JAVASCIRPT BASICS

Monday, December 3, 2018 11:12 AM

VARIABLES:
Values are provided to a application or a program using a variable.

'let' - is reserved by javascript to declare a variable.

```
let name = 'Vatsal Saglani';
console.log(name);
```

CONCATINATION:

```
// console.log(name);

let firstName = 'Vatsal'
let lastName = 'Saglani'
let fullName = firstName + ' ' + lastName

console.log(fullName);
```

Error when using the same variable again

```
let petName = 'None';
let petName = 'log';

console.log(petName);
```

```
let petName = 'log';
    ^

SyntaxError: Identifier 'petName' has already been declared
```

Can change the value assigned to the variable name to change the value

```
let petName = 'None';
petName = 'log';

console.log(petName);
```

```
✘ thevatsal@Vatsals-MacBook-Pro  /PROJECTS/Javascirpt/all-codes  node variables.js
log
```

NUMBERS:

```
let num = 40 + 2

console.log(num)
```

```
let age = 21
let dogYears = (age + 1)/7

console.log(dogYears)
```

Displaying number within a string using console log.

```
let studentScore = 14
let maxScore = 20

percent = (studentScore/maxScore)*100

console.log(`the percent is ${percent}`)
```

Temperature conversion using Javascirpt

```
let tempF = 92

let tempC = (tempF - 32)*(5/9)

console.log(`the temperature in celcius is ${tempC}`);
```

BOOLEANS: Works with both strings and integers(equality and inequality)

```
let temp = 31
let isFreezing = temp === 31

console.log(isFreezing);
```

Output: true

```
let temp = 31
let isFreezing = temp > 32

console.log(isFreezing);
```

Output: false

```
let temp = 32
let isFreezing = temp <= 32

console.log(isFreezing);
```

Output: true

```
let temp = 31
let isFreezing = temp >= 32

console.log(isFreezing);
```

Output: false

```
let temp = 31
let isFreezing = temp !== 32

console.log(isFreezing);
```

Output: true

```
// !== not equal operator

let temp = 31
let isFreezing = temp < 32

console.log(isFreezing);
```

Output: true

```
let age = 21
let isChild = age <= 7
let isSenior = age >= 65

console.log(isChild)
console.log(isSenior)
```

Output: false, false

CONDITIONAL STATEMENTS:

```
let temp = 44
let isFreezing = temp <= 32

if (isFreezing){
    console.log('It is freezing outside!')
}
```

```
let temp = 112
// let isFreezing = temp <= 32

if (temp <= 32){
    console.log('It is freezing outside!')
}

if (temp >= 110){
    console.log('It is hot outside')
}
```

```
let isAccountLocked = false

if (isAccountLocked){
    console.log('Is account locked')
} else {
    console.log('Not locked')
}
```

Output: Not locked

```
let isAccountLocked = true
let userRole = 'admin'
if (isAccountLocked){
    console.log('Is account locked')
} else if(userRole === 'admin') {
    console.log('Welcome Admin')
} else {
    console.log('Welcome')
}
```

```
let temp = 109
// let isFreezing = temp <= 32

if (temp <= 32){
    console.log('It is freezing outside!')
} else if(temp >= 110) {
    console.log('Its hot outside')
} else {
    console.log('Its fine to go outside')
}
```

```
let age = 2

if (age<=7){
    console.log('Child Discount')
}

if (age>=65){
    console.log('Senior Citizen discount')
}
```

LOGICAL AND and OR.

```
let temp = 33

if (temp >= 60 && temp <=95){
    console.log('Its nice outside');
} else {
    console.log('its not so great outside
}
```

```
let temp = 4

// if (temp >= 60 && temp <=95){
//     console.log('Its nice outside');
// } else {
//     console.log('its not so great outside')
// }

if(temp <= 0 || temp >=120){
    console.log('Its dangerous to go outside');
} else {
    console.log('Its not that dangerous to go outside');
}
```

```
let isGuestVegan_1 = true
let isGuestVegan_2 = true

if(isGuestVegan_1 && isGuestVegan_2){
    console.log('Offer only vegan food.');
} else if(isGuestVegan_1 || isGuestVegan_2) {
    console.log('Offer some vegan options.')
} else {
    console.log('Offer anything you want.')
}
```

```
let temp = 160

if (temp >=60 && temp <= 90) {
    console.log('It is pretty nice outside');
```

```
} else if(temp <= 0 || temp >= 120) {
    console.log('Its dangerous to go outside')
} else {
    console.log('Do whatever you want')
}
```

VARIABLE SCOPE IN JAVASCRIPT: If you define a variable inside a scope you can only us
inside that scope. If you try and use that variable out of that scope it will throw an error
For example look at the following code snippets:

```
let varOne = 'varOne';

if( true ){
    // if SCOPE
    console.log(varOne)
    // let varTwo = 'varTwo'

}
```

(WILL RUN SUCCESSFULLY)

```
let varOne = 'varOne';

if( true ){
    // if SCOPE
    console.log(varOne)
    let varTwo = 'varTwo'

}

console.log(varTwo);
```

(WILL TROW AN ERROR: varTwo not defined)
The above given code will throw an error because we are declaring the varable *varTwo* o
global scope and using it in the global scope.

```
let varOne = 'varOne';
```

e that variable
.

outside the

```
if( true ){
    // if SCOPE
    console.log(varOne)
    let varTwo = 'varTwo'
    console.log(varTwo);


}
```

(WILL RUN SUCCESSFULLY)

- JAVASCIPT USES THE CONCEPT OF LEXICAL SCOPE (STATIC SCOPE)
- ONE SHOULD WORK INSIDE OF THE CODE BLOCKS
- *In the above given snippets the **"if"** statement is a code block.*
- THERE ARE TWO TYPES OF SCOPES IN JS:
    - GLOBAL SCOPE: THE VARIABLES DEFINED/DECLARED OUTSIDE OF THE CODE UNDER THE GLOBAL SCOPE
    - LOCAL SCOPE: THE VARIABLES DEFINED/DECLARED INSIDE OF ANY CODE BLO UNDER THE LOCAL SCOPE
    - *In the above given code snippets varTwo is a variable defined inside a code b under a local scope, and varOne is a variable which is defined outside of any comes under the global scope*
    - *One can access the global scope variables from anywhere in the script.*
    - *But a variable inside a local scope can only be tampered or touched inside th*

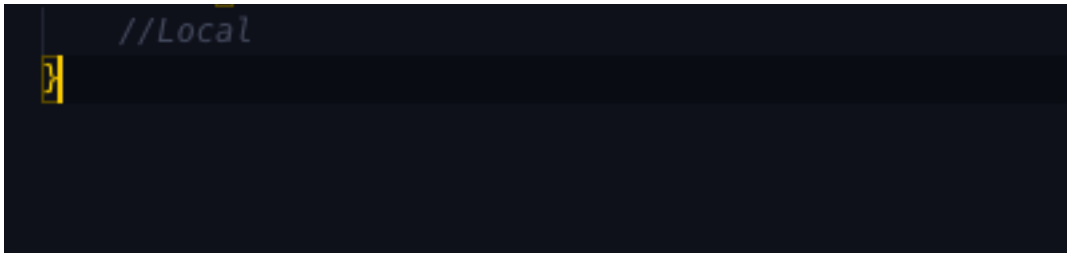
```
// Global

let name = 'Vatsal'

if(true) {
    // Local
    if(true) {
        // Local
        console.log(name)
    }
}


if(true) {
```

E BLOCKS COME

OCK COMES

lock so it comes
code block so it

at code block.



(WILL WORK SUCCESSFULLY-output-"Vatsal")

(WILL WORK - OUTPUT - "vat")

Variable Shadowing: When a variable in local scope uses its value in place of the varia
same name) in the parent scope.

ble (with the

```
    if(true) {
        // Local
        console.log(name)
    }
}


if(true) {
    //Local
    console.log(name)
}
```

log(message?: any, ...optionalPa

```
thevatsal@Vatsals-MacBook-Pro /PROJECTS/Javascirpt/all-codes node scope_2.js
vat
Vatsal
```

```
let name = 'Vatsal'

if(true) {
    // Local
    let name = 'vat'
    if(true) {
        // Local
        name = 'tat'
        console.log(name)
    }
}


if(true) {
    //Local
    console.log(name)
}
```

```
thevatsal@Vatsals-MacBook-Pro /PROJECTS/Javascirpt/all-codes node scope_2.js
tat
Vatsal
```

```
// Global

// let name = 'Vatsal'

if(true) {
    // Local
    // let name = 'vat'
    if(true) {
        // Local
        name = 'tat'
        console.log(name)
    }
}


if(true) {
    //Local
    console.log(name)
}
```

```
thevatsal@Vatsals-MacBook-Pro /PROJECTS/Javascirpt/all-codes node scope_2.js
tat
tat
```

***Note:** As one may have noted, in the program snippet above we haven't declared any nam using let, but in the output we are getting the correct result, because, when the *"log"* comm checks for name in the parent scope it doesn't find one and when goes above to check for th the global scope and again it doesn't find that so it **creates** a name variable by itself.

This something one must take care about. Leaked global.
Just declare your variables.

e variable
hand goes
he same in