IoCSS

On This Page

- Background
- distributeOptions format
- IoCSS Selectors
- Example: source
- Example: record

distributeOptions and IoCSS: Downward-Matching CSS-Like Context Selectors For Options Forwarding

The `distributeOptions` option is a top-level block supported by every IoC-configured component. It specifies how options should be distributed to subcomponents further down the tree.

Background

As component trees become larger, it will often happen that a high-level component will need to specify options for a component further down the component tree. Without `distributeOptions`, component configuration can become very, what we might call, "pointy":

```
// without developer's use of IoCSS, user writes

fluid.uiOptions(".my-uio-container", {
    components: {
        templateLoader: {
            options: {
                templatePrefix: "../../myTemplates"
                }
            }
        }
    }
);
```

The example above shows a simplified version of a situation with Infusion's "UI Options" component. In practice, the user of the component would have to write an even more deeply nested piece of configuration than this, if the developer had not made use of the `distributeOptions` directive in the component's options block, together with the use of "IoCSS" expressions to distribute the user's options to the right place in the component tree, as shown in the following example:

```
// developer writes:

fluid.defaults("fluid.uiOptions", {
    distributeOptions: {
        source: "{that}.options.templatePrefix",
        target: "{that > templateLoader}.options.templatePrefix"
    }
});

// user writes:

fluid.uiOptions(".my-uio-container", {
    templatePrefix: "../../myTemplates"
});
```

In the `distributeOptions` block above, the context `{that > templateLoader}` is an IoCSS expression which designates one or more of the child components of UI Options that are to receive the user's option. The syntax and meaning of these expressions is defined below.

As well as converting the exposed options structure of a component into a more compact form, `distributeOptions` is also a powerful tool for maintaining API stability for a component or family of components. Since the binding of IoCSS selectors such as `that > templateLoader` onto child components is flexible, the component tree could be refactored in quite an aggressive way without requiring changes in either the user's configuration, or even the `distributeOptions` block itself. If the refactoring was even more thorough (involving wholesale removal of the target component, or a change in its important grades), the developer could still maintain stability of the external user API just by changing the `distributeOptions` block. In terms of a standard discussion on Design Patterns, the use of `distributeOptions` could be seen as an automated and declarative scheme for achieving the ends of the Facade Design Pattern, without the need for either user or developer code.

`distributeOptions` format

The `distributeOptions` option is a top-level block supported by every IoC-configured component, holding an array of objects (or single object) containing the following properties:

Name	Description
`target`	(Required) An IoC expression describing the location in the component tree where the options are to be set. The "context" part of this expression will usually consist of an IoCSS selector (see below for format). However, it may also in specialised cases consist of a standard "upwards" IoC context expression, indicating that options are to be distributed to a parent component (this is only meaningful in the case the parent has not finished instantiating).
`source`	(Mutually exclusive with `record`) An IoC expression into the options structure of the source component, referencing what to copy to the target. If target's context is an IoCSS selector, the source must be the current component, i.e. `that`.
`record`	(Mutually exclusive with `source`) A record of options to set at the target.
`removeSource`	(Only possible if `source` is used) true/false: If true, the `source` options block is removed from its original site in the options structure when it is forwarded to the `target`.
`exclusions`	(Only possible if `source` is used) A list of EL paths into the source material which should not be forwarded. Whether or not `removeSource` is used, these will be retained in their original position in the source component's options.

IoCSS Selectors

Component matching rules:

Form	Description
`*`	matches any component - universal selector
`E`	matches any component holding a context name of E - special support for the string "that" as with standard IoC context matching
`E#myid`	matches any component with a context name of E with id equal to myid (of no use to developers since component `ids` cannot be predicted)

Descendent rules:

Form	Description
`E F`	Matches any F component that is a descendant of an E component
`E > F`	Matches any F component that is a direct child of an E component

Example: `source`

```
fluid.defaults("fluid.tests.uploader", {
    gradeNames: ["fluid.littleComponent", "autoInit"],
    components: {
        uploaderContext: {
            type: "fluid.progressiveCheckerForComponent",
            options: {componentName: "fluid.tests.uploader"}
        },
        uploaderImpl: {
            type: "fluid.tests.uploaderImpl"
        }
    },
    distributeOptions: [{
        target: "{that > uploaderImpl}.options" // Target a directly nested component matching the context
"uploaderImpl"
        source: "{that}.options",               // Distribute ALL of our options there, except exclusions:
        exclusions: ["components.uploaderContext", "components.uploaderImpl"], // options targetted directly at
these subcomponents are left undisturbed in place

    }],
    progressiveCheckerOptions: {
        checks: [{
            feature: "{fluid.test}",
            contextName: "fluid.uploader.html5"
        }]
    }
});
```

Example: `record`

```
fluid.defaults("fluid.moduleLayoutHandler", {
    gradeNames: ["fluid.layoutHandler", "autoInit"],
    ...
    distributeOptions: {
        target: "{reorderer}.options", // unusual: upward-matching selector distributes options back to parent
before instantiation ends
        record: {
            selectors: {
                movables: {
                    expander: {
                        func: "{that}.makeComputeModules",
                        args: [false],
                    }
                },
                dropTargets: {
                    expander: {
                        func: "{that}.makeComputeModules",
                        args: [false],
                    }
                },
                selectables: {
                    expander: {
                        func: "{that}.makeComputeModules",
                        args: [true],
                    }
                }
            }
        }
    }
});
```