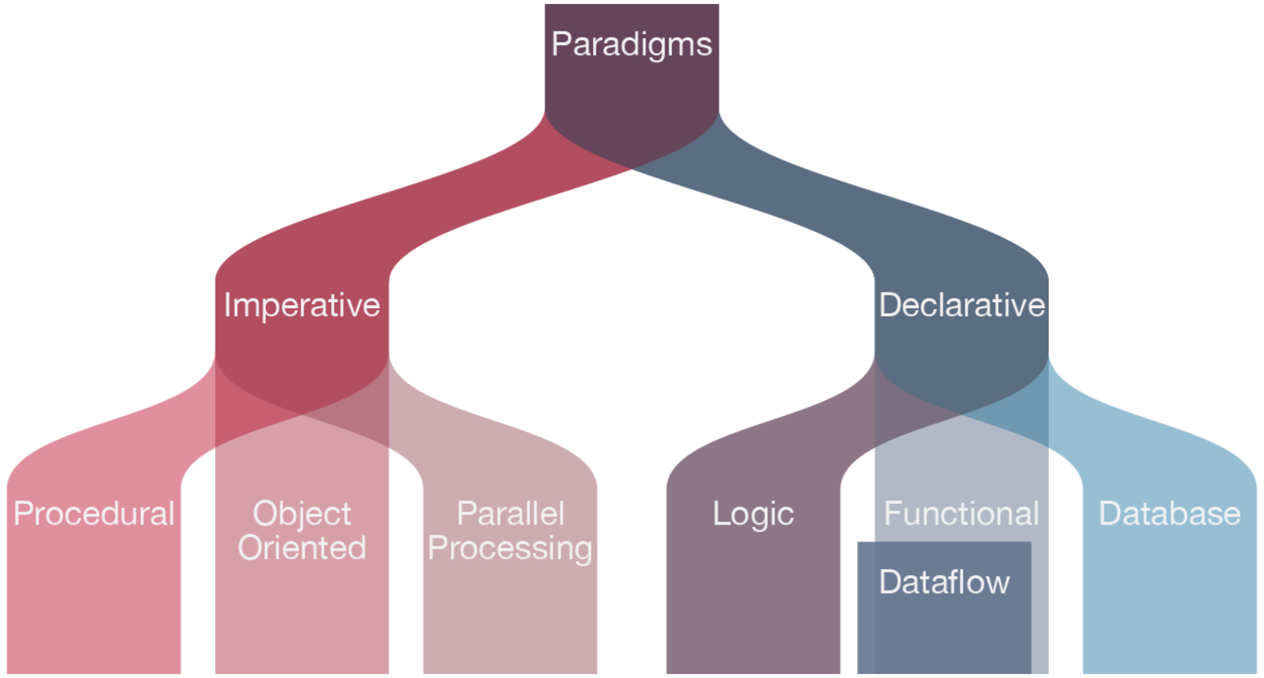
Scala - Day One

Programming Paradigms

Main Programming Paradigms:

- Imperative Programming
- Declarative Programming
- Functional Programming
- Object Oriented Programming

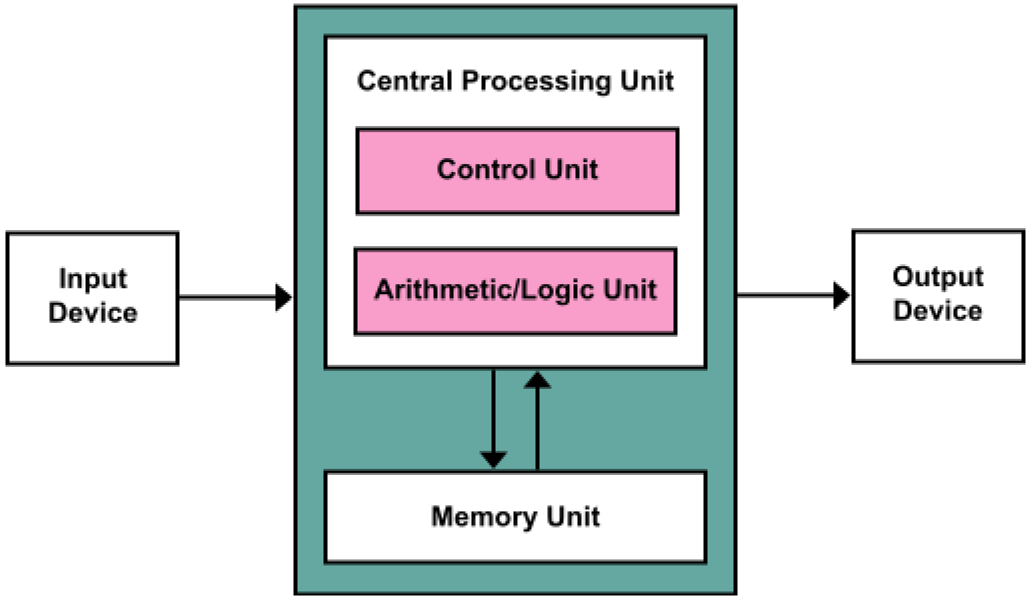
Programming Languages: C, C++, Java, Scala, Python and etc.



Imperative Programming

- It uses statements that change a program's state.
- It is about
  - modifying mutable variable using assignment
  - control structures such as if-then-else, loops, break, continue and return.
- In simple way to understand imperative programs is as instruction sequences for a Von Neumann Computer.

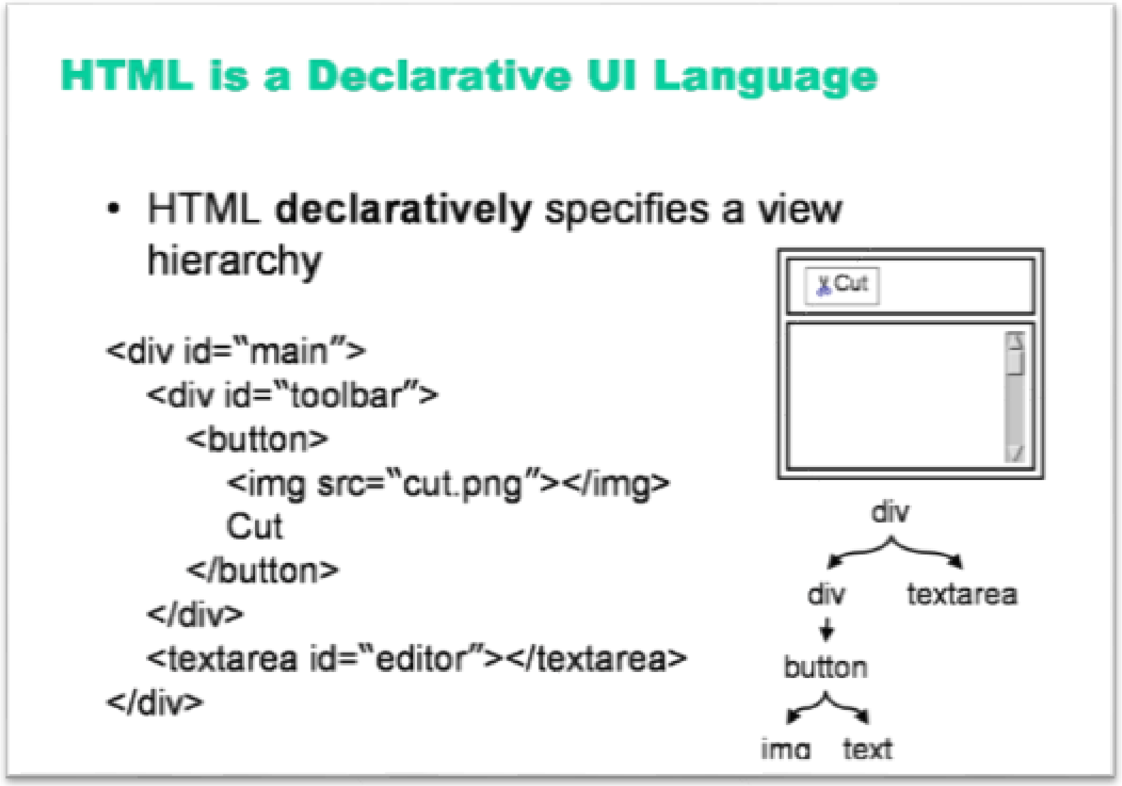
Procedural and object-oriented programming belong under imperative paradigm that you know from languages like C, C++, C#, PHP, Java and of course Assembly.



Declarative Programming

- Declarative code focuses on building logic of software without actually describing its flow.
- It expresses the logic of a computation without describing its control flow.
- Logic, functional and domain-specific languages belong under declarative paradigms.

Examples would be HTML, XML, CSS, SQL, Prolog, Haskell, F# and Lisp.



HTML is a Declarative UI Language

- HTML **declaratively** specifies a view hierarchy

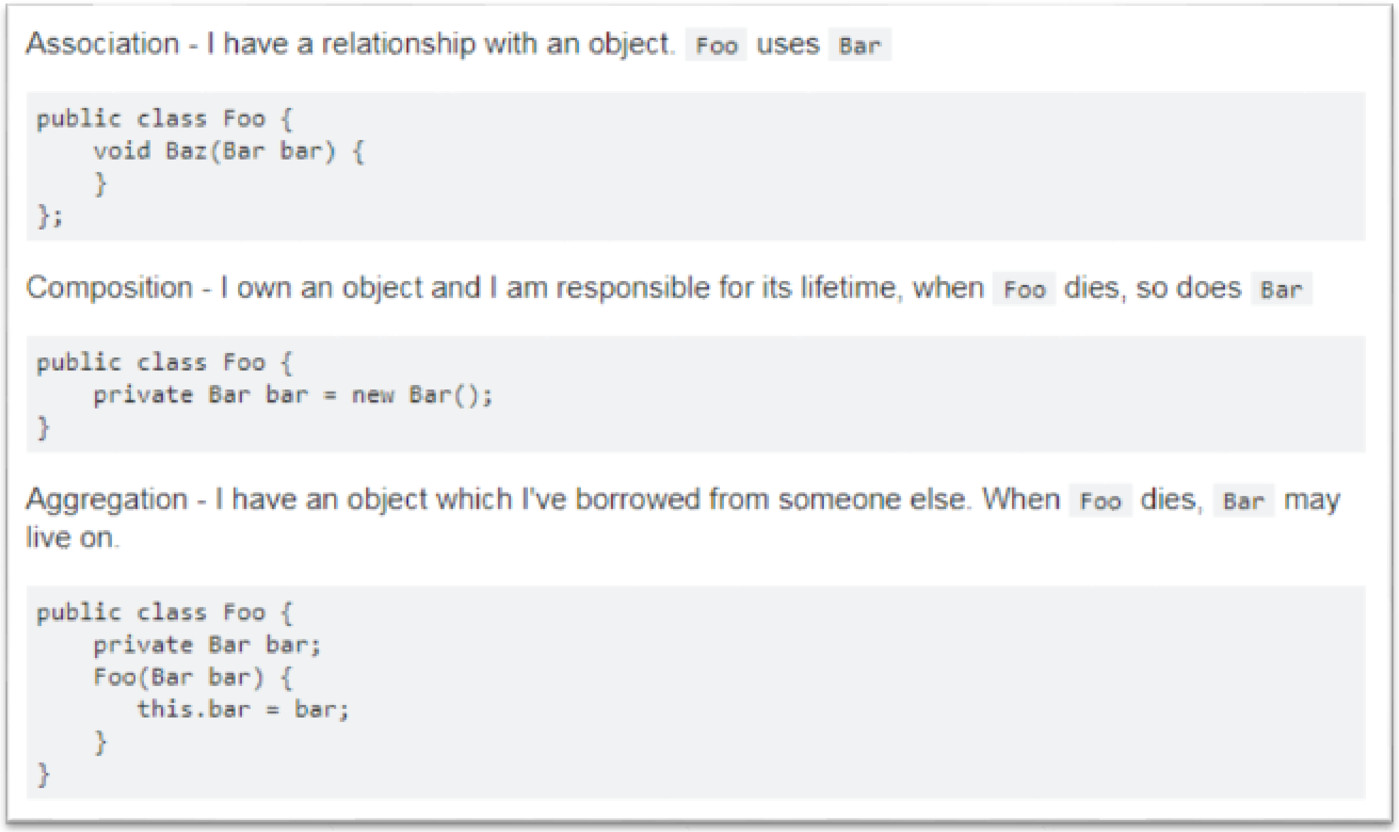
```
<div id="main">
  <div id="toolbar">
    <button>
      <img src="cut.png"></img>
      Cut
    </button>
  </div>
  <textarea id="editor"></textarea>
</div>
```

Object Oriented Programming

- It is based on the concept of "Objects", which may contain data in the form of fields and code in the form of procedures. *(fields -> attributes, procedures -> methods)*

Four principles of OOP

- Encapsulation – hiding the data by restricting access.
- Abstraction – abstracting the information which is not associated with any instances.
- Inheritance – In derived classes we implement abstract or super class.
- Polymorphism – overloading and overriding.



Association - I have a relationship with an object. `Foo` uses `Bar`

```
public class Foo {
    void Baz(Bar bar) {
    }
};
```

Composition - I own an object and I am responsible for its lifetime, when `Foo` dies, so does `Bar`

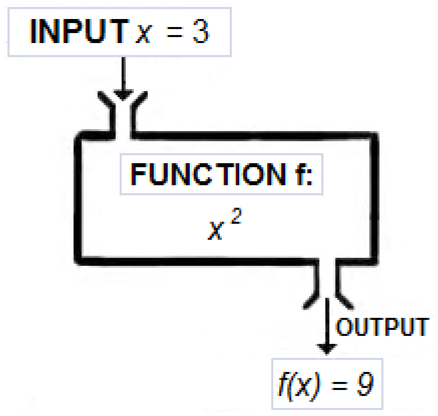
```
public class Foo {
    private Bar bar = new Bar();
}
```

Aggregation - I have an object which I've borrowed from someone else. When `Foo` dies, `Bar` may live on.

```
public class Foo {
    private Bar bar;
    Foo(Bar bar) {
       this.bar = bar;
    }
}
```

Functional Programming

- It is a style of building the structure and elements of computer programs, that treats computation as evaluation of mathematical functions.
- It avoids changing state and mutable data.
- Functional programming has its origins in lambda calculus and formal systems developed in the 1930s to investigate computability, function definition, function application and recursion.



Functional programming languages are categorized into two groups, i.e. −

- **Pure Functional Languages** − These types of functional languages support only the functional paradigms. For example − Haskell.
- **Impure Functional Languages** − These types of functional languages support the functional paradigms and imperative style programming. For example − LISP.

Functional programming is based on mathematical functions. Some of the popular functional programming languages include: Lisp, Python, Erlang, Haskell, Clojure, Scala etc.

Scala

- Scala is hybrid language. It bridges the gap between imperative, declarative, object oriented and functional programming.
- Scala is a *“multi paradigm”* language supporting both object oriented and functional programming.
- Scala source code is intended to be compiled to Java Bytecode, so that the resulting executable code runs on ***Java Virtual Machines*****.**
- Like Java, Scala is object oriented and uses a curly-brace syntax.
- Unlike Java, Scala has many features of functional programming including currying, type inference, immutability, lazy evaluation and pattern matching.
- It also has an advanced type system supporting algebraic data types, covariance and contra-variance, higher-order types and anonymous types.