

Variable Instantiation

let variableName = value

let required keyword to initialize a variable

variableName arbitrary name, no spacing, must contain letter, no punctuation, cannot use reserved keywords, camel-cased

= assign

value any data type

camel-case no spacing between words, every word except the first is capitalized e.g a red balloon -> aRedBalloon

the value assigned can be referenced in later parts of the program through variableName.

Only one value can be assigned to one variable, i.e one instantiation per variable, if let var1 = 'string' previously, cannot let var1 = 2 again later in the code

Operators

+ plus

- minus

* multiply

/ divide (5/2 = 2.5)

% remainder (5 % 2 = 1)

> more than

< less than

>= more than or equals

<= less than or equals

i++ i = i + 1

i-- i = i - 1

== equal value

=== equal value and data type

i (+, -, *, /) = shorthand for i = i + value, i = i - value, etc

Data Types

String 'string' Anything in quotes. If there are quotations inside the string, use a different type of quotation (' and ")

Number 1, 23, 400 number

Boolean true, false true/false value

Character 'a', '2' single input

Undefined let variable variable instantiated but not assigned a value

Null variable not defined(instantiated)

Array

Array ['string', 2, true]

- initialized with square brackets
- can contain all data types, including arrays and objects
- ordered list of values, starting from index 0 to refer to first element
- get item in array by referring to its index (array[0] gets 'string', array[1] gets 2)

Object

Object let object = { key1: value1, key2: value2 }

- similar to array, but replace index with key(string)
- can contain all data types, including arrays and objects
- refer to objects in 2 ways
- 1. object.key1 gets value1
- 2. object["key2"] gets value2. when using square brackets, put the key in string format

Function Example

```
let num1 = 1;
let num2 = 20;
let result = addTogether(num1, num2);
```

*num1 and num2 becomes firstNum and secondNum respectively
if no return value, calculations done in the function cannot be carried out of the*

Loops

if if condition is true, execute block
else must be used with if, executes if condition is false

if additional if statements after the else first if statement

while while condition is true, keep executing block

for for (let i = 0; i < 10; i++), a condensed while loop

for for (let number of numbers)

for for (let number in numbers)

```
loop (condition) {
  execute code in block
}
```

for

for has multiple uses

1. a condensed while loop

```
for (let i = 0; i < 10; i++) {}
```

2. To loop through an array/object

```
let numbers = [20, 30, 10, 50, 70];
```

```
for (let number of numbers) {
```

use value

each loop uses **number** = 20, then 30, 10 ...

```
}
```

```
for (let numbers in numbers) {
```

use index/key

each loop uses **number** = 0, then 1, 2 ...

to get value, use numbers[number] (object[index])

```
}
```

Function

```
function foo(param1, param2) { return param1 + param2 }
```

function keyword to instantiate function
foo function name, use to describe function purpose, camel-case

param1, param2 parameters to put into function (optional)

return value to get back from function (optional)

functions are called with brackets -> foo()
if function has parameters, function must be called with parameters -> foo(param)
parameters are assigned to new names for usage in the function (see below)

function

```
function addTogether(firstNum,  
secondNum) {  
return firstNum + secondNum;  
}
```

result will get the returned value of 21



By **tegdsd12**

cheatography.com/tegdsd12/

Published 14th April, 2020.

Last updated 14th April, 2020.

Page 1 of 2.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>