{JS}
MASTERY

{JS}
MASTERY

JavaScript CheatSheet Basic

Variables

`var variableName = value`

Can be reassigned and are only available inside the function they're created in. Its function scoped.

`const variableName = value`

Cannot be reassigned and not accessible before they appear within the code. Its block scoped.

`let variableName = value`

It can be reassigned but its similar to const i.e. block scoped.

If variables are not created inside a function or block they are globally scoped.

What is the block?

A block is a set of opening and closing curly brackets.

Variables

Primitive Data Types:

`Number`	5, 6.5, 7 etc
`String`	“Hello everyone” etc
`Boolean`	true or false
`Null`	represents null i.e. no value at all
`Undefined`	A variable that has not been assigned a value is undefined.
`Symbol`	used as an identifier for object properties.

Variables

Non-Primitive Data Types:

`Object`	instance through which we can access members
`Array`	group of similar values
`RegExp`	represents regular expression

Operators

Basic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
( )	Grouping operator
%	Modulus (remainder)
++	Increment numbers
- -	Decrement numbers

Operators

Comparison Operators

`==`	Equal to
`===`	Equal value and equal type
`!=`	Not equal
`!==`	Not equal value or not equal type
`>`	Greater than
`<`	Less than
`>=`	Greater than or equal to
`<=`	Less than or equal to

Operators

Logical Operators

&&	Logical and
||	Logical or
!	Logical not

Bitwise Operators

Bitwise operators in Javascript are mostly used for numerical conversions/computations, because sometimes they're much faster than their Math or parseInt equivalents

Operators

Bitwise Operators

&	AND statement
|	OR statement
~	NOT
^	XOR
<<	Zero fill left shift
>>	Signed right shift
>>>	Zero Fill right shift

Functions

Normal Function Declaration

```
function name (parameter) {
  // statements
}
```

Function stored in a variable

```
let name = function (parameter) {
  // statements
}
```

Arrow Function

```
const name = (parameter) => {
  // statements
}
```

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

If - Else Statements

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

Conditional Statements

If - Else If - Else Statements

```
if (condition1) {
  // code to be executed if the
     condition is true

} else if (condition2) {
  // code to be executed if the
     condition1 is false and
     condition2 is true

} else {
  // code to be executed if
     condition1 is false and
     condition2 is false
}
```

Conditional Statements

Switch Statement

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.
- If there is no match, the default code block is executed.

Conditional Statements

Ternary Operator

```
condition ? exprIfTrue : exprIfFalse
```

`condition`

An expression whose value is used as a condition.

`exprIfTrue`

An expression which is executed if the condition is truthy.

`exprIfFalse`

An expression which is executed if the condition is falsy.

Truthy / Falsy Values

FALSY Values

- false
- 0 (zero)
- "", '', `` (empty strings)
- null
- undefined
- NaN (not a number)

Note : Empty array ([]) is not falsy

TRUTHY Values

- Everything that is not FALSY

Strings

```
let variableName = “Hello world”
```

Escape Characters

`\'`	Single quote
`\"`	Double quote
`\\`	Backslash
`\b`	Backspace
`\f`	Form feed
`\n`	New line
`\r`	Carriage return
`\t`	Horizontal tabulator
`\v`	Vertical Tabulator

Want to Become a JavaScript Master ?

If you're serious about learning JavaScript and getting a job as a web developer. I highly encourage you to enroll in my **Complete Path to JavaScript Mastery** Course. Don't waste your time following disconnected, outdated tutorials. My Course has everything you need in one place:

This course has everything you need in one place:

- 10 hours of HD video
- Unlimited access
- Self-paced learning - take your time if you prefer
- Watch it online or download and watch offline
- 30-day money-back guarantee
- Certificate of completion

Special Coupon Code

The price for this course is $129 but the first 200 people who have downloaded this cheat sheet can get more than 50% off using the coupon code:

CHEATSHEET

https://www.completepathtojavascriptmastery.com