Article
Pravin Barton · Jan 13, 2017 2m read

Binding of the "this" variable in Zen custom components

I thought I'd share some issues I had today with Zen development in a custom component. Say we have created a Zen composite component with a few javascript methods that modify its child components. In order to modify these components, we can use something like the following method:

```
ClientMethod exampleFunction() [ Language = javascript ]
{
   var component = this.getChildById('Control1');
   component.setValue('');
}
```

In JavaScript methods on the client, *this* refers to the current component or page. The equivalent for expressions that are values of XML attributes is *zenThis* (see documentation).

However, say that we want a time interval before the method gets called. We can create a wrapper method that uses the Javascript setTimeout() method.

```
ClientMethod exampleTimer() [ Language = javascript ]
{
   setTimeout(this.exampleFunction, 1000);
}
```

If we run exampleTimer(), we get the following error:
TypeError: this.getChildById is not a function

Why can't we call getChildById? The problem is with object binding. Here's a good explanation of the issues with Javascript binding by Christophe Porteneuve. When we call a method directly through its object, *this* is bound to that object (in this case, the composite component). But when we pass the method into setTimeout(), it's being called indirectly as a reference. As a result we experience a binding loss. The method is no longer bound to its parent object, and *this* reverts to referencing the window object.

A possible solution is to use a zenPage method to get a reference to the composite component. But this defeats the purpose of a custom component, which is to encapsulate code in a black box. We shouldn't need to know how to locate the component on the Zen page. The Porteneuve article includes a few solutions, including explicitly specifying binding with the apply method. In my case, the easiest solution was to avoid binding by passing in the component as an argument.

```
ClientMethod exampleFunction(composite) [ Language = javascript ]
{
   var component = composite.getChildById('Control1');
   component.setValue('');
}

ClientMethod exampleTimer() [ Language = javascript ]
{
   setTimeout(this.exampleFunction, 1000, this);
}
```

#JavaScript #ZEN #Caché