BridgeLabz

Employability Delivered

NodeJs Programming Constructs

NodeJs – CLI (Command Line Interface) Installation

BridgeLabz
Employability Delivered

- **Step 1:** Check Homebrew Package Manager is installed using
  - ***brew –version*** - in the Terminal of Linux/Mac or in the Command Line of Windows
- **Step 2:** If not, Installation of Homebrew Package Manager
  - **For Windows –** Download Windows Subsystem for Linus using link https://docs.microsoft.com/en-us/windows/wsl/about
  - **For Linux & Windows -** https://docs.brew.sh/Homebrew-on-Linux
  - **For Mac -** https://docs.brew.sh/Installation
- **Step 3:** ***brew –version*** in command line to check for successful installation
- **Step 4:** ***brew –update*** – This updates Homebrew with a list of the latest version of Node.
- **Step 5:** ***brew install node*** **–** This will install node and npm
- **Step 6:** ***node -v*** **&** ***npm -v*** **–** This will check the installation of node and npm
- Refer https://treehouse.github.io/installation-guides/mac/node-mac.html for installation notes for Mac, Linux and Windows with Homebrew

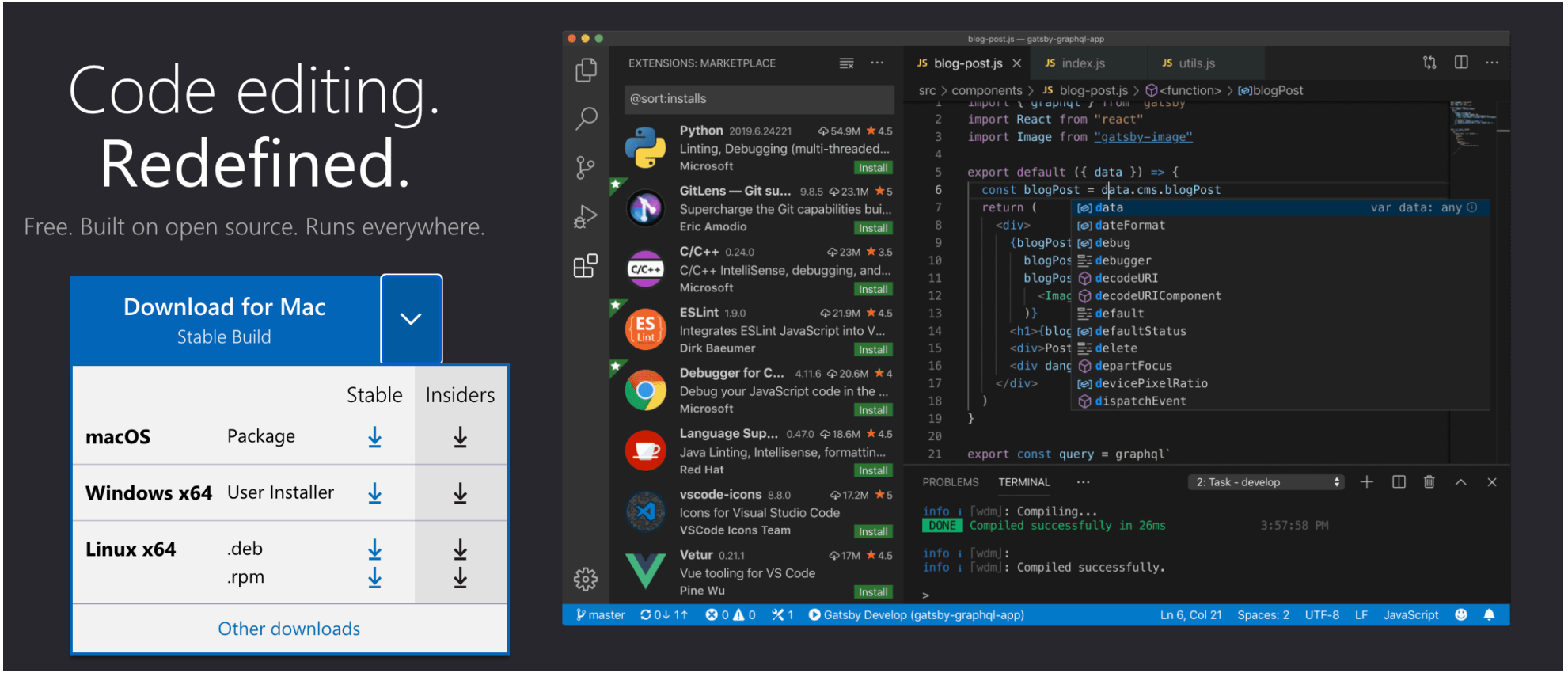
What is Node.js and npm

1. Node.js® is an environment which you can uses for compiling and running JavaScript code in command line and more importantly to create web-servers and networked applications.
2. NPM is a “package manager” that makes installing Node “packages” fast and easy.

VSCode Installation

BridgeLabz
Employability Delivered

- You can install VSCode from below link according to your OS: https://code.visualstudio.com/



BridgeLabz
Employability Delivered

Demonstrate Node REPL and Debugger

REPL - Read Eval Print Loop

BridgeLabz
Employability Delivered

Just like shell script/MySQL Client can be run in Command Line, Node comes with bundled REPL Environment

Step 1: Type ***node*** in your terminal

Step 2: In command prompt type

console.log("Hello")

Step 3: Use ***.exit*** or ***ctrlC twice*** to exit node command prompt.

Step 4: Random Function for single digit

Math.floor(Math.random() * 10);.

Step 4: Click to Practice.

```
NodeDevelopment $ node
Welcome to Node.js v13.5.0.
Type ".help" for more information.
> console.log("Hello");
Hello
undefined
> .exit
NodeDevelopment $
```

Node.js Debugger

BridgeLabz
Employability Delivered

Step 1: Type ***nano add.js*** and write code

```
let a = 10;
let b = 20;
let c = a + b;
add.js (END)
```

let – **Variable Declaration Local Scope. New Way**
var – **Variable Declaration Global Scope. Old Way**

Step 2: The latest debug version can also be installed independently (e.g. ***npm install -g node-inspect***)

Step 3: In Chrome Browser Open chrome://inspect

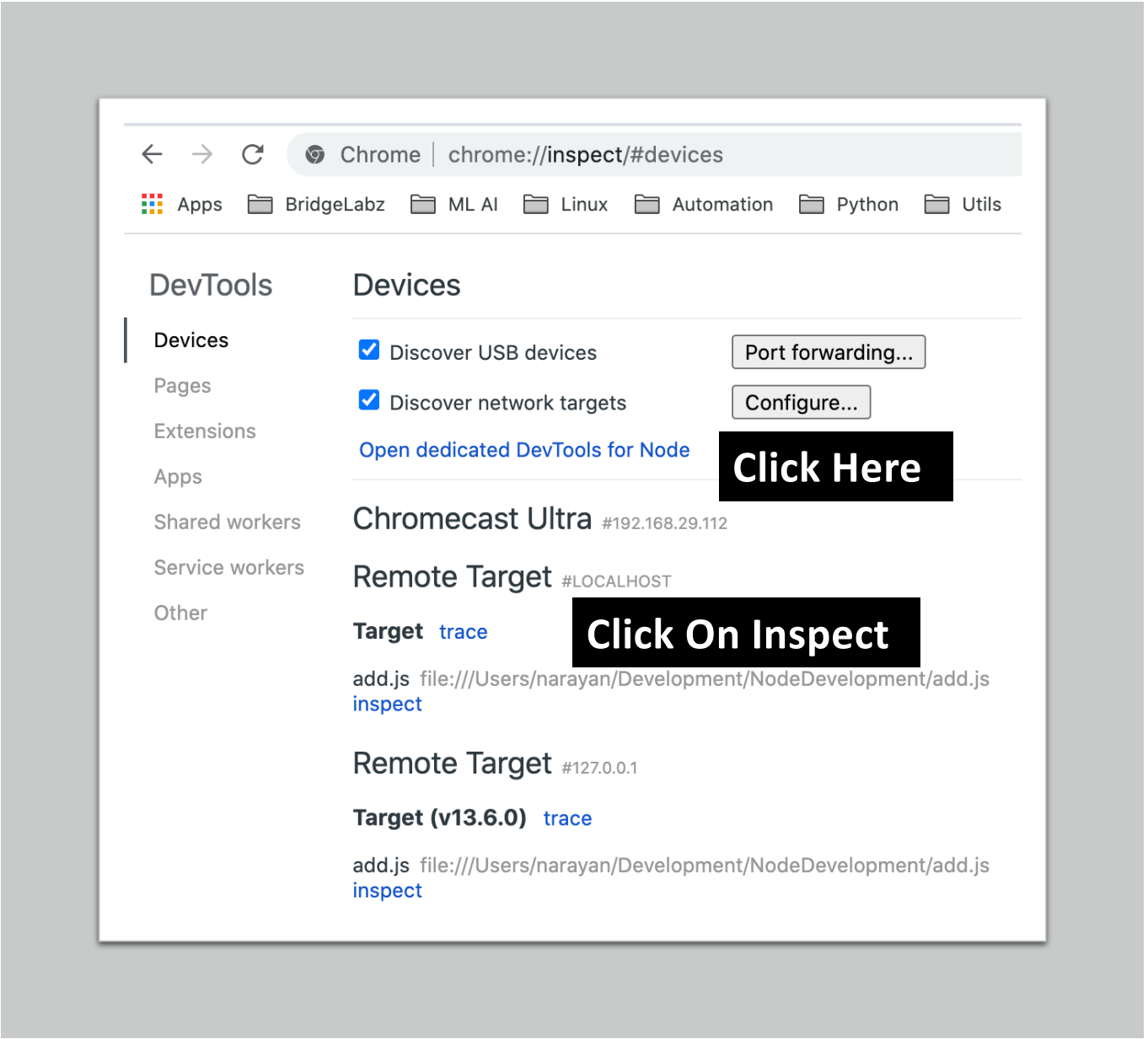
Step 4: Click the Open dedicated DevTools for Node link.

Step 5: Type ***node inspect add.js***

Step 6: Open Chrome and In URL Type ***chrome://inspect/#devices***

```
NodeDevelopment $ node inspect add.js
< Debugger listening on ws://127.0.0.1:9229/
< For help, see: https://nodejs.org/en/docs/
< Debugger attached.
Break on start in add.js:1
> 1 let a = 10;
  2 let b = 20;
  3 let c = a + b;
< Debugger attached.
< Debugger attached.
debug> next
break in add.js:2
  1 let a = 10;
> 2 let b = 20;
  3 let c = a + b;
  4
debug> next
break in add.js:3
  1 let a = 10;
  2 let b = 20;
> 3 let c = a + b;
  4
debug> next
break in add.js:3
  1 let a = 10;
  2 let b = 20;
> 3 let c = a + b;
  4
debug> cont
< Waiting for the debugger to disconnect...
debug> .exit
NodeDevelopment $
```

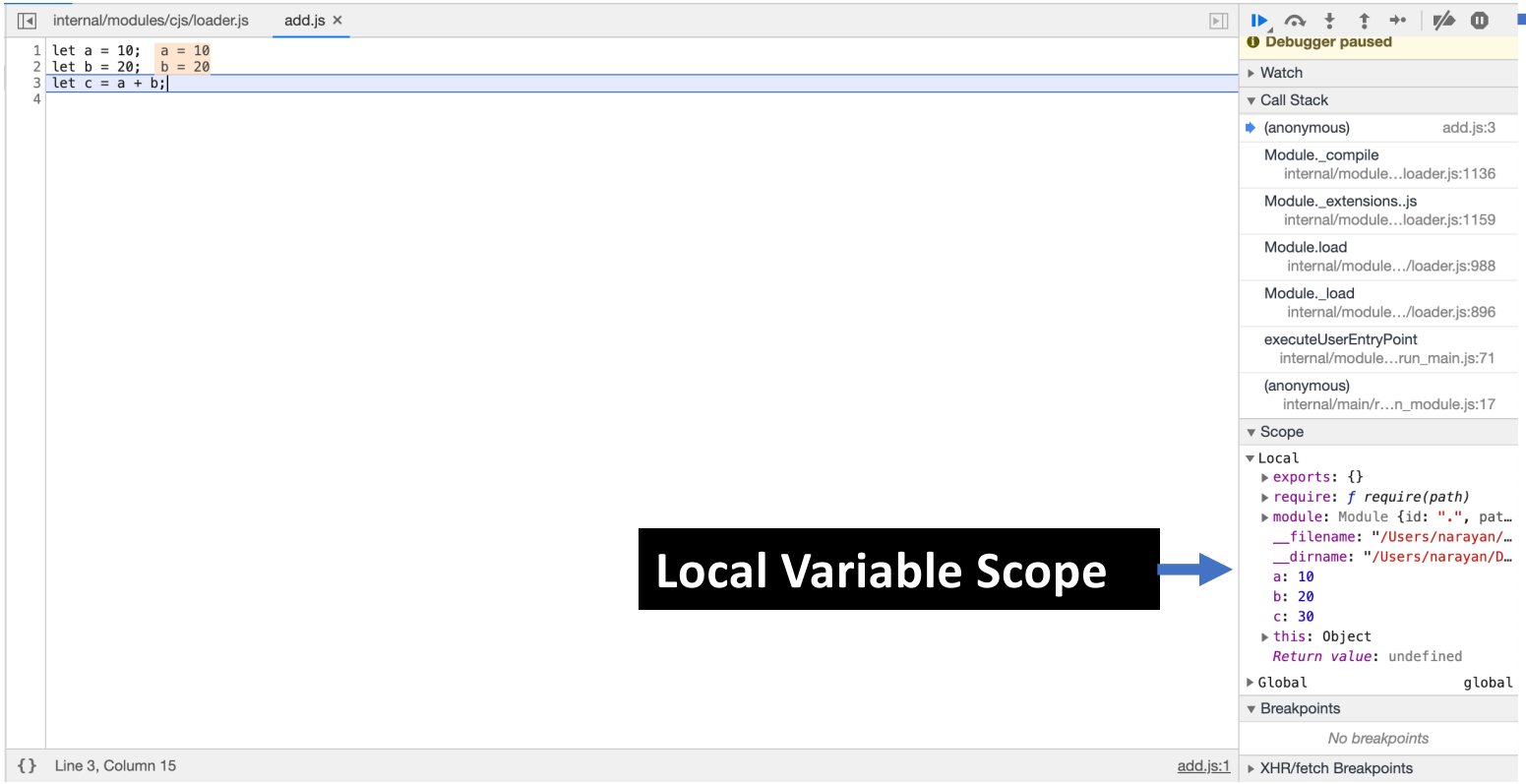
Node.js Debugger



Node.js Debugger Window

BridgeLabz
Employability Delivered

add.js debug console



Local Variable Scope

***F8* – Resume Script Execution**
F9* – *Execute the Current Line
F10* – *Step Over
F11* – *Step Into
Shift F11* – *Step Out

Step 5: Browser will open in Debug mode.

Step 6: On the debug prompt use ***next*** to proceed to next line

Step 7: ***cont*** to complete the Program execution and

Step 8: use ***.exit*** or ***ctrlC twice*** to exit node command prompt.

JS Data Types

BridgeLabz
Employability Delivered

```
var val;  val = "foo"
var datatype = typeof val;  datatype = "function"

var val = 0;  val = "foo"
var datatype = typeof val;  datatype = "function"

var val = 10n;  val = "foo"
var datatype = typeof val;  datatype = "function"

var val = true;  val = "foo"
var datatype = typeof val;  datatype = "function"

var val = "foo";  val = "foo"
var datatype = typeof val;  datatype = "function"

var datatype = typeof Symbol("id")  datatype = "function"

var datatype = typeof Math;  datatype = "function"

var datatype = typeof null;  datatype = "function"

let sayHi = function() {  sayHi = ƒ ()
    console.log.apply("Say Hi");
};
var datatype = typeof sayHi;  datatype = "function", sayHi =
console.log(datatype);
```

Summary

There are 8 basic data types in JavaScript.

- `number` for numbers of any kind: integer or floating-point, integers are limited by $\pm(2^{53}-1)$.
- `bigint` is for integer numbers of arbitrary length.
- `string` for strings. A string may have zero or more characters, there's no separate single-character type.
- `boolean` for `true`/`false`.
- `null` for unknown values – a standalone type that has a single value `null`.
- `undefined` for unassigned values – a standalone type that has a single value `undefined`.
- `object` for more complex data structures.
- `symbol` for unique identifiers.

The `typeof` operator allows us to see which type is stored in a variable.

- Two forms: `typeof x` or `typeof(x)`.
- Returns a string with the name of the type, like `"string"`.
- For `null` returns `"object"` – this is an error in the language, it's not actually an object.

In the next chapters, we'll concentrate on primitive values and once we're familiar with them, we'll move on to objects.

BridgeLabz
Employability Delivered

Demonstrate Node Programming Constructs

Programming Constructs Classification

1. Sequences
2. Selection
3. Functions
4. Repetition
5. Arrays
6. Map
7. Patterns

Note:  [js cheatsheet]

1. Sequence Statement

BridgeLabz
Employability Delivered

A sequence construct tells the CPU (processor) which statement is to be executed next.

Step 1: Create NodeDevelopment Directory using your terminal

Step 2: In Node Development Directory create hello.js file using ***nano –T 3 hello.js***

Step 3: Execute using ***node hello.js***

Step 4: Print hello using ***console.log*** function

```
console.log('hello');
hello.js (END)
```

Use *console.log* function to print to the terminal

2. Selection Statement

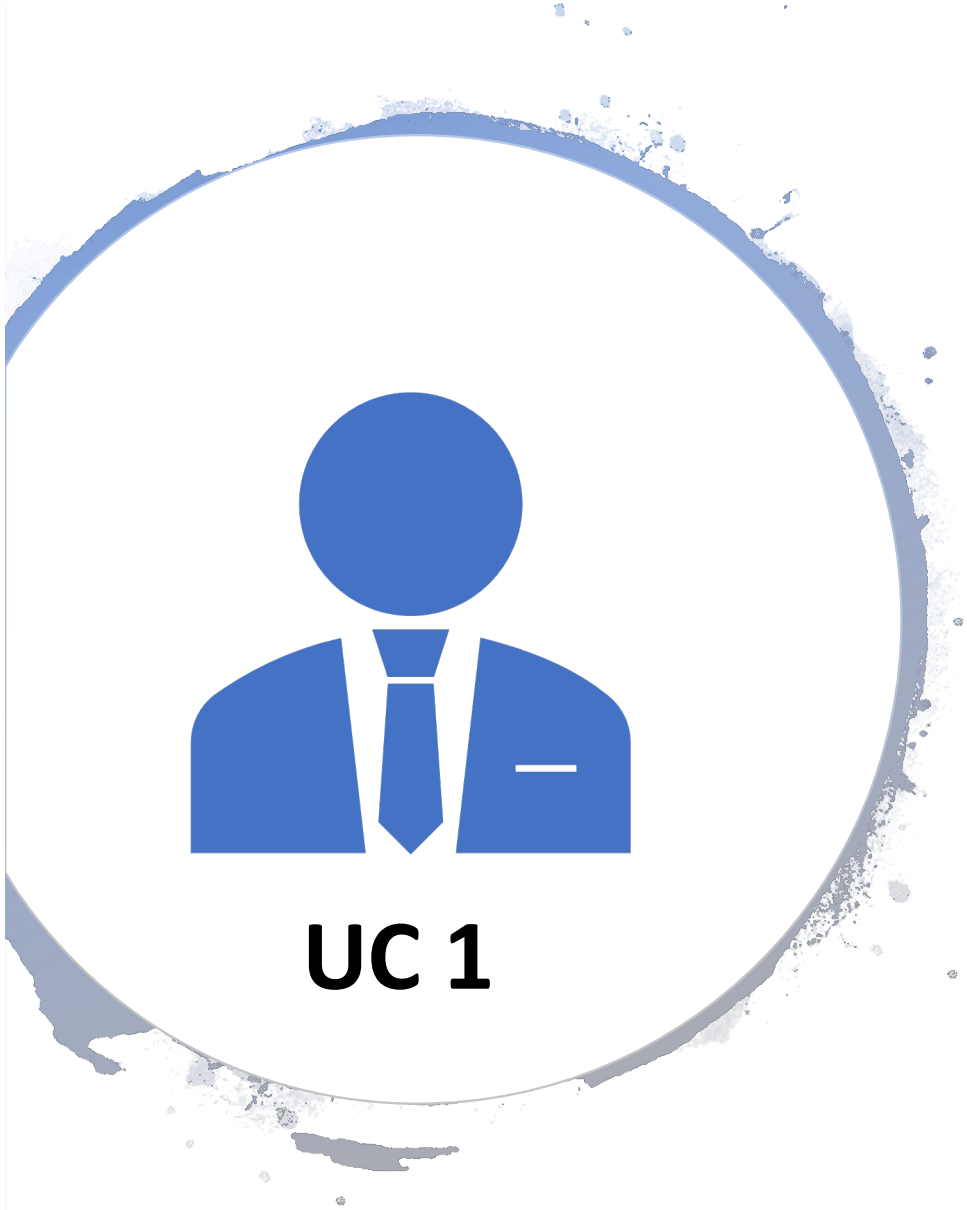
1. A selection statement provides for selection between alternatives
2. A program can take certain route depending on a situation and selection statements help in choosing between the routes.

2. Selection Statement Types

1. If statements
2. Case Statements
3. Pattern Matching

If Statements

```
if ((age >= 14) && (age < 19)) {          // logical condition
    status = "Eligible.";                 // executed if condition is true
} else {                                  // else block is optional
    status = "Not eligible.";             // executed if condition is false
}
```



BridgeLabz
Employability Delivered

Ability to Check Employee is present or Absent

- Use ***Math.Random*** to check Absent or Present
- Use ***const*** which signal that the identifier won't be reassigned. And use ***let*** where the variable may be reassigned. Both have Block Scope. Traditional ***var*** has function scope

BridgeLabz
Employability Delivered

Ability to Calculate Daily Employee Wage based on part time or full time work

- Use ***Math.Random*** to check No Time, Part Time or Full Time
- Assume Part Time is 4 Hrs and Full time is 8 Hrs and per hour wage is $20
- Solve Using Switch Statement

3. Functions

- Functions are a great way to reuse code.
- Think of a function as a small script within a script. It's a small chunk of code which you may call multiple times within your script. They are particularly useful if you have certain tasks which need to be performed several times.

```
function addNumbers(a, b) {
    return a + b; ;
}
x = addNumbers(1, 2);
```

BridgeLabz
Employability Delivered

UC 3

Refactor the Code to write a function to get work hours

4. Repetition Statement

BridgeLabz
Employability Delivered

A repetition construct causes a group of one or more program statements to be invoked repeatedly until some end condition is met.

4. Repetition Statement Types

1. **Fixed count loops** - repeat a predefine number of times.

   for (…) { … }

2. **Variable count loops** - repeat an unspecified number of times.

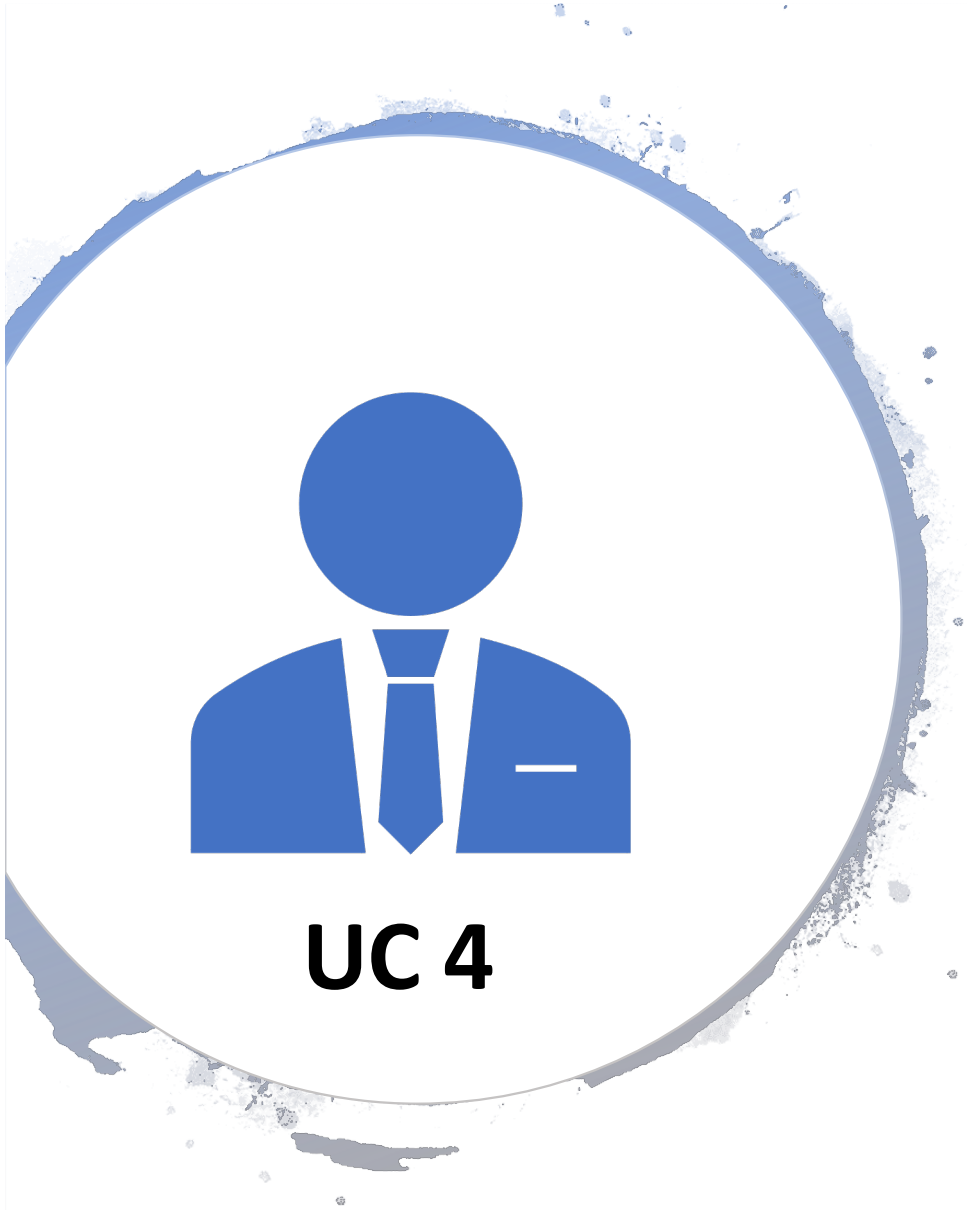
   while (…) { … }

for Loop Statement

```
let dogs = ["Bulldog", "Beagle", "Labrador"];

// OLD WAY
var allDogs = "";
for (var i = 0; i < dogs.length; i++) {
    allDogs += dogs[i] + " ";
}
console.log("OLD: " + allDogs)

// NEW WAY
allDogs = "";
for (let dog of dogs) {
   allDogs += dog + " ";
}
console.log("NEW : " + allDogs);
```



BridgeLabz
Employability Delivered

Calculating Wages for a Month assuming 20 Working Days in a Month

while Loop Statement

```
let dogs = ["Bulldog", "Beagle", "Labrador"];

let i = 0;
let allDogs = "";
while (i < dogs.length) {
    allDogs += dogs[i++] + " ";
}
console.log("while: " + allDogs)

i = 0;
allDogs = "";
do {
   allDogs += dogs[i++] + " ";
} while (i < dogs.length)
```

BridgeLabz
Employability Delivered

UC 5

Calculate Wages till a condition of total working hours of 160 or max days of 20 is reached for a month

BridgeLabz

Employability Delivered

Thank You