pui.widgets.add(config)

This function creates a new base widget from scratch. The implementation of the widget will require advanced JavaScript coding.

Parameter:

- config object - JavaScript object containing configuration properties

Configuration Options:

- name - the new base widget name; also referred to as the "field type"
- newId - the default id prefix - used to assign id's when the element is first created; for example if the newId is "Map", new elements created will have id's of Map1, Map2, Map3, etc.
- menuName - pull-down menu name for the widget
- defaults - an object with name/value pairs for default property settings for this base widget; these values can overridden by the pui.toolbox.add() function
- propertySetters - an object representing a set of functions that are triggered when a particular widget property is changed; the "field type" function can be used as the main function to construct the widget; each function will receive a configuration parameter with the following properties:
    - originalValue - original value before the property was changed
    - newValue - new value (before scripting evaluation)
    - value - new value (after scripting evaluation)
    - properties - object representing the current property names and values for the widget
    - dom - reference to the main DOM element for the widget
    - oldDom - reference to the old DOM element (this will only be different from dom if the change to the widget property required the dom element to be changed or recreated)
    - propertyName - the name of the widget property being changed
    - design - flag that specifies whether the widget property is being changed at design time or at run time
    - designItem - reference to the design item object for this widget
    - resizer - reference to the resizer object for this widget
    - evalProperty - function that can be used to evaluate scripted properties
- globalPropertySetter - a global function that is triggered every time any widget property is changed
- tag - the tag of the main DOM element; if omitted, "div" tag is assumed
- inputType - if the "input" tag is specified, this option specifies the input type, such as "text", "checkbox", "radio", "password", etc.
- resizable - if false is specified, the widget cannot be resized in the designer
- canBelongToGrid - if false is specified, the widget cannot be dragged into a subfile grid
- dependencies - Array of strings containing URLs pointing to JavaScript or CSS files. At run time, these will be loaded into your session (if not there already) only if your widget is used on the current format. At design time, they are loaded when the Visual Designer is loaded. *(Note that this configuration option is available starting with Profound UI Version 5, Fix Pack 8.0)*.

Example:

The following code creates a "google maps" base widget.

```
pui.widgets.add({
  name: "google maps",
  newId: "Map",
  menuName: "Google Maps Widget",

 // Scripts that are loaded only when this widget is first used.
  dependencies: ["/myproject/custom.css", "/myproject/custom.js",

    // A script that is loaded conditionally, only when the width property's value is 400px.
    {"script": "/myproject/custom2.js",
     "condition": function (properties, boundData, isDesigner){
        // The property is hard-coded in Designer or a Genie screen:
        if (typeof properties["width"] != "object") return properties["width"] == "400px";
        // The property is bound to an RPG field in a Rich Display File.
        else if (properties["width"] != null && typeof properties["width"]["fieldName"] ==
"string"){
          var value = boundData[ properties["width"]["fieldName"].toUpperCase() ];
          return value == "400px";
        }
        else if (isDesigner) return true; //Always load when in Visual Designer.
        return false;
      }
```

```
    }],

 // default properties for the widget
  defaults: {
    "width": "400px",
    "height": "300px",
    "z index": "25",
    "background color": "#FFFFFF"
  },

  // property setter functions
  propertySetters: {

    // this will fire when the field type property is set to "google maps"
    // all initialization code for the element goes here
    "field type": function(parms) {
      if (parms.design) {
        // all design mode content creation code goes here
        // add preview image
        var img = document.createElement("img");
        img.src = "/profoundui/proddata/images/GoogleMaps.jpg";
        img.style.position = "absolute";
        img.style.left = "0px";
        img.style.top = "0px";
        img.style.width = parms.dom.style.width;
        img.style.height = parms.dom.style.height;
        parms.dom.appendChild(img);
        parms.dom.style.overflow = "hidden";
      }
      else {
        // runtime content creation code goes here
        // clear
        parms.dom.innerHTML = "";
        // get url property - evalProperty will evaluate any js expressions that may have been
used to build the property
        address = parms.evalProperty("google maps address");
        // populate the widget's dom element
        if (address == null || address == "") {
          parms.dom.innerHTML = "Google Maps Address property not specified.";
        }
        else {
          // here, we build an iframe element and put it into the dom
          var widthHeight = "";
          var width = parseInt(parms.properties["width"]);
          var height = parseInt(parms.properties["height"]);
          if (width > 0) widthHeight += " width=" + width;
          if (height > 0) widthHeight += " height=" + height;
          var url = "http://maps.google.com/maps?hl=enl&amp;q=" + encodeURIComponent(address) +
"&amp;ie=UTF8&amp;hq=&amp;hnear=" + encodeURIComponent(address) + "&amp;output=embed";
          var html = "<iframe src=\"" + url;
          html += '" frameborder="0" scrolling="no" marginheight="0" marginwidth="0"';
          html += '"' + widthHeight + '>';
          parms.dom.innerHTML = html;
        }
      }
    },

    // this will fire when the width property of the element changes
    "width": function(parms) {
      if (parms.design) {
        // adjust preview image width
        parms.dom.firstChild.style.width = parms.value;
```

```
    }
  },

  // this will fire when the height property of the element changes
  "height": function(parms) {
    if (parms.design) {
      // adjust preview image height
      parms.dom.firstChild.style.height = parms.value;
    }
  }

}
```

```
});
```