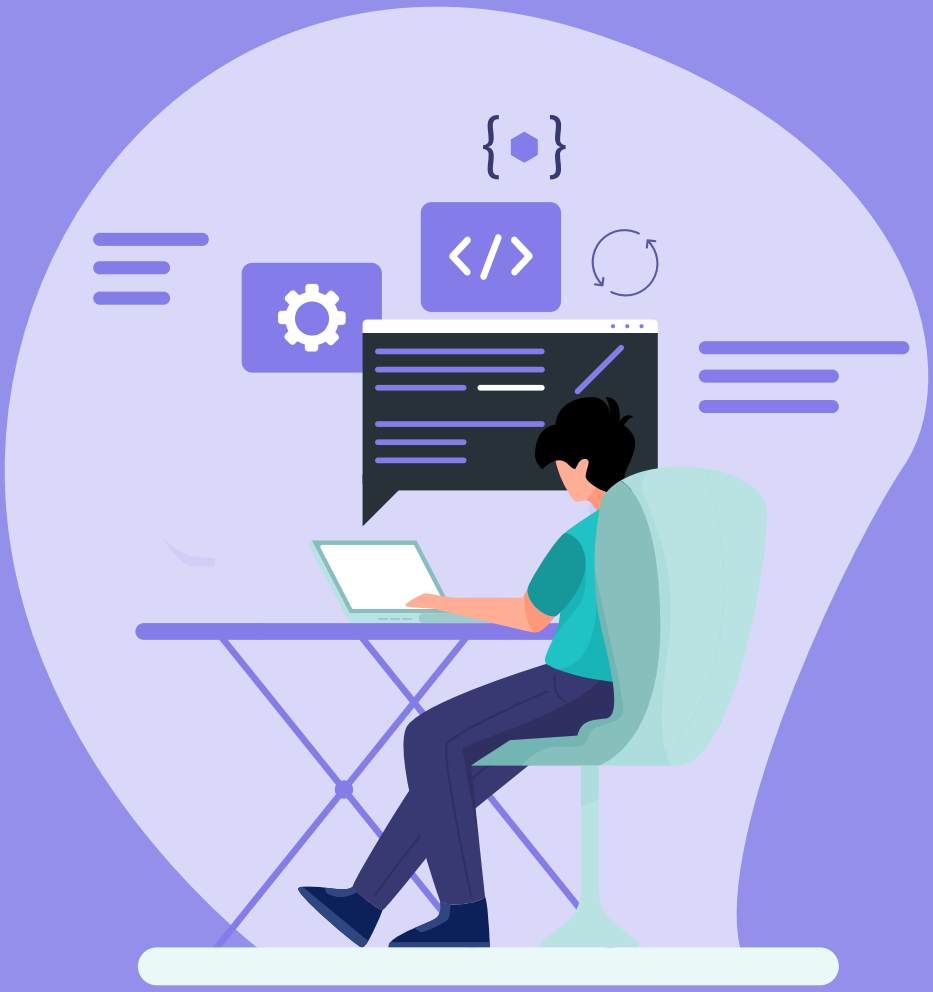
Asynchronous Programming in JS - 1

Pre-read



Relevel
by Unacademy

Things I need to know before this session

The session requires you to have basic knowledge of variables in Javascript. Make sure you have revised all the previous concepts.

What will be taught in the coming session?

Basic Terminology :

1. Declaring: Creating a new variable is called a variable declaration. The keywords used to declare variables are let, var, const. When a variable is declared, it will store undefined until a value is assigned to it if no initial value is given.

Eg : let a=10;
var b;
const c=20;

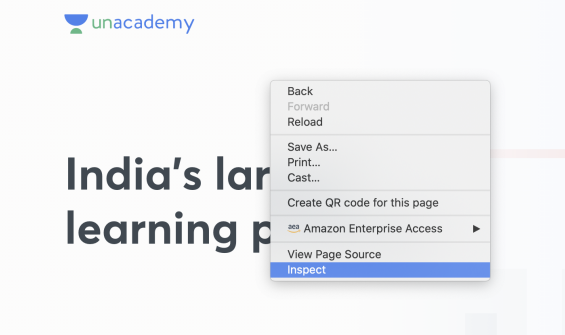
2. Instantiation : Creating an instance or object of a class is called instantiation

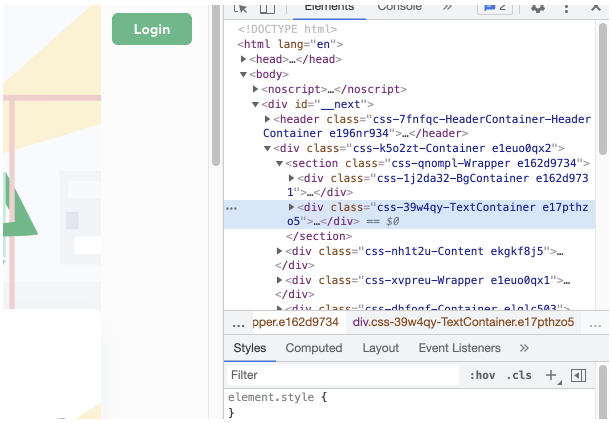
Eg :
```
class Sum {
    let a =10;
    let b=20;
}
let  obj1 = new Sum(); // Instantiating the class by creating class object
```

Introduction to HTML

HTML stands for HyperText Markup Language. You can consider HTML to be the essential ingredient used in building up a web page. Every webpage is associated with an HTML document that specifies the content and structure of the page.
Fun fact: You can very easily find the HTML document associated with any webpage!

- Right-click anywhere on the page.
- Click on Inspect (or, Inspect Element)
- You can find the HTML doc as shown in the image below
- Also show the script tags in the inspector to show the integration of js with html

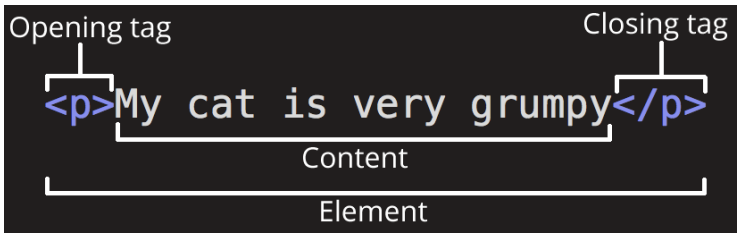




HTML basics

HTML (Hypertext Markup Language) is the code that is used to structure a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables. As the title suggests, this article will give you a basic understanding of HTML and its functions.

HTML is a markup language that defines the structure of your content. HTML consists of a series of elements, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicise words, can make the font bigger or smaller, and so on.



The main parts of our element are as follows:

- The opening tag: This consists of the name of the element (in this case, p), wrapped in opening and closing angle brackets. This states where the element begins or starts to take effect — in this case where the paragraph begins.
- The closing tag: This is the same as the opening tag, except that it includes a forward slash before the element name. This states where the element ends — in this case where the paragraph ends. Failing to add a closing tag is one of the standard beginner errors and can lead to strange results.
- The content: This is the content of the element, which in this case, is just text.
- The element: The opening tag, the closing tag, and the content together comprise the element.

Elements can also have attributes that look like the following:

Attributes contain extra information about the element that you don't want to appear in the actual content. Here, class is the attribute name and editor-note is the attribute value. The class attribute allows you to give the element a non-unique identifier that can be used to target it (and any other elements with the same class value) with style information and other things.
An attribute should always have the following:

- A space between it and the element name (or the previous attribute, if the element already has one or more attributes).
- The attribute name followed by an equal sign.
- The attribute value wrapped by opening and closing quotation marks.

Anatomy of an HTML document

That wraps up the basics of individual HTML elements, but they aren't handy on their own. Now we'll look at how individual elements are combined to form an entire HTML page. Let's revisit the code we put into our index.html example (which we first met in the Dealing with files article):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="My test image">
  </body>
</html>
```

Here, we have the following:

- **<!DOCTYPE html> —** doctype. It is a required preamble. In the mists of time, when HTML was young (around 1991/92), doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML, which could mean automatic error checking and other useful things. However these days, they don't do much and are basically just needed to make sure your document behaves correctly. That's all you need to know for now.
- **<html></html> —** the <html> element. This element wraps all the content on the entire page and is sometimes known as the root element.

- **<head></head> —** the <head> element. This element acts as a container for all the stuff you want to include on the HTML page that isn't the content you are showing to your page's viewers. This includes things like keywords and a page description that you want to appear in search results, CSS to style our content, character set declarations, and more.
- **<meta charset="utf-8"> —** This element sets the character set your document should use to UTF-8 which includes most characters from the vast majority of written languages. Essentially, it can now handle any textual content you might put on it. There is no reason not to set this and it can help avoid some problems later on.
- **<title></title> —** the <title> element. This sets the title of your page, which is the title that appears in the browser tab the page is loaded in. It is also used to describe the page when you bookmark/favorite it.
- **<body></body> —** the <body> element. This contains all the content that you want to show to web users when they visit your page, whether that's text, images, videos, games, playable audio tracks, or whatever else.

Async programming:

Asynchronous in general is not occurring at the same time.
Asynchronous in programming context is basically execution of code/function without waiting for the other code/functions to complete their execution.

CallBacks:

Callbacks are basically functions that are passed as arguments to other functions and then invoked in the outer function to perform its operation.
Also callbacks take time to execute and are not executed immediately; they are passed at first and are invoked later.

Call Stack: (20 mins)

Call Stack is the stack data structure maintained by the Javascript engine. It operates in LIFO manner (Last in First Out). It has the purpose of tracking the current function being executed. **Call Stack** is also known as **Function execution stack.**

Timing Events in JS:

In Javascript there are two **Timing Events**

- **setTimeout()**
- **setInterval()**

These timing events are not part of the JS engine but actually they are part of Browser and are available to us in the Global context associated with the Window object of the browser and in NodeJS they are associated with the Global object.

We will be learning all about the concepts mentioned above in this class.
At the end we will also create a **Countdown Timer App** by using these concepts.