# Renderer Component Trees

**Documentation Migration**

Beginning in version 1.3 of Infusion, we will be migrating our documentation over to our new Documentation space. These Renderer pages are being revamped in the new space, and we encourage you to refer to them. The new pages are not yet as comprehensive as these pages, but they are more focused, and include direct guidance on the recommended way to use the Renderer. They also include information about the new ProtoComponents.

If you have any comments on the new documentation format, feel free to let us know by emailing the infusion-users mailing list.

## Overview

A Component Tree is a JavaScript object that specifies the abstract properties of the View to be rendered and the behaviour of its components, independent from the rendering technology to be used.

For example, if your UI is going to allow users to select from a number of choices, your component tree will define the choices, but will be independent of whether or not the selection is rendered using checkboxes, a pull-down list, or some other method.

The Renderer includes a number of utility functions to generate component trees, but it is useful to have an understanding of the structure and purpose of the component tree. This page attempts to provide that understanding.

## Structure

A component tree is a tree of objects that represent the data to be rendered. The root node of the tree is typically an array of objects representing nodes in the tree, called `children`:

```
var myTree = {
  children: [
    {...},
    {...},
    ...
  ]
};
```

**On This Page**

**See Also**

- Renderer
- Renderer Templates
- Renderer Decorators
- Renderer API
- Fluid Renderer - Background

**Still need help?**

Join the infusion-users mailing list and ask your questions there.

Each component, or node in the component tree, is identified by an ID (more on these IDs fluid:later):

```
var myTree = {
  children: [
    {ID: "thing1",
     ...},
    {ID: "thing2",
     ...},
    ...
  ]
};
```

## Component Types

The Renderer categorizes component into different types depending on the nature of the data that is to be rendered. Different component types will have different fields, but in general, the tree for a component will contain either the actual data or a reference to a data model containing the data (see Renderer Data Binding for more information). The component types are not stated explicitly in the object, but the type is inferred by the Renderer based on the presence of the require fields.

For detailed information about Component types, see Renderer Component Types.

### IDs

Each component in the component tree has an ID. These are historically called RSF IDs because the Fluid Renderer is based on Reasonable Server Faces (RSF).

The simplest way to map the component tree onto an HTML template is to add the corresponding IDs to the relevant HTML elements using an `rsf:id` attribute:

Component tree snippet:

```
var myTree = {
  children: [
    {
      ID: "myDropDown",
      selection: "med",
      optionlist: ["xs", "s", "med", "l", "xl"]
      optionnames: ["Extra Small", "Small", "Medium",
                    "Large", "Extra Large"]
    }
    ....
]};
```

HTML template snippet:

```
....
<select rsf:id="myDropDown">
</select>
....
```

Alternatively, if you prefer not to place `rsf:id` attributes into the HTML template, you can map selectors to rsf ids.
This will require a selector map, which is an array of objects having selector and id keys. These objects are typically called "cutpoints" and you will see this name in the API documentation. This cutpoints array is passed into the renderer via the `cutpoints` option of the render function.

Component tree snippet:

```
var myTree = {
    children: [
        ...
        {
          ID: "myDropDown",
          selection: "med",
          optionlist: ["xs, "s", "med", "l", "xl"]
          optionnames: ["Extra Small", "Small",
"Medium",
                        "Large", "Extra Large"]
        }
        ...

var myCutpoints = [
    {id: "myDropDown", selector: ".dropDownBox"}
];
var myOptions = {
    cutpoints: myCutpoints
};
fluid.selfRender(renderContainer, myTree, myOptions);
```

HTML template snippet:

```
<select class="dropDownBox">
</select>
```

The format of an rsf:id gives the renderer guidance as to how to treat the element:

- if an `rsf:id` ends with a colon (e.g. `my-table-row:`), the element is a repeating element. That is, the renderer will duplicate the HTML element as many times as is necessary to display the data.