

WELCOME




JAVAFX

PRESENTED BY:
www.seminarlab.com

INTRODUCTION

- JavaFX is a rich client platform for building cross-device application & content.
- Designed to enable easy creation & deployment of RIAs.
- it was designed from the ground up to make GUI programming easy; it's declarative syntax, data binding model, animation support, and built in visual effects let you accomplish more work with less code.
- It provide software download installation instruction plus an appropriate development environment.

JAVAFX SCRIPT

-  Download and install the JDK.
-  Choose a development environment IDE.
-  Download and install the JavaFX compiler and runtime.



WRITING SCRIPTS

- ✍ Declaring script variable.
- ✍ Defining and invoking script functions.

```
def numOne = 100;
def numTwo = 2;
var result;
add();
subtract();
multiply();
divide();
function add() {
result = numOne + numTwo;
println("{numOne} + {numTwo} = {result}");
}
```

➤ Eg. Draggable MP3 Player.



```
...  
function stopCurrentSong():Void {  
    mediaPlayer.stop();  
    mediaPlayer.media = null;  
    if (playlist.currentPlayingSong != null) {  
        playlist.currentPlayingSong.closeMedia();  
    }  
}  
  
function playCurrentSong():Void {  
    playlist.currentPlayingSong = playlist.songs[playlist.currentSong];  
    mediaPlayer.media = playlist.currentPlayingSong.getMedia();  
    mediaPlayer.play();  
}  
...
```

➤ Passing arguments to script functions

```
var result;
```

```
add(100,10);
```

```
subtract(50,5);
```

```
multiply(25,4);
```

```
divide(500,2);
```

```
function add(argOne: Integer, argTwo: Integer)
```

```
{
```

```
    result = argOne + argTwo;
```

```
    println("{argOne} + {argTwo} = {result}");
```

```
}
```

➤ Returning values from script functions

➤ Accessing command-line arguments

```
var result;
function run(args : String[]) {
// Convert Strings to Integers
def numOne=java.lang.Integer.parseInt(args[0]);
def numTwo =ava.lang.Integer.parseInt(args[1]);
// Invoke Functions
add(numOne,numTwo);
subtract(numOne,numTwo);
multiply(numOne,numTwo);
divide(numOne,numTwo);
}
function add(argOne: Integer, argTwo: Integer) {
result = argOne + argTwo;
println("{argOne} + {argTwo} = {result}");
}
```

OBJECTS

- What is an object?
- Declaring an object
- Object Syntax

Creates an instance of the Address class (an object)

street, city, state, and zip are this object's instance variables

```
Address {  
    street: "1 Main Street";  
    city: "Santa Clara";  
    state: "CA";  
    zip: "95050";  
}
```

opening/closing braces required

DATATYPES

- String
- Number & Integer
- Boolean
- Duration - fixed unit of time
- Void
- Null

☀ SEQUENCES

➤ Creating Sequences

```
eg:-def weekdays=["mon","tue","wed","thu","fri"];
```

➤ Sequences declared within other sequences.

```
eg:-def days=[weekdays ,["sat","sun"]];
```

➤ Accessing Sequence Item

```
def days=["sat","sun"];  
println(days[0]);  
println(days[1]);
```

- **Inserting Items into a Sequence**
insert "mon" into days;
- **Deleting Items from a Sequence**
delete "mon" from days;
- **Reversing the Items in a Sequence**
var nums=[1.....5]
reverse nums;
- **Comparing Sequences**
- **Sequence Slices provide access to portions of a Sequence**

OPERATORS

- Assignment Operators
- Arithmetic Operators
- Unary Operators
- Equality & Relational Operators
- Conditional Operators
- Type Comparison Operator



EXPRESSIONS

➤ Block Expression

```
var nums = [5, 7, 3, 9];  
var total = {  
  var sum = 0;  
  for (a in nums) { sum += a };  
  sum;}  
println("Total is {total}.");
```

➤ The if Expression

➤ The Range Expression

- **The for Expression**
- **The while Expression**
- **The break and continue Expression**
- **The throw, try, catch, and finally Expression**

☀ DATABINDING & TRIGERS

- The bind keyword associates the value of a target variable with the value of a bound Expression.
- Binding & Objects

```
var x = 0;  
ddef y = bind x;  
x = 1;  
println(y); // y now equals 1  
x = 47;  
println(y); // y now equals 47
```

➤ Binding & Functions

```
var scale = 1.0;
bound function makePoint(xPos : Number, yPos :Number) : Point {
  Point {
    x: xPos * scale
    y: yPos * scale}      }
class Point {
  var x : Number;
  var y : Number;}
```

➤ Binding with Sequences

```
var seq1 = [1..10];
def seq2 = bind for (item in seq1) item*2;
printSeqs();
```

➤ Replace Triggers

CLASSES

```
➤ def customer = Customer {  
  firstName: "John";  
  lastName: "Doe";  
  phoneNum: "(408) 555-1212"  
  address: Address {  
  
    class Address {  
      var street: String;  
      var city: String;  
      var state: String;  
      var zip: String;  
    }  
  }  
}
```

➤ Inheriting from other Classes

```
abstract class Account {  
    var accountNum: Integer;  
    var balance: Number;  
    function getBalance(): Number {  
        return balance;}  
    function deposit(amount: Number): Void {  
        balance += amount;}}
```

PACKAGES

- Create a Package
- Add the Package Declaration

```
package addressbook;;  
class Address {  
    var street: String;  
    var city: String;  
    var state: String;  
    var zip: String;}
```

- Add the Access Modifier(public).

- **Compile the Source**
- **Use the Class**

ACCESS MODIFIERS

- Default Access
- The Package Access Modifier
- The protected Access Modifier
- The public Access Modifier
- The public-read Access Modifier
- The public-init Access Modifier

CONCLUSION

THANK YOU

By
www.seminarlab.com

www.seminarlab.com