Assignment For Day 2

1. What is lexical structure?

A. A programming language's lexical structure specifies set of some basic rules about how a code should be written in it. Rules like what variable names looks like, the delimiter characters for comments, and how one program statement is separated from the next.

2. What is Unicode?

A. Unicode is a standard character set that numbers and defines characters from the world's different languages, writing systems, and symbols.

Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.

3. Explain all the keywords present in the JavaScript with examples.

A. Keywords are reserved words that are part of the syntax in the programming language. For example,

```
const a = 'hello';
```

Here, const is a keyword that denotes that a is a constant.Keywords cannot be used to name identifiers.

The list of keywords available in JavaScript are

await	break	case	catch	class	const
Eg: let value = await promise;	if (i === 3) { break; }	switch(exp) {case val1: [break;]}	try { Function(); } catch (error) { console.error(error);}	class Human { }	const a = 1;
continue	debugger	default	delete	do	else
if (i === 3) { continue; }	function BuggyCode() {	default	delete expression	do statement while (cond);	if (cond) { state1 } else { state2}

Assignment For Day 2

	debugger; }				
enum	export	extends	false	finally	for

enum	let k;
	export default k = 12;
		class ChildClass extends ParentClass { /* ... */ }	if (false)
			{-----}
				try {
				.....}
				finally {
				.....
				}
					for (let i = 0; i < 9; i++) {
					.....
					}

function	if	implements	import	in	instanceof
function a() {----}	If(cond) {---}	implements	import "module-name";	'PI' in Math	let o = new C() o instanceof C
interface	let	new	null	package	private

interface	Let a = 10	const car1 = new Car();	const f = null;	A package is a file or directory that is described by a package.json file.	class ClassWithPrivateField {
					#privateField;
					}

protected	public	return	super	switch	static
protected	public	return -1;	super();	switch(exp) {case val1: [break;]}	static staticProperty = 'someValue';
this	throw	try	true	typeof	var

this.b = "JS";	throw 'Error2';
	
		try {
		Function();
		} catch (error) {
		console.error(error);}
			let a = true;	console.log(typeof 42);	var a =2;

void	while	with	yield		

void 2 == '2';
void (2 == '2')
	while(i<10)
	{---}
		function f(x, o) {
		with (o) {
		console.log(x);
		} }
			function* countAppleSales() {
			let saleList = [3, 7, 5]
			for (let i = 0; i < saleList.length; i++) {
			yield saleList[i]
			}}		

Assignment For Day 2

4. What are shorthand operators, explain with a suitable example?

A. A shorthand operator is a shorter way to express something that is already available in the programming language.

- x = x + y -> x += y
- x = x - y -> x -= y
- x = x * y -> x *= y
- x = x / y -> x /= y
- x = x % y -> x %= y

5. What is “use Strict” in JavaScript?

A. The purpose of "use strict" is to indicate that the code should be executed in "strict mode".

Strict mode makes several changes to normal JavaScript semantics:

- Eliminates some JavaScript silent errors by changing them to throw errors.
- Fixes mistakes that make it difficult for JavaScript engines to perform optimizations: strict mode code can sometimes be made to run faster than identical code that's not strict mode.
- Prohibits some syntax likely to be defined in future versions of ECMAScript.

To invoke strict mode for an entire script, put the exact statement "use strict"; (or 'use strict';) before any other statements.

Eg: // Whole-script strict mode syntax

```
'use strict';
var v = "Hi! I'm a strict mode script!";
```

Eg:

```
function strict() {
'use strict';                                   // Function-level strict mode syntax
function nested() { return 'And so am I!'; }
return "Hi!  I'm a strict mode function!  " + nested();
}
function notStrict() { return "I'm not strict."; }
```