Coq - a general purpose formal deduction system implemented on a computer.

There are different styles of using Coq to define objects and to prove theorems about these objects.

Once you learned Coq it is easier to move between different styles of using Coq.

There is, so far, no good book for learning to use Coq in the univalent style to define mathematical objects and prove theorems about them.

But there are books on using Coq in other styles:

1. Interactive Theorem Proving and Program Development by Ives Bertot and Pierre Casteran, also known as Coq'Art.

2. Certified Programming with Dependent Types by Adam Chlipala.

3. Software Foundations by Benjamin C. Pierce et al.

Also have a look at the author's websites.

There is yet no good reference for using Univalent Foundations to formalize classical mathematics. I am going to start working on it next year.

There is a book about Homotopy Type Theory and the emerging Univalent Synthesis. It is called "**Homotopy Type Theory**" and is very easy to find with Google or any other proof engine.

I will tell a little bit about the story of this book in my lecture at the Asian Science Camp next week.

Here is a list of things which I think someone who wants to work on the Univalent Synthesis should learn:

1. OCaml and Haskell programming languages.

   1a. More of a research topic - Fresh OCaml.

2. Coq and Agda proof assistants (at least one style of use).

3. Elementary classical category theory and homotopy theory (there are many math courses and books on these subjects).

   3a. More of a research topic but necessary - "Homotopy Type Theory" book.