JavaScript Objects

COEN 161

What is an Object?

- If JavaScript variables are containers for data values

```
var car = "Fiat";
```

- Then objects are containers for many values
- Objects are still variables that can be *contained*

```
var car = {type:"Fiat", model:"500", color:"white"};
```

- Each value in an object is written in a name:value pair
  - “JavaScript objects are containers for named values.”

Object Properties and Methods

- Each name:value pair in an object is called a **property**
- Objects also have properties that can perform actions on the object, these are called **methods**
    - Methods are stored in properties as *function definitions*
- “JavaScript objects are containers for named values called properties or methods.”

Object Properties and Methods

- Each name:value pair in an object is called a property

Properties	**Methods**
car.name = Fiat	car.start()
car.model = 500	car.drive()
car.weight = 850kg	car.brake()
car.color = white	car.stop()

Object Definition

- Objects can be defined using the *object literal* notation

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

var person = {

    firstName:"John",

    lastName:"Doe",

    age:50,

    eyeColor:"blue",

    fullName: function() {return this.firstName + " " + this.lastName;}

};
```

Accessing Object Properties

- There’s two ways to access object properties

```
objectName.propertyName      person.lastName; // "Doe"

objectName["propertyName"]   person["lastName"];// "Doe"
```

Accessing Object Methods

- Same idea as properties, two ways to access
- Remember that the object only has the function declaration, you need to invoke it to execute the method

```
objectName.propertyName()
```

```
person.fullName();
// "John Doe"
```

```
objectName["propertyName"]()
```

```
person["fullName"]();
// "John Doe"
```

```
objectName.propertyName
```

```
person["fullName"];
/* ƒ () {return this.firstName + " " +
this.lastName;} */
```

Object Constructors

- JavaScript allows creating object “blueprints” sometimes referred to as “classes”
- You can define these classes using a function
- This function is most commonly called a constructor

```
function Person(first, last, age, eye) {

    this.firstName = first;

    this.lastName = last;

    this.age = age;

    this.eyeColor = eye;

}
```

Creating Objects from Constructors

- To create a new object of that class, you use the ***new*** keyword

```
var myFather = new Person("John", "Doe", 50, "blue");

var myMother = new Person("Sally", "Rally", 48, "green");
```

The *this* Keyword

```
function Person(first) {

    this.firstName = first;

}
```

- In JS, *this* refers to the object that owns the code
- The value of *this* when used inside an object refers to the object itself
- In a constructor, the keyword *this* refers to the object that is being created

Types of Objects

- Primitives *can* be objects
  - Booleans can be objects (if defined with the new keyword)
  - Numbers can be objects (if defined with the new keyword)
  - Strings can be objects (if defined with the new keyword)
- It’s usually not a good idea to define primitives as objects (for performance reasons)
- Complex types are always objects
  - Arrays are always objects
  - Functions are always objects
  - Objects are always objects

Primitives vs Objects

- Primitives are ***immutable***
  - They’re values can’t be changed

Value	Type	Comment
"Hello"	string	"Hello" is always "Hello"
3.14	number	3.14 is always 3.14
true	boolean	true is always true
null	null (object)	null is always null
undefined	undefined	undefined is always undefined

- Objects contain values, including primitive values
  - You can change an object value by assigning a new primitive or complex value to it
  - The object as a whole stays the same

Built in JavaScript Objects

- var x1 = new Object(); // A new Object object
- var x2 = new String(); // A new String object
- var x3 = new Number(); // A new Number object
- var x4 = new Boolean(); // A new Boolean object
- var x5 = new Array(); // A new Array object
- var x7 = new Function(); // A new Function object

Object Prototypes

- The *prototype* is another property of all objects
- All of objects of the same type share the same prototype
- This is often called prototype inheritance
  - All Array objects inherit from the Array.prototype
  - All Person objects inherit from the Person.prototype
  - The Object.prototype is the top of the ***prototype chain***, all objects inherit from it

Object Prototypes

- If you want to add a new property to an object constructor, you can’t add it to the constructor directly

```
function Person(first, last, age, eyecolor) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eyecolor;
}
Person.nationality = "English"; // does not work
```

Object Prototypes

- You can modify properties on the property anytime

```
Person.prototype.nationality = "English"; // this works
```

- You can also add methods to the property
- All objects of that type then have access to that method

```
Person.prototype.name = function() {
    return this.firstName + " " + this.lastName;
};
```

Resources

https://www.w3schools.com/js/js_objects.asp

https://www.w3schools.com/js/js_object_definition.asp

https://www.w3schools.com/js/js_type_conversion.asp