JavaScript

Declare JavaScript Variables → var variableName;

▼ ***JS has 8 datatypes***

1. `undefined`
2. `null`
3. `boolean`
4. `string`
5. `symbol`
6. `bigint`
7. `number`
8. `object`

In JavaScript all variables and function names are case sensitive. → eg: myVar is not same as myvar or MYVAR

Differences Between the `var` and `let` Keywords → unlike `var`, when you use `let`, a variable with the same name can only be declared once.

```
var myName = Irene;
var myName = irene;
console.log(myName); → outputs “irene”

let myLastName = Joseph;
let myLastName = joseph; //results in an error, variable can only be declared once
```

const Keyword → `const` has all the features that `let` has, with the added bonus that variables declared using `const` are read-only.

```
/*A common practice when naming constants is to use all uppercase letters,
with words separated by an underscore.*/
```

```
const MY_LIFE = "Awesome ;)";
MY_LIFE = "Boring :c"; //display an error, reassigning is not allowed
```

Augmented Operations → +=, -=, *=, /=

In JavaScript, you can *escape* a quote from considering it as an end-of-string quote by placing a *backslash* (`\`) in front of the quote.

```
const sampleStr = "Alan said, \"Peter is learning JavaScript\".";
```

Find the Length of a String → stringName.length

Manipulate Arrays With :

1. **push() → push a value to the end of an array.**
2. **pop() → pop a value off of the end of an array.**
3. **shift() → removes the first element instead of the last.**
4. **unshift() → add elements in front of the array.**

Functions:

```
function functionName() {
  console.log("Hello World");
}

//function with Params
function testFun(param1, param2) {
  console.log(param1, param2);
}

//Return a Value from a Function with Return
function plusThree(num) {
  return num + 3;
}
```

Global Scope and Functions → Variables which are declared without the `let` or `const` keywords are automatically created in

the `global` scope.

Conditional Logic with if

```
function test (myCondition) {
  if (myCondition) {
    return "It was true";
  }
  return "It was false";
}
```

Strict Equality Operator → === | Strict Inequality Operator → !==

```
3 ===  3 --> TRUE
3 === '3'--> FALSE
3 !==  3 --> FALSE
3 !== '3'--> TRUE
```

Multiple Identical Options in Switch Statements

```
let result = "";
switch(val) {
  case 1:
  case 2:
  case 3:
    result = "1, 2, or 3";
    break;
  case 4:
    result = "4 alone";
}
```

JavaScript Objects

```
const cat = {
"name": "Whiskers",
"legs": 4,
"tails": 1,
"enemies": ["Water", "Dogs"]
};
//Accessing Object properties -> dot operator
```

```
cat.name;
cat.legs;

//Accessing Object Properties with Bracket Notation
cat["name"];

//Updating Object Properties
[cat.name](http://cat.name/) =  "Nikki";

const ourDog = {
"name": "Camper",
"legs": 4,
"tails": 1,
"friends": ["everything!"]
};

//Add New Properties to a JavaScript Object
ourDog.bark = "bow-wow";

//Delete Properties from a JavaScript Object
delete ourDog.bark;

//Testing Objects for Properties
ourDog.hasOwnProperty("top"); //returns false

//Nested object
const ourStorage = {
  "desk": {
    "drawer": "stapler"
  },
  "cabinet": {
    "top drawer": {
      "folder1": "a file",
      "folder2": "secrets"
    },
    "bottom drawer": "soda"
  }
};

ourStorage.cabinet["top drawer"].folder2;
ourStorage.desk.drawer;
```

Loops

```
const ourArray = [];
//For loop
for (let i = 0; i < 5; i++) {
  ourArray.push(i);
}
```

```
//While loop
let j = 0;
while(j < 5){
  ourArray.push(j);
  j=j+1;
}

//Do while loop
let i = 0;
do {
  ourArray.push(i);
  i++;
} while (i < 5);
```

Use the Conditional (Ternary) Operator → a ? b : c where `a` is the condition, `b` is the code to run when the condition returns `true` , and `c` is the code to run when the condition returns `false`.

splice function

syntax : array.splice(index, howmany, item1, ..., itemX);

return value of splice function is an array of the removed elements

slice function

syntax: array.slice(startIndex,stopIndex);

Spread operator

syntax : newArray = [...copyArray];

indexOf function

syntax : array.indexOf(element);

returns the index value if element exists in the array, else will return -1

Object.keys()