

CACTUSCREW DEVELOPPEMENT WEB

Var_dump en JS

Voici un outil pratique pour debugger en JS, et qui fonctionne sur la plupart des navigateurs.

Il est plus qu'utile de pouvoir farfouiller récursivement dans nos variables en JS.
3 formes possibles :

alert_r(elt); affiche dans une alert JS le contenu de la variable.

document_r(elt); dans le body de la page.

window_r(elt); dans un popup.

Il est possible de spécifier un second argument définissant la profondeur de récursivité (2 par défaut).

Ci dessous, le code réalisé par mes soins, donc si vous trouvez un bug (huhu), prévenez moi :
(testé avec succès sur Firefox, IE, et Opéra)

```
function debugJS() {  
  
    /**  
     * tabulation  
     */  
    this.tabul = '  ';  
  
    /**  
     * fonctions  
     */  
    this.dumpJS = dumpJS;  
  
    /**  
     * @desc decompose récursivement un element  
     * @param mixed elt element a decomposer  
     * @param int max nombre maxi de recurances  
     * @param string S_tab suivi des tabulations  
     * @param int rec suivi de reccursion  
     */  
    function dumpJS(elt, max, S_tab, rec) {  
        if (max == undefined) {  
            max = 2;  
        }  
    }  
}
```

```

}
rec++;
var S_result = "";
if (elt == 'undefined') {
    return "undefined";
}
switch (typeof elt) {
case 'object' :
    S_result += "object {\n";
    if (rec < max) {
        for (myl in elt) {
            try {
                S_result += S_tab + this.tabul + '[' + myl + '] => '
                S_result += this.dumpJS(elt[myl], max, S_tab + this.tabul, rec);

            } catch (e) {
                S_result += S_tab + this.tabul + '[' + myl + '] => ' + "*** ERROR **\n";
            }
        }
    }
} else {
    S_result += S_tab + this.tabul + "*** MAX RECURSION **\n";
}
S_result += S_tab + "}\n";
break;

case 'string' :
    S_result += typeof elt + ' "' + elt + "\"\n";
    break;

default :
    S_result += typeof elt + '(' + elt + ")\n";
    break;
}
return S_result;
}

}

___O_debugJS = new debugJS();
/**
 * @desc decompose récursivement un element et affiche une alerte
 * @param mixed elt element a decomposer
 * @param int max nombre maxi de recurances
 */
function alert_r(elt, max) {
    alert(___O_debugJS.dumpJS(elt, max, "", 0));
}

```

```

}

/**
 * @desc decompose récursivement un element et affiche dans le body
 * @param mixed elt element a decomposer
 * @param int max nombre maxi de recurrences
 */
function document_r(elt, max) {
    document.write('<pre>');
    document.write(____O_debugJS.dumpJS(elt, max, "", 0));
    document.write('</pre>');
}

/**
 * @desc decompose récursivement un element et affiche une nouvelle fenetre
 * @param mixed elt element a decomposer
 * @param int max nombre maxi de recurrences
 */
function window_r(elt, max) {
    win = window.open("", 'format',
'width=400,height=300,left=50,top=50,status,menubar,scrollbars,resizable');
    win.document.write('<pre>' + ____O_debugJS.dumpJS(elt, max, "", 0) + '</pre>');
    win.focus();
}

```

Les tests

```

/** tableau d'essai */
A_tablo = new Array();
A_tablo.push(10.6);
A_tablo.push(11);
A_tablo.push(12);
A_tablo.tablo = new Array();
A_tablo.tablo.tablo2 = new Array();
A_tablo.tablo.tablo2.tablo3 = new Array();
A_tablo.tablo.tablo2.tablo3.tablo4 = new Array();
A_tablo.tablo.tablo2.tablo3.tablo4.tablo5 = new Array();
A_tablo.tablo.tablo2.push('TADA');
A_tablo.tablo.tablo2.tablo3.push('TADA');
A_tablo.tablo.tablo2.tablo3.tablo4.push('TADA');
A_tablo.tablo.tablo2.tablo3.tablo4.tablo5.push('TADA');

alert_r(A_tablo, 2);
document_r(A_tablo, 10);
window_r(A_tablo, 2);

```

Voir un exemple vivant -->ici

Commentaires

2007-02-01 12:35:10 - NiKo - nperrault@gmail.com - <http://www.prendreuncafe.com>

Très interessant !

2007-04-11 09:59:38 - Phil

bravo, super efficace. Je me suis empressé de l'utiliser avec BONHEUR et d'intégrer ça à mes classes. Merci !!!