Announcements

- You may not use user-defined objects (like the ones you will see in this lecture) in Project #7.
- Make sure that your scores in grades.cs.umd.edu are correct (We will not address any errors after Wed May 9)
Project #7 (Sudoku Validator)

- How do we determine we have found all the digits for a row, column, square?
- How to we dynamically tag each `<input>` element?
- What functions should we have?
Objects

- **Object** – entity that aggregate multiple values in a single unit
- **Property** – Entity associated with an object that has a name and a value. They are like variables (you can store values in them and read values from them)
- **Object** (alternate definition) – unordered collection of properties, where each property has a name and a value.
- A property value can be any data type we have seen, including objects
Objects

- You use the . (period) operator to access an object’s properties
  `<OBJECT> . <PROPERTY>`
- A property value can be any data type we have seen, including objects
- You can create your own objects by either:
  ```javascript
  var myObj = {};
  var myOtherObj = new Object();
  ```
- You can create properties by assigning a value to it (we do not use `var`)
  ```javascript
  myObj.created = "Monday"
  ```
- You can update the property by assigning a new value
- You can delete a property with the delete operator
  ```javascript
  delete myObj.created;
  ```
- You can check for the existence of a property using the “in” operator
- **Example:** `ObjectEx.java`
for/in

- General form

```plaintext
for (property_name in object)
    statement
```

- Can be used to display the properties of an object (See forIn.html)
- `for/in` does not specifies the order in which properties of an object are visited.
- **Example:** `ObjectEx.java`
- The `for/in` does not loop through all the possible properties as some properties are considered non-enumerable. User-defined properties are enumerable.
Objects as a Maps

- We can also view an object as an entity that associates values with strings. How? Let’s first see how we can use the [ ] operator to access properties.
- You can use [ ] operator instead of . (period) operator
  myObj.created → myObj[“created”]

IMPORTANT: Notice that we have a string on the right side (“created”) whereas on the left side it is a property (variable)
- Using [ ] operator can provide a nice alternative to add properties to an object dynamically (when the program is executing).
- **Example:** AddingProperties.html
Global Object

- **Global object** – created by JavaScript interpreter when it starts up. Interpreter initializes the Global object with predefined values and functions. For example, parseInt, Infinity, etc.
- **Top-level code** – JavaScript that does not belong to a function
- **Global variables** – variables in top-level code
  - Global variables are properties of the Global object. When you define a variable outside any function you are defining a global variable (a property of the global object).
  - You should avoid using global variables in your code.
- In client-side JavaScript the Window object (window) represents the global object for all JavaScript code present in the browser window.
  - You can use the keyword `this` to refer to the Global object. Keep in mind that inside a function `this` does not refer to the global object.
- **Example**: GlobalObject.html
Functions as Data

- In JavaScript functions are considered data.
- That means they can be assigned to variables, stored as properties of objects or elements of arrays, passed as arguments to functions, etc.
- **Example:** FunctionsAreData.html
- Where have we seen this?
Object-Oriented Programming

- Two ways to design solutions
  - Procedural (functional) approach
  - Object-Oriented approach
- Object-Oriented design - Problem solving strategy where we visualize a design as a collection of objects interacting with one another.
- An object in this context is an entity with values and operations.
- We can use JavaScript to create object-oriented solutions to problems. Object properties in JavaScript allow us to add values and operations (methods) to an object.
- Object-oriented programming has several advantages
  - Groups together values and methods into a package which increases modularity
  - Promotes reusability of code
Break Statement

- Break causes the innermost loop to exit immediately.
- It is only legal if it appears inside a loop or switch statement.
- **Example:** Break.html
- Break can also be used with a label.
Recursion

- A recursive function is one that “calls itself”.
- One approach to see this process is to visualize that, instead of the function calling itself, the function calls another function that performs the same task as the original one.
- Nature has several examples of recursive phenomena.
- A typical recursive example is the computation of the factorial.
- Definition of factorial (non-recursive)
  \[ n! = n \times (n-1) \times (n-2) \times \ldots \times 1 \]
- Definition of factorial (recursive)
  \[ n! = n \times (n-1)! \]
- Let’s implement a recursive factorial function.
- Computation of the fibonacci series is another example of recursion
- Let’s draw a call tree that illustrates the set of calls that are taking place.