

Announcements

- ❖ Class Web Site:
- ❖ Check class announcements daily

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?
 - ❖ Everything we can do with a variable
- ❖ Which iteration statement is frequently used with arrays?
- ❖ Fundamental loop you should remember
 - `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.)
- ❖ Let's see different aspects of arrays via memory diagrams
 - ❖ How to create aliases
 - ❖ How to pass arrays to functions

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
- ❖ To remember → **!isNaN()** allow us to determine whether an expression is a number
- ❖ **Example:** NaN.html

null

- ❖ What is null?
 - ❖ Represents no value
 - ❖ Represents no address
 - ❖ `var a = null;`
- ❖ **Example:** Null.html
- ❖ When can use null?
- ❖ **Example:** ValidityCheck.html
 - ❖ Notice that using Number was not necessary; Why?

null and undefined

- ❖ null → indicates no value
- ❖ undefined
 - ❖ Value associated with uninitialized variables
 - ❖ `var x; // in a function`
 - ❖ When a function that is expected to return a value does not return one (IMPORTANT case)
 - ❖ 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

❖ **parseInt**

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

❖ **parseFloat**

- ❖ Takes a string as argument and converts the string to a floating point number. It stops parsing the string once it finds 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

❖ **Example: ParseStringNum.html**

❖ **Applications**

- ❖ Cleaning up data provided by prompt (e.g., 6Ft)

typeof Operator

- ❖ **Syntax:**

`typeof operand`

`typeof (operand)`

- ❖ **Semantics:**

Returns a string indicating the type of the operand

- ❖ **Example:** `TypeOf.html`

- ❖ Possible use

- ❖ Identifying the type of a parameter (e.g., is it a number or an array?)

eval

- ❖ Allow us to evaluate an expression
- ❖ **Example:** Eval.html
- ❖ Notice it can evaluation any JavaScript code
 - ❖ Provide “alert(‘Hello’)” to above code