# LYAH – Rough Weekly Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **W** | **B** | **Title** | **Difficulty** | **Date** |
| 1 |  | [Introduction](http://learnyouahaskell.com/introduction) | Beginner | 15/11/11 |
| 2 | 1 | [Starting Out](http://learnyouahaskell.com/starting-out) | Beginner | 22/11/11 |
| 3 | 2 | [Types and Typeclasses](http://learnyouahaskell.com/types-and-typeclasses) | Beginner | 29/11/11 |
|  |  | YOW |  | 6/12/11 |
| 4 | 3 | [Syntax in Functions](http://learnyouahaskell.com/syntax-in-functions) | Beginner | 13/12/11 |
| 5 | 4 | [Recursion](http://learnyouahaskell.com/recursion) | Beginner | 20/12/11 |
|  |  | Christmas |  | 27/12/11 |
|  |  | New Year |  | 3/01/12 |
| 6 | 5 | [Higher Order Functions](http://learnyouahaskell.com/higher-order-functions) | Beginner | 10/01/12 |
| 7 | 6 | [Modules](http://learnyouahaskell.com/modules) | Beg/Long | 17/01/12 |
| 8 | 7 | [Making Our Own Types and Typeclasses](http://learnyouahaskell.com/making-our-own-types-and-typeclasses) | Int | 24/01/12 |
| 9 | 8 | 1st half of [Input and Output](http://learnyouahaskell.com/input-and-output) | Int | 31/01/12 |
| 9b | 9 | 2nd half of [Input and Output](http://learnyouahaskell.com/input-and-output) (Book chapter 9 “More Input and Output” | Int | 7/02/12 |
| 10 | 10 | [Functionally Solving Problems](http://learnyouahaskell.com/functionally-solving-problems) | Int | 14/02/12 |
| 11 | 11 | 1st half of Functors, Applicative Functors and Monoids | Advanced | 21/02/12 |
| 11b | 12 | 2nd half of Functors, Applicative Functors and Monoids (Book Chapter 12 “Monoids”) | Advanced | 28/02/12 |
| 12 | 13 | [A Fistful of Monads](http://learnyouahaskell.com/a-fistful-of-monads) | Advanced | 6/03/12 |
| 13 | 14 | [For a Few Monads More](http://learnyouahaskell.com/for-a-few-monads-more) | Advanced | 13/03/12 |
| 14 | 15 | [Zippers](http://learnyouahaskell.com/zippers) | Advanced | 20/03/12 |

## LYAH – Website Outline

1. [Introduction](http://learnyouahaskell.com/introduction)
   * [About this tutorial](http://learnyouahaskell.com/introduction#about-this-tutorial)
   * [So what's Haskell?](http://learnyouahaskell.com/introduction#so-whats-haskell)
   * [What you need to dive in](http://learnyouahaskell.com/introduction#what-you-need)
2. [Starting Out](http://learnyouahaskell.com/starting-out)
   * [Ready, set, go!](http://learnyouahaskell.com/starting-out#ready-set-go)
   * [Baby's first functions](http://learnyouahaskell.com/starting-out#babys-first-functions)
   * [An intro to lists](http://learnyouahaskell.com/starting-out#an-intro-to-lists)
   * [Texas ranges](http://learnyouahaskell.com/starting-out#texas-ranges)
   * [I'm a list comprehension](http://learnyouahaskell.com/starting-out#im-a-list-comprehension)
   * [Tuples](http://learnyouahaskell.com/starting-out#tuples)
3. [Types and Typeclasses](http://learnyouahaskell.com/types-and-typeclasses)
   * [Believe the type](http://learnyouahaskell.com/types-and-typeclasses#believe-the-type)
   * [Type variables](http://learnyouahaskell.com/types-and-typeclasses#type-variables)
   * [Typeclasses 101](http://learnyouahaskell.com/types-and-typeclasses#typeclasses-101)
4. [Syntax in Functions](http://learnyouahaskell.com/syntax-in-functions)
   * [Pattern matching](http://learnyouahaskell.com/syntax-in-functions#pattern-matching)
   * [Guards, guards!](http://learnyouahaskell.com/syntax-in-functions#guards-guards)
   * [Where!?](http://learnyouahaskell.com/syntax-in-functions#where)
   * [Let it be](http://learnyouahaskell.com/syntax-in-functions#let-it-be)
   * [Case expressions](http://learnyouahaskell.com/syntax-in-functions#case-expressions)
5. [Recursion](http://learnyouahaskell.com/recursion)
   * [Hello recursion!](http://learnyouahaskell.com/recursion#hello-recursion)
   * [Maximum awesome](http://learnyouahaskell.com/recursion#maximum-awesome)
   * [A few more recursive functions](http://learnyouahaskell.com/recursion#a-few-more-recursive-functions)
   * [Quick, sort!](http://learnyouahaskell.com/recursion#quick-sort)
   * [Thinking recursively](http://learnyouahaskell.com/recursion#thinking-recursively)
6. [Higher Order Functions](http://learnyouahaskell.com/higher-order-functions)
   * [Curried functions](http://learnyouahaskell.com/higher-order-functions#curried-functions)
   * [Some higher-orderism is in order](http://learnyouahaskell.com/higher-order-functions#higher-orderism)
   * [Maps and filters](http://learnyouahaskell.com/higher-order-functions#maps-and-filters)
   * [Lambdas](http://learnyouahaskell.com/higher-order-functions#lambdas)
   * [Only folds and horses](http://learnyouahaskell.com/higher-order-functions#folds)
   * [Function application with $](http://learnyouahaskell.com/higher-order-functions#function-application)
   * [Function composition](http://learnyouahaskell.com/higher-order-functions#composition)
7. [Modules](http://learnyouahaskell.com/modules)
   * [Loading modules](http://learnyouahaskell.com/modules#loading-modules)
   * [Data.List](http://learnyouahaskell.com/modules#data-list)
   * [Data.Char](http://learnyouahaskell.com/modules#data-char)
   * [Data.Map](http://learnyouahaskell.com/modules#data-map)
   * [Data.Set](http://learnyouahaskell.com/modules#data-set)
   * [Making our own modules](http://learnyouahaskell.com/modules#making-our-own-modules)
8. [Making Our Own Types and Typeclasses](http://learnyouahaskell.com/making-our-own-types-and-typeclasses)
   * [Algebraic data types intro](http://learnyouahaskell.com/making-our-own-types-and-typeclasses#algebraic-data-types)
   * [Record syntax](http://learnyouahaskell.com/making-our-own-types-and-typeclasses#record-syntax)
   * [Type parameters](http://learnyouahaskell.com/making-our-own-types-and-typeclasses#type-parameters)
   * [Derived instances](http://learnyouahaskell.com/making-our-own-types-and-typeclasses#derived-instances)
   * [Type synonyms](http://learnyouahaskell.com/making-our-own-types-and-typeclasses#type-synonyms)
   * [Recursive data structures](http://learnyouahaskell.com/making-our-own-types-and-typeclasses#recursive-data-structures)
   * [Typeclasses 102](http://learnyouahaskell.com/making-our-own-types-and-typeclasses#typeclasses-102)
   * [A yes-no typeclass](http://learnyouahaskell.com/making-our-own-types-and-typeclasses#a-yes-no-typeclass)
   * [The Functor typeclass](http://learnyouahaskell.com/making-our-own-types-and-typeclasses#the-functor-typeclass)
   * [Kinds and some type-foo](http://learnyouahaskell.com/making-our-own-types-and-typeclasses#kinds-and-some-type-foo)
9. [Input and Output](http://learnyouahaskell.com/input-and-output)
   * [Hello, world!](http://learnyouahaskell.com/input-and-output#hello-world)
   * [Files and streams](http://learnyouahaskell.com/input-and-output#files-and-streams)
   * [Command line arguments](http://learnyouahaskell.com/input-and-output#command-line-arguments)
   * [Randomness](http://learnyouahaskell.com/input-and-output#randomness)
   * [Bytestrings](http://learnyouahaskell.com/input-and-output#bytestrings)
   * [Exceptions](http://learnyouahaskell.com/input-and-output#exceptions)
10. [Functionally Solving Problems](http://learnyouahaskell.com/functionally-solving-problems)
    * [Reverse Polish notation calculator](http://learnyouahaskell.com/functionally-solving-problems#reverse-polish-notation-calculator)
    * [Heathrow to London](http://learnyouahaskell.com/functionally-solving-problems#heathrow-to-london)
11. [Functors, Applicative Functors and Monoids](http://learnyouahaskell.com/functors-applicative-functors-and-monoids)
    * [Functors redux](http://learnyouahaskell.com/functors-applicative-functors-and-monoids#functors-redux)
    * [Applicative functors](http://learnyouahaskell.com/functors-applicative-functors-and-monoids#applicative-functors)
    * [The newtype keyword](http://learnyouahaskell.com/functors-applicative-functors-and-monoids#the-newtype-keyword)
    * [Monoids](http://learnyouahaskell.com/functors-applicative-functors-and-monoids#monoids)
12. [A Fistful of Monads](http://learnyouahaskell.com/a-fistful-of-monads)
    * [Getting our feet wet with Maybe](http://learnyouahaskell.com/a-fistful-of-monads#getting-our-feet-wet-with-maybe)
    * [The Monad type class](http://learnyouahaskell.com/a-fistful-of-monads#the-monad-type-class)
    * [Walk the line](http://learnyouahaskell.com/a-fistful-of-monads#walk-the-line)
    * [do notation](http://learnyouahaskell.com/a-fistful-of-monads#do-notation)
    * [The list monad](http://learnyouahaskell.com/a-fistful-of-monads#the-list-monad)
    * [Monad laws](http://learnyouahaskell.com/a-fistful-of-monads#monad-laws)
13. [For a Few Monads More](http://learnyouahaskell.com/for-a-few-monads-more)
    * [Writer? I hardly know her!](http://learnyouahaskell.com/for-a-few-monads-more#writer)
    * [Reader? Ugh, not this joke again.](http://learnyouahaskell.com/for-a-few-monads-more#reader)
    * [Tasteful stateful computations](http://learnyouahaskell.com/for-a-few-monads-more#state)
    * [Error error on the wall](http://learnyouahaskell.com/for-a-few-monads-more#error)
    * [Some useful monadic functions](http://learnyouahaskell.com/for-a-few-monads-more#useful-monadic-functions)
    * [Making monads](http://learnyouahaskell.com/for-a-few-monads-more#making-monads)
14. [Zippers](http://learnyouahaskell.com/zippers)
    * [Taking a walk](http://learnyouahaskell.com/zippers#taking-a-walk)
    * [A trail of breadcrumbs](http://learnyouahaskell.com/zippers#a-trail-of-breadcrumbs)
    * [Focusing on lists](http://learnyouahaskell.com/zippers#focusing-on-lists)
    * [A very simple file system](http://learnyouahaskell.com/zippers#a-very-simple-file-system)
    * [Watch your step](http://learnyouahaskell.com/zippers#watch-your-step)