Arrays & Objects

User Interfaces

420-WC4-AB

Arrays

- Allows you to store a collection of variables
- Usually stores the same type of data but it does not have to!
- Not a real data type.
  - Specialized object that contains date referred to by index value
- Create them with or without content
- Size and content can dynamically change

Creating Arrays

- Create array with no content

```
let cats = new Array();
let dogs = [];
```

- Create array content

```
let odds = new Array(3,5,7);
let countries = new Array("England", "Scotland");
let browsers = ["Safari", "Chrome", "FireFox", "Opera"];
let multi = [ [1,2,3], [4,5,6], [7,8,9] ];
let confused = ["shoe", 27, null, [42,17,19], Math.PI];
```

Length

```
let dogs = ["Pug", "Husky", "Poodle", "Hound"];
```

- The length property contains the number of elements in the array.

```
console.log(dogs.length); // 4
```

Array Indexes

```
let dogs = ["Pug", "Husky", "Poodle", "Hound"];
```

- Each element has an index.
- The first element is always at index 0.
- The last element is always at index length-1

```
dogs[0]; // "Pug"
let elem =  dogs[2]; // elem contains "Poodle"
dogs[ dogs.length – 1 ]; // "Hound"
```

Array Indexes – Find Specific Item

- Returns the first index at which a specified element it located
- Returns -1 if element is not found.

```
let dogs = ["Pug", "Husky", "Poodle", "Hound"];

console.log(dogs.indexOf('Poodle'));     // 2
console.log(dogs.indexOf('Husky', 2));  // -1
console.log(dogs.indexOf('PUG'));        // -1
```

Add New Item at Beginning

```
let dogs = ["Pug", "Husky", "Poodle", "Hound"];


dogs.unshift("Beagle");
// Beagle, Pug, Husky, Poodle, Hound


dogs.unshift("Boxer", "Corgi");
// Boxer, Corgi, Beagle, Pug, Husky, Poodle, Hound
```

Remove Item at Beginning

```
let dogs = ["Pug", "Husky", "Poodle", "Hound"];

let removedItem = dogs.shift();
// Husky, Poodle, Hound
// removedItem contains Pug

removedItem = dogs.shift();
// Poodle, Hound
// removedItem contains Husky
```

Add New Item at End

```
let dogs = ["Pug", "Husky", "Poodle", "Hound"];

dogs.push("Terrier");
// Pug, Husky, Poodle, Hound, Terrier

dogs.push("Whippet", "Greyhound");
// Pug, Husky, Poodle, Hound, Terrier, Whippet, Greyhound

dogs[dogs.length] = "Collie";
// Pug, Husky, Poodle, Hound, Terrier, Whippet, Greyhound, Collie
```

Remove Item at End

```
let dogs = ["Pug", "Husky", "Poodle", "Hound"];

let removedItems = dogs.pop();
// Pug, Husky, Poodle
// removedItem contains Hound

let removedItems  = dogs.pop();
// Pug, Husky
// removedItem contains Poodle
```

Alter Array - *Splice*

- The splice method changes the contents of an array by removing or replacing existing elements and/or adding new elements

```
array.splice(start, [deleteCount, [item1, [item2,[....] ] ] );
```

- start: origin with 0, pick up the position you want to splice, think array index
- [deleteCount] optional value for how many elements you want to delete
- [item*X*] optional value to add elements to array

Alter Array - *Splice*

- Start at index 2, remove 0 items and add following items

```
let dogs = ["Pug", "Husky", "Poodle", "Hound"];

dogs.splice(2,0,'Chihuahua', 'Bulldog');
// Pug, Husky, Chihuahua, Bulldog, Poodle, Hound
```

Alter Array - *Splice*

- Start at index 1, remove 3 items and add nothing

```
// Pug, Husky, Chihuahua, Bulldog, Poodle, Hound

let removedItems = dogs.splice(1,3);
// Pug, Poodle, Hound
// removedItems = [ Husky, Chihuahua, Bulldog ]
```

Merge Arrays - *concat*

- Merge two or more arrays into a new array
- Does not change existing arrays

```
let dogs = ["Pug", "Husky", "Poodle"];
let cats = ["Siemese", "Ninja"];
let pets = dogs.concat(cats);
//pets = [Pug, Husky, Poodle,Siemese, Ninja]
```

Array to String - *join*

- Creates new string by concatenating all the element of an array
- Define a parameter to the join to indicate what will separate each element of the array.

```
let dogs = ["Pug", "Husky", "Poodle"];

console.log (dogs.join(', ') );// Pug, Husky, Poodle
console.log (dogs.join('#') ); // Pug#Husky#Poodle
console.log (dogs.join(' et ') );  // Pug et Husky et Poodle
```

String to Array - *split*

- Create a new array from a string
- Define a parameter to the specified separator string to indicate where to make each split

```
let farm = "Chicken / Horse / Pig";
animals = farm.split(' / ');
console.log ( animals );// [ Chicken, Horse, Pig]
animals = farm.split('e');
console.log ( animals );// [ Chick, n / Hors,  / Pig]
```

Sorting - *sort*

- Sorts the current array based on it's string value (UTF-16 code unit)
- It will change the existing array

```
let months = ['March', 'Jan', 'Feb', 'Dec'];
months.sort();
console.log(months);
//  ["Dec", "Feb", "Jan", "March"]
```

Sorting - sort

- Remember that it sorts as string values.

```
let nums = [1, 30, 4, 21, 100000];
nums.sort();
console.log(nums);
//  [1, 100000, 21, 30, 4]
```

Reversing an Array - *reverse*

- Revers an array. First element, becomes the last. Last element becomes first
- It will change the existing array

```
let months = [Jan', Feb', Nov', 'Dec'];
months.reverse();
console.log(months);
//  [Dec, Nov, Feb, Jan]
```

Loops – for in

- The for in loop should used with arrays or objects, it automatically goes to the next element when the loop completes.
- The loop will occur once for every *element* of the *array or object*

```
let dogs = ["Pug", "Husky", "Poodle", "Hound"];
let allDogs = "";
for (let x in dogs) {
   allDogs += dogs[x];
}
// allDogs contains "PugHuskyPoodleHound"
```

Objects

- Objects let us store a collection of properties.

```
let objectName = {
  keys: value,
  keys: value
};
```

Objects

```
let teacher = {
  hometown: 'Montreal',
  eyes: 'Brown',
  likes: ['ninja cats', 'code'],
  birthday: {month: 11, day: 27}
};
let dog = {
  name: 'Shadow',
  breed: 'Jack Russel',
};
```

Accessing Objects

- You retrieve values from objects using the dot notation

```
let dog = {
  name: 'Shadow',
  breed: 'Jack Russel',
};
// RECOMMENDED to use using dot when possible
dog.name; // Shadow
dog.breed; // Jack Russel
```

Accessing Objects

- Sometimes it's not possible to use dot notation, so we can use bracket notation.
- Best practice to use dot notation as much as possible.

```
let passingGrade = {
   kindergarden:  0,
   11: '60%',
};
passingGrade.kindergarden; // 0
passingGrade['11']; // 60%
passingGrade[11]; // ERROR – unknown variable 11
```

TRY TO AVOID

Changing Objects

- Change existing values

```
dog.name = "Rex";
```

- Create new value

```
dog.age = 13;
```

- Remove value

```
delete dog.breed;
```

```
//before
   let dog = {
      name: 'Shadow',
      breed: 'Jack Russel',
   };
// after
   let dog = {
      name: 'Rex',
      age:  13,
   };
```

Object Methods

- Objects can also contain functions.
- We call these function using the same dot notation along with parentheses

```
let dog = {
  name: 'Shadow',
  breed: 'Jack Russel',
  bark: function(){
    console.log("Woof");
  }
};
dog.bark(); // Woof
```

Object Methods

```
let dog = {
  name: 'Shadow',
  breed: 'Jack Russel',
  bark: function(counter = 1){
    for (let i = 0; i < counter; i++){
      console.log("Woof");
    }
  }
};
dog.bark(); // Woof
dog.bark(4); // Woof Woof Woof Woof – (on separate lines)
```

Reusable Object

- What if you want to build several objects in with the same definition?
- Built using ordinary function that is referred to as a *constructor*

- Allows you to define an object pattern and reuse it.
- Usually starts with an uppercase letter
- Uses **this** keyword to define properties and methods
  - ***this.****key* = …
  - ***this.****method* = *function()* { … }

Create Reusable Object

```
function planet(name, diameter, distanceFromSun) {
    this.name = name
    this.diameter = diameter
    this.distanceFromSun  = distance
    this.showDetails = function() {
      return this.name + " has a diameter of "
                + this. diameter;
    }
}
```

Create Reusable Object

```
let Mercury = new planet("Mercury","3100 miles", "36 million miles");
let Venus = new planet("Venus", "7700 miles", "67 million miles");
let Earth = new planet("Earth", "7920 miles", "93 million miles");

Mercury.distanceFromSun;  //  36 million miles

Earth.showDetails();  // Earth has a diameter of 7920 miles
```

Iterating / Looping

```
let dog = {
  name: 'Shadow',
  breed: 'Jack Russel',
};

for(let x in dog ){
    console.log(dog[x]);
}
```

- We have to use bracket notation because we are accessing our object key using a variable

```
// Shadow
// Jack Russel
```

Questions

I'm always looking, and I'm always asking questions. – Anne Rice