Control structures are used when we want control to take a particular path through the code, depending on certain conditions.

Control structures include branches, cases and loops.

**if statements:**
- Conditional statements are used to perform different actions based on different conditions.
- Include if, if - else, if - else ladders and switch.

**Loops:**
- Loops can execute a block of code as long as a specified condition is true.
- Include while, do-while, for and for/in
Comparisons

- Boolean values in JavaScript can be true or false.
- Borrowing from C like languages, “real” values (non zero values) are considered true.
- 0, -0, null, undefined, empty, Nan and false evaluate to false.
- You can compare values using comparison operators like <, ==, etc.
- You can chain comparisons together using logical operators.
- Comparisons evaluate to true or false.
In JavaScript we have the following conditional 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

▶ Use **switch** to specify many alternative blocks of code to be executed
Loops

- **while** - loops through a block of code while a specified condition is true.
- **do/while** - also loops through a block of code while a specified condition is true.
- **for** - loops through a block of code a number of times.
- **for/in** - loops through the properties of an object.
break and continue

- break and continue are JavaScript keywords used to manipulate the loop iterations.
- The break statement “jumps out” of a loop.
- The continue statement “jumps over” one iteration in the loop.
A switch statement is often convenient for occasions in which there are multiple cases to choose from. The syntax format is:

```java
switch (expression)
{
    case constant:
        statements
    case constant:
        statements
    ...(as many case labels as needed)
    default: // optional label
        statements
}
```
The switch statement evaluates the expression, and then compares it to the values in the case labels. If it finds a match, execution of code jumps to that case label.

The values in case labels must be constants.

If you want to execute code only in the case that you jump to, end the case with a break statement, otherwise execution of code will ”fall through” to the next case.
Functions

- A JavaScript function is a block of code designed to perform a particular task. A JavaScript function is executed when it’s invoked.
- A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ()
- Function names can contain letters, digits, underscores, and dollar signs.
- The parentheses may include parameter names separated by commas: (parameter1, parameter2, ...)
- The code to be executed is placed inside curly brackets: {}
The code inside the function will execute when “something” invokes (calls) the function:
  - When an event occurs (when a user clicks a button)
  - When it is invoked (called) from JavaScript code
  - Automatically (self invoked)

When JavaScript reaches a return statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will “return” to execute the code after the invoking statement.

Functions often compute a return value. The return value is “returned” back to the “caller”.