# **Introduction to functional programming**

## Introduction to functional programming

In programming, there are two commonly used paradigms. Functional programming, sometimes abbreviated as FP, and object-oriented programming, sometimes abbreviated as OOP. You can think of these paradigms as different approaches to writing code.

*In functional programming, the data and functions that work on that data are combined inside objects.*

* *False*

## Function calling and recursion

Recursion is when a function calls itself. Have to be careful or can lead to an infinite loop.

Text

Description automatically generated

*Do you recall these comparison operators == and === ?*

* *Yes*

*Consider the following code:*

*function myDay() {*

*console.log('Morning');*

*console.log('Afternoon');*

*console.log('Evening');*

*myDay();*

*}*

*Which one of the following best describes what will happen if you run this code?*

* *The function will run in an infinite loop.*

## Introduction to scope

Scope is all about code accessibility. It determines which parts of the code are accessible and which parts are inaccessible. For example, what variables can a function access within code?

The different scope types: global and local.

A picture containing chart

Description automatically generated

The code that exists outside of a function is referred to as global scope, and all the code inside of a function is known as local scope or function scope. If a variable is defined within a function, then you can say it's scoped to that function. This is also known as local scope.

*You decide to create a variable within the local scope while scripting a new JavaScript file. Is this variable accessible to code at the global scope level?*

* *No*

## Scoping with var, let and const

The ES6 version of JavaScript introduced a new variety of scope known as the block scope. Block scope states that a variable declared in a block of code is only accessible inside that block. All the other code outside of the code block cannot access it. Block scope is built when you declare variables using let or const. In other words when you build variables with let or const, they become immediately scoped to the code block they were created in.

*You are performing an update to some code and you need to create a variable named quantity, which calculates the total number of items.*

*You need this variable to be accessible only inside a for loop.*

*You declare the variable using the let keyword and it is within a for loop that performs the calculation of the total number of items.*

*Does this variable declaration address your needs?*

* *Yes*

Use let if the value might change in the future, and use const if the value will never change.

## Comparing var, let and const

A variable declared with the var keyword can be accessed before initialization as long as the variable is eventually initialized somewhere in our code.

Let's move on to the let keyword. A key difference is that you cannot access a let variable before declaring it. While you cannot redeclare a let variable, you can reassign it.

the const keyword which is the strictest. A const variable must be initialized. As you can see, you can access a const variable before initialization or else you'll get a reference here.

*Which one of the following statements is true when declaring variables using either var, let or const?*

* *Variables declared with const must be assigned during declaration.*

# **Introduction to object-oriented programming**

## Introduction to object-oriented programming

The object oriented programming paradigm, often referred to as OOP. OOP revolves around the idea of organizing our programs using objects to group related data and functionality. This is in contrast to the functional programming approach, where the data used in the app needs to be kept separate from functions that operate on that data.

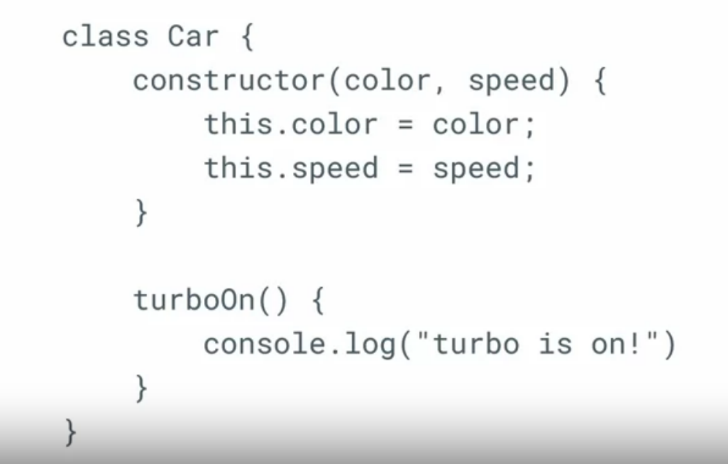
With the OOP approach, you create an object and store all data related to that object including variables, functions and output statements.

*You are coding in OOP style. Why would you want to use the "this" keyword?*

* *To refer to the object itself without specifying the object's name.*

## Classes

They are essentially a blueprint that you can repeatedly use to build new objects of a certain kind, as many times as you like.



Text

Description automatically generated with low confidence

NOTE: You don’t use the function keyword here, only the method name is needed.

*You are working with classes in JavaScript. Which of the following instructions should you adhere to? Check all that apply.*

* *Create an instance of the class using the keyword new and that class' name, followed by opening and closing parentheses, and optional arguments, based on how the class itself is defined.*
* *Add a constructor function to accept your parameters.*
* *Build your classes using the “class” keyword.*

## The Principles of OOP

The four fundamental OOP principles are inheritance, encapsulation, abstraction and polymorphism.

|  |  |
| --- | --- |
| **Inheritance** | * Inheritance is one of the foundations of object-oriented programming. * In essence, it's a very simple concept. It works like this: * There is a base class of a "thing". * There is one or more sub-classes of "things" that inherit the properties of the base class (sometimes also referred to as the "super-class") * There might be some other sub-sub-classes of "things" that inherit from those classes in point 2. * Note that each sub-class inherits from its super-class. In turn, a sub-class might also be a super-class, if there are classes inheriting from that sub-class.   **To setup the inheritance relation between classes in JavaScript, I can use the extends keyword, as in class B extends A.** |
| **Encapsulation** | * In the simplest terms, encapsulation has to do with making a code implementation "hidden" from other users, in the sense that they don't have to know how my code works in order to "consume" the code. |
| **Abstraction** | * Abstraction is all about writing code in a way that will make it more generalized. * The concepts of encapsulation and abstraction are often misunderstood because their differences can feel blurry. * It helps to think of it in the following terms: * An abstraction is about extracting the concept of what you're trying to do, rather than dealing with a specific manifestation of that concept. * Encapsulation is about you not having access to, or not being concerned with, how some implementation works internally. * While both the encapsulation and abstraction are important concepts in OOP, it requires more experience with programming in general to really delve into this topic. |
| **Polymorphism** | polymorphism is useful because it allows developers to build objects that can have the exact same functionality, namely, functions with the exact same name, which behave exactly the same. However, at the same time, you can override some parts of the shared functionality or even the complete functionality, in some other parts of the OOP structure. |

## Constructors

Constructor functions, commonly referred to as just "constructors", are special functions that allow us to build instances of these built-in native objects. All the constructors are capitalized.

For example, to create a new instance of the Date object, I can run: new Date(). What I get back is the current datetime.

However, not all the built-in objects come with a constructor function. An example of such an object type is the built-in Math object.

Running new Math() throws an Uncaught TypeError, informing us that Math is not a constructor.

*Instead of using Array, Function, and RegExp constructors, you should use their array literal, function literal, and pattern literal varieties: [], () {}, and /()/. (DUE TO BEING MORE PERFORMANT).*

## Inheritance

You may also be familiar with the concept of a prototype which is often referred to as an original model from which other forms are developed. In JavaScript, the prototype is an object that can hold properties to be shared by multiple other objects. And this is the basis of how inheritance works in JavaScript.

Text

Description automatically generated

*True or false? In JavaScript, you can use a prototype object to hold properties that can be shared with various other objects.*

* *True*

NOTE: It is better to use inheritance with classes and not prototypes.

# **Advanced JavaScript Features**

## De-structuring arrays and objects

Text

Description automatically generated with medium confidence

Note that I can only destruction something that already exists on an object. Using faulty spelling, including lower case won't work and will return an undefined value.

*True or False. In JavaScript, it's possible to extract the properties from objects into distinct variables using destructuring.*

* *True*

## For of loops and objects

Note: methods: Object.keys(), Object.values(), and Object.entries().

Graphical user interface, text, application, email

Description automatically generated

NOTE: Silly enough there was no “keys” defined in the provided example.

## For-of vs For-in

Text

Description automatically generated

In fact, as the common state, for-in loops are unreliable in this scenario because they iterate over not only the specified object, but also its prototype.

A for-of loop only iterates over the object's own properties and does not count the prototype at all.

*When working with objects, the for-of loop and for-in loop can be used to iterate over the object's properties. Which of the following statements are correct? Choose all that apply.*

* *The for-of loop will iterate over the object's own properties only when using the Object.keys() method to return an array to loop over.*
* *The for-of loop will not iterate over the object and its prototype properties.*

## Working with template literals

Template literals use backticks and allow for formula expressions and also multi line strings. To use variables or an expression you must use the ${} syntax.

*In what ways can template literals be used to write JavaScript code more efficiently? Check all that apply:*

* *You can combine strings with less code.*
* *You can interpolate variables.*
* *You can create multi-line strings.*

## Data Structures

|  |  |
| --- | --- |
| **Object** | Now you may recall that an object is unaltered, **noniterable** collection of key value pairs and you use objects when you need to store and later access a value under a key. |
| **Map** | Map is similar to array as **its iterable** however it consists of key-value pairs.  *It's important not to confuse a map with an object. With maps any value can be used as a key. With objects, keys can only be strings or symbols.* |
| **Set** | the last data structure I want you to know about is a set. This is another collection where each item in the collection must be unique. |

*Which one of these data structures consists of iterable key-value pairs?*

* *Maps*

## Spread operator

It is the shortest and simplest method to copy the properties of an object onto a newly created object. Think of the spread operator as a magical multi-purpose tool that can spread out array items and join objects together.

Note that the spread operator is characterized by three dots.

Text

Description automatically generated



Text

Description automatically generated with medium confidence

The advantage of this approach is that you don't have to list each individual member of the array that you want to pass to your function.

*The spread operator allows you to pass all array elements into a function without having to type them all individually. Is this true or false?*

* *True*

### Rest operator

Text

Description automatically generated with medium confidence A picture containing chart

Description automatically generated

First, second, and third will hold the first 3 items, but the secondVisit array will hold the remainder 4 items.

The rest operator, therefore, gives us what is left over of the source array, as a separate sub array.

*The rest operator allows you to take items from an array and use them to create a separate sub-array. True or false?*

* *True*

*Graphical user interface, text, application, email

Description automatically generated*

Rest parameter can also be used in functions (this looks more spread to me personally).

# **JavaScript in the Browser**

## JavaScript modules

CommonJS is designed to specify how modules should work outside of the browser environment. It is mostly used on server-side JavaScript namely node.js a downside of CommonJS is that browsers don't understand its syntax. That is certain keywords that CommonJS relies on, such as require and module.exports don't work as expected in browsers.

Which of the following statements about modules in JavaScript are true? Choose all that apply.

* To use an ES6 module in a browser, you need to set the script type attribute to “module”.
* Modules were added to ECMAScript ES6 specification.
* Modules allow for code to be reused and more easily replaced.

## JavaScript DOM manipulation

*True or false? Editing the local DOM of a webpage does not affect the document stored on the webserver.*

* *True*

## JavaScript selectors

Graphical user interface, text, application

Description automatically generatedText

Description automatically generated

Text

Description automatically generated Text

Description automatically generated

*Which of the following are JavaScript DOM selectors? Check all that apply .*

* *querySelector()*
* *getElementsByClassName()*
* *getElementById()*
* *querySelectorAll()*

## Event handling

In JavaScript, the button click and the like icon tap are examples of user triggered events. Events are happening all the time. You can use JavaScript code to listen for these events. You can even choose the parts of the page on which you are listening for events.

In JavaScript, the function that handles captured events is known as the event handler.

One of the easiest ways to listen for an event is to use the add event listener method. You can do that on a given HTML element.

*The next step is to run the addEventListener method on the target element. I type target.add EventListener. Then inside the parenthesis, I pass it two arguments.* ***The first is the event type*** *click as a string value, and the* ***second is my handle click function****.*

Graphical user interface, text, application, email

Description automatically generated

*You are using JavaScript code on your website to listen out for events. Which of the following are examples of events that your code can listen out for? Check all that apply.*

* *Icon tap*
* *Button click*

## JavaScript Object Notation – JSON

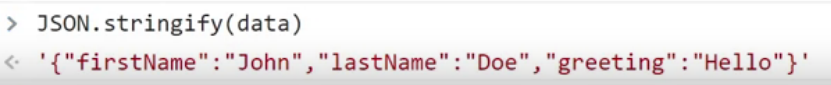
So to work with JSON, it is common to convert it back to a JavaScript object to work with its properties and methods.

To do this you need to use the global built in JSON object and its parse method.

Graphical user interface, text, application, email

Description automatically generated

You can also go the other way and convert a regular object to a JSON string. Using the string of my method of the JSON object.



*You can convert a JSON file to a JavaScript object so that you can work with that object's properties.*

* *True*

## ssss