# F# - RECORDS

http://www.tutorialspoint.com/fsharp/fsharp_records.htm

A **record** is similar to a tuple, however it contains named fields. For example,

```
type website =
   { title : string;
      url : string }
```

## Defining Record

A record is defined as a type using the **type** keyword, and the fields of the record are defined as a semicolon-separated list.

Syntax for defining a record is −

```
type recordName =
   { [ fieldName : dataType ] + }
```

## Creating a Record

You can create a record by specifying the record's fields. For example, let us create a *website* record named *homepage* −

```
let homepage = { Title = "TutorialsPoint"; Url = "www.tutorialspoint.com" }
```

The following examples will explain the concepts −

## Example 1

This program defines a record type named website. Then it creates some records of type website and prints the records.

```
(* defining a record type named website *)
type website =
   { Title : string;
      Url : string }

(* creating some records *)
let homepage = { Title = "TutorialsPoint"; Url = "www.tutorialspoint.com" }
let cpage = { Title = "Learn C"; Url = "www.tutorialspoint.com/cprogramming/index.htm" }
let fsharppage = { Title = "Learn F#"; Url = "www.tutorialspoint.com/fsharp/index.htm" }
let csharppage = { Title = "Learn C#"; Url = "www.tutorialspoint.com/csharp/index.htm" }

(*printing records *)
(printfn "Home Page: Title: %A \n \t URL: %A") homepage.Title homepage.Url
(printfn "C Page: Title: %A \n \t URL: %A") cpage.Title cpage.Url
(printfn "F# Page: Title: %A \n \t URL: %A") fsharppage.Title fsharppage.Url
(printfn "C# Page: Title: %A \n \t URL: %A") csharppage.Title csharppage.Url
```

When you compile and execute the program, it yields the following output −

```
Home Page: Title: "TutorialsPoint"
       URL: "www.tutorialspoint.com"
C Page: Title: "Learn C"
      URL: "www.tutorialspoint.com/cprogramming/index.htm"
F# Page: Title: "Learn F#"
      URL: "www.tutorialspoint.com/fsharp/index.htm"
C# Page: Title: "Learn C#"
      URL: "www.tutorialspoint.com/csharp/index.htm"
```

## Example 2

```
type student =
   { Name : string;
      ID : int;
      RegistrationText : string;
      IsRegistered : bool }

let getStudent name id =
   { Name = name; ID = id; RegistrationText = null; IsRegistered = false }

let registerStudent st =
   { st with
      RegistrationText = "Registered";
      IsRegistered = true }

let printStudent msg st =
   printfn "%s: %A" msg st

let main() =
   let preRegisteredStudent = getStudent "Zara" 10
   let postRegisteredStudent = registerStudent preRegisteredStudent

   printStudent "Before Registration: " preRegisteredStudent
   printStudent "After Registration: " postRegisteredStudent

main()
```

When you compile and execute the program, it yields the following output −

```
Before Registration: : {Name = "Zara";
   ID = 10;
   RegistrationText = null;
   IsRegistered = false;}
After Registration: : {Name = "Zara";
   ID = 10;
   RegistrationText = "Registered";
   IsRegistered = true;}
```