Day-23

What is JavaScript

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on web.

Features of JavaScript

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Application of JavaScript

JavaScript is used to create interactive websites. It is mainly used for:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

JavaScript Example

```
<script>
document.write("Hello JavaScript by JavaScript");
</script>
```

The **script** tag specifies that we are using JavaScript.

The **text/javascript** is the content type that provides information to the browser about the data.

3 Places to put JavaScript code

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)

1) JavaScript Example : code between the body tag

In the above example, we have displayed the dynamic content using JavaScript. Let's see the simple example of JavaScript that displays alert dialog box.

```
<script type="text/javascript">
 alert("Hello Javatpoint");
</script>
```

2) JavaScript Example : code between the head tag

Let's see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

In this example, we are creating a function msg(). To create function in JavaScript, you need to write function with function_name as given below.

To call function, you need to work on event. Here we are using onclick event to call msg() function.

```
<html>
<head>
<script type="text/javascript">
function msg(){
 alert("Hello Javatpoint");
}
</script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</body>
```

External JavaScript file

We can create external JavaScript file and embed it in many html page.

It provides **code re usability** because single JavaScript file can be used in several html pages.

An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

Let's create an external JavaScript file that prints Hello Javatpoint in a alert dialog box.

message.js

```
function msg(){
 alert("Hello Javatpoint");
}
```

Let's include the JavaScript file into html page. It calls the JavaScript function on button click.

index.html

```html
<html>
<head>
<script type="text/javascript" src="message.js"></script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</body>
</html>
```

Advantages of External JavaScript

There will be following benefits if a user creates an external javascript:

1. It helps in the reusability of code in more than one HTML file.
2. It allows easy code readability.
3. It is time-efficient as web browsers cache the external js files, which further reduces the page loading time.
4. It enables both web designers and coders to work with html and js files parallelly and separately, i.e., without facing any code conflictions.
5. The length of the code reduces as only we need to specify the location of the js file.

Disadvantages of External JavaScript

There are the following disadvantages of external files:

1. The stealer may download the coder's code using the url of the js file.
2. If two js files are dependent on one another, then a failure in one file may affect the execution of the other dependent file.
3. The web browser needs to make an additional http request to get the js code.
4. A tiny to a large change in the js code may cause unexpected results in all its dependent files.

5. We need to check each file that depends on the commonly created external javascript file.
6. If it is a few lines of code, then better to implement the internal javascript code.

Data Types-

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript non-primitive data types

The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

We will have great discussion on each data type later.

Browser Object Model

The **Browser Object Model** (BOM) is used to interact with the browser.

The default object of browser is window means you can call all the functions of window by specifying window or directly. For example:

```
1. window.alert("hello javatpoint");
```

is same as:

```
1. alert("hello javatpoint");
```

You can use a lot of properties (other objects) defined underneath the window object like document, history, screen, navigator, location, innerHeight, innerWidth,

Window Object

The **window object** represents a window in browser. An object of window is created automatically by the browser.

Window is the object of browser, **it is not the object of javascript**. The javascript objects are string, array, date etc.

Methods of window object

The important methods of window object are as follows:

Method	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.

close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

Example of alert() in javascript

It displays alert dialog box. It has message and ok button.

```
<script type="text/javascript">
function msg(){
 alert("Hello Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

Output of the above example

Example of confirm() in javascript

It displays the confirm dialog box. It has message with ok and cancel buttons.

```
<script type="text/javascript">
function msg(){
var v= confirm("Are u sure?");
if(v==true){
alert("ok");
}
else{
alert("cancel");
}

}
</script>

<input type="button" value="delete record" onclick="msg()"/>
```

Output of the above example

Example of prompt() in javascript

It displays prompt dialog box for input. It has message and textfield.

```
<script type="text/javascript">
function msg(){
var v= prompt("Who are you?");
alert("I am "+v);

}
</script>

<input type="button" value="click" onclick="msg()"/>
```

Output of the above example

Example of open() in javascript

It displays the content in a new window.

```
<script type="text/javascript">
function msg(){
open("http://www.facebook.com");
}
</script>
<input type="button" value="javatpoint" onclick="msg()"/>
```

Output of the above example

Example of setTimeout() in javascript

It performs its task after the given milliseconds.

```
<script type="text/javascript">
function msg(){
setTimeout(
function(){
alert("Welcome to Javatpoint after 2 seconds")
},2000);

}
</script>

<input type="button" value="click" onclick="msg()"/>
```

Output of the above example

JavaScript History Object

The **JavaScript history object** represents an array of URLs visited by the user. By using this object, you can load previous, forward or any particular page.

The history object is the window property, so it can be accessed by

```
window.history
```

OR

```
history
```

Methods of JavaScript history object

There are only 3 methods of history object.

No.	Method	Description
1	forward()	loads the next page.
2	back()	loads the previous page.
3	go()	loads the given page number.

Example of history object

Let’s see the different usage of history object.

```
history.back();//for previous page
history.forward();//for next page
```

3. history.go(2);//for next 2nd page
4. history.go(-2);//for previous 2nd page

JavaScript Navigator Object

The **JavaScript navigator object** is used for browser detection. It can be used to get browser information such as appName, appCodeName, userAgent etc.

The navigator object is the window property, so it can be accessed by:

1. window.navigator

Or,

1. navigator

Property of JavaScript navigator object

There are many properties of navigator object that returns information of the browser.

No.	Property	Description
1	appName	returns the name
2	appVersion	returns the version
3	appCodeName	returns the code name
4	cookieEnabled	returns true if cookie is enabled otherwise false
5	userAgent	returns the user agent
6	Language	returns the language. It is supported in Netscape and Firefox only.
7	userLanguage	returns the user language. It is supported in IE only.
8	Plugins	returns the plugins. It is supported in Netscape and Firefox only.
9	systemLanguage	returns the system language. It is supported in IE only.

10	mimeTypes[]	returns the array of mime type. It is supported in Netscape and Firefox only.
11	Platform	returns the platform e.g. Win32.
12	Online	returns true if browser is online otherwise false.

Methods of JavaScript navigator object

The methods of navigator object are given below.

No.	Method	Description
1	javaEnabled()	checks if java is enabled.
2	taintEnabled()	checks if taint is enabled. It is deprecated since JavaScript 1.2.

Example of navigator object

Let’s see the different usage of history object.

```
<script>
document.writeln("<br/>navigator.appCodeName: "+navigator.appCodeName);
document.writeln("<br/>navigator.appName: "+navigator.appName);
document.writeln("<br/>navigator.appVersion: "+navigator.appVersion);
document.writeln("<br/>navigator.cookieEnabled: "+navigator.cookieEnabled);
document.writeln("<br/>navigator.language: "+navigator.language);
document.writeln("<br/>navigator.userAgent: "+navigator.userAgent);
document.writeln("<br/>navigator.platform: "+navigator.platform);
document.writeln("<br/>navigator.onLine: "+navigator.onLine);
</script>
```

JavaScript Screen Object

The **JavaScript screen object** holds information of browser screen. It can be used to display screen width, height, colorDepth, pixelDepth etc.

The navigator object is the window property, so it can be accessed by:

```
window.screen
```

Or,

```
screen
```

Property of JavaScript Screen Object

There are many properties of screen object that returns information of the browser.

No.	Property	Description
1	width	returns the width of the screen
2	height	returns the height of the screen
3	availWidth	returns the available width
4	availHeight	returns the available height
5	colorDepth	returns the color depth
6	pixelDepth	returns the pixel depth.

Example of JavaScript Screen Object

Let’s see the different usage of screen object.

```
<script>
document.writeln("<br/>screen.width: "+screen.width);
document.writeln("<br/>screen.height: "+screen.height);
document.writeln("<br/>screen.availWidth: "+screen.availWidth);
document.writeln("<br/>screen.availHeight: "+screen.availHeight);
document.writeln("<br/>screen.colorDepth: "+screen.colorDepth);
document.writeln("<br/>screen.pixelDepth: "+screen.pixelDepth);
</script>
```

JavaScript Array

JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

1) JavaScript array literal

The syntax of creating array using array literal is given below:

```
var arrayname=[value1,value2.....valueN];
```

As you can see, values are contained inside [ ] and separated by , (comma).

Let's see the simple example of creating and using array in JavaScript.

```
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

7. The .length property returns the length of an array.

Output of the above example

```
Sonoo
Vimal
Ratan
```

2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

```
var arrayname=new Array();
```

Here, **new keyword** is used to create instance of array.

Let's see the example of creating array directly.

```
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Output of the above example

```
Arun
Varun
John
```

3) JavaScript array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

The example of creating object by array constructor is given below.

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Output of the above example

```
Jai
Vijay
Smith
```

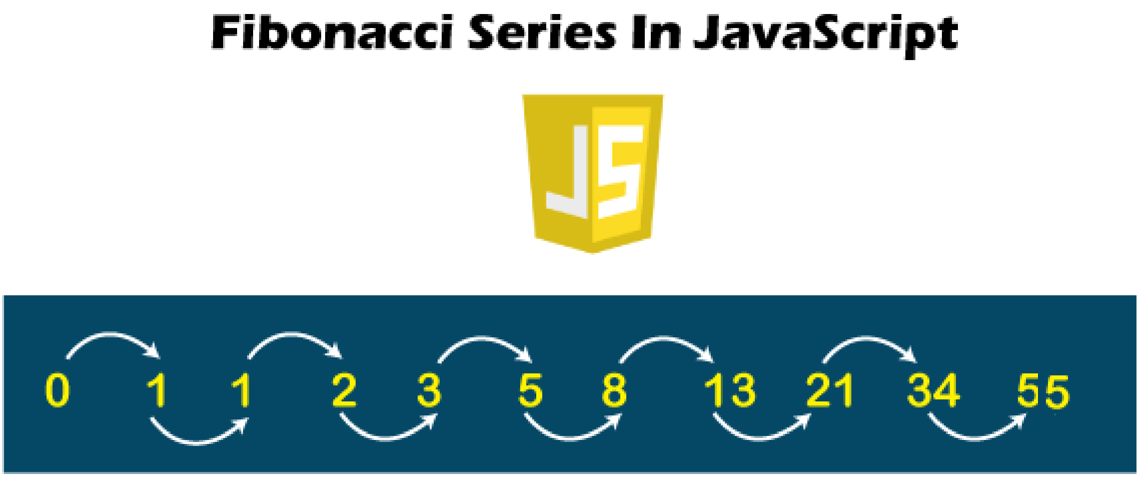
JavaScript Array Methods

Let's see the list of JavaScript array methods with their description.

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
copywithin()	It copies the part of the given array with its own elements and returns the modified array.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
filter()	It returns the new array containing the elements that pass the provided function conditions.
find()	It returns the value of the first element in the given array that satisfies the specified condition.
findIndex()	It returns the index value of the first element in the given array that satisfies the specified condition.
forEach()	It invokes the provided function once for each element of an array.

includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
isArray()	It tests if the passed value ia an array.
join()	It joins the elements of an array as a string.
keys()	It creates an iterator object that contains only the keys of the array, then loops through these keys.
lastIndexOf()	It searches the specified element in the given array and returns the index of the last match.
map()	It calls the specified function for every array element and returns the new array
of()	It creates a new array from a variable number of arguments, holding any type of argument.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
reduce(function, initial)	It executes a provided function for each value from left to right and reduces the array to a single value.
reduceRight()	It executes a provided function for each value from right to left and reduces the array to a single value.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.

toLocaleString()	It returns a string containing all the elements of a specified array.
toString()	It converts the elements of a specified array into string form, without affecting the original array.
unshift()	It adds one or more elements in the beginning of the given array.
values()	It creates a new iterator object carrying values for each index in the array.



Fibonacci series is a series that generates subsequent series of numbers by the addition of the two previous numbers. The first two terms of the Fibonacci series are **zero** and **one**, respectively. And the next terms are the addition of the two previous terms.

Representation of the Fibonacci series

Fn = (Fn -1) + (Fn - 2)

Fn represents the addition of the previous terms **(Fn - 1)** and (**Fn - 2)**. Here Fn-1 is the first terms, and Fn-2 is the second terms of the Fibonacci series.

Example:

- First terms of the series is: 0

- Second term of the series is: 1
- Third terms of the series is: (0 + 1) = 1
- Fourth terms of the series is: (second + third) term = (1 + 1) = 2
- Fifth term of the series is: (Third + fourth) = 1 + 2 = 3

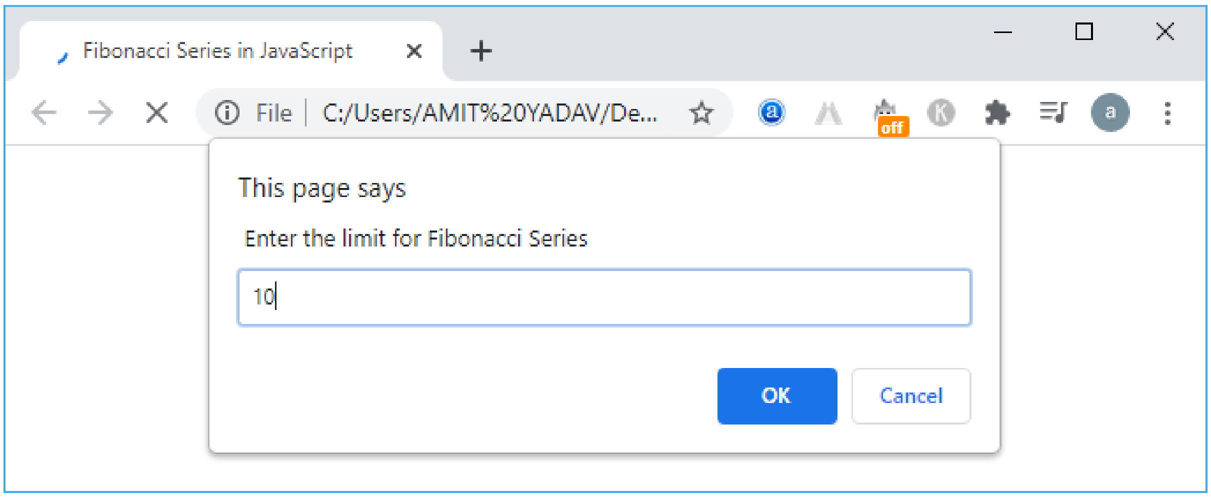
Here is generated series: 0, 1, 1, 2, 3, ... Similarly, we can find the series of the next terms.

Get the Fibonacci series up to n terms

Let's consider an example to get the Fibonacci series up to a desired numbers in JavaScript using for loop.

Program1.html

```
<html>
<head>
<title> Fibonacci Series in JavaScript </title>
</head>
<body>
<script>
// declaration of the variables
var n1 = 0,  n2 = 1, next_num, i;
var num = parseInt (prompt (" Enter the limit for Fibonacci Series "));
document.write( "Fibonacci Series: ");
for ( i = 1; i <= num; i++)
{  document.write (" <br> " +  n1); // print the n1
   next_num = n1 + n2; // sum of n1 and n2 into the next_num

   n1 = n2; // assign the n2 value into n2
   n2 = next_num; // assign the next_num into n2
}

</script>
</body>
</html>
```

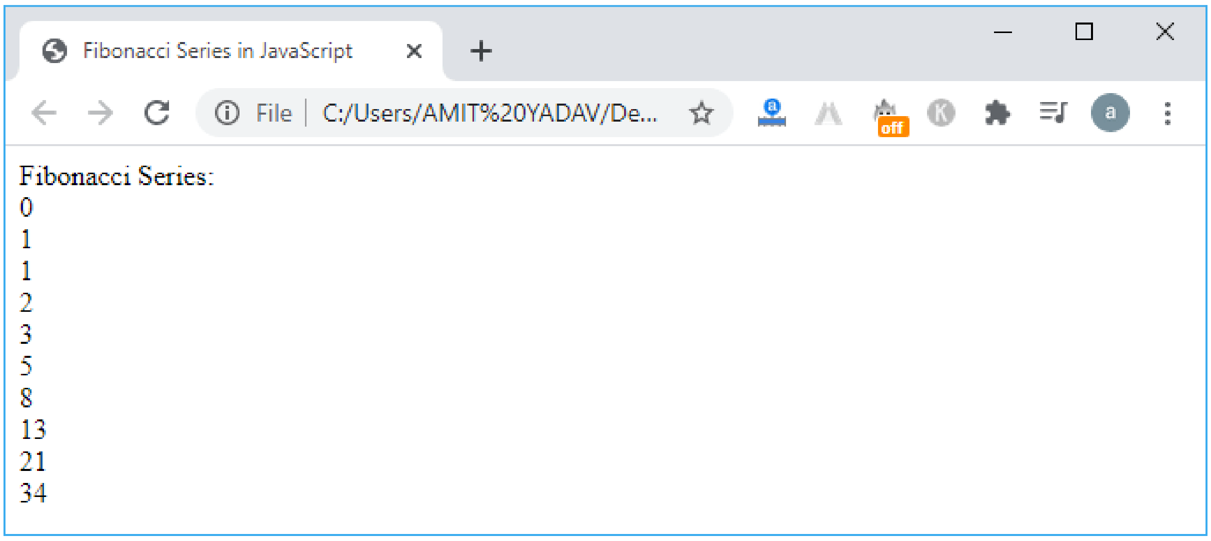

Fibonacci Series in JavaScript
File | C:/Users/AMIT%20YADAV/De...
OK
Cancel

This page says

Enter the limit for Fibonacci Series

10

After that, it displays the Fibonacci series that starts with 0 and 1. And the next term is the addition of its previous two terms, as shown below.


Fibonacci Series in JavaScript
File | C:/Users/AMIT%20YADAV/De...

```
Fibonacci Series:
0
1
1
2
3
5
8
13
21
34
```

Ugly number Java(this is JAVA code Not Java script) Understand logic and find program for java script for same.

The **Ugly** number is another special positive number in Java. If a number has only 2, 3, or 5 prime factors and by convention 1 is also included, the number is called Ugly number.

Let's take some examples of Ugly numbers.

1. The simple method using for loop
2. Use Dynamic Programming

Let's understand both the methods one by one implement the logic of each one.

The simple method using for loop

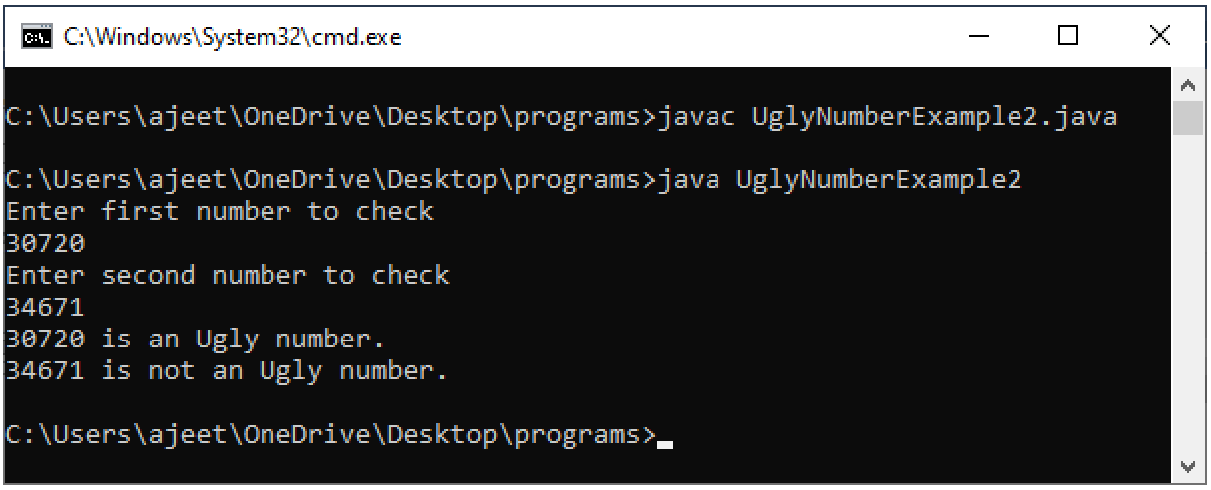
UglyNumberExample1.java

```java
//import required classes and packages
import java.io.*;
import java.util.*;

//create UglyNumberExample1 class to get Nth Ugly number
class UglyNumberExample1 {

    // create a method divideByGreatestDivisible() method for dividing the number by the given greatest divisible power
    static int divideByGreatestDivisible(int number, int gdp)
    {
        while (number % gdp == 0)
            number = number / gdp;
        return number;
    }

    // create checkUglyNumber() method that returns true when the number is an Ugly
    static boolean checkUglyNumber(int number)
    {
        number = divideByGreatestDivisible(number, 2);
        number = divideByGreatestDivisible(number, 3);
        number = divideByGreatestDivisible(number, 5);

        return (number == 1) ? true : false;
    }

    //create findNthUglyNumber() method for getting the nth Ugly number
    static int findNthUglyNumber(int NthNumber)
    {
        int number = 1;

        // initialize counter
        int counter = 1;

        // using  while loop to get Nth Ugly number
        while (NthNumber > counter) {
            number++;
            if (checkUglyNumber(number))
                counter++;  //increment counter value
        }
        return number;
    }

    //main() method start
    public static void main(String args[])
    {

        int NthNumber1, NthNumber2;

        //create scanner class object to get input from user
        Scanner sc = new Scanner(System.in);

        //show custom message
        System.out.println("Enter first Nth value");

        //store user entered value into variable NthNumber1
        NthNumber1 = sc.nextInt();

        //show custom message
        System.out.println("Enter second Nth value");

        //store user entered value into variable NthNumber2
        NthNumber2 = sc.nextInt();

        System.out.println("The " + NthNumber1 + "th Ugly number is: " + findNthUglyNumber(NthNumber1));
        System.out.println("The " + NthNumber2 + "th Ugly number is: " + findNthUglyNumber(NthNumber2));
    }
}
```

UglyNumberExample2.java

```java
//import required classes and packages
import java.io.*;
import java.util.*;

//create UglyNumberExample2 class to check whether the given numbers are Ugly or not
class UglyNumberExample2 {

    // create a method divideByGreatestDivisible() method for dividing the number by the given greatest divisible power
    static int divideByGreatestDivisible(int number, int gdp)
    {
        while (number % gdp == 0)
            number = number / gdp;
        return number;
    }

    // create checkUglyNumber() method that returns true when the number is an Ugly
    static boolean checkUglyNumber(int number)
    {
        number = divideByGreatestDivisible(number, 2);
        number = divideByGreatestDivisible(number, 3);
        number = divideByGreatestDivisible(number, 5);

        return (number == 1) ? true : false;
    }

    //main() method start
    public static void main(String args[])
    {

        int number1, number2;

        //get input from user
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter first number to check");
        number1 = sc.nextInt();

        System.out.println("Enter second number to check");
        number2 = sc.nextInt();

        if (checkUglyNumber(number1))
                System.out.println(number1 + " is an Ugly number.");
            else
                System.out.println(number1 + " is not an Ugly number.");

        if (checkUglyNumber(number2))
                System.out.println(number2 + " is an Ugly number.");
            else
                System.out.println(number2 + " is not an Ugly number.");
        }
}
```

Output:



```
C:\Users\ajeet\OneDrive\Desktop\programs>javac UglyNumberExample2.java

C:\Users\ajeet\OneDrive\Desktop\programs>java UglyNumberExample2
Enter first number to check
30720
Enter second number to check
34671
30720 is an Ugly number.
34671 is not an Ugly number.

C:\Users\ajeet\OneDrive\Desktop\programs>
```

Using Dynamic Programming

There is another way of getting the Nth Ugly number that takes O(n) extra space. The series of starting few Ugly numbers is 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, ……..

We can split the above sequence into tree subsequences in such a way that each group in an ugly sequence like as

1. 1×2, 2×2, 3×2, 4×2, 5×2, …
2. 1×3, 2×3, 3×3, 4×3, 5×3, …
3. 1×5, 2×5, 3×5, 4×5, 5×5, …

In order to get each ugly number from the sequences, we use the merge method as merge sort.

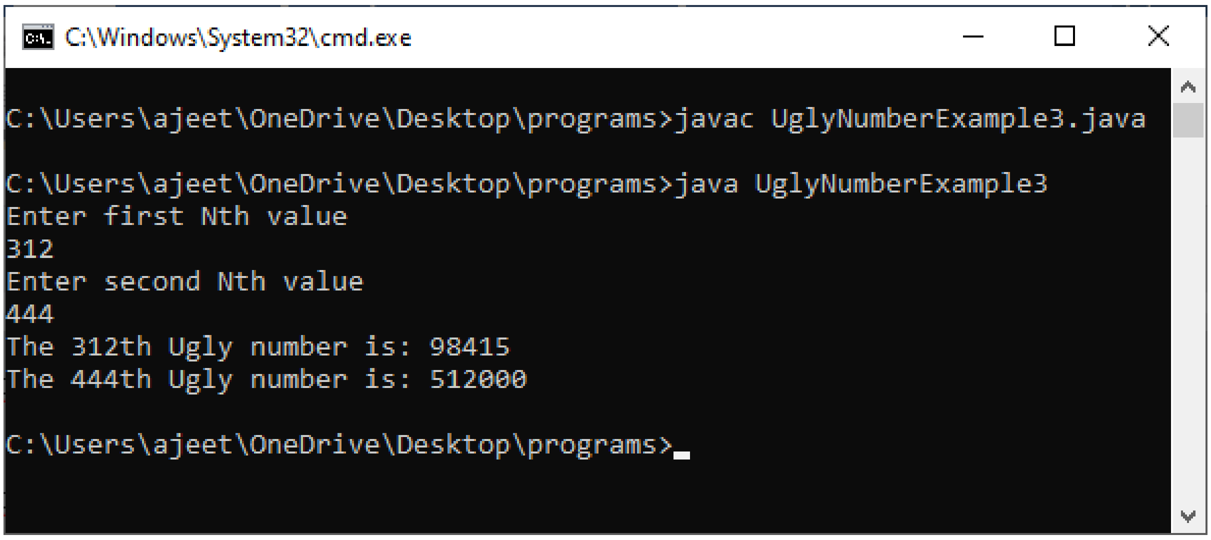
These are the following steps which we use to get the Nth ugly number using dynamic programming:

1. First, we declare an array for ugly numbers.
2. 1 is the first ugly number, so we initialize it our program as first ugly number.
3. Next, we initialize i2, i3, i5 as array index variables for pointing to the 1st element of the ugly array:
   i2 = i3 = i5 =0;
4. Initializing choices to next ugly number like as
   mulitple2 = ugly[i_2]*2;
   mulitple3 = ugly[i_3]*3
   mulitple5 = ugly[i_5]*5;
5. We use the looping statement for filling all the ugly numbers to the given range:

```
//import required classes and packages
import java.io.*;
import java.util.*;

//create UglyNumberExample1 class to get the Nth ungly number
class UglyNumberExample3 {

    //create findNthUglyNumber() method for getting the nth Ugly number
```

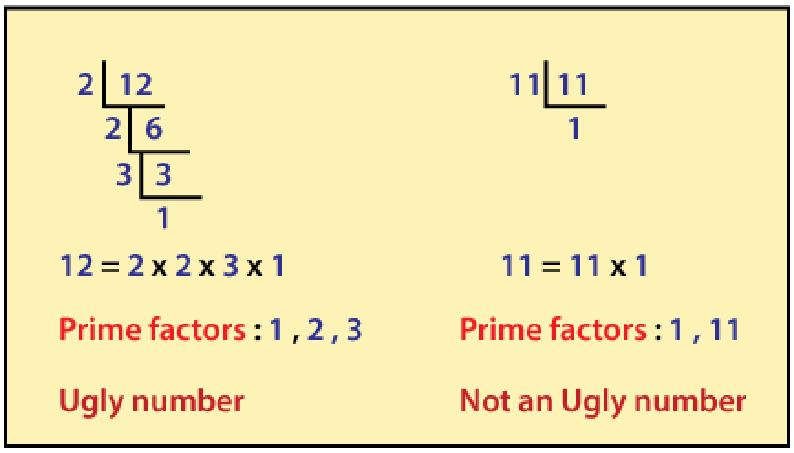
```java
    static int findNthUglyNumber(int number)
    {
         // declare an array for ugly numbers
        int uglyNumbers[] = new int[number];

        //we initialize i2, i3, i5  as array index variables for pointing to the 1st element of the ugly array:
        int i2 = 0, i3 = 0, i5 = 0;

        int multiple2 = 2;
        int multiple3 = 3;
        int multiple5 = 5;

        int nextUglyNumber = 1;

        uglyNumbers[0] = 1;

        for (int i = 1; i < number; i++)
        {
            nextUglyNumber = Math.min(multiple2, Math.min(multiple3, multiple5));

            uglyNumbers[i] = nextUglyNumber;
            if (nextUglyNumber == multiple2)
            {
                i2 = i2 + 1;
                multiple2 = uglyNumbers[i2] * 2;
            }
            if (nextUglyNumber == multiple3)
            {
                i3 = i3 + 1;
                multiple3 = uglyNumbers[i3] * 3;
            }
            if (nextUglyNumber == multiple5)
            {
                i5 = i5 + 1;
                multiple5 = uglyNumbers[i5] * 5;
            }
        }

        // End of for loop (i=1; i<n; i++)
        return nextUglyNumber;
    }

    //main() method start
    public static void main(String args[])
    {

        int NthNumber1, NthNumber2;

        // use Scanner class for taking value of NthNumber1 and NthNumber2
        Scanner sc = new Scanner(System.in);

        //show custom message
        System.out.println("Enter first Nth value");

        //store user entered value into variable NthNumber1
        NthNumber1 = sc.nextInt();

        //show custom message
        System.out.println("Enter second Nth value");

        //store user entered value into variable NthNumber2
        NthNumber2 = sc.nextInt();

        System.out.println("The " + NthNumber1 + "th Ugly number is: " + findNthUglyNumber(NthNumber1));
        System.out.println("The " + NthNumber2 + "th Ugly number is: " + findNthUglyNumber(NthNumber2));
    }
}
```

Output:



```
C:\Users\ajeet\OneDrive\Desktop\programs>javac UglyNumberExample3.java

C:\Users\ajeet\OneDrive\Desktop\programs>java UglyNumberExample3
Enter first Nth value
312
Enter second Nth value
444
The 312th Ugly number is: 98415
The 444th Ugly number is: 512000

C:\Users\ajeet\OneDrive\Desktop\programs>_
```

1. 27 is not an Ugly number because its prime factors contain 7.
2. 12 is an Ugly number because its prime factors are 2 and 3.
3. 28 is also not an Ugly number because its prime factors also contain 7.
4. 16 is an Ugly number because its prime factors only contain 2.


2 | 12
2 | 6
3 | 3
1
12 = 2 x 2 x 3 x 1
Prime factors : 1 , 2 , 3
Ugly number
11 | 11
1
11 = 11 x 1
Prime factors : 1 , 11
Not an Ugly number