Variables & Constants

A variable is a data container which holds some data and in JS we create a variable by using let keywork.

Let username = ‘Nadeem’

Now with a variable create, we can always reassign it and store a new value in there

Username = ‘Khawja’

Constant

The **const** declaration creates block-scoped constants, much like variables declared using the let keyword. The value of a constant can't be changed through reassignment (i.e. by using the assignment operator), and it can't be redeclared (i.e. through a variable declaration). However, if a constant is an object or array its properties or items can be updated or removed

1. const number = 42;

2.

3. try {

4. number = 99;

5. } catch (err) {

6. console.log(err);

7. // Expected output: TypeError: invalid assignment to const `number'

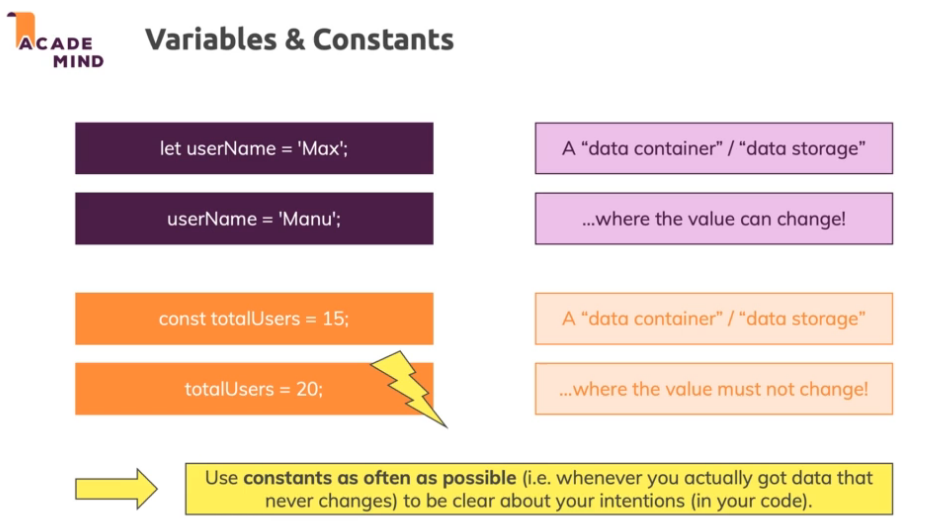
8. // (Note: the exact output may be browser-dependent)

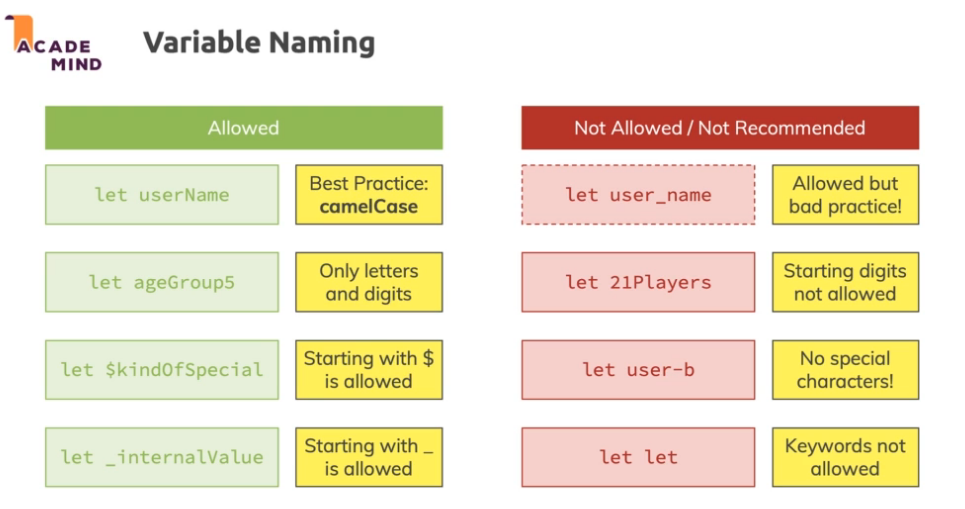
9. }

10.

11. console.log(number);

12. // Expected output: 42





**Grammar and Types**

Basics

JavaScript is **case-sensitive** and uses the **Unicode** character set.

const fruh = ‘fooBar’;

const Fruh = ‘fooTwo’;

the variable fruh is not the same as Fruh because JavaScript is case sensitive.

In JavaScript, instructions are called statements and are separated by semicolons (;).

A semicolon is not necessary after a statement if it is written on its own line. But if more than one statement on a line is desired, then they must be separated by semicolons.

**Comments**

// a one line comment

/\* this is a longer,

\* multi-line comment

\*/

String

const defaultResult = 0;

let currentResult = defaultResult;

currentResult = (currentResult + 10) \* 3 / 2 -1;

//let calculationDescription = ' ( ' +defaultResult + ' + 10) \* 3 / 2 -1';

// Whats the use of `` --> it avoids the use of manual string concatination with variables so the above line can also be written as

let calculationDescription = `( ${defaultResult} + 10) \* 3 / 2 -1`;

// using `` is called as template literal in JavaScript

outputResult(currentResult,calculationDescription);

**Declarations**

JavaScript has three kinds of variable declarations.

var : Declares a variable, optionally initializing it to a value.

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

const : Declares a block-scoped, read-only named constant.

* With the keyword var. For example, var x = 42. This syntax can be used to declare both **local** and **global** variables, depending on the *execution context*.
* With the keyword const or let. For example, let y = 13. This syntax can be used to declare a block-scope local variable.

In a statement like let x = 42, the let x part is called a *declaration*, and the = 42 part is called an *initializer*. The declaration allows the variable to be accessed later in code without throwing a ReferenceError, while the initializer assigns a value to the variable. In var and let declarations, the initializer is optional. If a variable is declared without an initializer, it is assigned the value **undefined**.

let x;

console.log(x); // logs "undefined"

const declarations always need an initializer, because they forbid any kind of assignment after declaration, and implicitly initializing it with undefined is likely a programmer mistake.

const x; // SyntaxError: Missing initializer in const declaratio