Week 5

Week 4 Review

- A callback function facilitates asynchronous programming by implementing a polling strategy. [true][false]
- The array .map method uses a function to change an array [true][false]
- The difference between forEach and map is …

Associative Arrays

- The index to an array can be a string!
- This creates a key-value pair

```
flowers = [];
flowers["daisy"] = 12          //daisy is the key, 12 is the value
flowers["rose"] = 15
flowers["carnation"] = 8
document.write(flowers["rose"])  //displays 15

for (key in flowers)
   document.write(key + " $" + flowers[key] + " - ")
//Output:  daisy $12 - rose $15 - carnation $8
```

Try it

- Create an associative array of (5) cities and the state they are in
  The cities should be the keys.
- Display cities only
- Create a function to locate a city and return its state using the array as a parameter to the function

Simple Objects

- An object can be a set of name - value pairs:

```
var flowers = {
      daisy: 12,
      rose: 15,
      carnation: 8
}
document.write (flowers['rose’])  //displays 15
document.write (Object.keys(flowers))
// displays: daisy,rose,carnation
```

Or is it an associative array?

```
flowers["tulip"] = 7;
document.write (Object.keys(flowers))
```

Try it

- Create a simple object using the same city / state data
- Display cities only using Object.keys()
- Rewrite your function to check if a city is a valid key and then return its state.

Destructuring an Array

- Destructuring assigns parts of the array to other variables
- Given:
  x = [33,5,44,63];

  We could do:
  first = x[0];
  second=x[1];
  third=x[2];
- Using destructuring:
  var[first, second, third] = x;

Try It - 1

Destructuring can be a mechanism to quickly create several variables.

Use destructuring to assign the age of Tom, Bill, Pat, Jen, and Tess the values: 45, 34, 28, 29, and 38 respectively.

Try It -2

You can get the current date and time using
new Date()
there are several methods to extract the various parts of the date
see: https://www.w3schools.com/jsref/jsref_obj_date.asp

- Create an array consisting of the numeric month, day, and year
- Assign the array to variables month, day, year using destructuring
- Display in the form 1/1/2020
- How would you assign the month and year only?

The Spread operator ...

- The spread operator takes a subset of an array
- Given

```
x = ['apples', 'pears', 'berries', 'oranges’];
var [pie, ...fruits] = x;
document.write (fruits);    // pears,berries,oranges
```

- Or combine two arrays

```
fruitPies= ['apple','blueberry','peach’];
creamPies=['lemon','chocolate','banana’];
allPies = [...fruitPies,...creamPies];
```

Try it - 1

- The spread operator can be used in conjunction with a function parameter Follow these instructions to create a function that uses the spread operator to create a function with a variable number pf arguments.
  - Create a function with two arguments: the first is a value, the second uses the spread operator
  - Initialize sum to the start value
  - Use forEach with the second argument (which is an array) to iterate through the values in the array and add them to the start value
  - Display the sum at the end

Try it - 2

- You have a list of students in class 1 (Jane, Alex, Harvey) and a list from class 2 (Seymour, Emily, Frank).
- Pass them to a function as two strings with each name separated by a space (ex "Jane Alex")
- In the function split the strings into two arrays (use split)
- Combine into one array using the spread operator
- Sort the list of names
- Use .map to display them as a numbered list

Putting it Together

```
function findLess(arr, target)
{
	matches=[];
	for (var i=0; i<arr.length; i++)
	{
		if (arr[i] < target)
		matches.push(arr[i]);
	}
	return matches;
}
//find the largest number that is also less than 35
x = [33,5,40,44,3,40];
var [lastMatch] = findLess(x, 35).sort().reverse();
```

Try It

- Rework findLess using forEach

But wait! There's a method for that.

- .filter() — create a new array based on a filter
- arguments — a function that provides the criteria for which items to "keep"

- For example, the findLess function could be simplified to:

```
function findLess(array, target)
{
      return array.filter(item => item < 35);
}
```

Objects using JSON

- JavaScript Object Notation
- For data representation and transmission
- Comprised of key - value pairs
- Text based

- Minimal and portable
- Based on conventions seen in many languages
- Code for parsing JSON is available in many languages

Simple JSON object

```
{
    "first name"    : "Julie",
    "last name"     : "Smith",
    "course"        : "Web Apps",
    "grade"         :  92
}
```

Notes:

- "key" : value
- Start and end with { }
- Keys are quoted
- Values can be strings, numbers, booleans, and null or an array or other object containing these types
- Commas between pairs
- Validator: https://jsonlint.com/

Using an array for a value

- Use [ ] notation to indicate an array

```
{
    "first name"    : "Julie",
    "last name"     : "Smith",
    "course"        : "Web Apps",
    "grades"        :  [88, 95, 91, 92]
}
```

Nesting JSON objects

- A JSON object can contain other objects
- Example: The student name can be another JSON object

```
{
    "name" : {
                    "first"  : "Julie",
                    "last"   : "Smith"
    }
    "course"        : "Web Apps",
    "grades"        :  [88, 95, 91, 92]
}
```

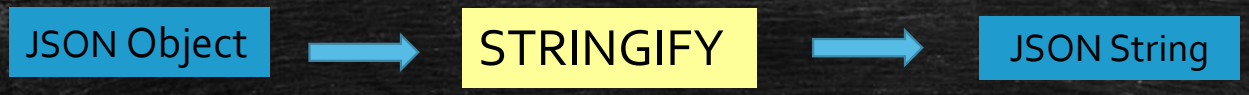
Array of JSON objects

- You can create a collection / array of JSON objects using the [ ] notation

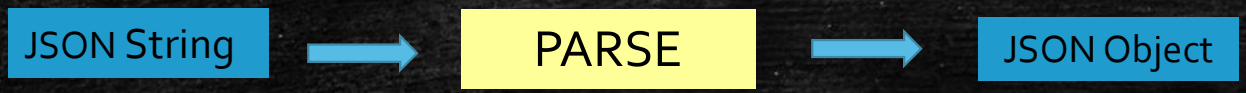
```
[
  {“id” : 1,  “type” : “rose”},
  {“id” : 2,  “type” : “carnation”},
  {“id” : 3,  “type” : “sunflower”}
]
```

Built in JS Functions: parse() & stringify()

- **stringify**() : serializes a JSON object
  - Turns a JSON or JavaScript Object into a JSON string



- **parse**() : parses a JSON string
  - Turns a JSON serialized string back into a JSON object



Example

- student = {
  "name":"Suzie",
  "course":"Web Apps"
  }

- strStudent = JSON.stringify(student);
  // result is the string: {"name":"Suzie","course":"Web Apps"}

- objStudent = JSON.parse(strStudent); //restores to an object

Is the “simple object” a JSON?

These will get the same result …

```
var flowers = {
      daisy: 12,
      rose: 15,
      carnation: 8
      }
console.log(flowers["daisy"]);
```

```
var flowers = {
       "daisy": 12,
       "rose": 15,
       "carnation": 8
       }
console.log(flowers["daisy"]);
```

Using stringify() with the simple object

```
var flowers = {
        daisy: 12,
        rose: 15,
        carnation: 8
}
```

Turn it into a JSON string

```
s = JSON.stringify(flowers)
document.write(s)
// output:
{"daisy":12,"rose":15,"carnation":8,"tulip":7}
```

And back to an object …

```
obj= JSON.parse(s)
document.write(obj['daisy’])
// output
12
```

Objets in JavaScript

- A class is the definition for a set of objects
- Create a class using the keyword "class" or via a function
- Classes in JavaScript have:
  - Properties: Characteristics, State
  - Methods: Things the object can do
  - Events: Things the object can respond to

- Use "new" to create an *instance* or *object* from a class (sometimes there are shortcuts that do not require *new*)
  - *new* implicitly calls the *constructor* for the class

Creating Objects using Functions

- The function defines the class and is the *constructor* method.
- Use the keyword "this" to refer to the current instance

```
function Rectangle(len, wid)
{
    this.length = len;
    this.width = wid;
}
```

- Instance the class using *new*:

```
r = new Rectangle(3,4);
```

- Access the data members using the dot notation

```
area = r.length * r.width;
```

Try it

- Create a class in JavaScript called Flower
- Flower should have two properties: name and price, that are set in the constructor
- Instance the three flowers we created previously.
- Display the name and price of one of the flower objects.

Methods – help the object to do something!

- The real utility of objects is when it has methods
- A method is a function that uses the keyword "this" to refer to members of the object.

```
function area()
{
   return this.length * this.width;
}
```

- Attach the method to the class by assigning it in the constructor

```
function Rectangle(len, wid)
{
     this.length= len;
     this.width= wid;
     this.area= area;
}
```

Call the method using the dot notation

```
r = new Rectangle(3,4);
rectArea = r.area();
```

Try it

- Modify the Flower class:
- Add a method called show() which returns a string with the name and price (ie: "Rose: $9")
- Display one of the flower objects using the show() method

Example: Array of Objects

- Create an array of the three Flower instances

```
flowers= [daisy,rose,carnation];
```

Or

flowers = [new Flower('rose', 9), new Flower('carnation', 7), new Flower('daisy', 3)]

- Use .map() to create a new collection with the Flower data

```
collection = flowers.map(f =>f.show());
```

- Display the collection on separate lines:

```
document.write (collection.join("<br />"))
```