JavaScript For Absolute Beginners

(Daniyal Nagori)

JavaScript

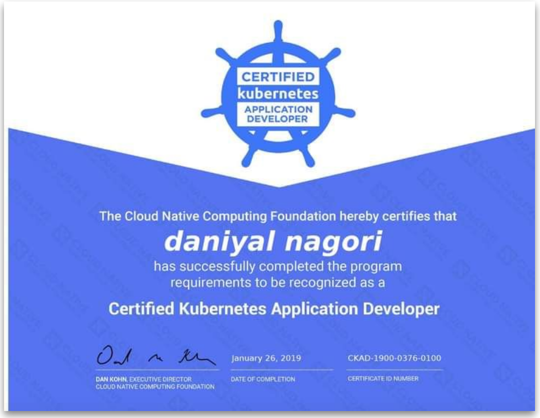
fb.com/daniyalnagori1237

linkedin.com/in/daniyalnagori

github.com/daniyalnagori

twitter.com/daniyalnagori1

About Instructor



CERTIFIED kubernetes APPLICATION DEVELOPER

The Cloud Native Computing Foundation hereby certifies that

daniyal nagori

has successfully completed the program requirements to be recognized as a

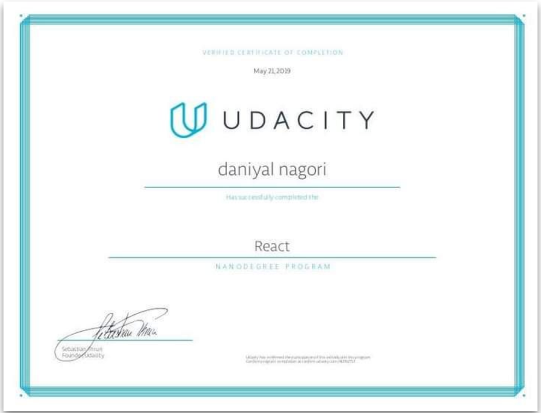
Certified Kubernetes Application Developer

[illegible]

DAN KOHN, EXECUTIVE DIRECTOR
CLOUD NATIVE COMPUTING FOUNDATION

January 26, 2019
DATE OF COMPLETION

CKAD-1900-0376-0100
CERTIFICATE ID NUMBER



VERIFIED CERTIFICATE OF COMPLETION

May 21, 2019

UDACITY

daniyal nagori

Has successfully completed the

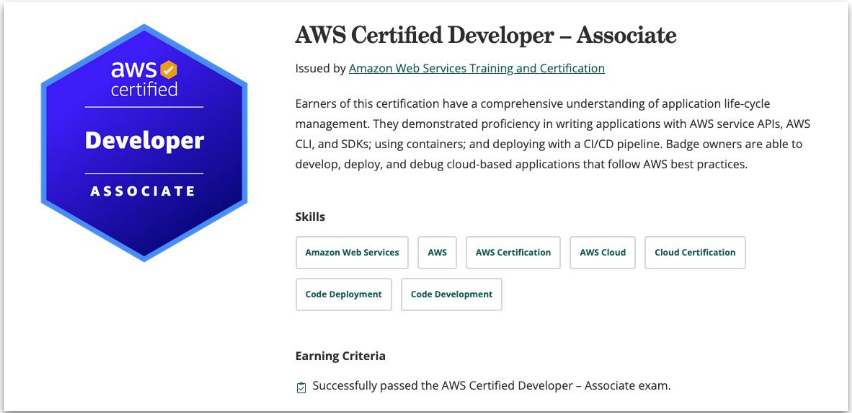
React

NANODEGREE PROGRAM

[illegible]

Sebastian Thrun
Founder, Udacity

[illegible]



aws certified
Developer
ASSOCIATE

AWS Certified Developer – Associate

Issued by Amazon Web Services Training and Certification

Earners of this certification have a comprehensive understanding of application life-cycle management. They demonstrated proficiency in writing applications with AWS service APIs, AWS CLI, and SDKs; using containers; and deploying with a CI/CD pipeline. Badge owners are able to develop, deploy, and debug cloud-based applications that follow AWS best practices.

Skills

Amazon Web Services | AWS | AWS Certification | AWS Cloud | Cloud Certification | Code Deployment | Code Development

Earning Criteria

Successfully passed the AWS Certified Developer – Associate exam.

Integrated Development Environment

Setting up your environment

- There are many ways in which you can set up a JavaScript coding environment. Such as:
    - Integrated Development Environment (IDE). Example: VS Code, Sublime Text, Atom, etc.
    - Web browser. Example: Chrome, Firefox, etc.
    - Online editor (optional). Example: StackBlitz, Replit, etc.

Adding Javascript to a Web Page

Adding JavaScript to a web page

- There are two ways to link JavaScript to a web page.
  - The first way is to type the JavaScript directly in the HTML between two <script> tags.

```
<html>
    <script type="text/javascript">
        alert("Hello World!");
    </script>
</html>
```

  - The second way is to create a file with extension of .js and link it to our web page.

```
<html>
    <script type="text/javascript" src="hello_world.js"></script>
</html>
```

ALERT

ALERT

- The alert() method displays an alert box with a message and an OK button.
- The alert() method is used when you want information to come through to the user.
- The alert box takes the focus away from the current window, and forces the user to read the message.
- Do not overuse this method. It prevents the user from accessing other parts of the page until the box is closed.

CONSOLE LOG

CONSOLE LOG

- The **console.log()** method writes (**logs**) a message to the console.
- The **console.log()** method is useful for testing purposes.

Document Write

Document Write

- The **document.write()** method writes directly to an open (**HTML**) document stream.
- The **document.write()** method deletes all existing HTML when used on a loaded document.

VARIABLES

VARIABLES

- Variable means anything that can vary.
- A JavaScript variable is simply **a name of storage location**.
- A variable must have a unique name.

Variables

- Variables are values in your code that can represent different values each time the code runs.
- The first time you create a variable, you declare it. And you need a special word for that: `let` , `var` , or `const` .
  Example: `let firstname = "Ali";`
- The commonly used naming conventions used for **variables** are camel-case.
  Example: `let firstName = "Ali";`

Variables Scope

- **LOCAL**
  - Variables declared within a JavaScript function, become LOCAL to the function.
- **GLOBAL**
  - A variable declared outside a function, becomes GLOBAL.

VARIABLE Names Legal & Illegal

VARIABLE Names

- A variable name can't contain any spaces
- A variable name can contain only letters, numbers, dollar signs, and underscores.
- The first character must be a letter, or an underscore (-), or a dollar sign ($).
- Subsequent characters may be letters, digits, underscores, or dollar signs.
- Numbers are not allowed as the first character of variable.

Comments

Comments

- Single line Javascript comments **start with two forward slashes (//)**.
- All text after the two forward slashes until the end of a line makes up a comment
- Even when there are forward slashes in the commented text.
- Multi-line Comments
- Multi-line comments start with /* and end with */.
- Any text between /* and */ will be ignored by JavaScript.

Statements

Statements

- A computer program is a list of "instructions" to be "executed" by a computer.
- In a programming language, these programming instructions are called statements.
- A JavaScript program is a list of programming statements.
- JavaScript applications consist of statements with an appropriate syntax. A single statement may span multiple lines. Multiple statements may occur on a single line if each statement is separated by a semicolon.

Data types

Primitive data types

- String
  - A string is used to store a text value.
    Example: `let firstName = "Ali";`
- Number
  - A number is used to store a numeric value.
    Example: `let score = 25;`
- Boolean
  - A boolean is used to store a value that is either `true` or `false`.
    Example: `let isMarried = false;`
- Undefined
  - An undefined type is either when it has not been defined or it has not been assigned a value.
    Example: `let unassigned;`
- Null
  - null is a special value for saying that a variable is empty or has an unknown value.
    Example: `let empty = null;`

Template Literals

Template Literals

A new and fast way to deal with strings is **Template Literals or Template String.**

How we were dealing with strings before ?

```
var myName = "daniyal" ;
var hello = "Hello "+ myName ;
console.log(hello); //Hello daniyal
```

Template Literals

What is Template literals ?

As we mentioned before , it’s a way to deal with strings and specially dynamic strings ; so you don’t need to think more about what’s the next quote to use single or double.

How to use Template literals

It uses a `backticks` to write string within it.

typeof Operator

Analyzing and modifying data types

- You can check the type of a variable by entering **`typeof`**.
  Example:
  ```
  let testVariable = 1;
  console.log(typeof testVariable);
  ```
- The variables in JavaScript can change types. Sometimes JavaScript does this automatically.
  Example:
  ```
  let v1 = 2;
  let v2 = “2”;
  console.log(v1 * v2); // 4 ← Type Number
  console.log(v1 + v2); // “22” ← Type String
  ```

Analyzing and modifying data types

- There are three conversion methods:
  - `String()` ← `converts to string type`
  - `Number()` ← `converts to number type`
  - `Boolean()` ← `converts to boolean type`

Operators

Operators

- Arithmetic operators:
  - Addition

    Example:

    -
      ```
      let n1 = 1;
      let n2 = 2;
      console.log(n1 + n2); // 3
      ```

    -
      ```
      let str1 = “1”;
      let str2 = “2”;
      console.log(str1 + str2); // “12”
      ```

Operators

- Arithmetic operators:
  - Subtraction
    Example:

    -
      ```
      let n1 = 5;
      let n2 = 2;
      console.log(n1 - n2); // 3
      ```
  - Multiplication
    Example:

    -
      ```
      let n1 = 5;
      let n2 = 2;
      console.log(n1 * n2); // 10
      ```

Operators

- Arithmetic operators:
  - Division

    Example:

    -
      ```
      let n1 = 4;
      let n2 = 2;
      console.log(n1 / n2); // 2
      ```
  - Exponentiation

    Example:

    -
      ```
      let n1 = 2;
      let n2 = 2;
      console.log(n1 ** n2); // 4
      ```

Operators

- Arithmetic operators:
  - Modulus
    Example:
    - **let n1 = 10;**
      let n2 = 3;
      console.log(n1 % n2); // 1

Operators

- Assignment operators:
    - Assignment operator are used to assigning values to variables.
      Example:
        - **let n = 5;**
          console.log(n); // 5
          n += 5;
          console.log(n); // 10
          n -= 5;
          console.log(n); // 5

Operators

- Comparison operators:
    - Comparison operator are used to compare values of variables.
      Example:

        -
          ```
          let n = 5;
          console.log(n == 5); // true
          console.log(n === 5); // true
          console.log(n != 5); // false
          console.log(n > 8); // false
          console.log(n < 8); // true
          console.log(n >= 8); // false
          console.log(n <= 8); // true
          ```

Math Expressions
Familiar Operators

Expressions

- An Expression is a combination of values, variables, function call and operators, which computes to a value.
- The computation is called an evaluation.
- “Daniyal” + “Nagori”

Math Expressions Familiar Operators

- Wherever you can use a number, you can use a math expression.
- **"+", "-", "*", "/"** and **"%"** are commonly used operators.
- **"%" (Modulus)** operator works similar to "/" but instead of the result, It gives you the remainder when the division is executed.
- Examples:
  - let add = 2 + 3; // 5
  - let subtraction = 8 - 4; // 4
  - let multiplication = 2 * 2; // 4
  - let division = 4 / 2; // 2
  - let modulus = 9 % 3; // 0

Math Expressions
UnFamiliar Operators

Math Expressions UnFamiliar Operators

- There are several specialized math expressions such as “++”, “--” and “**”.
  - **“++” :** It increments the variable by 1.
  - **“--” :** It decrements the variable by 1.
  - **“**” :** Exponentiation is one of the newer operators in JavaScript, and it allows us to calculate the power of a number by its exponent.

Math Expressions UnFamiliar Operators

Post Increment vs Pre Increment

- Post Increment
  - The operator increases the variable var1 by 1 but returns the value before incrementing.
  - Example:
    - let i = 1;
      let num = i++ // 1
- Pre Increment
  - The operator increases the variable var1 by 1 but returns the value after incrementing.
  - Example:
    - let i = 1;
      let num = ++i // 2
- Same rule for the **Decrement**

Math Expressions
Eliminating Ambiguity

Math Expressions Eliminating Ambiguity

Complex arithmetic expressions can pose a problem, one that you may remember from high school algebra.

- Examples:
  - var totalVal = (5 + 2) * 3 + 6; // 27
  - var totalVal = (2 * 4) * 4 + 2; // 34

Concatenating Text String

Concatenating Text Strings

- The **concat()** method joins two or more strings.
- The **concat()** method does not change the existing strings.
- The **concat()** method returns a new string.
- You can also use **“+”** operator to concatenate multiple strings.
- Examples:
  - let userName = Daniyal
    console.log("Thanks, " + userName + "!")

Prompts

Prompts

- The **prompt()** method displays a dialog box that prompts the user for input.
- The **prompt()** method returns the input value **(String)** if the user clicks **"OK"**, otherwise it returns **null**.
- When a **prompt box** pops up, the user will have to click either **"OK"** or **"Cancel"** to proceed.
- Do not overuse this method. It prevents the user from accessing other parts of the page until the box is closed.

If, Else, Else If Statements

If, Else and Else If Statements

- Use **if** to specify a block of code to be executed, if a specified condition is true.
- Use **else** to specify a block of code to be executed, if the same condition is false.
- Use **else if** to specify a new condition to test, if the first condition is false.

If, Else and Else If Statements - Examples

- If Example:

  -
    ```
    let x = prompt("Where does the Pope live?");
    let correctAnswer = "Pakistan";
    if (x == correctAnswer ) {
            alert("Correct!");
     }
    ```
- else - Example

  -
    ```
    let x = prompt("Where does the Pope live?");
    let correctAnswer = "Pakistan";
    if (x == correctAnswer ) {
            alert("Correct!");
     } else {
            alert("Wrong!");
    }
    ```

If, Else and Else If Statements - Examples

- Else if - Example

  -
    ```
    let x = prompt("Where does the Pope live?");
    let correctAnswer = "Pakistan";
    if (x == correctAnswer ) {
            alert("Correct!");
    } else if (x=="Pakista") {
            alert("Close!");
    } else {
            alert("Wrong!");
    }
    ```

Comparison Operators

Comparison Operators

- Comparison and Logical operators are used to test for **true** or **false**.
- Comparison operators are used in logical statements to determine equality or difference between variables or values.
- **“==”, “===”, “!=”, “!==”, “>”, “<”, “>=”** and **“<=”** are some of the comparison operators.

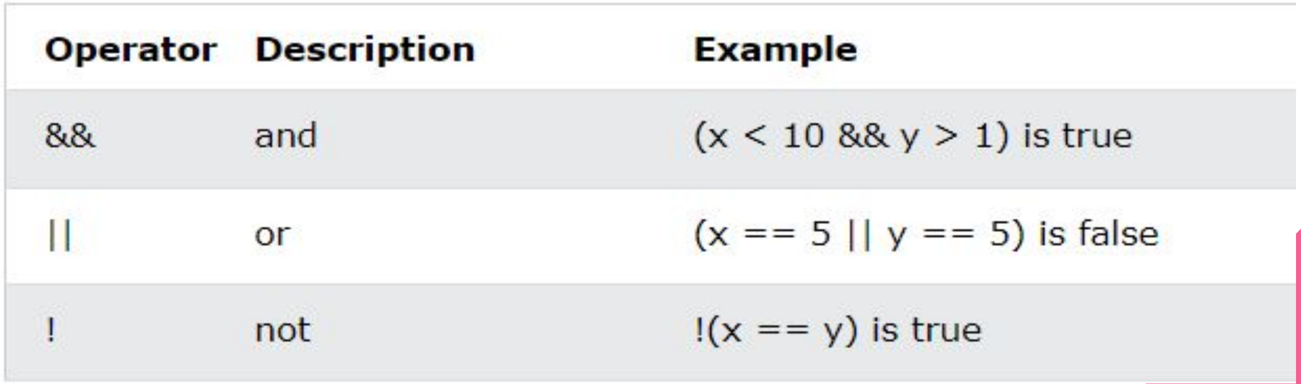
Comparison Operators - Examples

- let a = 2 + 2 == "4" // true
- let b = 2 + 2 === "4" // false
- let c = 2 + 2 > 4 // false
- let d = 2 + 2 >= 4 // true
- let e = 2 + 3 !== 5 // false

Testing Sets Of Conditions (Logical Operators)

Testing Sets Of Conditions (Logical Operators)

- Logical operators are used to determine the logic between variables or values.
- Given that `x = 6` and `y = 3`, the table below explains the logical operators:



Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

Testing Sets Of Conditions (Logical Operators) - Examples

-
  ```
  let x = 6
  let y = 10

  let a1 = x < y && x === 6 // true
  let a2 = x < y && x !== 6  // false
  let a3 = x === y || y === 10 // true
  let a4 = (x===6 && y===4) || x < y  // true
  ```

If Statement Nested

If Statement Nested

- JavaScript allows us to nest if statements within if statements. i.e, we can place an if statement inside another if statement.

If Statement Nested - Example

```
// Ticketing system
let country = prompt("Where do you live?")
// Number() function is used to convert the string to number
let age = Number(prompt("What's your age?"))

if (country === "pakistan") {
    if (age >= 18) {
        console.log("Here is your ticket")
    } else {
        console.error("Age restriction")
    }
} else {
    console.log("Invalid country")
}
```

Array

Array

- **The Problem:** Suppose you have five fruits and you want to store them in the variable, But you have to create five variables to store the fruits which is not an efficient approach, what if you have thousands of fruits?

  -
    ```
    let fruit1 = "apple"
    let fruit2 = "banana"
    let fruit3 = "grapes"
    let fruit4 = "strawberry"
    let fruit5 = "orange"
    ```

- **The Solution:** Here the array comes into play which helps to store multiple data in a single variable.

  -
    ```
    let fruits = ["apple","banana", "orange", "grapes", "strawberry"]
    ```

Array - More About Array

- An array is a special variable, which can hold more than one value.
- An array can hold many values under a single name, and you can access the values by referring to an index number.
- In JavaScript, arrays always use numbered indexes.
- Array indexes start with 0.
- Examples:
    - let fruits = ["apple","banana", "orange", "grapes", "strawberry"]
      fruits[0] // apple
      fruits[3] // grapes
    - let x = [1, 2, "daniyal"] // Arrays can store multiple types of data

Arrays: Adding and removing elements

Arrays: Adding and removing elements

- When you work with arrays, it is easy to **remove elements** and **add new elements**. This is what **popping** and **pushing** is.
- The **pop()** method removes the last element from an array:
- The **pop()** method returns the value that was **"popped out"**
- The **push()** method adds a new element to an array (at the end).
- The **push()** method returns the new array length.

Arrays: Adding and removing elements - Examples

- var pets = [];
  pets[0] = "dog"; // adds “dog” to an array at 0 index
  pets[1] = "cat"; // adds “cat” to an array at index 1

  pets.pop(); // removes the last element of an array which is cat in our case
  pets.push("parrot"); // adds a new element to an array

Arrays: Removing, inserting, and extracting elements

Arrays: Removing, inserting, and extracting elements

- Shifting is equivalent to popping, but working on the **first element** instead of the **last**.
- The **shift()** method removes the first array element and "**shifts**" all other elements to a lower index.
- The **shift()** method returns the value that was "**shifted out**".
- The **unshift()** method adds a new element to an array (at the beginning), and "**unshifts**" older elements:
- The **unshift()** method returns the new array length.

Arrays: Removing, inserting, and extracting elements - Example

- var pets = [];
  pets[0] = "dog"; // adds “dog” to an array at 0 index
  pets[1] = "cat"; // adds “cat” to an array at index 1

  pets.shift(); // removes the first element of an array which is cat in our case
  pets.unshift("parrot"); // adds a new element to an array (at the beginning)

Arrays: Removing, inserting, and extracting elements

Splicing and Slicing Arrays

- The **splice()** method adds new items to an array.
  - Example:
    ```
    const fruits = ["Banana", "Orange", "Apple", "Mango"];
    fruits.splice(2, 0, "Lemon", "Kiwi");
    // adds elements to an array at 2nd index
    // deleted 0 elements
    ```
- The **slice()** method slices out a piece of an array.
  - Example:
    ```
    const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
    const citrus = fruits.slice(1); // [Orange,Lemon,Apple,Mango]
    ```
  - **Notes:**
    The **slice()** method creates a new array.