Introduction to Programming

Relevel
by Unacademy

Topics Covered

- Introduction
- Different types of Programming Paradigm
- Difference between Scripting & Compiled Language
- What is JavaScript?
- Tips & Tricks
- Practice Questions

What is Programming:

- Programming is the set of instructions that we give to our computers to carry out a certain task.
- We know that computers don’t understand English, they only understand binary i.e. 0 & 1, but humans cannot speak binary, we speak languages that contain many words, vowels, phrases and letters.
- But we need the computer to understand our instructions so that we can have the computer perform a task, since the computer doesn’t understand English and since we cannot speak binary, we had to find a middle ground.
- To make computers understand our instructions, we take the help of compilers(programs responsible for translating human language to machine code).

Why is Programming important?

- Programming allows us to make use of computers to solve complex mathematical equations or render a 4K video to be uploaded on YouTube or using Google Docs to write this document that you are reading currently.

- Everything and anything that can be done on a computer is done using programming, anything you do on a computer, be it a game you play(Counter Strike/ Project IGI), video you watch on YouTube or an application you run like MS Word, it is done using programming, each of these items contains thousands of lines of instructions telling the computer what to do and thus we get to run them smoothly on our computers.

- Not everything can be done manually by human beings, certain tasks require automation, and there are computations that the human mind cannot always accurately state.

- Thus to reduce manual work, improve efficiency and automate labor, programming was introduced as a means to solve the problem.

- Programming can be done using a variety of computer programming languages, such as JavaScript, Python, and C++.

Different types of Programming Paradigms?

There are basically four types of programming:

- Procedural
- Object Oriented
- Functional
- Declarative

Procedural Programming:

- Procedural programming is a programming paradigm that evolved from structured programming and is based on the concept of invoking procedures. Procedures, often known as routines, subroutines, or functions, are essentially a set of instructions to be followed. Any procedure in a programme can be invoked at any time during execution, either by other procedures or by the programme itself.
- As the name implies, a procedural language relies on specified and well-organized procedures, functions, or sub-routines in a program's architecture to express all the steps that the computer must follow to attain a particular state or result.

- A programme is divided into variables, functions, statements, and conditional operators using the procedural language. To complete a task, procedures or functions are applied to the data and variables. These procedures can be called/invoked from any point in the programme hierarchy, as well as from other procedures. One or more processes can be found in procedural language software.

- Procedural language is one of the most common types of programming languages in use, with notable languages such as C/C++, Java, ColdFusion and PASCAL.

Procedural Programming:

When it comes to the following situations, procedural programming is often the best option:

- There's a complicated operation with inter - dependencies and a need for clear visibility of multiple program states ('SQL loading,"SQL loaded,'Network online,'No audio hardware,' and so on). This is usually the best way to start and stop an application.
- The program is one-of-a-kind, with only a few elements in common.
- The program is set in stone and is unlikely to change significantly over time.
- Over time, it is envisaged that no or just a few features would be added to the project.

Object Oriented Programming:

- Object-oriented programming is a programming paradigm based on the concept of objects, which contain data as well as code to manipulate it. Object-oriented programming imitates many of the characteristics of real-world objects.
- Objects are made up of data and methods that describe their state and behaviour. These two entities are encapsulated in each object.
- The user cannot see how object methods are implemented inside. This allows objects to alter their abstract state using a simple external API.
- Classes have instances, which are objects. Classes serve as blueprints for constructing objects. An object's type is also its class.
- Other classes' state and behaviour can be inherited by classes. Objects of the subclass can be cast into objects of the parent class based on this concept.
- Polymorphism is a result of this type of casting. An object of a class can be implicitly cast to an object of the class's predecessors by the programme.

Object Oriented Programming:

OOP is frequently the better choice when:

- We have a team of programmers who don't need to know everything about every component.
- There's a lot of code out there that can be shared and reused.
- The project is expected to alter frequently and be expanded over time.
- Different resources, such as datasets or hardware, can assist different portions.

When we have numerous developers that need to share implementation details, we probably don't want to use it. For example, if we were creating a keyboard driver, we wouldn't want to break it up into pieces that would hide implementation details from a developer working on the driver.

Functional Programming:

- Functional programming languages were created specifically for symbolic computing and list processing. Mathematical functions are the foundation of functional programming. Lisp, Python, Erlang, Haskell, Clojure, and other functional programming languages are some of the most popular.

- The following are the most notable features of functional programming:

- Functional programming languages are based on the concept of mathematical functions that execute computations using conditional expressions and recursion.

- Higher-order functions and slow evaluation are supported by functional programming.

- Flow is not supported by functional programming languages. Loop statements and conditional statements such as If-Else and Switch Statements are examples of controls. They use functions and functional calls directly.

Functional Programming:

- When should functional programming be used:
    - Domain specific languages (DSLs): If the issue domain is so complex that traditional industry languages like Java and C/++ fail to address it efficiently, a DSL might be the answer.
    - We wouldn't use a DSL to construct our entire system, but we could use it, like the 5ESS switch, to code a critical function in a way that is easier to understand and maintain, ensuring its quality.
    - DSL creation is a breeze with functional languages.

Declarative Programming:

- Declarative programming is defined as writing code that defines what you want to achieve rather than how you want to do it. It's up to the compiler to find out how to do it.
- Declarative programming refers to a set of problems for which the language implementation is responsible for finding a solution. Some parallel processing systems benefit from declarative programming since it simplifies the programming.
- SQL and Prolog are two declarative programming languages.

Declarative Programming:

Declarative programming is useful in the following situations:

- When we want to keep the code factorable and clean.
- When readability of the code is to be given utmost importance.
- When database queries need to be placed.

Difference between Scripting vs Compiled Language:

- Scripting languages are, in essence, programming languages. The distinction between the two is that scripting languages do not require compilation and instead are interpreted. A *C program*, for example, must be compiled before running, whereas a scripting language like *JavaScript* or *PHP* does not require compilation.

- Compiler programs are generally faster than interpreter programs since they are transformed from native machine code first. Furthermore, compilers examine and interpret code just once, reporting any faults that the code may contain collectively, whereas the interpreter reads and analyses code statements each time it encounters them, stopping immediately if there is a mistake. In fact, the gap between the two is blurring due to modern hardware's better calculation capabilities and innovative coding approaches.
- Some scripting languages traditionally used without an explicit compilation step are JavaScript, PHP, Python, VBScript.

- Some programming languages traditionally used with an explicit compilation step are C, C++.

Difference between Scripting vs Compiled Language:

Scripting Languages in Practice:

1. To programmatically automate some operations
2. Information extraction from a data set
3. Requires less code than standard programming languages.

Programming Language Applications:

1. They are usually run within a parent application, similar to scripts.
2. More consistent with mathematical models when integrating code
3. Programming languages such as JAVA can be compiled and run on any platform.

What is Javascript?:

- *JavaScript* is a programming language that enables developers to create interactive web pages. It Is commonly used as part of web pages.
- It is lightweight and can be interpreted in an object-oriented manner.
- JavaScript was initially called LiveScript, but Netscape changed it to JavaScript, which is because

Of the excitement, it has generated.

- JavaScript is a programming language that provides an interface to the server-side. It allows developers to create interactive and non-standard web applications.
- Client-side JavaScript is a type of programming language that is commonly used for web apps.
- A web page is not a static HTML document but can include interactive software that can interact with the user.
- JavaScript is a programming language that enables developers to create interactive and non-static web pages. It serves as the third layer of web.
- The three layers are built on top of one another.
- HTML is the markup language that enables us to structure and give meaning to the web content that we create. It provides various markup constructs for our web pages.

What is Javascript?

HTML
CSS
JAVASCRIPT

- CSS is a language that we use to set the styling rules for our HTML content.
- JavaScript is an incredibly powerful scripting language that enables anyone to create almost anything they can imagine.

Why is it important ?

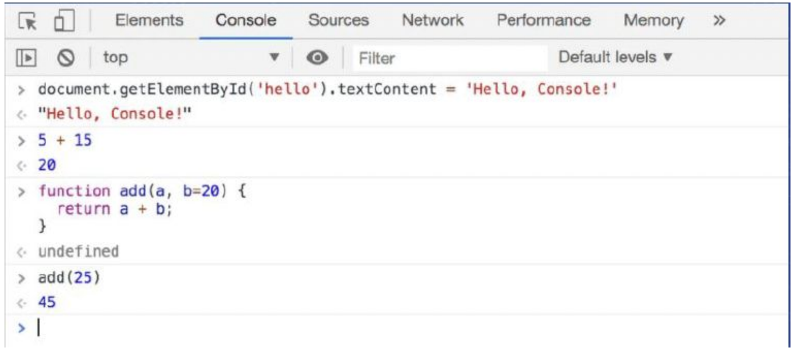
JavaScript makes the web dynamic, some of the dynamic behavior are:

- Autocomplete
- Loading new content or data onto the page without reloading the page
- Rollover effects and dropdown menus
- Animating page elements such as fading, resizing or relocating
- Playing audio and video
- Validating input from Web forms
- Repairing browser compatibility issues

Where is it used?

- It allows developers to create dynamic web pages and is a must have language for website development, being on client side it reduces the demand of website servers. Some of the popular javascript frontend frameworks and libraries are: React, Angular, VueJS, Svelte, etc.
- JavaScript allows users to add dynamic elements to a website, such as changing text and content or graphics that resize themselves.
- The following are examples of front-end modifications that you may apply to a website using JavaScript :
  - ➔ When a link is pressed, data is shown.
  - ➔ Create a drop-down menu that appears when a button is clicked.
  - ➔ Demonstrate a website's tabs.
  - ➔ On a website show animations.
  - ➔ On a website’s page, you may play video or audio.

How to run JavaScript?



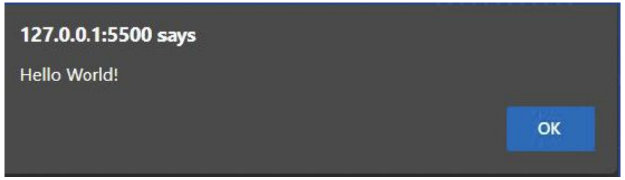
```
> document.getElementById('hello').textContent = 'Hello, Console!'
<· "Hello, Console!"
> 5 + 15
<· 20
> function add(a, b=20) {
    return a + b;
  }
<· undefined
> add(25)
<· 45
> |
```

- JavaScript is a scripting language so it cannot run on it's own.
- The code executes in a browser environment like internet explorer, google chrome, firefox etc.

First JavaScript program: “Hello World”

```
<html>
<head>
  <title>Hello World</title>
  <script type=”text/javascript”>
  alert(“Hello World”):
   </script>
</head>
<body>
</body>
</html>
```

Output:



127.0.0.1:5500 says

Hello World!

- JavaScript code can be included within script tags contained within an HTML document.
- Always use the type attribute to mention which scripting language you are using (type attribute is not necessary in HTML5)

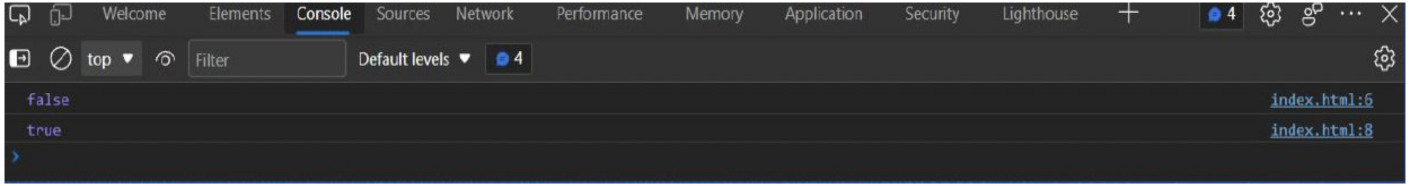
Comparison operators:

- You can ask questions and compare any two values together.
- You will get a boolean value in output: “TRUE” or “FALSE”
- We will use console.log() :
- This shows the result in the web console.
- To open web console:
- type Ctrl + SHift + I on windows or linux
- Type Cmd + Shift + K on mac
- You can write click on the browser => inspect => console

Example :

```
<Script>
    // are these two values equal
       console.Log(2 == 3) ;
   // is the first value less than second
       console.Log(2 <== 3) ;
</Script>
```

Output :



```
false                                                    index.html:6
true                                                     index.html:8
```

JavaScript Data Types:

- Data types are used to store specific types of data and can be used to manipulate it accordingly.
- The types are dynamic.
- Primitive Data Types: can only store a single value.
  - ➢ String
  - ➢ Number
  - ➢ Boolean

- Composite Data Types: can hold complex values and collection of values.
  - ➢ Objects
  - ➢ Array
  - ➢ Function

- Special data Types
  - ➢ Undefined
  - ➢ Null

Example :

```
var x;        // x is undefined
   X = 12;        // x is a Number
   X = “John”;  // x is a String
```

String :

- Use to store text value
- Enclosed by quotes
- Can use single quotes : ‘ ’
- Can use double quotes : “ ”
- Can use backticks: ` `

Example :

```
var name = "John";
var lastName = "Doe";
var fullName = `My name is ${name} ${lastName}`;
console.log(fullName);
// output:
//My name is John Doe
```

Number

- Use to store numbers (positive and negative numbers)
- Infinity values represents the mathematical Infinity, the greatest number
- Infinity value represent the smallest
- Nan : Not A Number: invalis numeric value

Example :

```
var a = 12; // number, integer
   var b = 12.1; // number ,floating point number
   var c = 16/0; // Infinity
   var d = 16/-0; // Infinity
   var e = “text”/2 // NaN
```

Thank You.