ProtoComponent Types

The Renderer categorizes components 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 values of the fields will contain either the actual data or a reference to a data model containing the data. The component types are not stated explicitly in the object, but the type is inferred by the Renderer based on the presence of particular fields.

The following tables describe the different types of components and the fields used by each component. In these tables, field names shown in bold text are the definitive fields that will indicate which type of component is being described.

Type	Description	Format	Example
Bound	A control which holds a single value, such as headers, labels, etc.	`componentID: { fieldName: valueOrBinding }`	`var protoTree = { mainHeader: "Carving Woods", sectionHeader1: "Sassafras", sectionHeader1: "Butternut", sectionHeader1: "Basswood" };`
Array of Bound	repeated Bound fields	`componentID: { fieldName: [valueOrBinding1, valueOrBinding2, ...] }`	`var protoTree = { mainHeader: "Carving Woods", sectionHeaders: ["Sassafras", "Butternut", "Basswood"] };`
Selection	A selection control where a user chooses either one or many options from a set of alternatives, such as a drop-down	`componentID: { selection: valueOrBinding, optionlist: [array of internal values], optionnames: [array of display strings] }` For information on how to create trees for radio buttons and checkboxes, see Renderer Component Tree Expanders.	`var protoTree = { contact-addressType1: { selection: "${fields.addressType1}", optionlist: ["Home", "Work"], optionnames: ["home", "work"] } };`
Link	A reference to a URL, such as a hyperlink	`componentID: { target: destinationUrl, linktext: stringToDisplay }`	`var protoTree = { contact-addressType1: { target: "http://company.com/help/${topic.url}", linktext: "${topic.name}" } };`

Container	A component that contains other components in a free-form way	`componentID: {`
		`  children: [array of other protocomponents]`
		`}`	

Message	A component that encapsulates the data needed to resolve a localised message. Similar to a Bound, but the value is a key into a string bundle	`componentID: {`
		`  messageKey: key,`
		`  args: [array of arguments to`
		`  be interpolated into`
		`  the message format]`
		`}`
		
		See fluid.formatMessage for more information about message formatting.
			`var protoTree = {`
			`  instructions: {`
			`  messageKey: "instructionKey",`
			`  args: ["thing", 3, "%path1"]`
			`  }`
			`};`