→ **Conditional statements :**

If-else : → use to check condition whether the given condition is true or false

Syntax -

```
    if (condition){
       // Body - task 1
    } else {
         // if not task1, then task 2
    }
```

Nested if-else : → multiple conditions & statements

Syntax -

```
    if (condition) {
       // Body
    } else if (condition 2) {
       // Body
    } else if (codition 3) {
       // Body
    } else {
       // Body
    }
```

values
.equals → compare content
'==' → address comparison
references

→ **Loops :** iteration

for loop : → used when number of loops/iterations are known.

Syntax -

```
    for (initialization; condition; incr|dec){
       // body
    }
```

while loop :→ used when no. of loops are unknown

Syntax -

```
while (condition){
      // body
      // incr/decr
}
```

do while loop :→ used when we want to execute statement at least once.

:→ Also known to be exit control loop as it checks condition after execution of statement.

Syntax -

```
do {
    //body
    //update statement / incr or dcr
}while(condition);
```

while loop	Do while loop
* no iteration fixed	* ~~iteration~~ executes statement at least once
* entry controlled loop	* exit controlled loop
* no semicolon at while(condition)	* semicolon at while(condition);

.trim() → removes spaces ~~characters~~ between strings & characters.

.charAt() → returns a char value at the given index number.

→ **Switch case** : jump to various cases based on expression.
: duplicate cases not allowed.

Syntax – switch (expression) {

```
switch (expression) {
        case one :
              // code block
              break;    — terminate the sequence
        case two :         (if no break, it
              // code block   will continue
              break;          with other
        default :             cases)
              // code block
}
```

→ it works when none of the given cases executed / or inputted. ← default :

→ if default is b/w cases use break;

: **Nested switch case** –

Syntax –

```
switch (expression) {
      case one :
            // code
            break;
      case two :
            switch (expression) {
                  case one :
                        // code
                        break;
                  default :
                        // code
            }
            break;
      default :
            // code
}
```