# JS

- What does const mean?

  it does not mean that the variable is a constant, it just means that you cannot re-assign it

  e.g. given an object: the variable is always an object, but its content (properties) can change

- With const you can update...

  the property value of an object

- What is dot notation?

  To read or update the value of a property, you can use the dot notation.

  e.g. user.age

- What is an object?

  a data type that allows you to group several variables together into one variable that contains keys and values.

- What is a parameter?

  a variable in a function definition

- What are arguments?

  When a function is called, the arguments are the data you pass into the method's parameters.

- What happens when you call a function without providing a value for an expected argument?

  it will default to undefined

- What are two benefits of arrow functions?

- It uses lexical scope
- It can benefit from implicit return

▼ What is arrow function syntax?

Arrow functions always start with the parameters, followed by the arrow ⇒ and then the function body.

▼ What does arrow function syntax look like?

```
const sum = (a, b) => {
    return a + b;
}

// one line example
const tripleIt = value => value * 3;
```

▼ How to write a function by defining a function and giving it a name?

```
function sum(a, b) {
    return a + b;
}
```

▼ How to write a function by defining a variable and assigning it to an anonymous function?

```
const sum = function(a, b) {
    return a + b;
}
```

▼ How to write the Array forEach() with arrow function?

```
grades.forEach(grade => {
    console.log(grade);
});
```

▼ How to write the Array filter() with arrow function?

```
let numbersAboveTen = numbers.filter((number) => {
    return number >= 10;
});
```

▼ What is a callback function?

a "call-after" function, is any executable code that is passed as an argument to other code

▼ How can you do an implicit return that isn't undefined?

1. Your function should be an arrow function.
2. The function body should be only **one line**. This means you have to remove the curly braces.
3. You have to remove the `return` keyword because the function body is one line.

```
const sum = (a, b) => a + b;

sum(1, 3); // 4
```

▼ How to write Array.filter(callback) as a one-liner?

```
const numbers = [9, 5, 14, 3, 11];

const numbersAboveTen = numbers.filter(function(number) {
    return number >= 10;
});
console.log(numbersAboveTen); // [14, 11]

// instead
const numbersAboveTen = numbers.filter(number => number >= 10);
```

▼ How to write Array.find(callback) as a one-liner?

```
const names = ["Sam", "Alex", "Charlie"];

const result = names.find(function(name) {
  return name === "Alex";
});
```

```
console.log(result); // "Alex"

// instead
const result = names.find(name => name === "Alex");
```

▼ How to write Array.map(callback) as a one-liner?

```
const numbers = [4, 2, 5, 8];

const doubled = numbers.map(function(number) {
    return number * 2;
});
console.log(doubled); // [8, 4, 10, 16]

// instead
const doubled = numbers.map(number => number * 2);
```

▼ How to write Array.map(callback) with a conditional?

```
export const getRaisedGrades = grades => {
    // return all the grades raised by 1 (grades should not exceed 20)
    return grades.map(grade => {
        if (grade < 20) {
            grade = grade + 1;
        }
        return grade;
    });
}
```

▼ Reduce sum example

```
export const getSumGrades = grades => {
    // return the sum of all the grades
    return grades.reduce((a, b) => a + b, 0);
}
```

▼ What does String.trim() do?

String .trim() removes all leading and trailing space characters.

▼ What does String.startsWith() do?

String .startsWith(substring) returns true when the substring is found at the beginning of the string, and false otherwise.

▼ What does String.endsWith() do?

String .endsWith(substring) returns true when the substring is found at the end of the string, and false otherwise.

▼ What does String.includes() do?

String .includes(substring) returns true when the substring can be found anywhere in the string, and false otherwise

▼ What does String .split(separator) do?

The .split(separator) method divides the string into an array by splitting it with the separator you provide.

```
let apps = "Calculator,Phone,Contacts";
let appsArray = apps.split(",");
console.log(appsArray); // ["Calculator", "Phone", "Contacts"]
```

▼ What does Array.join(glue) do?

It joins an array into a string on a glue character e.g. a space

▼ What does String.replace(search, replace) do?

The .replace(search, replace) method returns a new string where the **first** occurrence of the search parameter you provide is replaced by the replace parameter.

```
const message = "You are welcome.";
message.replace(" ", "_"); // "You_are welcome";
console.log(message); // "You are welcome" (original string is not changed)
```

▼ What does String.replaceAll(search, replace) do?

It replaces **all** occurrences of the search parameter.

```
const message = "You are welcome.";
message.replaceAll(" ", "_"); // "You_are_welcome";
console.log(message); // "You are welcome" (original string is not changed)
```

▼ What is a slug?

In computer science, a *slug* is a string used to identify a certain item.

Oftentimes, this *slug* is used in the URL for Search Engine Optimization and better user experience.

Let's say you've got a product called: `"Easy assembly dining table"`. We cannot use this name in the URL bar because it contains spaces (it won't look nice, for example `https://example.com/item/Easy assembly dining table`).

This is why we use a slug that looks like this:

`easy-assembly-dining-table` so the URL becomes: `https://example.com/item/easy-assembly-dining-table`.

▼ How to generate an HTML string from an array?

```
const html = `<ul>
    ${users.map(user => `<li>${user.name}</li>`).join("")}
    </ul>`;
console.log(html); // <ul> <li>Sam Doe</li><li>Alex Blue</li> </ul>
```

What's very important here is the `.join("")`. If you forget this, you will get the following HTML:

```
<ul><li>Sam Doe</li>,<li>Alex Blue</li></ul>
```

That's because the array returned from .map() will automatically be converted to a string by the browser.

▼ What does Array.every(callback) do?

The Array .every(callback) method returns true when **every item in the array satisfies the condition** provided in the callback.

```
const numbers = [15, 10, 20];

const allAbove10 = numbers.every(number => number >= 10); // true
const allAbove15 = numbers.every(number => number >= 15); // false
```

▼ What does Array.some(callback) do?

The Array .some(callback) method returns true when **at least one item in the array satisfies the condition** provided in the callback.

```
const numbers = [15, 10, 20];

const someOver18 = numbers.some(number => number >= 18); // true
const someUnder10 = numbers.some(number => number < 10); // false
```

▼ How can you empty an array by changing its length?

```
const items = ["Pen", "Paper"];
items.length = 0;

console.log(items); // []
```

▼ What does Array.splice(start[, deleteCount]) do?

The .splice(start[, deleteCount]) method removes items from the array starting from the start index that you specify.

If no deleteCount is provided, it will remove all the remaining items as of the start index.

the deleteCount parameter is optional

When you specify a deleteCount, then it will remove as many items as you provided in the deleteCount from the start index.

▼ What does the reduce() method do?

The goal of the reduce() method is to calculate a single value from an array.

In other terms, you reduce an array into a single value.

e.g. summing an array: We can reduce the array [5, 10, 5] to the number 20.

▼ What is the reducer?

a callback which allows you to configure the logic of how the array will be reduced into a single number

▼ What arguments does reduce() take?

The reducer (callback that reduces array to a single number) and the initial value

▼ What does reduce() look like?

```
const grades = [10, 15, 5];

const sum = grades.reduce((total, current) => {
    return total + current;
}, 0);

// The total is always referring to the last computed value by the reduce function.
// total aka accumulator
```

▼ How to use reduce() to multiply?

```
const numbers = [5, 2, 10];

const result = numbers.reduce((total, current) => {
    return total * current;
}, 1);
```

▼ What is NaN?

Not a Number

▼ What does array destructuring syntax look like?

```
const dimensions = [20, 5]

// create variables
const [width, height] = dimensions;

// log them
console.log(width); //20
console.log(height); //5
```

▼ What does array destructuring syntax look like in React hooks?

```
const [counter, setCounter] = useState(0);
```

▼ How can you concatenate/merge several arrays into a new array using the ... spread syntax?

```
const lat = [5.234];
const lng = [1.412];
const point = [...lat, ...lng];
console.log(point); // [5.234, 1.412];

const items = ["Tissues", "Oranges"];
const otherItems = [...items, "Tomatoes"];
console.log(otherItems); // ["Tissues", "Oranges", "Tomatoes"]
```

▼ Is JavaScript a dynamic language?

Even statically typed languages can have a dynamic or variant data type that can contain different data types.

JavaScript is called a dynamic language because it doesn't just have a few dynamic aspects, pretty much everything is dynamic.

▼ What does the Object global variable contain?

methods that are relevant to objects

e.g. Object.keys(user);

▼ What does Object.keys(obj) method do?

```
const user = {
    id: 1,
    name: "Sam Green",
    age: 20
};

const keys = Object.keys(user);
console.log(keys); // ["id", "name", "age"]
```

▼ What does Object.values(obj) method do?

```
Object.values(someObject).forEach(val => console.log(val));
// this will console.log() every value in the given object.
```

Object.values() returns an array whose elements are the enumerable property values found on the object.

▼ Why do you get this error?: Uncaught TypeError: Cannot read property 'toUpperCase' of undefined

This is one of the **most common errors** that you will see in JavaScript. **TypeScript** does a great job at preventing this kind of errors, though it comes with its own overhead.

It's important to be able to read this error message and understand that the issue is not `.toUpperCase()` but instead, the expression that came before it `person.age`.That's because you end up calling `undefined.toUpperCase()` which does not exist.

```
person.age.toUpperCase();
```

▼ What has happened when you see [object Object]?

If you see [object Object], it means you tried to use an object in a context that expects a string

```
const person = {
    id: 1,
    firstName: "Sam"
};

console.log(`Hello ${person}`); // "Hello [object Object]"
```

▼ What does Object.entries(obj) do?

returns an array of arrays representing every key/value pair.

```
const user = {
    id: 1,
    name: "Sam Green",
    age: 20
};

const entries = Object.entries(user);

[
```

```
    ["id", 1],
    ["name", "Sam Green"],
    ["age", 20]
]
```

▼ When you access a property that does not exist on an object, you will get...?

undefined

▼ What does Object shorthand look like?

```
const age = 18;
const person = {
    name: "John",
    age
}


const isAdmin = false;
const darkMode = true;

const settings = {
    isAdmin,
    darkMode
};

console.log(settings); //{isAdmin: false, darkMode: true}
```

Because the property name is the same as the name of the variable used as its value, then you can drop the : age so you're left only with age.

▼ Awesome console.log and object shorthand debugging tip

```
const getProduct = (a, b) => {
    console.log({a, b});

    let product = a * b;
    console.log({product});

    return product;
}

getProduct(2, 3);
/*
{a: 2, b: 3}
{product: 6}
*/
```

▼ How to do object destructuring?

```
const config = {
    id: 1,
    isAdmin: false,
    theme: {
        dark: false,
        accessibility: true
    }
};

// instead of this
const id = config.id;
const isAdmin = config.isAdmin;
const theme = config.theme;

// do this
const {id, isAdmin, theme} = config;

// if only need some variables
const {isAdmin, theme} = config;
```

▼ How to destructure with a default value?

```
const user = {
    id: 1,
    name: "Sam"
};

const {name, isAdmin = false} = user;
console.log(isAdmin); // false
```

▼ How to concatenate objects with ... spread operator?

```
const firstPerson = {
    name: "Sam",
    age: 18
}

const secondPerson = {
    age: 25,
    type: "admin"
}
```

```
const mergedObjects = {...firstPerson, ...secondPerson};
console.log(mergedObjects); // {name: "Sam", age: 25, type: "admin"}
```

the order of the objects matters here. the second age persisted

▼
▼
▼
▼
▼
▼
▼
▼
▼