PROG8020

Week 1

Course Outline and Overview

- Primary topics:
  - JavaScript
  - jQuery
  - Node.js
  - MongoDB

Overview: JavaScript

- Data Types and Operation
- Objects, Properties, Methods
- Functions and Events
- Logical and Conditional Operators
- Making Decisions
- Loops
- Arrays
- Forms - **Data Validation**

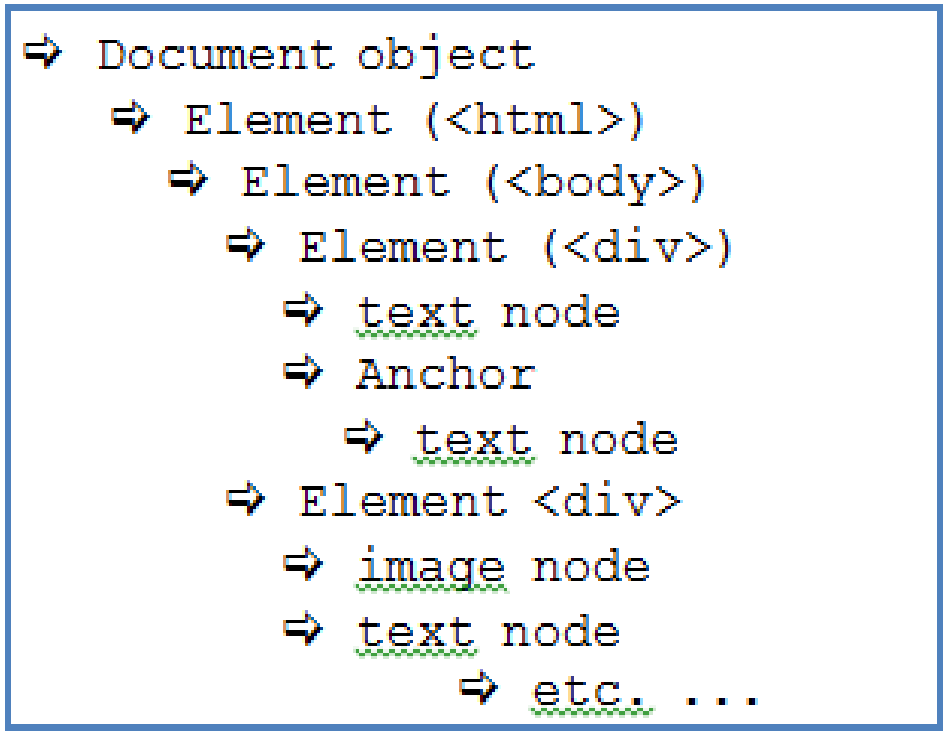
Overview: jQuery

- JavaScript and the relationship to jQuery
- Special Data Types and Structures
- Executing functions after DOM has been built
- Creating an Element with jQuery()
- Accessing the elements of a web page
- Formatting with Style Sheets under jQuery

- AJAX support in jQuery

Overview: DOM

- DOM: Document Object Model
- DOM tree: the hierarchical structure of how tags are rendered
- Example: `<h1>` tags are a higher level than `<p>` tags



Overview: XML

- XML: Extensible Markup Language
- Designed to transport and store data
- Uses tags like HTML
- Example:

```
<customer>
    <name>Sally Smith</name>
    <greeting>Final Notice!</greeting>
    <message>Please remit your balance
    immediately</message>
</customer>
```

Javascript

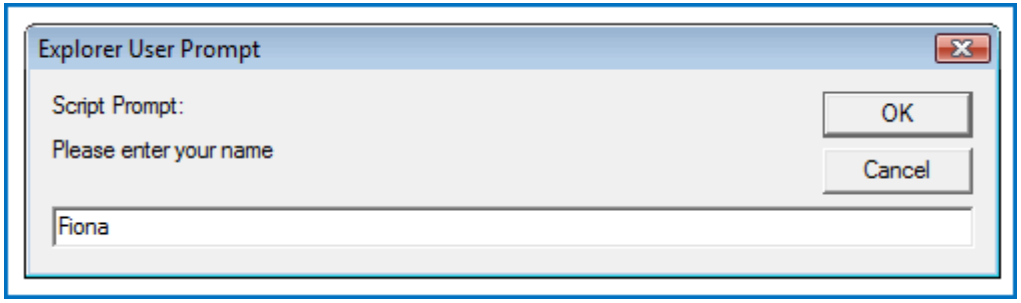
- Used to make web pages dynamic (interactive with the user)
- A dynamic programming language
  - executes at runtime
- Contains first-class functions
  - supports passing functions as arguments to other functions, returns them as values from other functions, assigns them to variables, or stores them in data structures
- A multi-paradigm language
  - Allows for greater flexibility

Javascript: Prompts

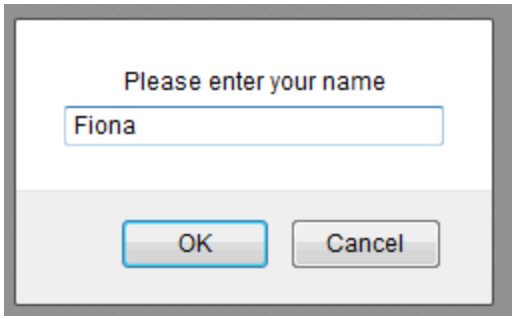
Prompt:

```
var name = prompt("Please enter your name"," ");
```

User sees:



(Internet Explorer)



(Firefox)

After entering "`Fiona`", the variable `name` holds the value "`Fiona`"

Prompts: Processing the Input

Example:

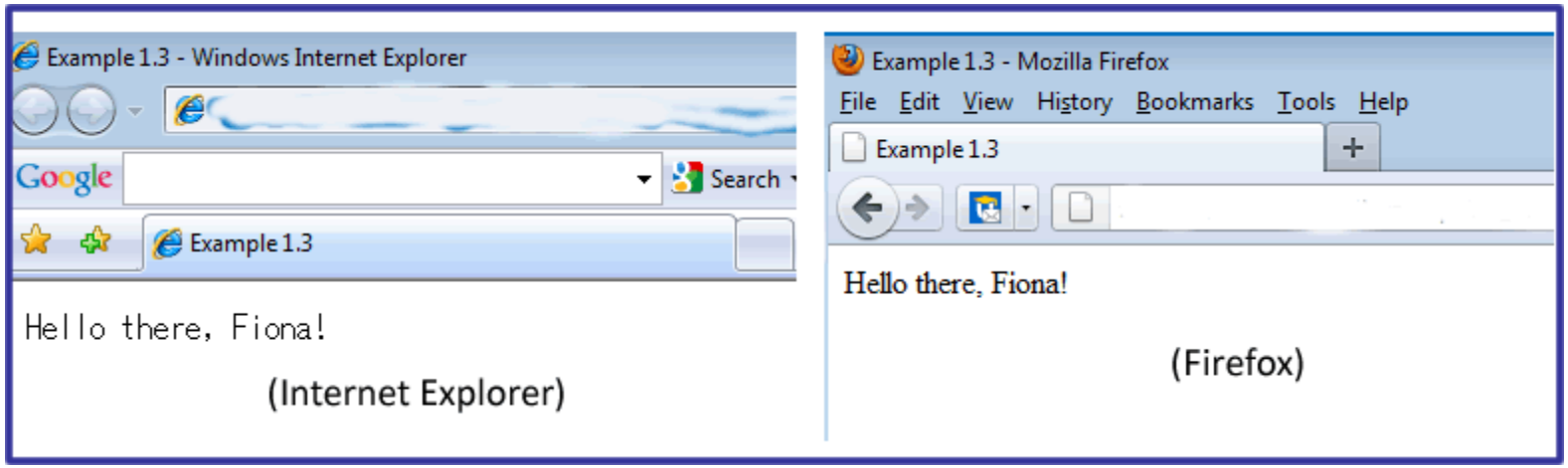
```
<script type="text/javascript">
    var name = prompt("Please enter your name"," ");
    var greeting = "Hello there, " + name + "!";
</script>
```

If the user enters `"Fiona"`, the variable `name = "Fiona"` and the variable `greeting = "Hello there, Fiona!"`

Prompts: Output

Example:

```
<script type="text/javascript">
    var name = prompt("Please enter your name"," ");
    var greeting = "Hello there, " + name + "!";
    document.write(greeting);
</script>
```



Control Structures

The sequential (or sequence) structure

statements execute in sequence

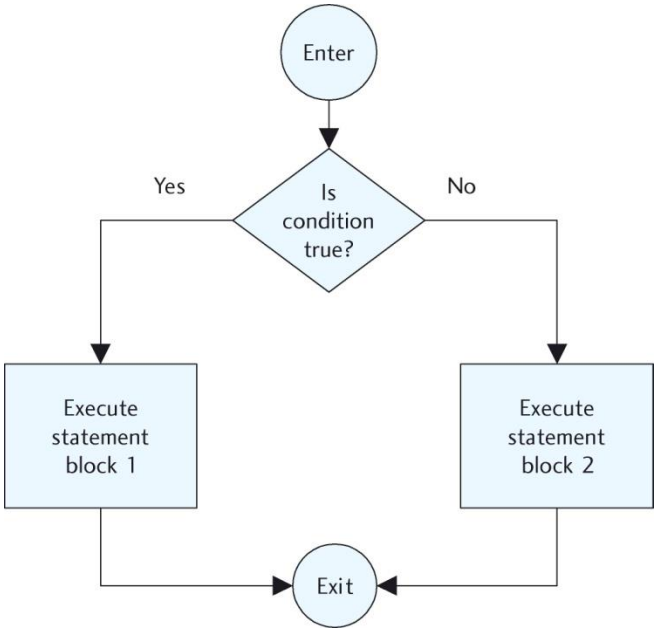
The decision (or selection) structure

statements execute if a condition is true

if not, either nothing happens or other statements execute

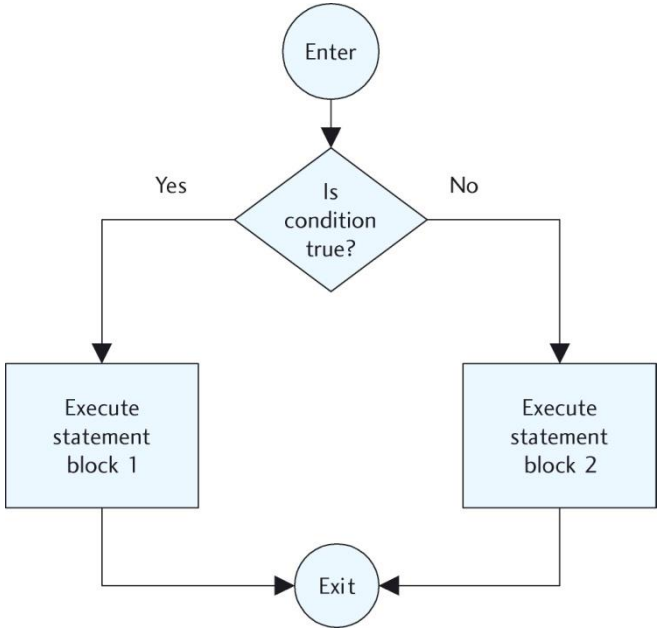
The loop (or repetition) structure

statements execute until a condition is

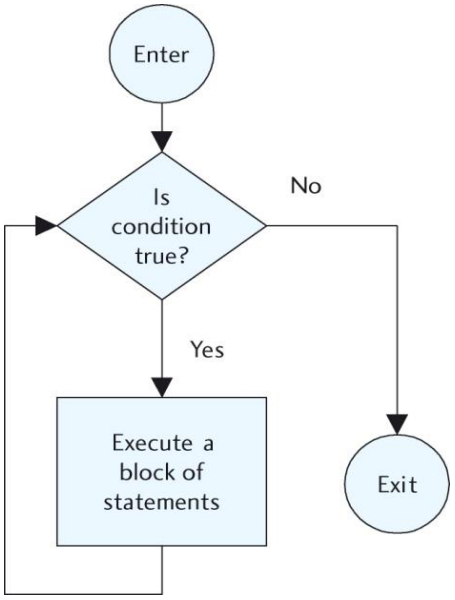


Control Structures (2)

- The decision (or selection) structure



The loop (or repetition) structure



Data Types and Operations on Data

Numeric Data

- Numbers are values that can be processed and calculated.
- Many languages make a distinction between integers and floating point numbers.
- JavaScript: when a number is stored in a variable, it is initially treated as a floating point number.
- All numbers in JavaScript are initially stored as the **numeric** data type.
- When a number is entered into a prompt box, it is initially stored as a **text** value.
  - It cannot be used in a calculation.
  - It must be turned into a numeric value to use in a calculation.

String Data

- Strings are a series of keyboard characters enclosed in double or single quotation marks.
- Strings can consist of words, phrases, sentences, and even whole paragraphs.
- A string can also be a single character such as a letter or a punctuation character.
- When a number is stored as a string, it cannot be used in a numerical calculation or process.

Variables and Named Constants

- A variable is called a variable because it can vary.
- A quantity that can change value during the execution of a program.
- Any time we need to refer to that data, we refer to its **variable name.**
- A named constant is a value that is used often in a program but will not change value throughout the program, such as the number in a dozen or the tax rate charged on purchases.

Assignment Statements

Declaring variables: Use the `let` keyword

```
let age;
```

creates a variable named `age`

```
let age = 23;
```

creates a variable named **`age`** which is assigned an initial value of `23`

It is also possible to use the `var` keyword.

Use `const` to declare constants.

Operations on Data

Arithmetic Operators:

Operator	Description	Example	Result, if y = 3
+	Addition	x = y + 2	x = 5
-	Subtraction	x = y - 2	x = 1
*	Multiplication	x = y * 2	x = 6
/	Division	x = y / 2	x = 1.5
%	Modulus	x = y % 2	x = 1

The Concatenation Operator

- Concatenation Operator: joins two strings together
- The symbol is **+** but, by the context, the computer knows that it is not used to add values.

Example:

```
greeting = 'Good morning'
name = "Robbie"
```

The following statement concatenates the variables and other text and stores it as one string in a third variable named **welcome**:

```
welcome = greeting + ', ' + name;
```

After the execution of this statement, the variable **welcome** contains :

```
"Good morning, Robbie"
```

More Concatenation

- Latest versions of JavaScript allow for a different concatenation approach:

Example:

```
greeting = 'Good morning'
name = "Robbie"
```

The following statement concatenates the variables and other text and stores it as one string in a third variable named **welcome**:

```
welcome = `${greeting}, ${name}`;
```

After the execution of this statement, the variable **welcome** contains :

```
"Good morning, Robbie"
```

Problem Solving: The importance of Logical Thinking

JavaScript in the Web Page

The <script></script> Tag Pair
The <noscript></noscript> Tag Pair

<script></script>:

Used to define a client-side script like JavaScript

<noscript></noscript>:

Used to provide alternate content for users who have disabled scripts in their browsers

Used to provide alternate content for browsers that don't support client-side scripting – rare today

```
<noscript>
    Sorry, your browser doesn't support JavaScript.
</noscript>
```

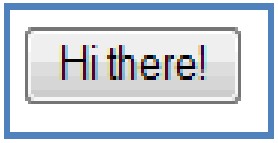
JavaScript in a web page <body>

Using inline JavaScript with a button:

Code to add a button to a web page:

```
<input type="button" id="myButton" value="Hi there!"
      onclick="alert('Well, hello my friend.');" />
```

Creates a button that looks like this:



When clicked, you get an alert that says:

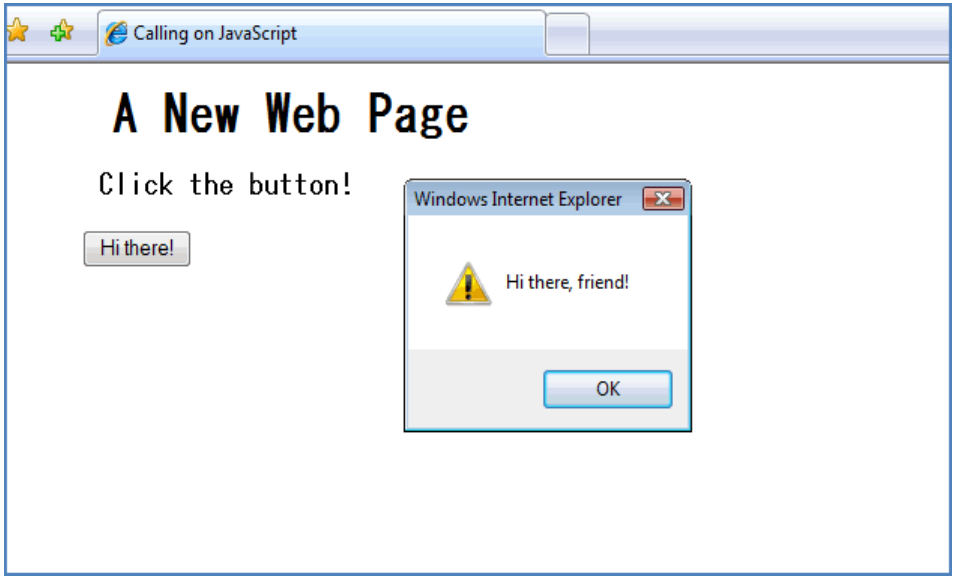
```
Well, hello, my friend.
```

JavaScript in the Document <head> Section

Most JavaScript we will write will be in <head> section

JavaScript executes when user does something in web page

```
<html>
<head>
<title>Example</title>
<script>
function welcome(){
    alert('Hi there, friend!');
}
</script>
</<head>>
<body>
   <h1>A New Web Page</h1>
   <h3>Click the button! </h3>
   <p><input type="button"
        id="myButton" value="Hi
        there!" onclick =
        "welcome();" /></p>
</body>
</html>
```



The <body onload> Event

Loads JavaScript before user views the page, as it is loading

```
<html>
  <head>
    <title>Example</title>
    <script>
      function welcome(){
        alert("Hi there, friend!");
      }
    </script>
  </head>
  <body onload = "welcome()">
   <h1>A New Web Page</h1>
   <h3>This site is nothing but fun!</h3>
  </body>
</html>
```

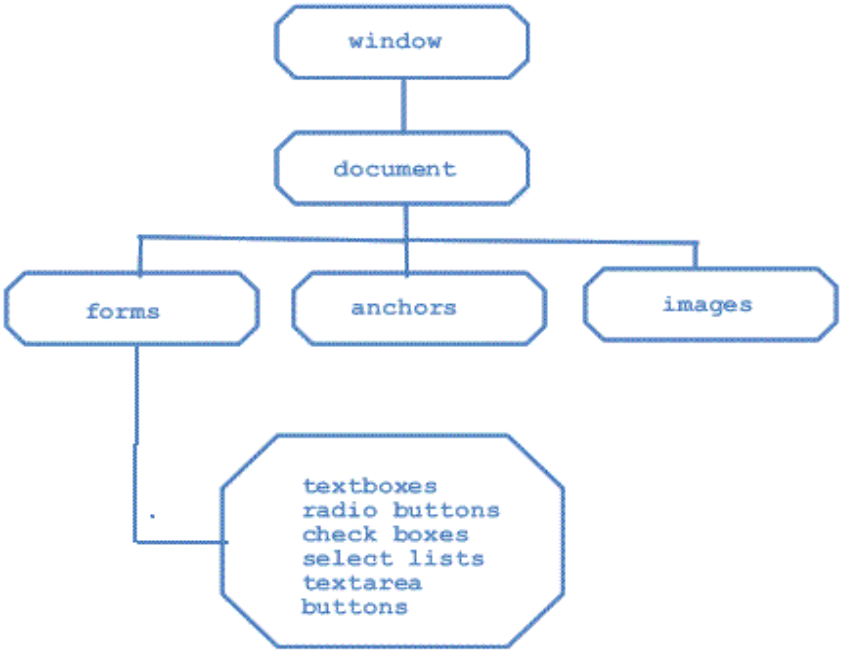
Introduction to Objects

Objects: Properties and Methods

- Anything that has properties and a function (or functions) is an **object.**
- Properties are qualities, traits, or attributes common to a specific thing—or object.
  - Properties (also called attributes) describe the object
- A function, in this context, is a process or operation executed by or to the object.
  - Methods (also called functions) are the things the object can do or have done to it.

Object Example: The Document Object

- An HTML document is an object.
- It uses the **Document Object Model** (DOM)



Objects: Dot Notation

- You instruct the browser where to place content by using **dot notation**.
- The object is accessed, then a dot, and then any further instructions (methods or attributes) are appended.

```
document.write('Welcome to my first JavaScript page!');
```

The `document` object is accessed. The dot says to use the `write()` method on the `document` object.

Objects: `getElementById()`

- Each part of a web page is called an element
- To access an element use the `getElementById()` method
- Allows access a particular container within a document
- Each container must be marked with an identifier
- Add an `id` attribute to the HTML tag

Objects: `InnerHTML` Property

The `innerHTML` property sets or returns the inner HTML of an element.

```
1. <html>
2.   <head>
3.     <title>Example</title>
4.     <script type="text/javascript">
5.       function getValue(){
6.         let dog = document.getElementById('puppy');
7.         let dogName = dog.innerHTML;
8.         document.write('<h1>Your dog is not a terrier </h1>');
9.         document.write('<h2>It is a ');
10.        document.write(dogName);
11.        document.write('</h2>');
12.      }
13.    </script>
14.  </head>
15.  <body>
16.    <h1 id = "puppy" onclick="getValue()">Poodle</h1>
17.  </body>
18.</html>
```

Objects: jQuery

We can do the same with jQuery

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
    <script  src=https://code.jquery.com/jquery-3.3.1.min.js></script>
    <script type="text/javascript">
      function writePage(){
        let puppyName = $('#puppy').text();
        let page = `<h1>Your dog is not a terrier.</h1><h2>It is a ${puppyName}</h2>`;
        document.write(page);
      }
    </script>
  </head>
  <body>
    <h1 id="puppy" onclick="writePage()">Poodle</h1>
  </body>
</html>
```

Objects: `open()` and `close()`

- These methods open or close a new window

The `innerHTML` property sets or returns the inner HTML of an element.

```
1.  <html>
2.  <head>
3.  <title>Using the open() and close() Methods</title>
4.  <script type="text/javascript">
5.  function openWin()
6.  {
7.     smallWindow = window.open('','', 'width=300, height=200');
8.     smallWindow.document.write('<p>Hi again, old friend!<br />Glad to see you today</p>');
9.  }
10. function closeWin()
11. {
12.    smallWindow.close();
13. }
14. </script>
15. </head>
16. <body>
17.   <input type="button" value="Open a small window" onclick="openWin()" />
18.   <input type="button" value="Close the small window" onclick="closeWin()" />
19. </body>
20. </html>
```

Introduction to JavaScript Functions and Events

Functions

- A function is used to isolate a group of instructional statements so that other parts of the program can use that code.
- Functions and methods can normally be used interchangeably.
- Two main categories of functions: user-created and built-in
- Syntax to create your own function:
  - type the `function` keyword
  - Follow with the function's name
  - Put all statements within opening and closing curly brackets ({ }).

```
function name(){
        JavaScript statements...;
}
```

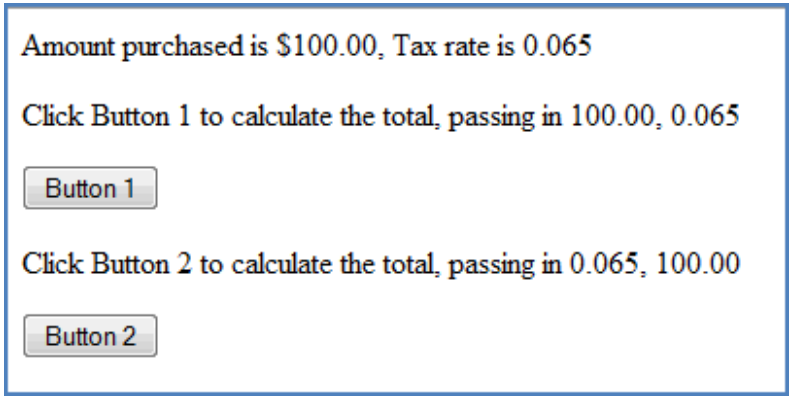
Functions: Built-in Functions

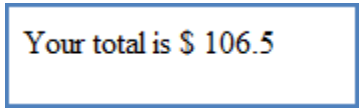
- Some built-in JavaScript functions that we have used so far:
  - `alert()`
  - `write()`
  - `open()`
  - `close()`
  - `getElementById()`

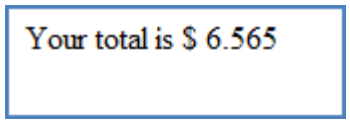
Functions: Parameters

Parameters are values that are passed into a function.

```
<head>
<title>Using parameters</title>
<script type="text/javascript">
function calculateTotal(purchaseAmt, taxRate)
{
   tax = purchaseAmt * taxRate;
   total = purchaseAmt + tax;
   document.write(`Your total is $ ${total}`);
}
</script>
</head>
<body>
<p>Amount purchased is $100.00, Tax rate is 0.065</p>
<p>Click Button 1 to calculate total, passing  in
100.00, 0.065<p>
<input type="button" value="Button 1"  onclick =
"calculateTotal(100, .065)" />
<p>Click Button 2 to calculate the total, passing  in
0.065, 100.00<p>
<input type="button" value="Button 1"  onclick =
"calculateTotal(0.065, 100)" />
</body>
```







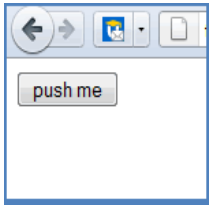
Functions: Parameters (2)

The `prompt()` Function

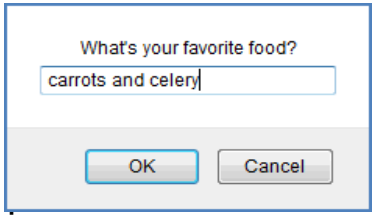
Allows us to prompt the user to input values which can then be used in any way

```
<head>
<title>The prompt() Function</title>
<script type="text/javascript">
function showPrompt()
{
          let food = prompt('What's your favorite
food?', 'carrots and celery');
          document.write('It's your lucky day! ' +
food + ' is on today's lunch menu!');
}
</script>
</head>
<body>
<input type="button" onclick = "showPrompt()"
value = "push me" />
</body>
```

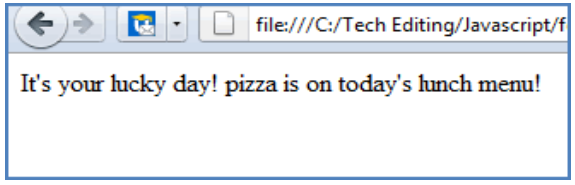
You first see:



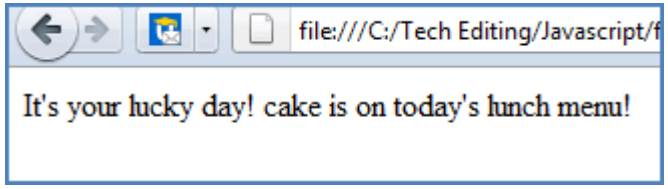
After pushing the button:



If user enters pizza:



If user enters cake:



Javascript Events

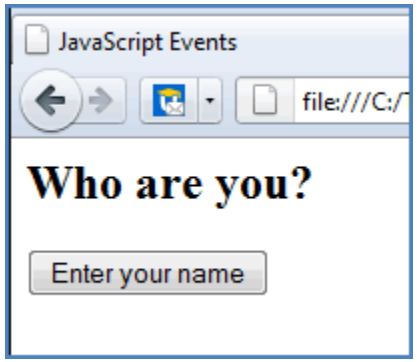
- An **event** is an action that can be detected by JavaScript
- Usually events are used in combination with functions
- When an event occurs, the function is executed
- Called event-driven programming
- Events:
  - a mouse click
  - a web page or image loading
  - rolling a mouse over a link, an image, or another hot spot on a web page
  - selecting an element or a field on a form

Prompt() and Events

```
1. <html>
2. <head>
3. <title> JavaScript Events</title>
4. <script type="text/javascript">
5. function greet()
6. {
7.    let name = prompt('Please enter your name',' ');
8.    document.write(`<h2>Hello ${name}!<br/>How are you?</h2>`);
9. }
10.</script>
11.</head>
12.<body>
13.<h2 id ="hello">Who are you?</h2>
14.<button type="button" onclick="greet()">Enter your
name</button>
15.</body>
16.</html>
```

`Prompt()` and Events (2)

Initially, this page has a single line and a button and looks like this:



If the user presses the button, types in `Helmut Lindstrom` at the prompt, and presses OK, the page will now look like this:

Thank You!