Maydm

Web Development

Maydm

Day 11: Programming Paradigms and ES6

Classes, Scopes, and Arrows

Maydm

Warm up!

Maydm

Today's Schedule

Morning:

- Review of JS so far
- Functional Programming
- OO Programming
- Classes
- Constructors
- Properties
- Methods
- Project: Build a Zoo

Afternoon:

- Intro to ES6
- Const & let vs var
- Scope
- Arrow Functions
- Reading Documentation

Maydm

JavaScript So Far

- Comments
- Variables
- Data Types
- Operating on Variables
- Functions
- Conditionals
- Arrays
- Methods
- Loops
- Objects & JSON
- The DOM and Classes

Maydm

Functional Programming

Functional programming is a programming paradigm that avoids changing state and mutable data.

It's programming with expressions (functions)

A function's return value depends on its arguments. Calling a function with the same value for an argument will always have the same output.

Imperative vs. Functional Programming

```
const arr = [1, 2, 3, 4, 5]
function getOdds(arr){
  let odds = [];
  for(let i = 0; i < arr.length + 1; i++){
    if(i % 2 !== 0){
      odds.push(i)
    };
  };
  return odds
};

console.log(getOdds(arr))
// logs [1, 3, 5]
```

Maydm

Imperative vs. **Functional** Programming

```
functional getOdds2(arr){
  Return arr.filter(num => num % 2 !== 0)
}

console.log(getOdds(arr))
// logs [1, 3, 5]
```

Maydm

Imperative vs. Functional Programming

In the first example we must state what to do each step of the way in order to get our desired result. In the second example we use methods to express our intention. The .filter method handles everything we expressed in the first example.

As an analog example, this is the difference in setting a car's thermostat to a fan setting to blow cold or hot air and adjusting the thermostat to a specific degree and the fan to auto.

Maydm

Object Oriented Programming

Object-oriented programming is a programming paradigm in which data structures become objects. In OOP, objects can inherit attributes from other other objects.

- Objects are self-contained entities that consist of both data and procedures to manipulate the data

Maydm

JavaScript Properties

- JavaScript Properties are the values associated with an object.
- An object is a collection of unordered properties.
- Properties can be changed, added, and deleted.
- Properties can be accessed using “dot” or “bracket” notation

```
objectName.property

objectName[“property”]

objectName[expression]
```

Maydm

Classes

Classes

- Category of objects
- Defines common properties of objects that belong to that class

Maydm

Constructors

Constructors are special methods used for creating an object created in a class

- Constructors can be initiated using the `new` keyword or `extends`

Maydm

Constructor Example

```
class Polygon {
  constructor() {
    this.name = “Polygon”;
  }
}

var poly1 = new Polygon();

console.log(poly1.name);
// “Polygon”
```

Maydm

Constructor Example with Extends

```
class Square extends Polygon {
  constructor() {
    super();
    this.name = "Square";
  }
}

var poly1 = new Polygon();
console.log(poly1.name);
// “Polygon”
```

Maydm

Constructors

Constructors are special methods used for creating an object created in a class

- Constructors can be initiated using the `new` keyword or `extends`

Maydm

Keyword “this”

In JS, “this” is a reserved keyword. **It refers to the object it belongs to**. In the below example, **this.name** refers to the name property of the object being created.

```
class Polygon {
  constructor() {
    this.name = “Polygon”;
  }
}
```

Maydm

Keyword “this”

In this Polygon class, **this.name** will receive whatever argument is passed in with a new object. In this case, it’s name will be ‘square.’

```
class Polygon {
  constructor(name) {
    this.name = name;
  }
}

var square = new Polygon(‘square’);
```

Maydm

Keyword “this”

The keyword **this** can only be accessed after a constructor method is called.

```
class Polygon {
  constructor(name) {
    this.name = name;
  }
}
```

Maydm

Methods

JavaScript methods are actions that can be performed on objects

You can think of methods as ready-made function statements.

In fact, a JavaScript method is a property containing a function definition.

Building a JS Object Method

```
var car = {
  make: “Tesla”,
  model: “3”,
  vin: “1234abcd”,
  makeAndModel: function() {
    return this.make + “ “ + this.model;
  }
};

car.makeAndModel();
```

In this example, `this` refers to the owner of the function, i.e. the “car” object.

Methods are Functions Stored as Object Properties

Remember that Objects store property: value pairs.

In the example in the previous slide the function is stored as a value inside of the property "makeAndModel"

Property	**Value**
make	Tesla
model	3
vin	1234abcd
makeAndModel	function() {return this.make + " " + this.model;}

Maydm

Accessing Methods

Access methods using the syntax of:

objectName.methodName()

The property storing a function will execute the function when it is invoked using the object name and ().

car.makeAndModel() will invoke the function inside the property called makeAndModel

Calling the property without the () will return the function definition.

JavaScript’s Built-In Methods

JavaScript has several built in functions that can be using without creating user definitions. Additional libraries, such as jQuery, can be incorporated into JavaScript programs to take advantage of additional built-in methods.

Popular vanilla JavaScript methods include toUpperCase() and toLowerCase()

```
var firstString = “Hello, World!”;
var capsString = firstString.toUpperCase();
var smallString = firstString.toLowerCase();
```

Try this in your console to see the results.

Maydm

Add a New Method to an Established Object

Developers can add new methods to established objects by assigning a function to an object name and a new property.

```
Car.passengers = “5”; //adds a new “passengers”
                        property with a value of 5.
car.passengerWeight = function() {
  return this.passengers * 160;
};

car.passengerWeight();
```

Try this in your console to see the results.

Maydm

How to Identify Methods

Identifying a JavaScript method is fairly simple. If you see a statement in dot or bracket notation with a () at the end then that is most likely a method.

Maydm

Methods & “this”

In methods, the keyword **this** always refers to the object that owns the method. Here, the method makeUpperCase takes **this** (the object) and returns it as upper case.

```
class Animal {
    constructor(type) {
        this.type = type;
    }
    makeUpperCase(this) {
        return this.toUpperCase();
    }
}
```

Maydm

Scope

There are two categories of scope in JavaScript:

- Local Scope
- Global Scope

Functions create scope in JavaScript. Variables outside of functions are said to have global scope. Anything inside that program can access and edit their values. Variables inside of scopes have a scope that is local to that function. Local variables cannot be accessed outside of that function.

Maydm

Scope Example

Try this snippet in your console to experience scope.

```
var myVar = “I’m global”;

function myFunction() {
  var myVar = “I’m local”;
  return myVar
}

console.log(myVar);
myFunction();
```

Maydm

Lessons about Scope

As you found in that example, the same variable name produces two different results because of Scope. So, we can name variables similar to one another, but this is high inadvisable as it can make debugging more complicated.

Global variables live forever and are suspect to manipulation.

Local variables live inside a function and are deleted once the function is complete.

Objects and functions also have scope.

Previously Undeclared Variables are Global

Assigning a value to a variable that has not been declared creates a global variable.

```
function globalFunction() {
  newVariable = “Global Variable”;
  return newVariable;
}

console.log(newVariable); // not defined
globalFunction();         // ”Global Variable”
console.log(newVariable); //”Global Variable”
```

Maydm

Scope & “this”

The keyword **this** is always in scope. To test this out, try typing it in the console.

What does it return?

Project Build a Zoo

Build a zoo using what you know about JavaScript Objects. Find the starter code in today's repository to get your project started.

Introduction to ES6

JavaScript is also known as ECMAScript. Standards are maintained by ECMAScript Language Specifications. ECMAScript 2015 introduced ES6 which provided a number of new features for JavaScript developers.

In 2015, a transpiler called Babel was required for developers to take advantage of ES6 features in web browsers. However, Since that was four years ago, most browsers are ES6 compatible. In fact, ECMAScript has accepted two new standards. The most current version of JavaScript at this time is known as ES9 or ECMAScript 2018

ES6 Features

Some of the most popular ES6 features include:

- let
- const
- Arrow functions
- Promises

let

We just learned that there are two types of scope in JavaScript. That's not quite true. ES6 introduced Block Scope with the `let` and `const` keywords.

In **Block Scope** every block of code can be its own scope. As discussed previously, functions can have their own scope, but now other blocks such as if-statements and loops can have their own scope.

let initiates block scope (var never does) and will not overwrite a var by the same name. let assignments are limited to its block.

let can have global or block scope.

Maydm

let Example

```
var x = 10;
console.log(x);
{
  let x = 2;
  console.log(x);
}
console.log(x);
```

Maydm

let loop Example

```
var i = 5;
for (var i = 0; i<10; i++) {
  console.log(i);
}
console.log(i);

REFRESH BROWSER

let i = 5;
for (let i = 0; i<10; i++) {
  console.log(i);
}
console.log(i);
```

Maydm

const

const is very similar to let except const variables cannot be reassigned.

const initiates block scope similar to let.

const variables must be assigned a value when they are declared.

const creates a consistent, immutable reference to a value.

Properties in const objects and values in const arrays can be changed and added, but constant objects and arrays cannot be reassigned

Maydm

const Example

```
const zero = 0;
zero = “zero”; // Will return an error
Zero = zero++  // Will return an error; immutable
```

const Block Scope Example

```
var x = 5;
console.log(x);

{
  const x = 2;
  console.log(x);
}

console.log(x);
```

Maydm

const Assignments

```
INCORRECT ASSIGNMENT
const zero;
zero = 0;

CORRECT ASSIGNMENT
Const zero = 0;
```

Maydm

const Assignments

```
INCORRECT ASSIGNMENT
const zero;
zero = 0;

CORRECT ASSIGNMENT
Const zero = 0;
```

Maydm

const Object properties are mutable

```
const car = {make:”Tesla”,model:”3”,color:”black”};
// We can change and add properties
car.color = “white”;
car.owner = “Karanja”;

// We cannot reassign the const object
car = {make:”Kawasaki”,model:”versys”,color:”red”};
// Throws an error
```

Maydm

const Array values are mutable

```
const city = [“Madison”,”Milwaukee”,”Chicago”];
// change and add an array element
city[0] = “St. Paul”;
city.push(“San Francisco”);

// Cannot reassign the const array
city = [“Seattle”,”San Diego”, “Los Angeles”];
// Throws an error
```

Maydm

Arrow Functions

Acts mostly like a standard function, but with a more concise syntax

- No need for the **function** or **return** keywords or the curly brackets.
- Do not have their own **this** so they are not suited for defining methods
- Must be defined before they are used
- Prefer to use const w/ arrow functions because expressions are constant
- NOTE: Can only omit “return” and curly brackets if the function is a single statement. You can choose to use them in simpler functions

Maydm

Arrow Function Example

```
ES5 Function

var x = function(x, y) {
  return x * y;
}

x(4,5) //Refresh browser before building Arrow Func

ES6 Arrow Function

const x = (x, y) => x * y;

x(4,5)
```

Dissecting Arrow Function Example

```
var x = function(x, y) { return x * y; }

ES6 Arrow Function

const x = (x, y) => x * y;
```

Define Function

Omit “function” keyword,
show args in ()

Arrow replaces return statement

Maydm

Arrow Functions with more than one Statement

Remember that arrow functions with more than statement must use the return keyword and curly brackets

```
const x = (x, y) => { console.log(x); return x * y };
x(4,5)
```

Function keyword is not needed

show args in ()

Because function has more than one statement we must use return keyword and curly brackets

Maydm

Reading Documentation

Learning to read documentation is a critical skill of all developers. We read documentation to gain a deeper understanding of APIs, methods and how to write proper syntax in our programs.

We've explored some documentation already such as Bootstraps and jQuery's. We've also used MDN's documentation from time to time. In the next few slides we'll work together to explore documentation in greater detail.

Maydm

Reading Documentation

Learning to read documentation is a critical skill of all developers. We read documentation to gain a deeper understanding of APIs, methods and how to write proper syntax in our programs.

We've explored some documentation already such as Bootstraps and jQuery's. We've also used MDN's documentation from time to time. In the next few slides we'll work together to explore documentation in greater detail.

For further reading, there's an excellent blog article by Cassandra Wilcox called A Beginner's Guide to Using Developer Documentation

Maydm

Reading Markdown Documentation

Commonly, Documentation is written using Markdown syntax so let's start our documentation journey there.

Google "Markdown Documentation" and find the hit titled "Markdown Cheetsheet" located on github.com

Answer the following questions:

- How many Headers are there?
- How do you create a link?
- How do you link to an image?
- How do you create blockquotes?

Maydm

Reading Markdown Documentation

Let’s look at documentation for something a little more challenging. Google “JavaScript Arrays” and select the MDN link.

Practice by creating an array of three of your heroes in the console and then practice the syntax provided in the documentation

- How can we loop over an array?
- How can we remove an item from the front of an array?
- How can we find the index of any item in the array?
- What is the relationship between `length` and numerical properties?
- How many mutator methods are available for arrays?
- Is support available for Array.prototype.filter in Chrome and MS IE?

Maydm

Reading Markdown Documentation

Practice reading JavaScript Documentation by looking up a JavaScript keyword on MDN or W3Schools. Write down three things about that keyword along with examples.

For example, you might read about functions, promises, loops, if statements or Objects.

Maydm

Reflection

Write in your journal about how you feel or what you learned today.

Prompts:

- What are some differences between object-oriented programming and functional programming
- How can classes be useful when programming with objects?
- What was challenging about reading documentation? Why is it beneficial for developers to be able to read documentation?

Maydm