Announcements

- Class Web Site:
  - You can find this link at the end of the main passport site
Introduction to Functions

- Function - An entity that completes a particular task for us.
- It can take values necessary to complete a particular task.
- It can return values.
- After completing a task it returns to the point after the call.
- Examples of JavaScript functions.
  - `document.writeln`
  - `alert()`
- You can define your own functions.
- **Example:** `Function.html`
Introduction to Functions

- General form of a function is:

  function name (<comma-separated list of parameters>)
  {
    statements
  }

- Functions are invoked by using the () operator.
- A function can receive values via parameters.
- Some functions may not return a value.
- Some functions may not take any parameters.
- There are other approaches to define functions.
main() Function

- The organization for code dealing with functions will be as specified in the following example.
- **Example:** MainFunction.html
JavaScript (Functions)

- Advantages of functions are:
  - Allows you factor out common code.
  - Allows you to reuse code.
  - Allows you to control the code complexity.

- While designing a solution to a problem you can divide a problem into sub-problems each represented by a function.
Passing Values to Function

- Mechanism used to pass values to function is called pass-by-value
- Parameters – variables in the function that receive data
  - There are normal variables
- Arguments – values you pass to a function
- **Example**: PassingValues.html
- Does it matter how we name the parameters?
Functions Returning Values

- A function can return a value via the return statement
  
  \[
  \text{return expression;}
  \]

- A call to a function that returns a value can be used as an expression

**Example:** (See FunctionReturn.html)

- The function execution terminates when a return statement is executed

- A return statement with no return value terminates the function execution

- Can we return more than one value?
Scope of Variables

- Variables declared in a function are called local variables.
- They are created on entry to the function and destroyed on exit.
- You can use the same name in different functions as they are different variables.
- Variables declared outside of a function are called global variables.
Generation of Random Values

- Example: RandomValues.html
Events

- **Event** – Notification that something has occurred
  - Example of situations that make the web browser generate an event
    - Browser finishes loading a document
    - When the user clicks on a button
    - When the user moves the mouse
    - Others

- **Event handler** (also known as event listener)
  - JavaScript function or code fragment that is executed when a particular event occurs

- **Event handler registration**
  - Associating an event handler with a particular event

- **Example**: EventEx.html
Event-driven Programming

- Normal (control flow-based) programming
  - Approach
    - Start at main()
    - Continue until end of program or exit()

- Event-driven programming
  - Start at main()
  - Register event handlers
  - Await events & perform associated computation

- GUIs (Graphical User Interfaces)
  - Example of event-driven software
Event Handler Attributes for most HTML

- **Mouse Related**
  - **onclick** – mouse button is pressed and released
  - **ondblclick** – mouse button is double-click over element
  - **onmouseover** – mouse moves over element
  - **onmouseout** – mouse moves off element
  - **onmousemove** – mouse pointer is moved
  - **onmousedown** – mouse is pressed down while cursor is over the element
  - **onmouseup** – mouse is released while the cursor is over the element

- **Keyboard Related**
  - **onkeypress** – key pressed and released
  - **onkeydown** – key is pressed
  - **onkeyup** – key is released

- **Other**
  - Keep in mind that there are additional handlers that are specific to certain tags. We will address those later on.
For Loop

- Iteration statement
- General form

\[
\text{for (initialize; test; expression)} \\
\text{statement}
\]

basically equivalent to

\[
\text{Initialize} \\
\text{while (test)} \{ \\
\text{statement} \\
\text{expression} \\
\}
\]

If more than one statement use {} 

- **Example:** ForLoops.html 
- **Example:** ForLoopVariations.html
Arrays

- **Problem** - You need to keep track of the scores of students in a class
  - Declaring and handling 50 variables is not an easy task
  - Arrays come to the rescue

- **Array** – Collection of values that can be treated as a unit or individually.
- You can visualize an array as a set of variables one after another
- There are several ways to define arrays.

```javascript
var scienceScores = new Array(); // Creates an empty array
var mathScores = new Array(3); // Creates an array with 3 entries
var englishScores = [77, 88, 65]; // Creates an array with 3 entries
  // having the specified values
```
Arrays

- To access elements of an array
  - Use the [ ] operator
  - We will use index values **starting at zero** to represent each element

- Accessing array elements

  ```javascript
  mathScores[0] = 70; // Assigning 70 to the first array element
  mathScores[1] = 80; // Assigning 80 to the second array element
  var total = mathScores[0] + mathScores[1]; // reading the first
  // and second elements
  ```

- The array length property defines the number of elements

- Several functions are associated with arrays.
  - sort() – sorts elements of an array
  - reverse() – reverse elements of an array
  - join() – converts elements of an array to string and concatenates them
  - Others

- For loops are frequently use to iterate through arrays

**Example:** ArrayEx.html