JavaScript Variables

Session-1

CLARUSWAY
WAY TO REINVENT YOURSELF

Table of Contents

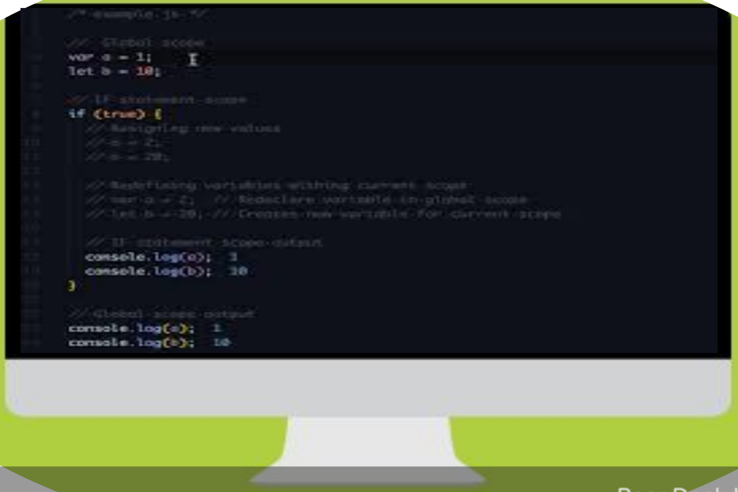
- JavaScript Variables
- The Assignment Operator
- Naming Rules
- Reserved Words in JavaScript
- let vs var vs const
- Stack and Heap

Play Kahoot!

1

What is Variable?

What is Variables?

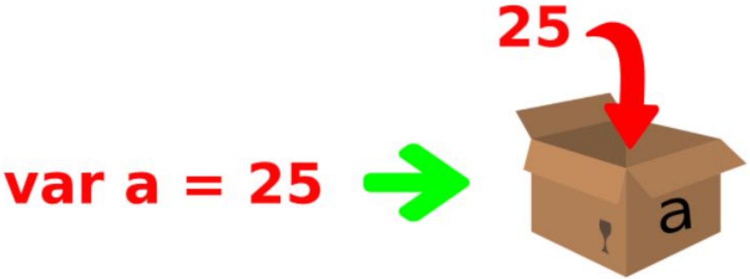


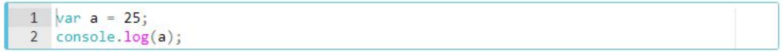
```
/* example.js */

// Global scope
var a = 1;
let b = 10;

// If statement scope
if (true) {
  // Assigning new values
  // a = 2;
  // b = 20;

  // Redefining variables withing current scope
  // var a = 2;  // Redeclare variable in global scope
  // let b = 20; // Creates new variable for current scope

  // If statement scope output
  console.log(a);  1
  console.log(b);  10
}

// Global scope output
console.log(a);  1
console.log(b);  10
```

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

What is Variable?



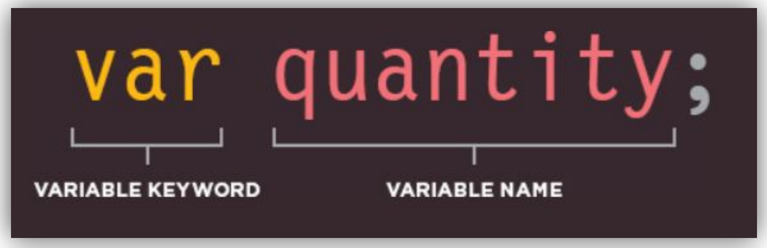


```
var a = 25;
console.log(a);
```

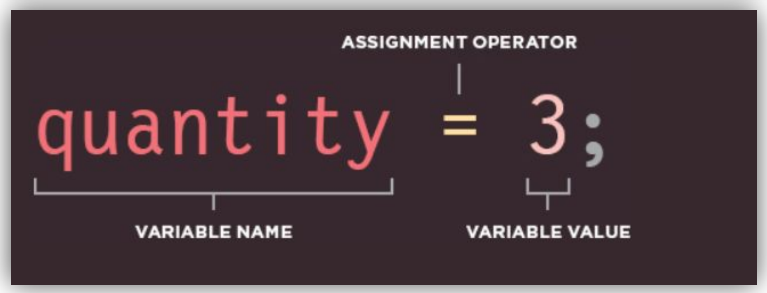
Variables are used to store for data values

What is Variable?

```
var myNumber = 3;
```



```
var quantity;
```



```
quantity = 3;
```

In this example,
*** var is a variable keyword.**
*** myNumber is a variable.**
*** 3 is a value.**

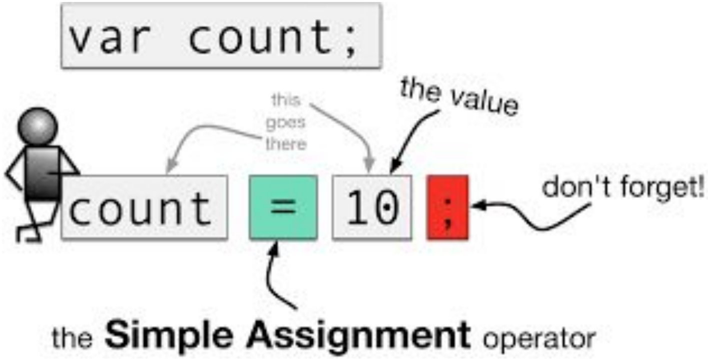
Warning ! :JavaScript is case sensitive. This means that the variables myNumber, mynumber or MYNUMBER are not same variables. All of them are different variables.

2 The Assignment Operator

The Assignment Operator

In JavaScript, the equal sign (=) is used for the assignment operator

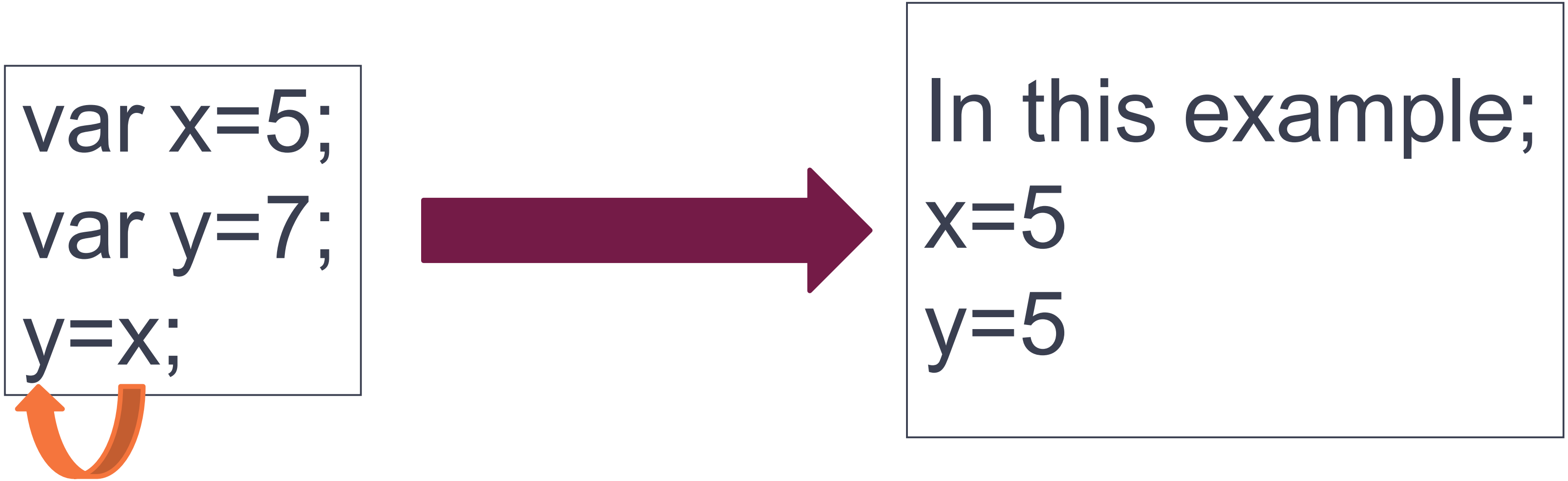
We can store a value in a variable with the assignment operator.



CLARUSWAY
WAY TO REINVENT YOURSELF

The Assignment Operator

Assignment always goes from right to left



```
var x=5;
var y=7;
y=x;
```

In this example;

x=5

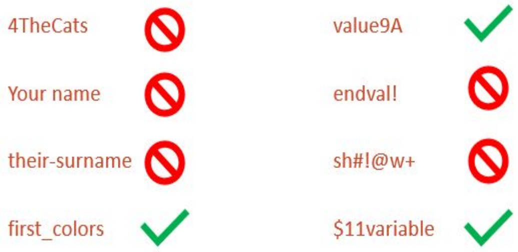
y=5

3

Naming Rules

Naming Rules

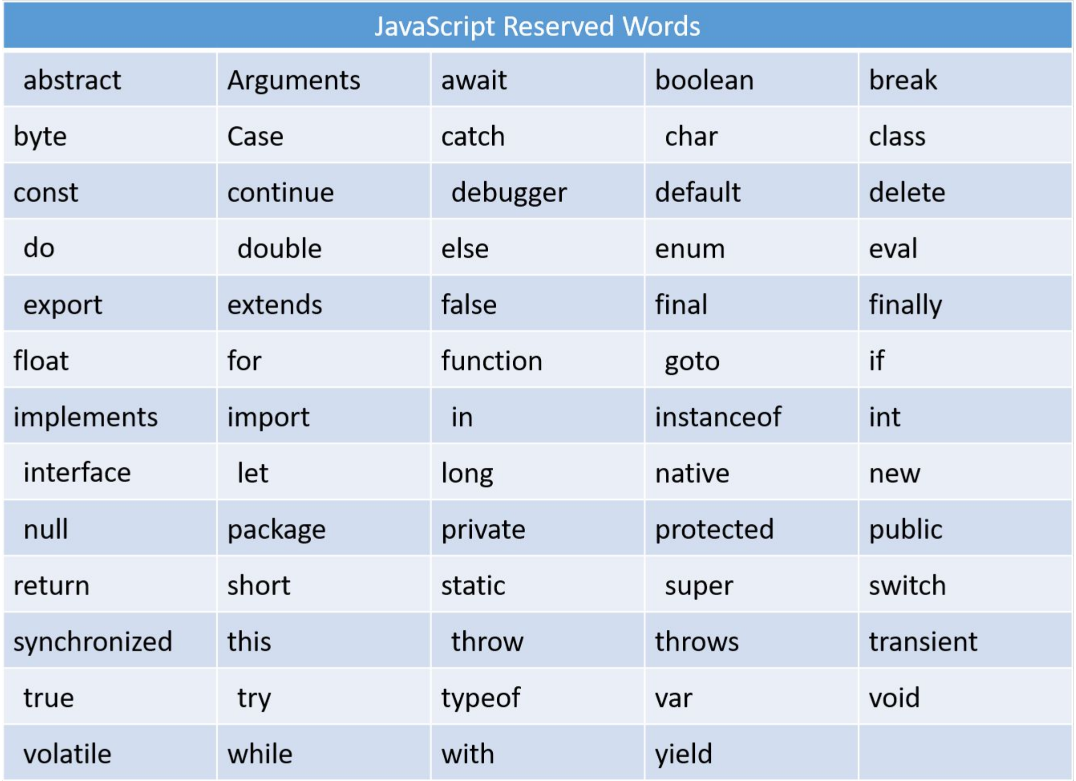
- Names can composed of letters, digits, underscores, and dollar signs
- Numbers are not allowed as the first character
- The first character must be;
  a letter, an underscore (_), a dollar sign ($)
- JavaScript names must not contain spaces, mathematical or logical operators
- Reserved words cannot be used as names



4TheCats	🚫	value9A	✓
Your name	🚫	endval!	🚫
their-surname	🚫	sh#!@w+	🚫
first_colors	✓	$11variable	✓

Naming Rules

Reserved words cannot be used as names



JavaScript Reserved Words				
abstract	Arguments	await	boolean	break
byte	Case	catch	char	class
const	continue	debugger	default	delete
do	double	else	enum	eval
export	extends	false	final	finally
float	for	function	goto	if
implements	import	in	instanceof	int
interface	let	long	native	new
null	package	private	protected	public
return	short	static	super	switch
synchronized	this	throw	throws	transient
true	try	typeof	var	void
volatile	while	with	yield	

4

let vs var vs const

let vs var vs const

Before ES6 we used to define a variable using the **var** keyword.

let and **const** keyboards are added to JavaScript with ES6.

Scope

- Three types of scope:
  - Global scope
  - Function scope
  - Block scope

- Global scope

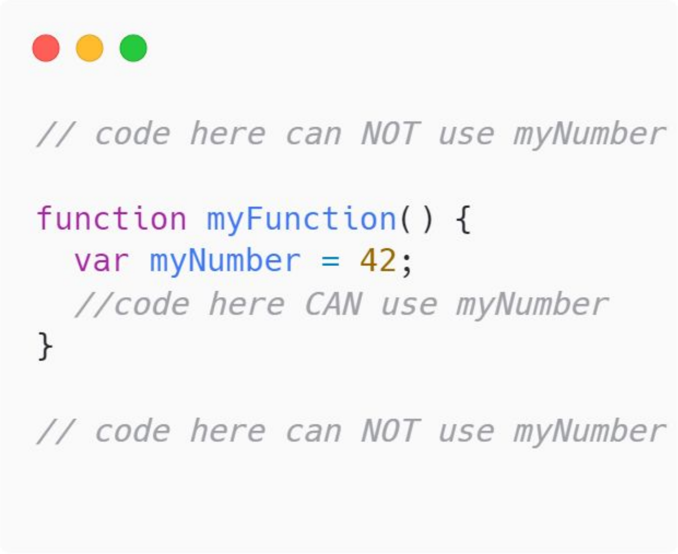
  Outside any function

  Variables can be accessed from ***anywhere in the program***

Scope

Function Scope

- ➤ Variables defined anywhere *inside a function* are *local* to that function
- ➤ Can be used anywhere inside that function
- ➤ Cannot be used outside that function

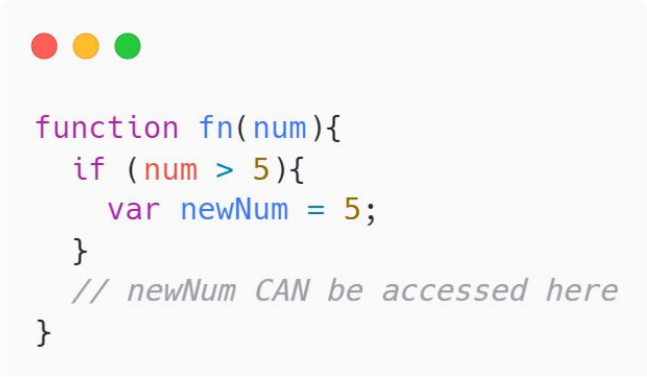


```
// code here can NOT use myNumber

function myFunction() {
  var myNumber = 42;
  //code here CAN use myNumber
}

// code here can NOT use myNumber
```

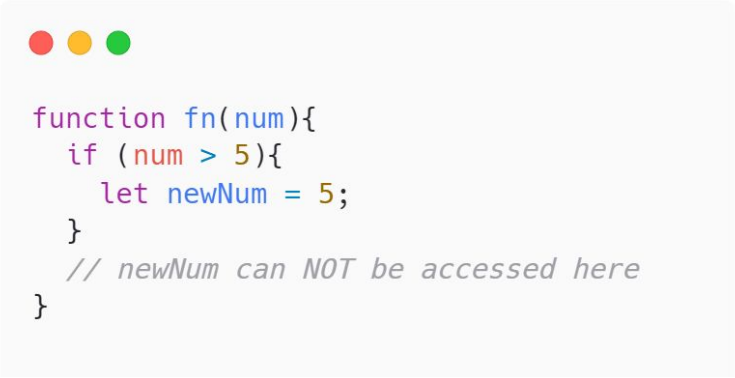
Scope

Block Scope

➤ To limit a variable to its block inside the function, use *let*



```
function fn(num){
  if (num > 5){
    var newNum = 5;
  }
  // newNum CAN be accessed here
}
```



```
function fn(num){
  if (num > 5){
    let newNum = 5;
  }
  // newNum can NOT be accessed here
}
```

let vs var

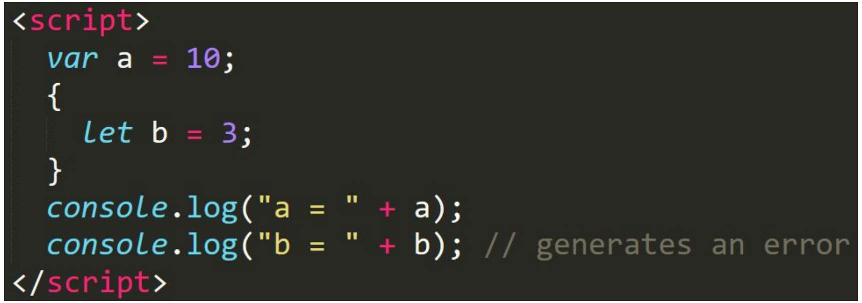
- At global and function scopes, *let* and *var* work *almost* the same
- *var* supports redeclaration, while *let* does not
- Both support re-assignment. Use *const* to disallow it
- *let* is more like regular variables in other languages

  Preferred over *var*

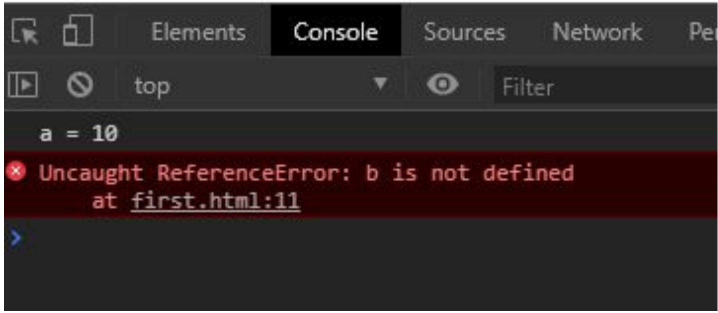
let vs var vs const

let The let statement enables you to declare a variable with block scope.

let Scope is the fundamental concept that defines a variable's visibility in all programming languages.

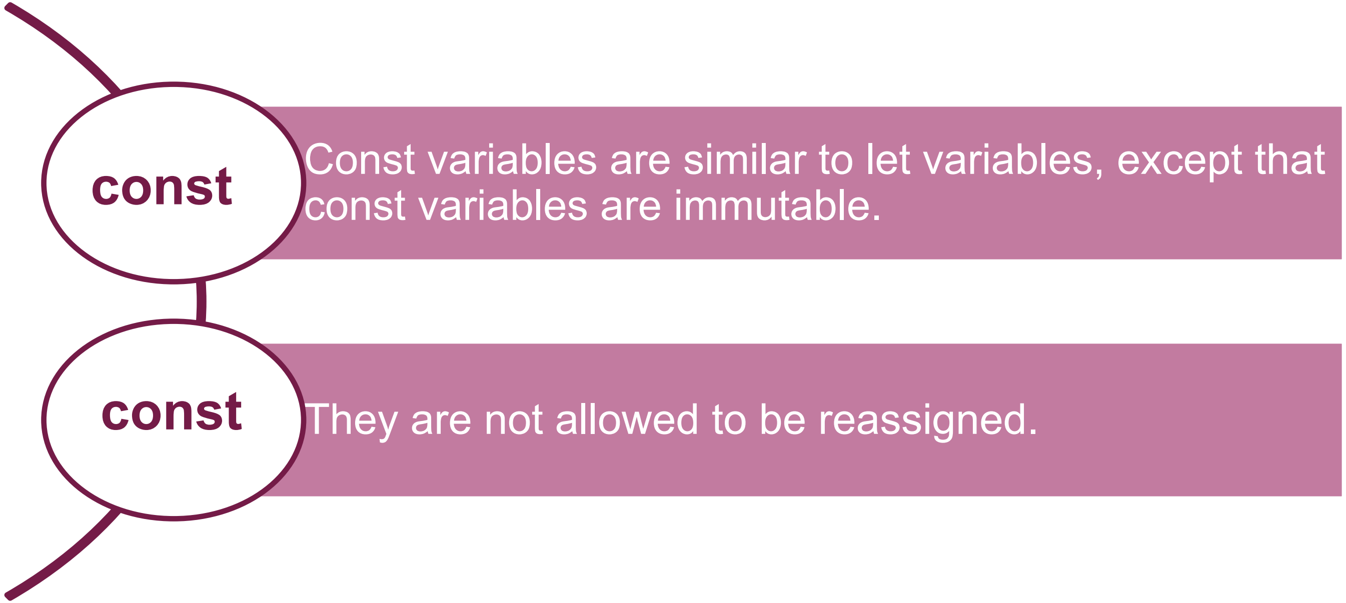


```
<script>
  var a = 10;
  {
    let b = 3;
  }
  console.log("a = " + a);
  console.log("b = " + b); // generates an error
</script>
```



```
a = 10
Uncaught ReferenceError: b is not defined
    at first.html:11
```

let vs var vs const



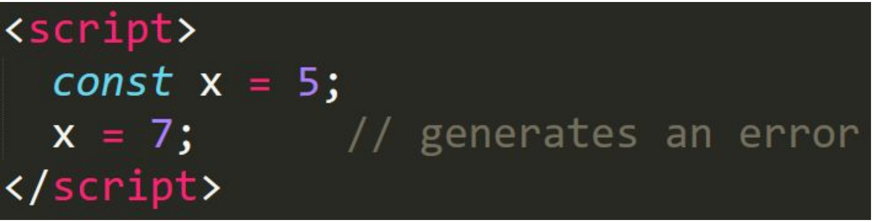
const

Const variables are similar to let variables, except that const variables are immutable.

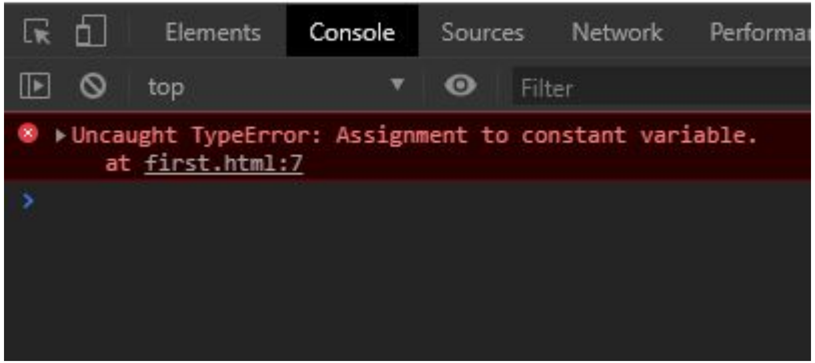
const

They are not allowed to be reassigned.

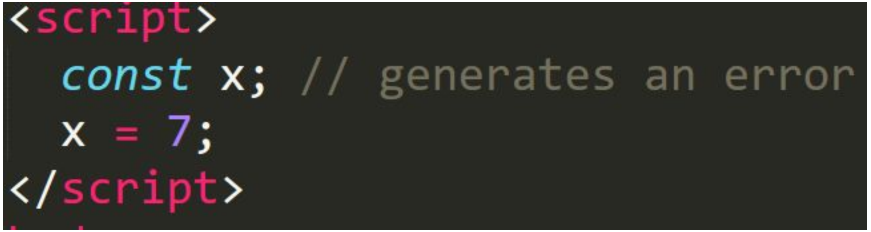
let and const



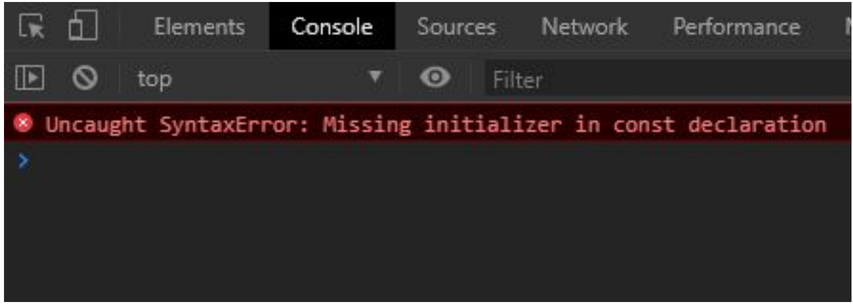
```
<script>
  const x = 5;
  x = 7;        // generates an error
</script>
```



```
Uncaught TypeError: Assignment to constant variable.
    at first.html:7
```



```
<script>
  const x; // generates an error
  x = 7;
</script>
```



```
Uncaught SyntaxError: Missing initializer in const declaration
```

4

Stack and Heap

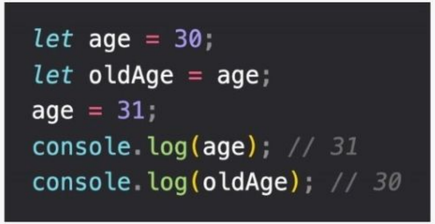
Stack and Heap

Variables in JavaScript (and most other programming languages) are stored in two places: stack and heap.

- A stack is usually a continuous region of memory allocating local context for each executing function.
- Heap is a much larger region storing everything allocated dynamically.
- This separation is useful

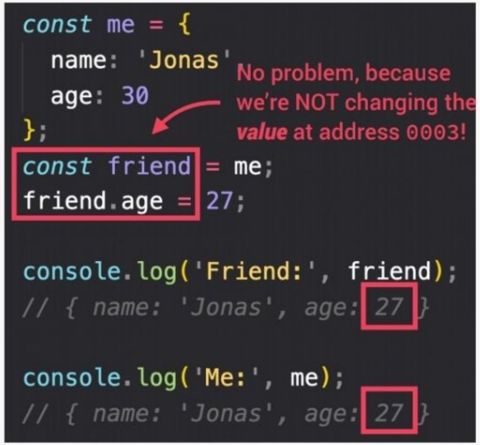
  Stack is more protected and faster, no need for dynamic garbage collection.

Primitive values example:



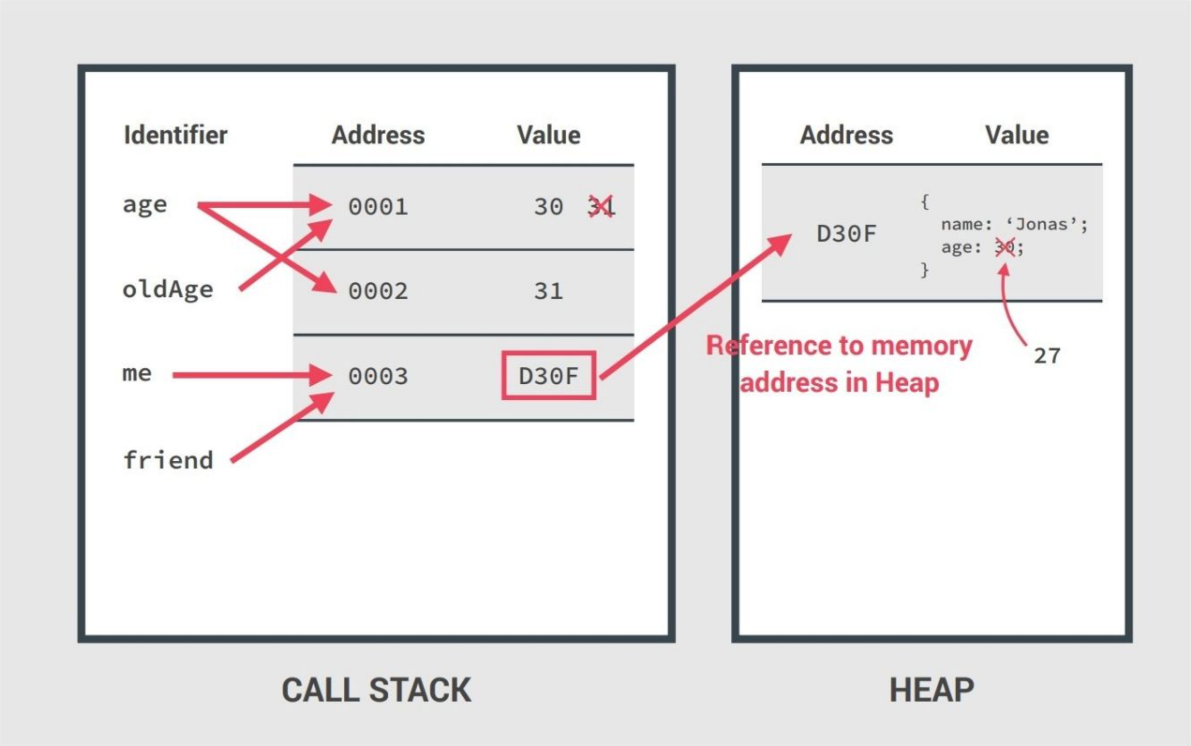
```
let age = 30;
let oldAge = age;
age = 31;
console.log(age); // 31
console.log(oldAge); // 30
```

Reference values example:



```
const me = {
  name: 'Jonas'
  age: 30
};
const friend = me;
friend.age = 27;

console.log('Friend:', friend);
// { name: 'Jonas', age: 27 }

console.log('Me:', me);
// { name: 'Jonas', age: 27 }
```

No problem, because we're NOT changing the *value* at address 0003!



Identifier	Address	Value
age	0001	30 ~~31~~
oldAge	0002	31
me	0003	D30F
friend		

CALL STACK

Address	Value
D30F	{ name: 'Jonas'; age: ~~30~~; }

HEAP

Source: Jonas Schmedtmann, The Complete JavaScript Course 2022: From Zero to Expert!

THANKS!

Any questions?