JavaScript and ReactJS

October 23, 2018

Outline

1. What are JavaScript and ReactJS?
2. JavaScript basics
3. JavaScript custom objects
4. ReactJS basics
5. ReactJS tutorial

What are JavaScript & ReactJS?

- JavaScript:
  - A scripting language designed to run in a host environment, which provides tools for communication with the outside world (e.g. client-side browser, server-side Node.js)
- ReactJS
  - A JavaScript library for building user interfaces using small, isolated components
- Library = collection of routines for a program to use

JavaScript Basics

1. Types
2. Variables
3. Operators
4. Control structures
5. Objects
6. Arrays
7. Functions

Types (though there are a few more...)

- Number
- String
- Boolean (e.g. true or false)
- Null = indicates a deliberate non-value
- Object
  - Function
  - Array

Types

- Don’t need to explicitly check *null*
  - **let electionWinner = null;**
    if (electionWinner) {/ do stuff */}**

- Declaring an Array
  - **let a = Array(9); OR let a = new Array(9);**
- Declaring a function

  -
    ```
    function add(num1, num2) {
        return num1 + num2;
    }
    ```

Variables

- In JS, you must use a keyword to declare a variable
- Can use *let*, *const*, or *var*
  - The scope of *let*/*const* is the block in which it was declared.
  - The scope of *var* is the function in which it was declared.
  - *const* is used to declare variables whose value won't change.

Operators

- *+, -, *, /, %* available
- String concatenation with *+*
  - **var age = 25;**
    var str = “My age is: “ + age
- Comparisons with *<, >, <=, >=*
  - **if (age < 50) { /** do stuff */ }**
- Use *===* and *!==* to be safe.
  - **‘123’ == 123 but ‘123’ !== 123**

.

Operators

- Ternary operator an alternative to short if-else statements

```
let result;
let balance = 20;
if (balance === 20) {
    result = balance;
} else {
    result = balance - 10;
}
```

```
let result = (balance === 20) ?
    balance : balance - 10;
```

Operators

- Negating boolean values
  - **let isTuesday = true;**
  - **isTuesday = !isTuesday;**
  - **if (!isTuesday) {/** do stuff */}**
- Use && (and), // (or) to create complex boolean values
  - **if (isTuesday && isFall) { … }**
  - **if (balance > 50 || balance < 100) {...}**

Control Structures

- JS has for, while, and do-while loops, switch keyword
  - **for (let i = 0; i < 5; i++) {**

    console.log(i);

    }

Objects

- JavaScript is based on objects.
- An object is simply a collection of key-value pairs.
    - **let cat = {**

```
let cat = {
        name: “Mittens”,
        legs: 4,
        hasTail: true,
        makeSound: function() { console.log(“meow”)},
}
```

Objects

- Creating a new, empty object
  - **let obj = {};**
  - **let obj = new Object();**
- Access and modify properties with dot notation.
  - **let nLegs = cat.legs;**
  - **cat.name = “Garfield”;**

Arrays

- An object with the built-in property *length*
    - **let length = arr.length;**
- Creating a new, empty Array
    - **let arr = Array(); OR let obj = new Array();**
- Can optionally specify number of items and initialize
    - **let arr = new Array(9).fill(‘0’);**

Arrays

- Many array methods available
  - **a.push(item);**
  - **let item = a[i];**
- Can nest arrays
  - **let nested = [[1,2,3], [4,5,6], ‘abc’];**

Functions

- A computational structure that takes input and whose return value specifies the output (if any).

  -
    ```
    function add(x, y) {
        var total = x + y;
        return total;
    }
    ```
  - `let result = add(3, 4);`

Functions

- Functions can be anonymous.

  -
    ```
    let myFn = function() {
        return “I am anonymous!”;
    };

    myFn();
    ```

  -
    ```
    let myOtherFn = () =>  “I am also anonymous.”;

    myOtherFn();
    ```

Custom Objects

- JavaScript uses prototype-based inheritance.
  - Functions are objects, have properties, and can store state.
  - **function Cat(name, color) {**

    ```
    function Cat(name, color) {
        this.name = name;
        this.color = color;
    }
    ```
  - **let cat1 = Cat("Mittens", "black");**
  - **let cat2 = Cat("Simon", "grey")**
  - *this* refers to the specific context of Cat() we are using.

Custom Objects

- The *class* keyword wasn't introduced until the ES6 version of JavaScript.
  - Limited compared to *class* in other languages.
  - **class Cat {**

```
class Cat {
    constructor(name, color) {
        this.name = name;
        this.color = color;
    }
}
```

ReactJS Basics

- React is built with Component classes
  - Think of a Component like an object.
    - Must have a *render()* method.
    - Has properties stored in the *props* variable.
- Components render, or display, UI elements (think HTML).
  - Elements have properties, too.
- Components can pass data to other components.
- Our custom objects extend Component classes (share some behavior).

ReactJS Basics

- Components can store their state in a *state* variable.
  - Change *state* by using *setState()*
  - *state* is an object, and it has properties.
- In React, any JavaScript expression can be placed inside curly braces.
  - **{“hello”}**
  - **{function() {return “Anonymous function again.”;}}**

ReactJS Tutorial

Head over to https://reactjs.org/tutorial/tutorial.html

Resources

1. A re-introduction to Java Script (JS tutorial)
2. Tutorial: Intro to React