Exploring Career Options in Engineering and Science (ECOES)

Mijeong Ban

Stevens Institute of Technology

July 29, 2019

Mijeong Ban: Adjunct Instructor for ECOES Session2

1. Education
   - B.S. in Computer Science and
   - M.S. in Computer Science - Machine Learning Program at *Stevens Institute of Technology*
2. TA for
   - CS105: Intro to Scientific Computing
   - CS347: Software Development Process
   - CS383: Computer Org. and Programming
   - CS523: Programming the IoT using iOS
   - CS546: Web Programming I

David Kim: Teaching Assistant

- B.S. in Computer Science at *Stevens Institute of Technology*
- Cloud Computing Intern at Prudential Financial

Computer Science Department at Stevens

Computer Science at Stevens

- Largest department at Stevens
- We have two majors in CS
  - Computer Science
  - Cybersecurity
- Despite growth, we maintain small class sizes
- Start in the major as a freshman
  - Develop more skills, go in-depth in specific areas, graduate courses available
  - Ready for industry, start-ups, graduate school
- Co-op, internship opportunities
- Summer research internships within the department, funded by Stevens and CS

Computer Science and Cybersecurity

2018 Computer Science and Cybersecurity Graduates	
Employed	80%
Graduate School	14%
Returning to Home Country/Traveling	2%
Military	2%
Outcomes Finalized	**98%**
Seeking Employment	2%
Computer Science and Cybersecurity Average Salary	
Stevens Average	$86,300

AP exams to consider

- Calculus AB
- Calculus BC
- Computer Science A
- Chemistry
- Biology
- Physics I & II
- Any humanities
- Not Statistics

Introduction

Overview

- We will learn some basic programming in JavaScript
- Objectives:
  - Get an idea of what programming is about
  - Get to know a programming language that is really cool and easy to learn
  - We will use a programming language: JavaScript!

Node.js Compiler

- Node.js is a JavaScript run-time environment that executes JavaScript code outside of a browser.
- For now we will use an online interpreter.
- `https://www.tutorialspoint.com/execute_nodejs_online.php`

What we will cover

- Output/Comments
- Data Types
- Variables
- Operators
  - Arithmetic Operators
  - Assignment Operators
  - Comparison Operators
  - Logical Operators
- Conditional Statements
- Functions
- Data Structures - Arrays
- Loop For

Output/Comments

Output

To display data,

```
1 console.log("Hello, World!");
```

Comments

Comments are lines of code that JavaScript will intensionally **ignore**. They are usually used for notes about what the code does.

- **Single Line Comments**
- **Multi-line Comments**

```
1 console.log("Hello, World!"); // single line
      comment
2 console.log("Hello, World!");
3 /*
4 This
5 is
6 multi-line comment
7 */
```

Data Types

Data Types

- **undefined**: something that hasn't been defined
- **null**: "nothing". Something that doesn't exist
- **boolean**: true or false
- **number**: number
- **string**: a series of characters
- **object**: Store a lot of different key-value pairs

*** You can use typeof() operator to check data types

Data Types


```
1 var x; // declare variable x
2 console.log(typeof(x)); // undefined
3
4 var x = false; // Boolean
5 var length = 10; // Number
6 var firstName = "John"; // String
7 var name = {firstName: "John", age: 18}; // Object
```

Variables

Variables

Set data into a variable

- Variables allow computers to store and manipulate data in dynamic fashion.
- Variables allow values to be reused
- There are three ways to declare a variable in JavaScript
  - **var**: It can be used throughout your whole program
  - **let**: It will only be used within the scope of where you declare that
  - **const**: It is for a variable that should never change

Variables


```
1  /* Declare a variable: you don't need to specify data
       type in JavaScript */
2  var name = "Jason";
3  console.log(name); // Jason
4
5  /* Manipulate data */
6  name = "David";
7  console.log(name) // David
8
9  /* Declare a variable using let */
10 let i = 0;
11
12 /* Declare a variable using const */
13 const pi = 3.14;
14 pi = 100; // you will get an error because you tried
       to manipulate const variable
```

Variables - String


```
1 var name = "David";
2 var str = "My name is " + name + ", how are you?";
3 console.log(str);
```

Operators

Arithmetic Operators

To perform arithmetic on numbers,

Operator	**Description**
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Division Remainder)
++	Increment
- -	Decrement

Arithmetic Operators


```
1  /* Addition */
2  var number1 = 10;
3  var number2 = 20;
4  var sum = number1 + number2;
5  console.log(sum); // 30
6
7  /* Subtraction */
8  var number1 = 20;
9  var number2 = 10;
10 var difference = number1 - number2;
11 console.log(difference); // 10
```

Arithmetic Operators


```
1  /* Multiplication */
2  var number1 = 10;
3  var number2 = 10;
4  var product = number1 * number2;
5  console.log(product); // 100
6
7  /* Division */
8  var number1 = 66;
9  var number2 = 33;
10 var quotient = number1 / number2;
11 console.log(quotient); // 2
```

Arithmetic Operators


```
1 /* Remainder */
2 var remainder = 11 % 3;
3 console.log(remainder); //2
4
5 // Remainder is often used when you need to check if a
      number is odd or even
6 var a = 10 % 2; // 0 -> even number
7 var b = 11 % 2; // 1 -> odd number
```

Arithmetic Operators


```
1 /* Increment */
2 var myNumber = 1;
3 myNumber++;
4 console.log(myNumber); // 2
5
6 /* Decrement */
7 myNumber--;
8 console.log(myNumber); // 1
```

Assignment Operators

To assign values to variables,

Operator	Same As
x = y	x = y
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y

```
1 var x = 10;
2 x += 5; // same as x = x + 5;
3 console.log(x); // 15
```

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or value.

Operator	**Description**
==	equal to
!=	not equal
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

```
1 console.log(true && true); // true
2 console.log(true && false); // false
3
4 console.log(true || true); // true
5 console.log(true || false); // true
6
7 console.log(!true) // false
8 console.log(!false) // true
```

Conditional Statements

Conditional Statements

Conditional statements are used to perform different actions based on different conditions.

- if Statement
- else Statement
- else if Statement

The if Statement

Syntax:

```
1 if (condition) {
2     // code to be executed if the condition is true
3 }
```

Example:

```
1 var today = "Friday";
2 if (today == "Friday") {
3     console.log("Today is Friday. TGIF!");
4 }
```

The else Statement

Syntax:

```
1 if (condition) {
2     // code to be executed if the condition is true
3 } else {
4     // code to be executed if the condition is false
5 }
```

Example:

```
1 var today = "Friday";
2 if (today == "Friday") {
3     console.log("Today is Friday. TGIF!");
4 } else {
5     console.log("Today is not Friday :( ");
6 }
```

The else if Statement

Syntax:

```
1 if (condition1) {
2     // code to be executed if the condition1 is true
3 } else if (condition2){
4     // code to be executed if the condition1 is false
      and condition2 is true
5 } else {
6     // code to be executed if the condition1 is false
      and condition2 is false
7 }
```

The else if Statement

Example:

```
1 var number = 0;
2 if (number > 0) {
3     console.log(number + " is positive");
4 } else if (number < 0) {
5     console.log(number + " is negative");
6 } else {
7     console.log(number + " is zero");
8 }
```

Conditional Statements - Exercise

Write a program:

```
1 /*
2  * If the number is <= 10 then display "The variable
       value is less than or equal to 10"
3  * If the number is <= 20 then display "The variable
       value is less than or equal to 20"
4  * If the number is <= 30 then display "The variable
       value is less than or equal to 30"
5 */
6 var number = 23;
7 // Write conditional statements here
```

Functions

Function

- Functions allow us to create **reusable** code.
- Instead of writing the same code repeatably, we can write a function to perform a particular task and use it whenever we want.

Syntax:

```
1 function name(parameter1, parameter2, ...) {
2     // code to be executed
3 }
```

** Parameters are inputs for function

Example:

```
1 /* Declare a function */
2 function sayHi() {
3     console.log("Hi there!");
4 }
5
6 /* Call a function */
7 sayHi();
8 sayHi();
9 sayHi();
```

Example: Passing values to functions with arguments

```
1 function add(num1, num2) {
2     return num1 + num2; // return statement: output
3 }
4 var sum = add(4, 3);
5 console.log(sum); // 7
```

Function - Exercise

Write a function that compares two values.

- If they are equal, return true
- If they are not equal, return false

```
1 function compare(a, b) {
2     // Write code here
3 }
4
5 console.log(compare(2, 2)); // should return true
6 console.log(compare(3, 10)); // should return false
```

Function - Exercise

Write a function that checks if the input number is even or odd. Assume that the input number is greater than/ equal to 0.

```
1 function evenOrOdd(num) {
2     // Write code here
3 }
4
5 evenOrOdd(11) // should display "This number is odd"
6 evenOrOdd(20) // should display "This number is even"
```

Function - Exercise

Write a function that returns largest of three numbers. Assume three numbers are all different.

```
1 function largest(num1, num2, num3) {
2     // Write code here
3 }
4
5 console.log(largest(1, 2, 3)) // should return 3
6 console.log(largest(4, 22, 5)) // should return 22
```

Data Structures - Arrays

A data structure is a particular way of organizing data in a computer so that it can be used effectively

- **Array**
- Linked List
- Stack
- Queue
- Tree
- Hashing
- Graph
- Matrix

We will look at only one of them, Arrays.

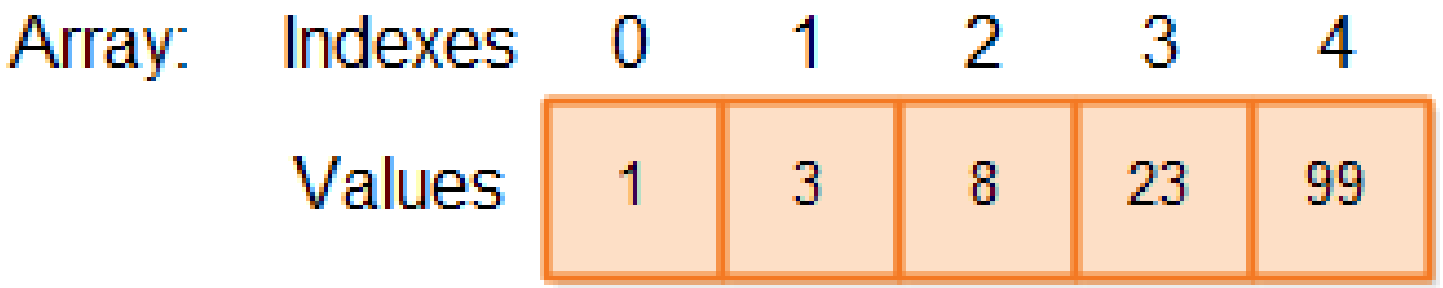
Arrays allow you to store several pieces of data in one place.
Syntax:

```
1 var array_name = [element1, element2, ...]
```

Example:

```
1 var arr1 = [1, 2, 3];
2 var arr2 = ["John", 23]; // elements can be any data
      type
```

You can access an array element by referring to the **index number**. Array indexes start with 0, not 1.



Arrays - Access/Modify array data with indexes


```
1  var myArray = ["a", "b", "c", "d"];
2
3  /* Access to the first data in array */
4  var myData = myArray[0];
5  console.log(myData); // a
6
7  /* Modify the first element of the array */
8  myArray[0] = "f";
9  console.log(myArray); // ['f','b','c','d']
10
11 /* Get the array length */
12 var arraySize = myArray.length;
13 console.log(arraySize); // 4
```

These are functions that you can use to manipulate arrays.

- pop()
- push()
- shift()
- unshift()

Array Methods - **pop()**

The pop() method **removes** the last element from an array.

```
1 var students = ["John", "David", "Irene", "Alex"];
2
3 /* pop the last element out of an array */
4 var removed = students.pop();
5 console.log(students);
6 console.log(removed); // Alex: pop() method returns
      the value that was popped out
```

Array Methods - **push()**

The push() method **adds** a new element to an array **at the end**

```
1 var students = ["John", "David", "Irene", "Alex"];
2
3 /* push a new element into an array */
4 var added = students.push("Steve");
5 console.log(students);
6 console.log(added); // 5: push() method returns the
      new array length
```

The shift() method **removes** the first array element and shifts all other elements to a lower index.

```
1 var students = ["John", "David", "Irene", "Alex"];
2
3 /* removes the first element */
4 var removed = students.shift();
5 console.log(students);
6 console.log(removed); // John: shift() method returns
      the element that was shifted out
```

Array Methods - **unshift()**

The unshift() method **adds** a new element to an array **at the beginning**, and unshifts older elements.

```
1 var students = ["John", "David", "Irene", "Alex"];
2
3 /* adds a new element "Steve" */
4 var added = students.unshift("Steve");
5 console.log(students);
6 console.log(added); // 5: unshift() method returns the
      new array length
```

Array Methods - Exercise

Given non-empty array, write a function that takes an array as an input and

- If the length of array is even, then add an element with value ”even” at the end and return the array
- If the length of array is odd, then add an element with value ”odd” at the beginning and return the array

```
1 function oddOrEvenArray(arr) {
2     // Write your code here
3 }
```

Loops

The For Loop

Loops are useful when you want to run the same code over and over again.
Instead of writing:

```
1 console.log("Hello, World!");
2 console.log("Hello, World!");
3 console.log("Hello, World!");
4 console.log("Hello, World!");
5 console.log("Hello, World!");
```

You can write:

```
1 var i = 0;
2 for (i = 0; i < 5; i++) {
3     console.log("Hello, World!");
4 }
```

The For Loop

Syntax:

```
1 for (statement1; statement2; statement3) {
2     // code to be executed
3 }
```

- **statement1** sets a variable before the loop starts
- **statement2** defines the condition for the loop to run
- **statement3** changes the variable from statement1 each time the code block in the loop has been executed

The For Loop

Example:

```
1  /* Print out numbers 0 to 4 */
2  for (var i = 0; i < 5; i++) {
3      console.log("The number is " + i);
4  }
5
6  /* Assign 0 to 4 numbers to an array */
7  var myArray = [];
8  for (var i = 0; i < 5; i++) {
9      myArray[i] = i;
10 }
11 console.log(myArray); // [0, 1, 2, 3, 4]
```

The For Loop

Since we learned arrays, loops are very handy when you want to iterate each element in an array.

```
1 var myArray = ["John", "David", "Irene", "Alex", "
      Steve"]
2
3 /* Print out each element in array */
4 /* Range of i here: 0 to 4 since the array length is 5
       */
5 for (var i = 0; i < myArray.length; i++) {
6     console.log("now i is " + i);
7     console.log(myArray[i]);
8 }
```

Example:
Push numbers 1 - 10 into an array using push() method

```
1 var myArray = [];
2 for(var i = 0; i < 10; i++) {
3     myArray.push(i+1); // since i starts at 0
4 }
5 console.log(myArray);
```

Write a function that checks if there is a 3 in the given array as input using for loop

```
1 function checkThree(array) {
2     // Write your code
3 }
4
5 console.log(checkThree([1,2,3])) // should return true
6 console.log(checkThree([1,1,1,1,1,1])) // should
    return false
```

Write a function that checks if the sum of the elements is even. Return false otherwise. Assume the sum of the elements will be greater than zero.

```
1 function evenSum(arr) {
2     // Write your code
3 }
4
5 console.log(evenSum([2,1,3,4])) // sum = 10, return
    true
6 console.log(evenSum([-1,4,2])) // sum = 5, return
    false
```

More Challenges

- Make sure you make an account for leetcode website!
  - `https://leetcode.com/tag/array/`
- Get this slide
  - `https://www.dropbox.com/s/a5uzkwsungcnb2q/ECOES_Presentation.pdf?dl=0`

References

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