analogous to the grammar of a 'human' language.
  - If the language rules are violated then the typical outcome is the program cannot execute.
- Style requirements:
  - Approaches for producing a well written program.
  - (The real life analogy is that something written in a human language may follow the grammar but may still be poorly written).
  - If style requirements are not followed then the program can execute but there may be other problems (e.g., it is difficult to understand because it's overly long and complex - more on this during the term).
    - Your style mark may suffer for the assignments

## Naming Variables: JavaScript Requirements

- JavaScript language requirement (syntax ~ grammar)
  - Name can be alphanumeric but the *name cannot start with a number*  
**2**num (Not OK) vs. num1 (OK)
  - Spaces are not allowed  
First **█**name (Not OK) vs. firstName (OK) OR first\_name (OK)

## Naming Variables: Style Requirements

- Style requirement (~quality writing)
- Variable names should be self-descriptive and lower case (except for multi-word variable names)  
x, foo, AGE (Not OK)  
vs. firstName, age, height (OK)

## Variable Naming Conventions: Bottom Line

- Both the language and style requirements must be followed when declaring your variables.
  - Otherwise your program won't work or your style mark component will suffer.

## Operators

- The common operators that you will see in JavaScript are the same as other languages and applications (such as MS-Excel)

Operation	JavaScript operator
Division	/
Modulo (remainder)	%
Multiplication	*
Subtraction	-
Addition <sub>1</sub>	+

1 Note:

Performs mathematical **addition** only if all the inputs are numeric

e.g.,  $12 + 12 = 24$

If one or more inputs are strings then a **concatenation** is performed

e.g., `alert("Num = " + num);`



## Review: Lookup Tables (For Constants)

- Excel: Lookup tables are used to define values that do not typically change but are referred to in multiple parts of a spreadsheet.

**Lookup Tables**

- As the name implies it contains information that needs to be referred to ("looked up") in a part of the spreadsheet.
- Can be used to address some of the issues related to the previous example:
  - Clarity
  - Entering the same data multiple times

$$=(B2*G2)+(C2*G3)$$

	A	B	C	D	E	F	G
1	Student	Assignment grade point	Exam grade point	Term grade point		Component	Weight
2	1	4.2	3.3	3.66		Assignment	0.4
3	2	3.3	3.7	3.54		Exam	0.6
4	3	2.3	1	1.52			
5	4	4	4	4			

## Named Constants

- They are similar to variables: a memory location that's been given a name.
- Unlike variables their contents *cannot* change.
- The naming conventions for choosing variable names generally apply to constants but constants should be all UPPER CASE. (You can separate multiple words with an underscore).
- Example **const PI** = 3.14
  - **PI** = **Named constant**, 3.14 = Unnamed constant
- They are capitalized so the reader of the program can quickly distinguish them from variables.

## Declaring Named Constants

### Note:

This program will not work in older web browsers that don't support named constants (IE11 and earlier, older versions of Firefox and likely Chrome and Safari).

- **Format:**

– Constants should be declared at the very start of the function (right after you start defining the function "function <function name>()")

**const <Name of constant> = <Expression><sup>1</sup>**

JT: the assignment is preceded by the keyword 'const' to indicate that it's a constant

- **Name of example:** 6constants.htm

```
function buttonPress()
{
  const AGE_MULTIPLIER = 7;
  var humanAge = 10;
  var catAge = humanAge * AGE_MULTIPLIER;
  alert("Age in person years: " + humanAge);
  alert("Age in cat years: " + catAge);
}
```

<sup>1</sup> The expression can be any mathematical operation but can't be the result of a function call

## Why Use Named Constants: Clarity

- They can make your programs easier to read and understand

- Example:

income = 315 \* 80 **No** ☹️

Vs.

income = WORKING\_DAYS\_PER\_YEAR \* DAILY\_PAY **Yes** 😊

## Why Use Named Constants: Easier Maintenance

- Changing the constant once will change the value throughout the program.
- Abstracted example (not complete JavaScript code but enough to give you an idea of how using constants can be beneficial)

```
const GST = 0.05;

tax = price * gst;
Alert(tax); "
Alert(gst);
If (gst <= 0)
{
  alert("Tax cannot be negative!");
}
```

## Program Documentation

- It's used to explain the workings of a program to the reader of the program (other people who write programs 'programmers').
- It's not meant for users of the program (or web page visitors)
  - Typically it's fairly technical in nature.
  - It must be distinguished from JavaScript instructions (which are executed) because documentation should not execute.

```
<script>
  Author: James Tam
  Version: March 20, 2016

  Learning concept:
  * Creating and using variables

function buttonPress()
{
  Etc.
```

**When documentation not differentiated from instructions:**

**Error: I don't know how to "Author: James Tam"**

## Single Line Documentation

- Single line
  - Everything after the quote until the end of the line will not be treated as a JavaScript instruction (not translated into machine language/binary).

```
// Everything to the end of the line is not counted as a
// JavaScript instruction (used to explain details of the
// program to other programmers.
```

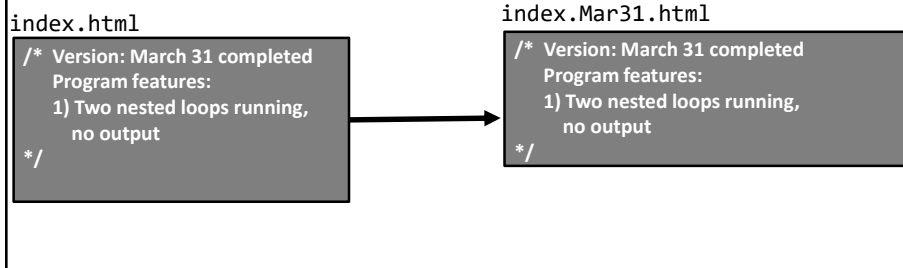
## Multi-Line Documentation

- Multiple line
  - Start of documentation uses `/*`
  - End of documentation uses `*/`
  - Everything between is not counted as a JavaScript instruction

```
/*
   Author: James Tam
   Tutorial: 888
*/
```

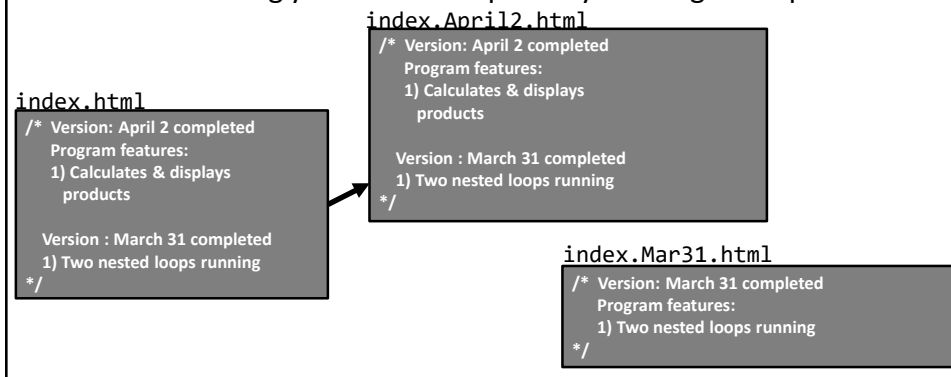
## Program Documentation: Requirements For This Class

- Your JavaScript assignment submission must include:
  - Some evidence of a versioning system.
  - As significant program features have been completed (tested and the errors removed/debugged) a new version should be saved in a separate file.
  - List the program features (from the assignment description) and clearly indicate if the feature was completed or not completed.



## Program Documentation: Requirements For This Class (2)

- DO NOT make edits to the working backup file(s).
  - In the example the backup files are the ones with the date suffix.
- Add new features to the copy.
- That way if your current version becomes non-functional or is malfunctioning you still have a partially working backup.



## Program Documentation: Requirements For This Class (3)

- You only need to hand in the last version of your assignment (DOUBLE CHECK your submitted file to ensure that this truly is the case) to fulfill the 'evidence of versioning' requirement.
  - We won't have time to go through multiple versions (just check the last)

index.html

```

/* Version: April 2 completed
   Program features:
   1) Calculates & displays
      products

   Version : March 31 completed
   1) Two nested loops running
*/

```

**Example of a student submission (JavaScript instructions excluded here for obvious reasons).**

- However even if you aren't directly marked on the previous versions making them gives you a fall-back in case disaster strikes.
  - You should back your current and previous versions in a place other than your computer.

## Minimum Documentation Needed For CPSC 203

```
// Version number (or date)
```

```
/*
```

A list of each program feature and an indication if the feature was completed (yes or no).

Alternatively: just list in the documentation the features that you actually completed (previous slides\_.

```
*/
```

```
// (If applicable): explicit instructions on the name and
// location of any data files required. For a
// detailed example see the example from the next section
// (requires images to be in a specific folder).
```

## Required Header Documentation: **Where**

- Within the 'script' tag but before your function.

```

<script>
/*
  Author: James Tam
  Version: March 20, 2016

  Learning concept:
  * Creating and using variables
*/
function buttonPress()
{
  var num1 = -1;
  var num2 = -1;
  num1 = 1;
  num2 = num1;
  alert("num1=" + num1 + " " + "num2=" + num2);
  num1 = 2
  alert("num1=" + num1 + " " + "num2=" + num2);|
}
</script>
<input type="button" value="Press" onclick="buttonPress()"/><br>

```

**Example documentation**

## Additional Documentation: **Where**

- Can be used to explain in detail how parts of a complex program work (your discretion if needed for the assignment).

```

<script>
// Single line documentation

/*
  Multi-line documentation
*/
function <function name>()
{
  // Extra:
  // If needed additional documentation
  // can be added here.
}
</script>

```

**Location of required documentation**

**Extra additional documentation (only if you think it's needed)**

## Documentation: Common Mistake #4

- Take care not to nest multi-line comments
- This is an error!

```

/*
  /* Author: James Tam */
  Tutorial: 888
*/

```

Marks end of documentation

Invalid statement (not JavaScript)

## Multiple Controls

- If you need to **identify one control** from another then you can use the 'id' attribute when creating a control in the html tags

- **Format:**

**id = "<Descriptive string>"**

- **Example:**

```
<input type="text" id="text1"/>
```

- When choosing a descriptive string to identify the control it's a good idea to apply the same conventions used as when you are naming variables e.g., no spaces, good and self-descriptive names

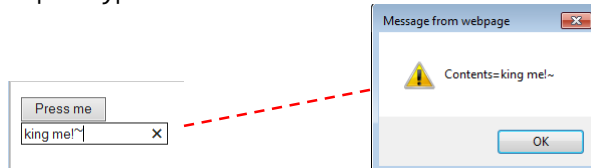


## Example Of Using Multiple Controls: **Identifying Controls**

- **Name of example:** 7multipleControls.htm

```
<script>
function buttonPress()
{
    var text1 = document.getElementById("text1").value;
    alert("Contents=" + text1)
}
</script>
<input type="button" value="Press me"
onclick="buttonPress()"/><br>

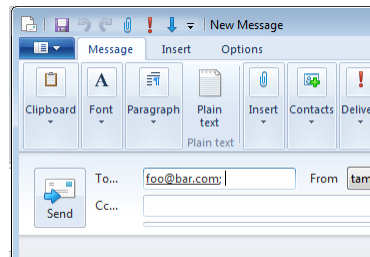
<input type="text" id="text1" value="default1"/><br>
```



## Sending Email: Basic Version (Same Constant Email Address)

- **Name of example:** 8sendingEmail.htm

```
<script>
function buttonPress()
{
    window.open("mailto:foo@bar.com");
}
</script>
<input type="button" value="Send mail" onclick="buttonPress()"/>
```

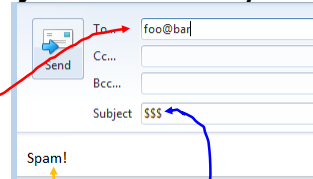


## Sending Mail: Constant Email, Subject And Body

- Name of example: 9sendingEmail.htm

```
<script>
function buttonPress()
{
    window.open("mailto:foo@bar" + "?subject=$$$"
        &body=Spam!");
}
</script>
```

```
<input type="button" value="Send mail"
onclick="buttonPress()"/>
```



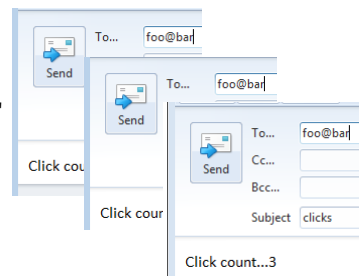
## Sending Mail: Constant Email, Subject And Body Is Variable

- Name of example: 10sendingEmail.htm

```
<script>
var count = 0;
function buttonPress()
{
    count = count + 1;
    window.open("mailto:foo@bar" +
        "?subject=clicks&body=Click count..." + count);
}
</script>
```

Creating 'count' outside of the function means that the previous count is retained

```
<input type="button" value="Send mail"
onclick="buttonPress()"/>
```



## Sending Mail: Assignment

**Constant fixed data: with some example email**

**Constant fixed data: subject is always the same**

**Variable data**

Title: Sushi-master

First Name: James

Last Name: Tam

- Example:** 6multipleControls.htm
- One example of accessing the data value of a control

## Accessing Data From Multiple GUI Controls

- **Name of example:** 11getData.htm

```

<script>
function buttonPress()
{
    var login = document.getElementById("textControl1").value;
    var password =
        document.getElementById("passwordControl1").value;
    alert("Login name " + login);
    alert("Secret password " + password);
}
</script>

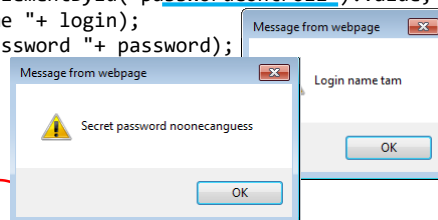
```

Show data

Login: tam

Password: ●●●●●●●●

Irrelevant: not used



```

<input type="button" value="Show data" onclick="buttonPress()"/><br>
Login: <input type="text" id="textControl1" /><br>
Password: <input type="password" id="passwordControl1" /><br>
<br>
Irrelevant: <input type="text" value="not used" id="textControl2" /><br>

```

## Acknowledgments

- The functional requirements of the email program were based on the assignment description produced by James Tam
- The different versions of the email program were based on a solution produced by Omar Addam

## Getting **User Input**

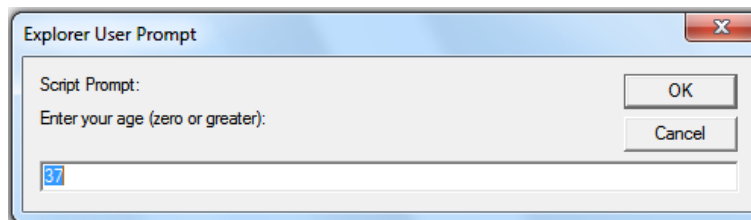
- Getting Input:

**Format:**

```
<variable> = prompt("<Prompting message>", "<default input>")
```

**Example** (assume we create a variable called 'age'):

```
age = prompt("Enter your age (zero or greater):", "37");
```



## Getting JavaScript To **Run Automatically**

- **Name of example:** 12getInputAutoRun.htm

```
<script>
function main()
{
    var age = -1;
    age = prompt("Enter your age (zero or greater):", "37");
};
window.onload=main;
</script>
```

## Additional Online Resources

- <http://www.w3schools.com/js/>
- [https://msdn.microsoft.com/en-us/library/ie/ms535262\(v=vs.85\).aspx](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

- How to use html to create a graphical controls such as buttons
- How to include JavaScript instructions via the `<script>` `</script>` tag
- How to define a 'sub-part' of a program using a function definition
- Getting a program to react to events (e.g., `onclick`) by defining a function
- Displaying output via `alert()`
- Defining variables with 'var'
- Variable naming conventions
- Common operators: mathematical (+, -, \*, /) and concatenation (+)

## After This Section You Should Now Know (2)

- How to declare and access named constants
- What are the benefits of employing named constants
- Single and multi-line documentation
- What should be included in the documentation for the programs that you write for this class
- In HTML: How to use the 'id' property to uniquely identify controls
- JavaScript: How to use the `document.getElementById("<string>")` method to access a graphical control

### After This Section You Should Now Know (3)

- Different variations when sending email using JavaScript instructions
- How to get a JavaScript program to automatically run when the web page is loaded into a web browser via `window.onload()=<function name>`
- Getting input with a `prompt()`