Variables in JavaScript

Variables in JavaScript are containers for storing data values. They can hold values of various data types such as numbers, strings, objects, etc.

Syntax:

```
var variableName = value;
let variableName = value;
const variableName = value;
```

Variable Declarations:

var: Declares a variable, optionally initializing it to a value. It's function-scoped or globally scoped.

let: Declares a block-scoped local variable, optionally initializing it to a value. Introduced in ES6.

const: Declares a block-scoped, read-only named constant. It must be initialized at the time of declaration. Introduced in ES6.

Example:

```
var x = 10;
let y = 20;
const z = 30;
```

Variable Scope:
Global Scope: Variables declared outside any function are in the global scope and can be accessed from anywhere in the code.
Local Scope: Variables declared within a function are in the local scope and can only be accessed within that function.
Block Scope: Variables declared with let or const inside a block {} can only be accessed within that block.

Hoisting:
JavaScript hoists variable declarations (not initializations) to the top of their containing scope.

Control Statements in JavaScript

Control statements control the flow of execution of the code. They include conditional statements and loops.

Conditional Statements:

if statement

Evaluates a condition and executes the block of code if the condition is true.

```
if (condition) {
    // code to be executed if condition is true
}
```

if...else statement

Executes one block of code if the condition is true, and another if the condition is false.

```
if (condition) {
    // code to be executed if condition is true
} else {
    // code to be executed if condition is false
}
```

else if statement

Allows for multiple conditions to be checked.

```
if (condition1) {
    // code to be executed if condition1 is true
} else if (condition2) {
    // code to be executed if condition2 is true
} else {
    // code to be executed if all conditions are false
}
```

switch statement

Evaluates an expression and executes code based on matching case clauses.

```
switch(expression) {
    case value1:
        // code to be executed if expression === value1
        break;
    case value2:
        // code to be executed if expression === value2
        break;
    default:
        // code to be executed if expression doesn't match any case
}
```

Loops in JavaScript
Loops are used to execute a block of code repeatedly as long as a specified condition is true.

Types of Loops:

for loop

Loops through a block of code a number of times.

```
for (initialization; condition; increment) {
    // code to be executed
}
```

Example:

```
for (let i = 0; i < 5; i++) {
    console.log(i);
}
```

while loop

Loops through a block of code as long as a specified condition is true.

```
while (condition) {
    // code to be executed
}
```

Example:

```
let i = 0;
while (i < 5) {
    console.log(i);
    i++;
}
```

do...while loop

Also loops through a block of code as long as a specified condition is true, but the block of code is executed at least once.

```
do {
    // code to be executed
} while (condition);
```

Example:

```
let i = 0;
do {
    console.log(i);
    i++;
} while (i < 5);
```

for...in loop

Loops through the properties of an object.

```
for (variable in object) {
    // code to be executed
```

```
}
```
Example:

```
const person = {fname: "John", lname: "Doe", age: 25};
for (let key in person) {
    console.log(person[key]);
}
```
for...of loop
Loops through the values of an iterable object such as an array or a string.

```
for (variable of iterable) {
    // code to be executed
}
```
Example:

```
const array = [1, 2, 3, 4, 5];
for (let value of array) {
    console.log(value);
}
```