Announcements

- Check class announcements daily
- You must implement programming projects by yourself
Reviewing One-Dimensional Arrays

- How do we define an array?
- How do we represent arrays?
- How can we access the elements of an array?
- What can we do with the elements of an array?
- Which iteration statement is frequently used with arrays?
- Fundamental loop you should remember
  
  ```java
  for (k = 0; k < a.length; k++) {
  }
  
  where a is an array
  ```
- Arrays are created in a memory area called the heap
- Array variable holds address of array
- How are array elements accessed?
- We can create aliases to arrays via assignments
- Arrays are objects
  - Object → Entity that has values and operations (functions)
Passing Arrays to Functions

- Let’s review how we pass numbers to functions
- How we pass arrays to functions?
  - **Example:** PassReturnArrays.html
- **Memory Diagram**
  - Tool we will use to illustrate the associations between variables and entities (e.g., objects, arrays, etc.)
NaN

- **NaN** → Not-A-Number (Same as `Number.NaN`)
  - Unequal to any number including itself
  - Use `isNaN` function → determines (returns true or false) whether an argument is not a number. It attempts to convert the argument to a number
    - The following comparisons return false
      - `NaN == NaN`, `NaN === NaN`

- **Example:** `NaN.html`
null

- What is null?
  - Represents no value
  - Represents no address

- Example: Null.html
- When can use null?
  - Example: ValidityCheck.html
null and undefined

- null → indicates no value
- undefined
  - Value associated with uninitialized variables
  - Value associated with object properties that do not exist
- == considers null and undefined equal
- === considers null and undefined different
Parsing Strings into Numbers

- **Number**
  - Returns `NaN` if the argument does not represent a well-formed numeric literal; otherwise a number

- **parseFloat**
  - Takes a string as argument and converts the string to a floating point number. It stops parsing the string once it founds a character that cannot be part of a floating point number
  - Returns `NaN` if a number cannot be generated
  - Leading and trailing spaces are allowed

- **parseInt**
  - Takes two parameters: a string and a radix (defaults to 10)
  - With a string parameter behaves like parseFloat but returning an integer

- **Example:** ParseStringNum.html
typeof Operator

- **Syntax:**
  - typeof `operand`
  - typeof `(operand)`

- **Semantics:**
  Returns a string indicating the type of the operand

- **Example:** TypeOf.html
**eval**

- Allow us to evaluate an expression
- **Example:** Eval.html
Exercise

- Let’s consider writing a JavaScript program that defines a web page for a user based on provided values (e.g., name, personal information, photo, etc.)