HTML   CSS

Java Recursion

Java Recursion

Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.

Recursion may be a bit difficult to understand. The best way to figure out how it works is to experiment with it.

Recursion Example

Adding two numbers together is easy to do, but adding a range of numbers is more complicated. In the following example, recursion is used to add a range of numbers together by breaking it down into the simple task of adding two numbers:

Example

Use recursion to add all of the numbers up to 10.

```java
public class Main {
  public static void main(String[] args) {
    int result = sum(10);
    System.out.println(result);
```

```java
  }
  public static int sum(int k) {
    if (k > 0) {
      return k + sum(k - 1);
    } else {
      return 0;
    }
  }
}
```

Try it Yourself »

Example Explained

When the `sum()` function is called, it adds parameter `k` to the sum of all numbers smaller than `k` and returns the result. When k becomes 0, the function just returns 0. When running, the program follows these steps:

```
10 + sum(9)
10 + ( 9 + sum(8) )
10 + ( 9 + ( 8 + sum(7) ) )
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0
```

Since the function does not call itself when `k` is 0, the program stops there and returns the result.

ADVERTISEMENT

Halting Condition

Just as loops can run into the problem of infinite looping, recursive functions can run into the problem of infinite recursion. Infinite recursion is when the function never stops calling itself. Every recursive function should have a halting condition, which is the condition where the function stops calling itself. In the previous example, the halting condition is when the parameter `k` becomes 0.

It is helpful to see a variety of different examples to better understand the concept. In this example, the function adds a range of numbers between a start and an end. The halting condition for this recursive function is when **end** is not greater than **start**:

Example

Use recursion to add all of the numbers between 5 to 10.

```java
public class Main {
  public static void main(String[] args) {
    int result = sum(5, 10);
    System.out.println(result);
  }
  public static int sum(int start, int end) {
    if (end > start) {
      return end + sum(start, end - 1);
    } else {
      return end;
```

```
      }
   }
}
```

Try it Yourself »

The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.
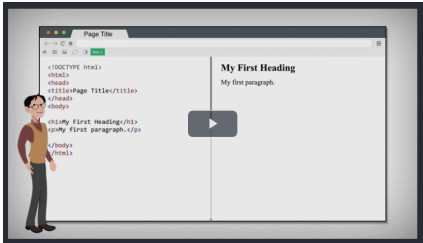
‹ Previous   Next ›

ADVERTISEMENT

NEW

We just launched
W3Schools videos

Explore now

COLOR PICKER

**Get certified
by completing
a Java
course today!**

Get started

CODE GAME

Play Game

Report Error

Spaces

Pro

Buy Certificate

Top Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial

PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
Angular Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

Get Certified

HTML Certificate
CSS Certificate
JavaScript Certificate
Front End Certificate
SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate

FORUM | ABOUT

W3Schools is optimized for learning and training. Examples might be simplified to improve reading and learning. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant