Test.index

December 8, 2025

Contents

1 Test.Hierarchy.Sub.Base **2**

2 Test.Hierarchy.Sub **3**

3 Test.Plain.Test **4**

4 Test.Plain.Test2 **6**

5 Test.index **7**

6 Test.Literate.LaTeX **8**

1 Test.Hierarchy.Sub.Base

```
module Test.Hierarchy.Sub.Base where

-- Testing a level 4 module

module Test-TOC-1 where

  module Test-TOC-11 where

    module Test-TOC-111 where

      module Test-TOC-1111 where

        module Test-TOC-11111 where

  module Test-TOC-12 where

module Test-TOC-2 where

-- Testing in-page navigation
```

2 Test.Hierarchy.Sub

```
module Test.Hierarchy.Sub where

-- Testing a level 3 module
```

3 Test.Plain.Test

```
-- This test file currently lacks module-related stuff.

{- Nested
   {- comment. -} -}

module Test.Plain.Test where

infix  12 _!
infixl 7 _+_ _-_
infixr 2 -_

postulate x : Set

f : (Set → Set → Set) → Set
f _*_ = x * x

data ℕ : Set where
  zero : ℕ
  suc : ℕ → ℕ

_+_ : ℕ → ℕ → ℕ
zero + n = n
suc m + n = suc (m + n)

postulate _-_ : ℕ → ℕ → ℕ

-_ : ℕ → ℕ
- n = n

_! : ℕ → ℕ
zero ! = suc zero
suc n ! = n - n !

record Equiv {a : Set} (_≈_ : a → a → Set) : Set where
  field
    refl  : forall x    → x ≈ x
    sym : {x y : a} → x ≈ y → y ≈ x
    _`trans`_ : forall {x y z} → x ≈ y → y ≈ z → x ≈ z

data _≡_ {a : Set} (x : a) : a → Set where
  refl : x ≡ x

subst : forall {a x y} →
  (P : a → Set) → x ≡ y → P x → P y
subst {x = x} .{y = x} _ refl p = p

Equiv-≡ : forall {a} → Equiv {a} _≡_
Equiv-≡ {a} =
  record { refl = λ _ → refl
         ; sym = sym
         ; _`trans`_ = _`trans`_
```

```
          }
  where
  sym : {x y : a} → x ≡ y → y ≡ x
  sym refl = refl

  _‘trans‘_ : {x y z : a} → x ≡ y → y ≡ z → x ≡ z
  refl ‘trans‘ refl = refl

postulate
  String : Set
  Char   : Set
  Float  : Set

data Int : Set where
  pos    : ℕ → Int
  negsuc : ℕ → Int

{-# BUILTIN STRING String #-}
{-# BUILTIN CHAR   Char   #-}
{-# BUILTIN FLOAT  Float  #-}

{-# BUILTIN NATURAL ℕ  #-}

{-# BUILTIN INTEGER       Int  #-}
{-# BUILTIN INTEGERPOS pos #-}
{-# BUILTIN INTEGERNEGSUC negsuc #-}

data [_] (a : Set) : Set where
  [] : [ a ]
  _::_ : a → [ a ] → [ a ]

{-# BUILTIN LIST [_] #-}
-- {-# BUILTIN NIL  []  #-}
-- {-# BUILTIN CONS _::_ #-}

primitive
  primStringToList : String → [ Char ]

string : [ Char ]
string = primStringToList "∃ apa"

char : Char
char = '∀'

anotherString : String
anotherString = "¬ be\
     \pa"

nat : ℕ
nat = 45

float : Float
float = 45.0e-37
```

4 Test.Plain.Test2

```
module Test.Plain.Test2 where

open import Test.Plain.Test

-- Testing the inter-file goto facility.

test : ℕ
test = 12 + 34 + 56

-- Testing qualified names.

Eq = Test.Plain.Test.Equiv {Test.Plain.Test.ℕ}
```

5 Test.index

```
module Test.index where

import Test.Hierarchy.Sub
import Test.Hierarchy.Sub.Base

import Test.Literate.LaTeX
import Test.Literate.Markdown
import Test.Plain.Test
import Test.Plain.Test2
```

6 Test.Literate.LaTeX

This literate Agda file has extension ‘lagda‘.

```
module Test.Literate.LaTeX where

-- Just testing...
```

In generated web pages, prose in files with extension ‘lagda‘ or ‘lagda.tex‘ is rendered *verbatim* in a fixed-width font:

* Line breaks and alignment are preserved.

* Markup is displayed without rendering.

* URLs do not become active links.

See ‘Test.Literate.Markdown‘ for an example of a literate Agda file where the prose is rendered as Markdown in generated web pages.

LaTeX files with highlighted Agda code can be generated from files with extension ‘lagda‘ or ‘lagda.tex‘. However, extensive use of LaTeX markup in literate Agda source files can significantly reduce the readability of the prose in generated web pages.