UIElements

- Lost My Mind
  - PreventDefaultAction ?
- The visual tree
  - VisualElement
    - Base class for all nodes in the visual tree
    - May have child elements
    - You can customize the look and behavior
  - Connectivity
    - VisualElement.Add()
    - EditorWindow.rootVisualElement
  - Drawing order
    - parents before children
    - children are drawn according to their sibling list
  - Coordinate systems
    - World: relative to the panel space. The panel is the highest element in the visual tree.
    - Local: relative to the element itself.
    - VisualElement.layout (which the layout system calculate)
      - layout.position
      - layout.transform
      - static class VisualElementExtensions
        - WorldToLocal
        - LocalToWorld
        - ChangeCoordinatesTo - (local space A to local space B)
- The layout engine
  - Yoga — implements a subset of Flexbox
  - Flexbox — a HTML/CSS layout system.
  - Default layout
    - A container distributes its children vertically.
    - A visual element with text uses the text size in its size calculations.
    - The position of a container rectangle includes its children rectangles.
    - *This behaviour can be restricted by other layout properties.*
  - Standard UI controls
    - button
    - toggle
    - text field
    - label .etc
- The UXML format
  - familiar with HTML / XAML / XML
  - Allows you to
    - define the structure of the user interface (UI) in XML
    - define the UI layout with USS stylesheets
  - UXML Template

```
<?xml version="1.0" encoding="utf-8"?>
<UXML
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="UnityEngine.UIElements"
  xsi:noNamespaceSchemaLocation="../UIElementsSchema/UIElements.xsd"
  xsi:schemaLocation="UnityEngine.UIElements ../UIElementsSchema/UnityEngine.UIElements.xsd">

  <Label text="Select something to remove from your suitcase:"/>
  <Box>
    <Toggle name="boots" label="Boots" value="false" />
    <Toggle name="helmet" label="Helmet" value="false" />
  </Box>
  <Button name="ok" text="OK" />
</UXML>
```

  - Defining new elements

```
class StatusBar : VisualElement
{
  public class Factory : UxmlFactory<StatusBar> { }

  public StatusBar()
  {
    m_Status = String.Empty;
  }

  string m_Status;
  public string status { get; set; }
}
```

  - Defining attributes on elements

```
class StatusBar : VisualElement
{
  public class Factory : UxmlFactory<StatusBar, Traits> {}

  public class Traits : VisualElement.UxmlTraits
  {
    UxmlStringAttributeDescription m_Status = new UxmlStringAttributeDescription { name = "status" };

    public override IEnumerable<UxmlChildElementDescription> uxmlChildElementsDescription
    {
      get ;   // $$$$$$$$$$ This not work $$$$$$$$$ https://docs.unity3d.com/Manual/UIE-UXML.html
    }

    public override void Init(VisualElement ve, IUxmlAttributes bag, CreationContext cc) { }
  }

  // ...
}
```

  - Defining a namespace prefix — [assembly: UxmlNamespacePrefix("My.First.Namespace", "First")]
  - Reusing UXML — using the <Template> and <Instance> elements, See " Eve.uxml"
  - Loading UXML from C#
    - var template = EditorGUIUtility.Load("path/to/file.uxml") as VisualTreeAsset;
    - var template = AssetDatabase.LoadAssetAtPath<VisualTreeAsset>("path/to/file.uxml");
    - template.CloneTree()
  - UXML Elements Reference
    - Base elements
      - BindableElement — can be bound to a SerializedProperty
      - VisualElement
        - class
        - style
        - name
        - tooltip
        - tabindex .....
    - Utilities
      - Box \ TextElement \ Label \ Image \ Foldout
      - IMGUIContainer
    - Templates
      - Template — A reference to another UXML template that can be instantiated using the Instance element.
      - Instance — Instance of a Template.
      - TemplateContainer — A template container
    - Controls
      - BaseField<T>
        - BaseFieldTrait<ValueType, UxmlType>
        - Button \ RepeatButton \ Toggle \ Slider \ Scroller .....
      - TextInputBaseField<TValueType>
    - TextInput
      - TextField
      - IntegerField \ LongField \ FloatField \ DoubleField \ Vector2Field \ Vector2IntField \ Vector3Field .....
    - Complex widgets
      - PropertyField
      - PropertyControl<Int \ long \ float \ strings>
      - ColorField \ CurveField \ ObjectField \ InspectorElement .....
    - Toolbar
      - Toolbar
      - ToolbarXX (Button,Toggle,Menu,SearchField.....)
    - Views and windows
      - ListView
      - ScrollView
      - TreeView
      - PopWindow
  - UQuery
    - root.Query<Button>("foo").First();
    - root.Query("foo").Children<Button>().ForEach(Ido stuff);
- Styles
  - Unity style sheet : USS
    - Definition — is an asset; have the .uss extension; The USS syntax matches the W3C specifications for CSS3.
    - Attaching USS to visual elements
      - Load "StyleSheet" objects
      - VisualElement.styleSheets.Add()
    - Matching
      - C# class name
      - A name property that is a string
      - A class list represented as a set of strings
      - The ancestry and position of the VisualElement in the visual tree.
    - USS selectors
      - TypeName (...) — C# type.
      - #name { ... } — VisualElement.name .
      - .class { ... }
      - * { ... } — Matches any element.
      - :pseudo-state { ... } — pseudo-states like : Button:hover , :active,:inactive, focus,selected,disabled,enabled,checked,root
      - Complex selectors
        - #name .class { ... } — The empty (or space) delimiter matches all descendants of an element.
        - #name > .class { ... } — The symbol (>) matches visual elements that are the direct descendant of an element matched by the previous selectors.
        - #name , .class { ... } — Selector List -- Same as : #name { ... } .class { ... }
    - USS Properties types
      - Length — pixel unit, [width:200px;]
      - Numeric — float or int, [flex:1.0]
      - Keywords — For example: position:absolute.
      - Color — rgb(255, 255, 0) / rgba(255, 255, 0, 1.0) / #92FF00
      - Assets — background-image: resource("Images/my-image"); background-image: url("Images/my-image.png"); .....
      - strings
- Event System
  - Event Target
    - PickingMode.Position / Ignore
    - element.CaptureMouse() / ReleaseMouse()
    - focus order controlled by tabindex
  - Event propagation
    - Trickles down:
      - send to Target
    - Bubbles up:
  - Responding to Events
    - by registering an event callback,
      - Callbacks must be registered on instances of the class.
      - Callbacks are executed for all elements along the propagation path
      - myElement.RegisterCallback<MouseDownEvent>(MyCallback);
    - by implementing a default action.
      - Default actions are implemented as virtual functions on the class.
      - Default actions are executed for the event target only.
      - override void ExecuteDefaultActionAtTarget(EventBase evt)
    - event handling
      - Event.currentTarget is the element on which the callback was registered
      - Event.target is the element on which the event occured.
    - Stopping event propagation and cancelling default actions
      - StopImmediatePropagation()
      - StopPropagation()
      - PreventDefaultAction()
    - Synthesizing Events
    - Event Type Reference
- Built-in Controls
- Bindings
  - Condition
    - derives from IBindableElement .
    - implements the IBindable interface..
  - Binding Steps
    - specify the 【bindingPath】 from the IBindable interface
    - Create a 【SerializedObject】 for the object being bound.
    - 【Bind】 this object to the UIElements Control or one of its parents.
- IMGUIContainer
  - var mLeftContainer = new IMGUIContainer(OnMyGUI());
  - root.Add(mLeftContainer);

XMind:ZEN | 试用模式