# UI控件dat.GUI的使用

原文地址：<https://workshop.chromeexperiments.com/examples/gui/>（可能需翻墙）。

dat.gui是一个用于JavaScript的轻量级控件库。

## 1．基本用法

只需要使用少量代码，dat.GUI就可以创建一个用户界面。

<script type="text/javascript" src="dat.gui.js"></script>

<script type="text/javascript">

var FizzyText = function() {

this.message = 'dat.gui';

this.speed = 0.8;

this.displayOutline = false;

this.explode = function() { ... };

// Define render logic ...

};

window.onload = function() {

var text = new FizzyText();

var gui = new dat.GUI();

gui.add(text, 'message');

gui.add(text, 'speed', -5, 5);

gui.add(text, 'displayOutline');

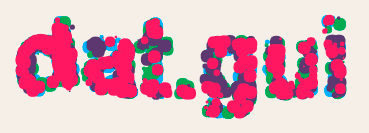
gui.add(text, 'explode');

};

</script>

* 属性必须是公有的，即this.prop = value。
* dat.GUI会根据属性的初始值确定控件的类型。
* 按下H键可以显示/隐藏所有控件。

注意：水平有限，不知道示例的代码是如何实现的，要看到本例以及其他9个例子的动态效果请至原地址查看。下面是效果的截图：



## 2．约束输入内容

你可以对输入的数值进行一些限制。若一个数值被设置了min或max值，则成为一个滑动条。

gui.add(text, 'noiseStrength').step(5); // Increment amount

gui.add(text, 'growthSpeed', -5, 5); // Min and max

gui.add(text, 'maxSize').min(0).step(0.25); // Mix and match

你也可以从下拉菜单中选择数值和字符串。

// Choose from accepted values

gui.add(text, 'message', [ 'pizza', 'chrome', 'hooray' ] );

// Choose from named values

gui.add(text, 'speed', { Stopped: 0, Slow: 0.1, Fast: 5 } );

## 3．文件夹

你可以将任意数量的控件归为一组，用户界面会表示为一个折叠的文件夹。

var gui = new dat.GUI();

var f1 = gui.addFolder('Flow Field');

f1.add(text, 'speed');

f1.add(text, 'noiseStrength');

var f2 = gui.addFolder('Letters');

f2.add(text, 'growthSpeed');

f2.add(text, 'maxSize');

f2.add(text, 'message');

f2.open();

## 4．颜色控件

dat.GUI包含一个颜色选择器，这个选择器可以识别不同的颜色表示，下面的代码展示了使用不同颜色格式的颜色选择器控件。

var FizzyText = function() {

this.color0 = "#ffae23"; // CSS字符串格式

this.color1 = [ 0, 128, 255 ]; // RGB数组格式

this.color2 = [ 0, 128, 255, 0.3 ]; // 包含alpha值的RGB数组格式

this.color3 = { h: 350, s: 0.9, v: 0.3 }; // Hue, saturation, value

// Define render logic ...

};

window.onload = function() {

var text = new FizzyText();

var gui = new dat.GUI();

gui.addColor(text, 'color0');

gui.addColor(text, 'color1');

gui.addColor(text, 'color2');

gui.addColor(text, 'color3');

};

dat.GUI会根据颜色初始的格式设置对应的值。

## 5．保存菜单设置

通过调用gui.remember方法可以在GUI界面上添加一个保存菜单，这样就可以保存菜单上的所有对象。

var fizzyText = new FizzyText();

var gui = new dat.GUI();

gui.remember(fizzyText);

// Add controllers ...

点击图标可以改变保存的设置，你可以将这些值保存到localStorage，也可以使用下面的代码将将一个JSON对象复制到源代码中：

var fizzyText = new FizzyText();

var gui = new dat.GUI({ load: JSON });

gui.remember(fizzyText);

// Add controllers ...

## 6．预设值

你可以将所有设置保存起来作为预设值。点击**Save**按钮可以修改当前的预设值，点击New按键可以创建一个新的预设值，点击Revert按钮会清除所有未经保存的更改。

使用下来菜单可以切换不同的预设值。以下代码可以设置默认的预设值：

var gui = new dat.GUI({

load: JSON,

preset: 'Flow'

});

使用localStorage要当心：最好能经常将保存的JSON数据粘贴到你的源码中。使用localStorage保存你的预设值非常方便且快捷，但它也很容易丢失，丢失的现象常常发生在以下场景中：清空浏览器数据，换用浏览器，甚至在修改当前网页的URL时也有可能发生。

## 7.事件

你可以通过一个事件监听器监听控件引发的事件。

var controller = gui.add(fizzyText, 'maxSize', 0, 10);

controller.onChange(function(value) {

// Fires on every change, drag, keypress, etc.

});

controller.onFinishChange(function(value) {

// Fires when a controller loses focus.

alert("The new value is " + value);

});

## 8. 自定义面板放置位置

默认情况下，dat.GUI面板使用的是固定位置，并自动添加到创建时的DOM元素。

将参数autoPlace设置为false，你就可以改变默认的设置。

var gui = new dat.GUI({ autoPlace: false });

var customContainer = document.getElementById('my-gui-container');

customContainer.appendChild(gui.domElement);

## 9. 自动更新显示

在控件上使用listen方法可以让它接收来自外界的变化。

var fizzyText = new FizzyText();

var gui = new dat.GUI();

gui.add(fizzyText, 'noiseStrength', 0, 100).listen();

var update = function() {

requestAnimationFrame(update);

fizzyText.noiseStrength = Math.random();

};

update();

由于每一帧都会修改对应的属性，因此往往会比较慢。

## 10．手动更新显示

如果你想手动控制控件的更新，可使用updateDisplay方法。

var fizzyText = new FizzyText();

var gui = new dat.GUI();

gui.add(fizzyText, 'noiseStrength', 0, 100);

var update = function() {

requestAnimationFrame(update);

fizzyText.noiseStrength = Math.cos(Date.getTime());

// Iterate over all controllers

for (var i in gui.\_\_controllers) {

gui.\_\_controllers[i].updateDisplay();

}

};

update();

## 示例

见网页：<http://shiba.hpe.sh.cn/jiaoyanzu/WULI/Article2324>。

### js代码

var canvas, ctx, w, h;

var x = 0, y = 0, width = 50, height = 50, fillColor = "#ff0000",

strokeColor = "#ffae23", shapeType = "椭圆", showBorder = true;

init();

requestAnimationFrame(animate);

function init() {

// 初始化canvas

canvas = document.getElementById("canvas");

w = canvas.width;

h = canvas.height;

ctx = canvas.getContext("2d");

ctx.lineWidth = 8;

var obj = {

message: '使用例子',

显示边框: true,

图形: '椭圆',

x坐标: x,

y坐标: y,

宽: width,

高: height,

绘制信息: function () {

if (shapeType == "椭圆")

alert("你在坐标(" + x + "," + y + ")绘制了一个长轴为" +

width + "、短轴为" + height + "的" + shapeType);

else if (shapeType == "矩形")

alert("你在坐标(" + x + "," + y + ")绘制了一个宽为" +

width + "、高为" + height + "的" + shapeType);

},

边框颜色: "#ffae23", // CSS string

填充颜色: "#ff0000"

};

var gui = new dat.GUI({ autoPlace: false });

var customContainer = document.getElementById('my-gui-container');

customContainer.appendChild(gui.domElement);

gui.add(obj, 'message');

gui.add(obj, '显示边框').onChange(function (e) {

showBorder = e;

});

gui.add(obj, '绘制信息');

gui.add(obj, 'x坐标').min(-200).max(200).step(5).onChange(function (e) {

x = e;

});

gui.add(obj, 'y坐标').min(-200).max(200).step(5).onChange(function (e) {

y = e;

});

gui.add(obj, '宽').min(20).max(100).step(5).onChange(function (e) {

width = e;

});

gui.add(obj, '高').min(20).max(100).step(5).onChange(function (e) {

height = e;

});

gui.add(obj, '图形', ['椭圆', '矩形']).onChange(function (e) {

shapeType = e;

});

var f1 = gui.addFolder('颜色');

f1.addColor(obj, '边框颜色').onChange(function (e) {

strokeColor = e;

});

f1.addColor(obj, '填充颜色').onChange(function (e) {

fillColor = e;

});

}

function drawShape() {

ctx.save();

ctx.fillStyle = fillColor;

if (showBorder) {

ctx.strokeStyle = strokeColor;

ctx.stroke();

}

if (shapeType == "矩形") {

// 清空画圆可能带来的线条

ctx.beginPath();

ctx.closePath();

ctx.fillRect(x - width / 2, y - height / 2, width, height);

if (showBorder)

ctx.strokeRect(x - width / 2, y - height / 2, width, height);

}

else if (shapeType == "椭圆") {

ctx.translate(x, y);

ctx.scale(width, height);

ctx.beginPath();

ctx.arc(0, 0, 1, 0, Math.PI \* 2, true);

ctx.closePath();

ctx.fill();

}

ctx.restore();

}

function render() {

ctx.clearRect(0, 0, w, h);

ctx.save();

ctx.translate(w / 2, h / 2);

ctx.scale(1, -1);

drawShape();

ctx.restore();

}

function animate(time) {

requestAnimationFrame(animate);

render();

}