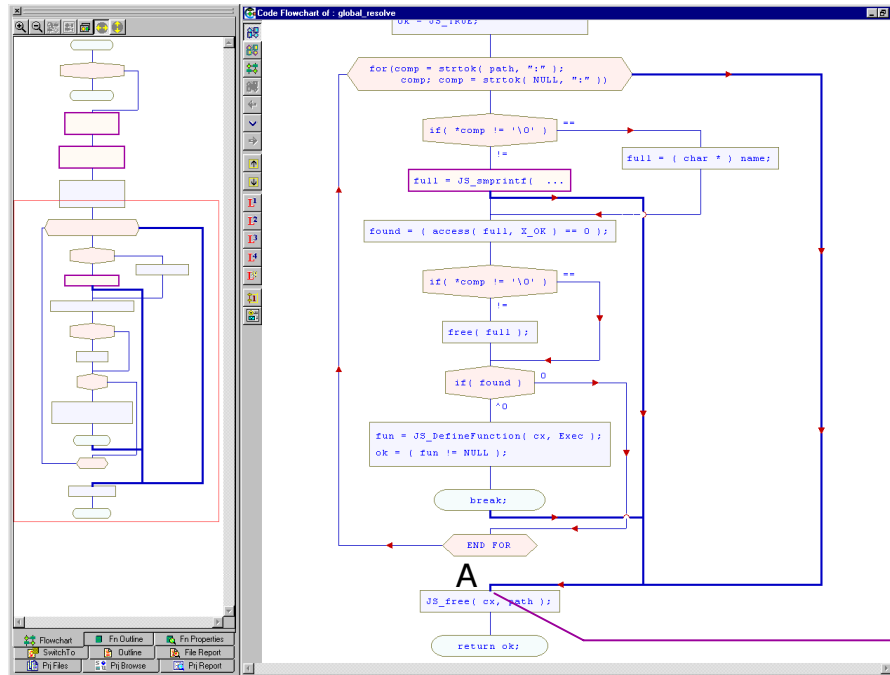


## Understand Any Function in less time



← With the flowchart, you can understand this 50-line function in half the time.

How do you find all paths that go to point A in the function?

Answer:

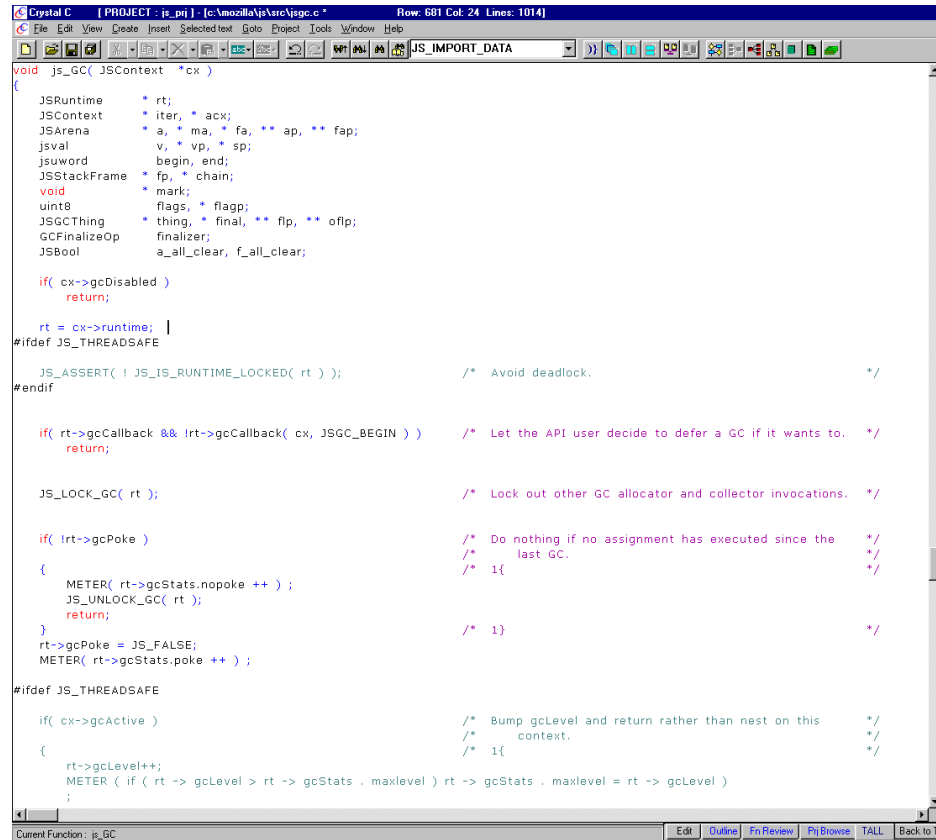
Click on point A in the flowchart.

Crystal C/C++ automatically creates the flowchart of any function.

*If you do not have flowcharts,*

- how do you find all such paths?
- or examine them during code-review?

## How to Attack a Very Long Function



```
void js_GC( JSContext *cx )
{
    JSRuntime      *rt;
    JSContext      *iter, *acx;
    JSArena        *a, *ma, *fa, **ap, **fap;
    jsval          v, *vp, *sp;
    jsuword        begin, end;
    JSStackFrame   *fp, *chain;
    void           *mark;
    uint8          flags, *flagp;
    JS GCThing     *thing, *final, **flp, **oflp;
    GCFinalizeOp   finalizer;
    JSBool         a_all_clear, f_all_clear;

    if( cx->gcDisabled )
        return;

    rt = cx->runtime;
#ifdef JS_THREADSAFE
    JS_ASSERT( ! JS_IS_RUNTIME_LOCKED( rt ) ); /* Avoid deadlock. */
#endif

    if( rt->gcCallback && !rt->gcCallback( cx, JSGC_BEGIN ) ) /* Let the API user decide to defer a GC if it wants to. */
        return;


    JS_LOCK_GC( rt ); /* Lock out other GC allocator and collector invocations. */

    if( !rt->gcPoke ) /* Do nothing if no assignment has executed since the last GC. */
    { /* 1{ */
        METER( rt->gcStats.nopoke ++ );
        JS_UNLOCK_GC( rt );
        return;
    } /* 1} */
    rt->gcPoke = JS_FALSE;
    METER( rt->gcStats.poke ++ );

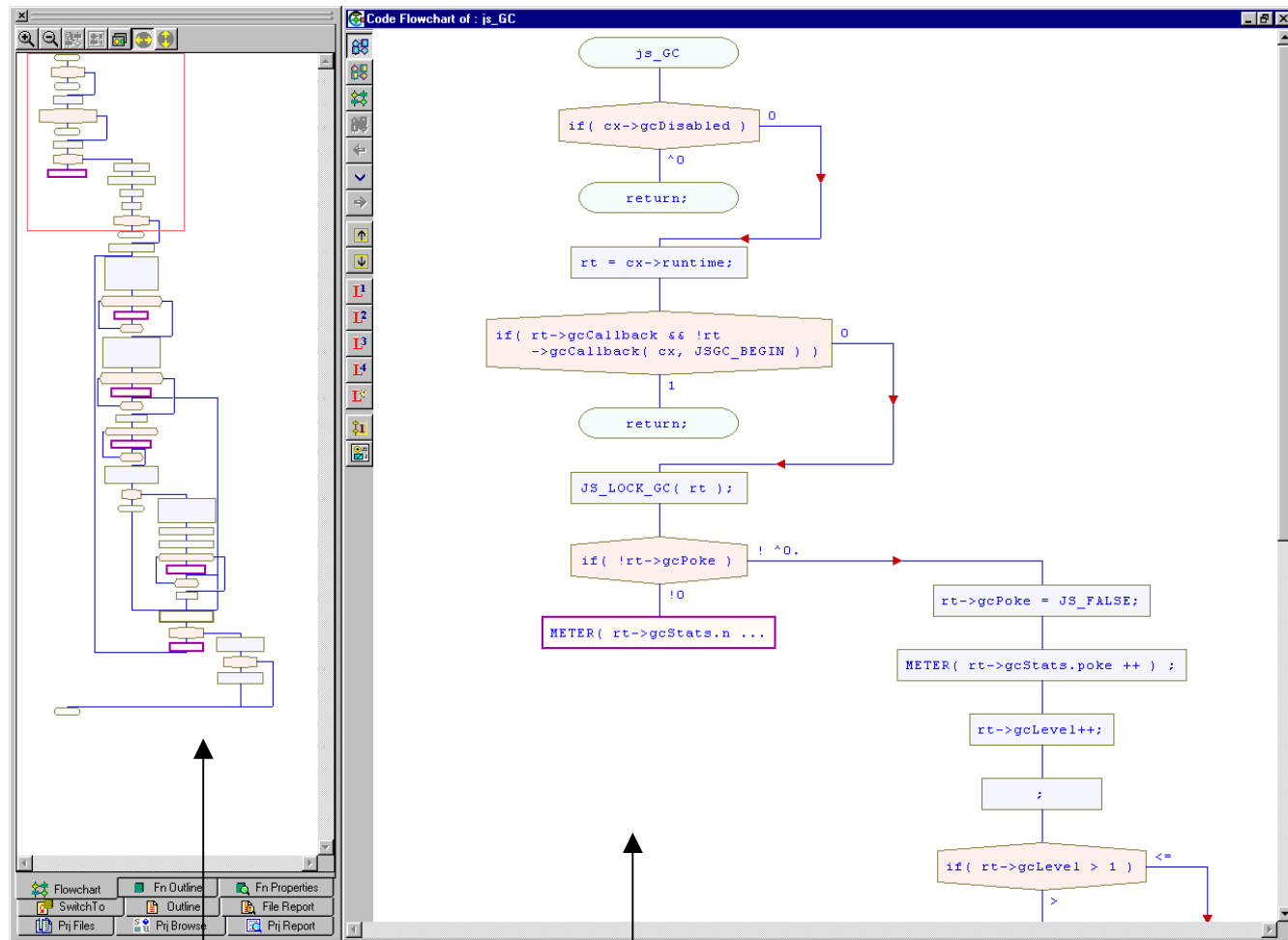
#ifdef JS_THREADSAFE
    if( cx->gcActive ) /* Bump gcLevel and return rather than nest on this context. */
    { /* 1{ */
        rt->gcLevel++;
        METER ( if ( rt -> gcLevel > rt -> gcStats . maxlevel ) rt -> gcStats . maxlevel = rt -> gcLevel )
        ;
    }
}
```

← This is `js_GC`, a 350-line function from Mozilla source code. ( seven pages of your window)

How would you review and understand this function's code?

Answer: Click the flowchart icon .

## Understand the Top-Level Logic in less than 5 minutes



For large functions, Crystal C/C++ automatically creates a top-level flowchart.

← Top-level flowchart of the function `js_GC`.

You will need **less than five minutes** to go through this flowchart.

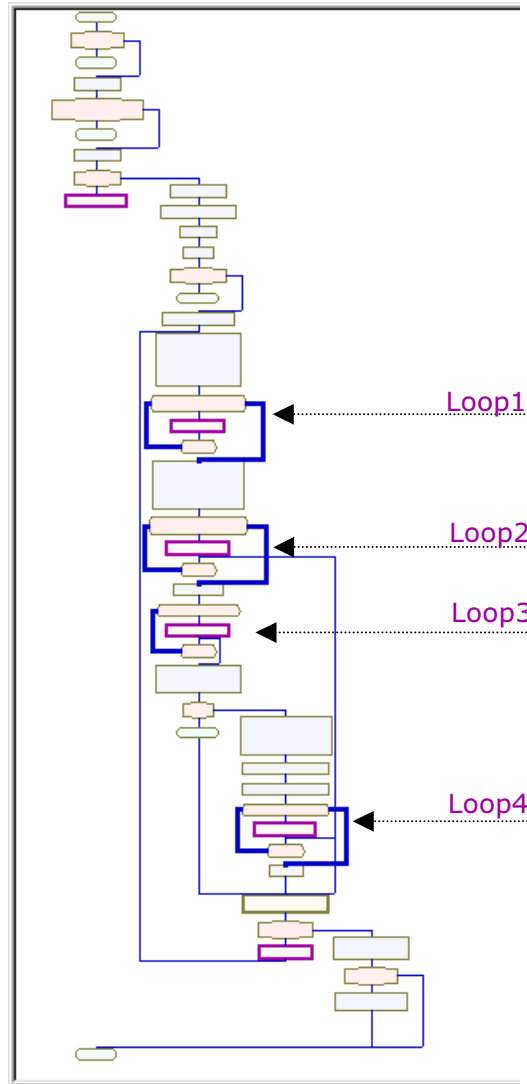
*If you do not have flowcharts, how do you construct the top-level view of a long function?*

## Divide and Conquer:

You just viewed the top-level flowchart;

**Next:** Flowcharts of the Inner Parts.

Click on a loop or switch symbol to create its flowchart.

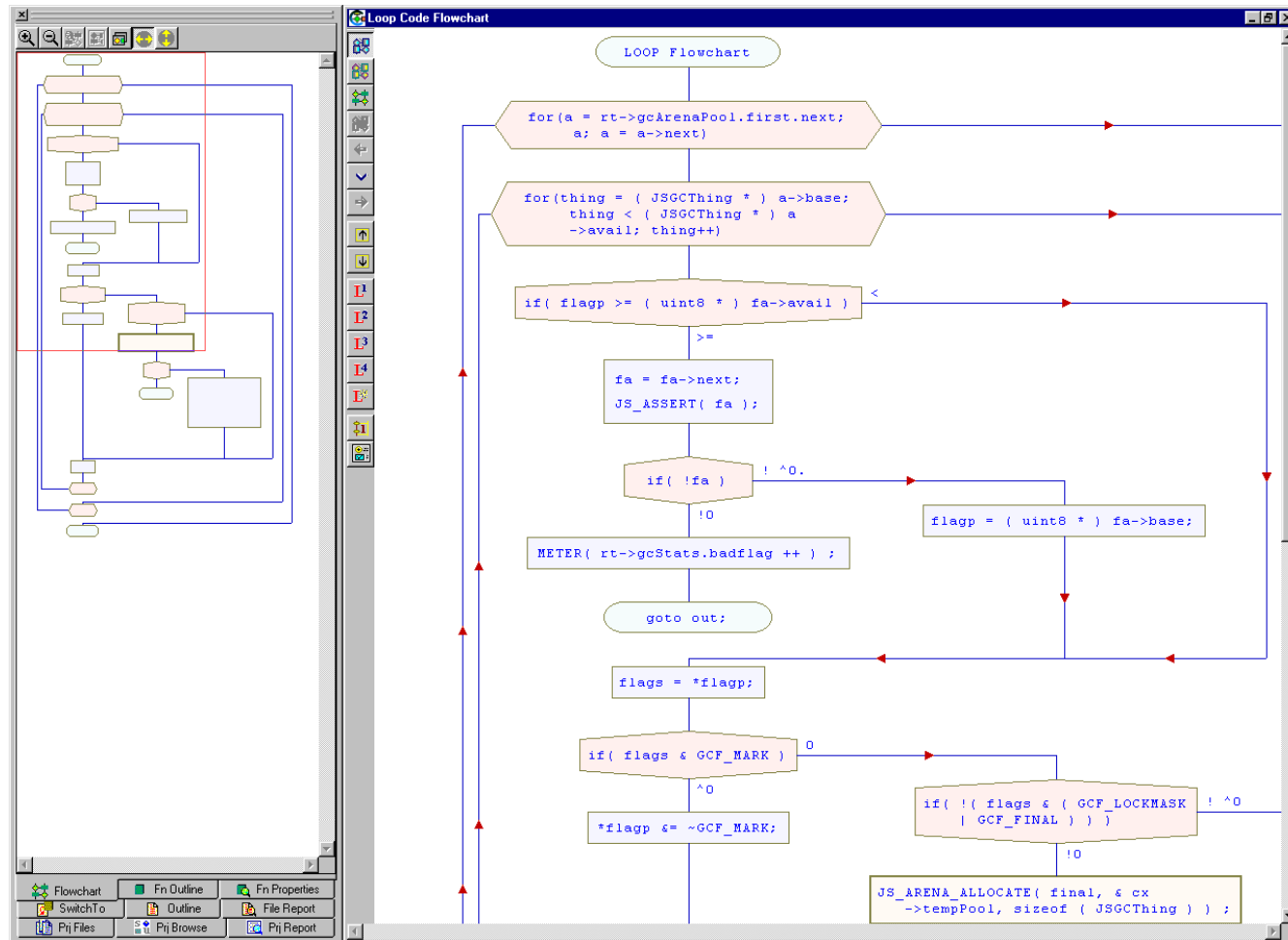


← The Major Loops in js\_GC

A purple outline  indicates a high-level symbol.

It hides the internal details of a loop, switch etc.

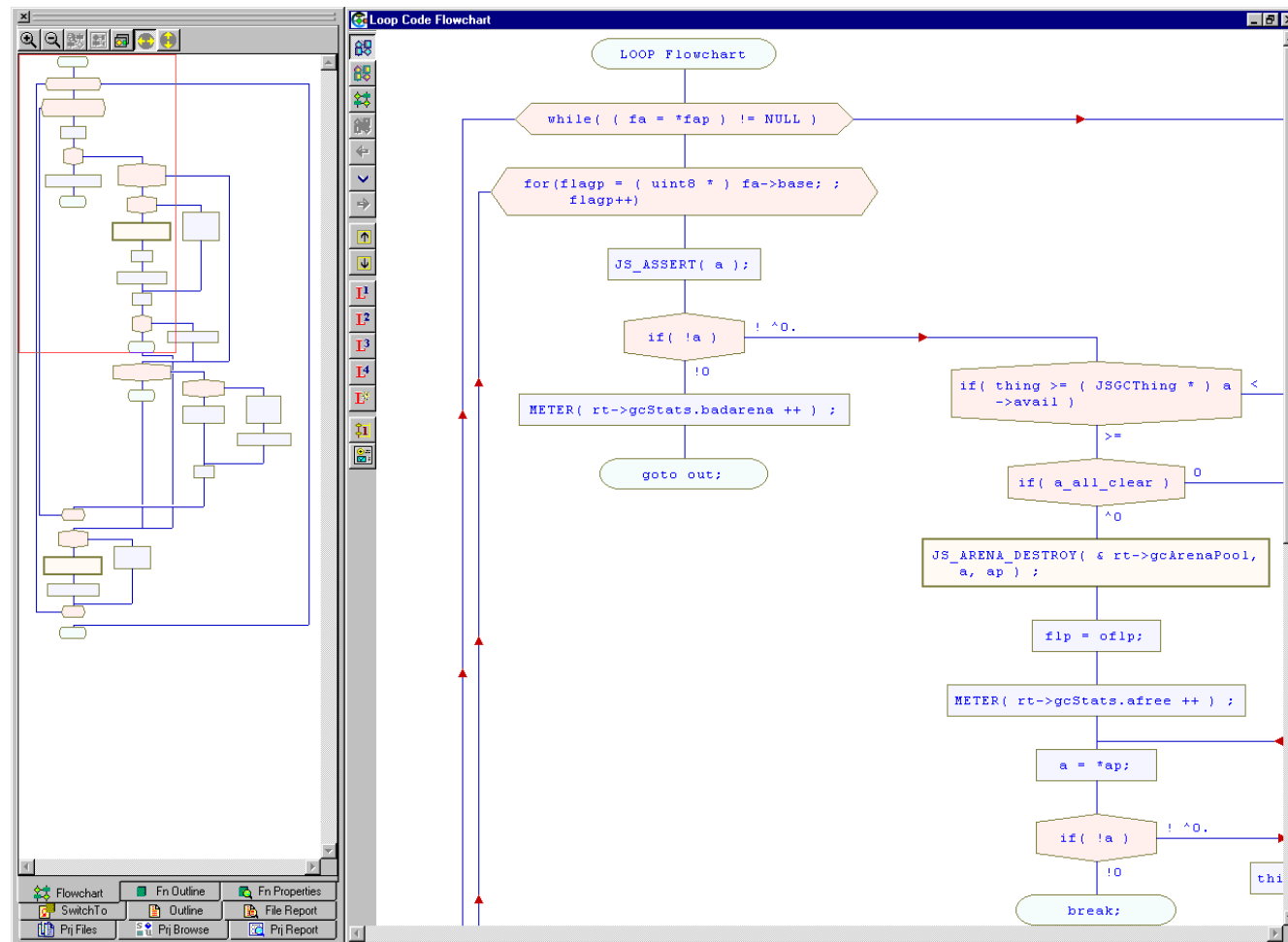
## The 1st Major Loop in js\_GC



← Flowchart of 1st major loop in js\_GC

You will need just two minutes to understand this loop.

## The Next Major Loop in js\_GC



You will need  
less than five minutes  
to understand this loop.

The graphical view is  
easy to understand and  
easy to recall.

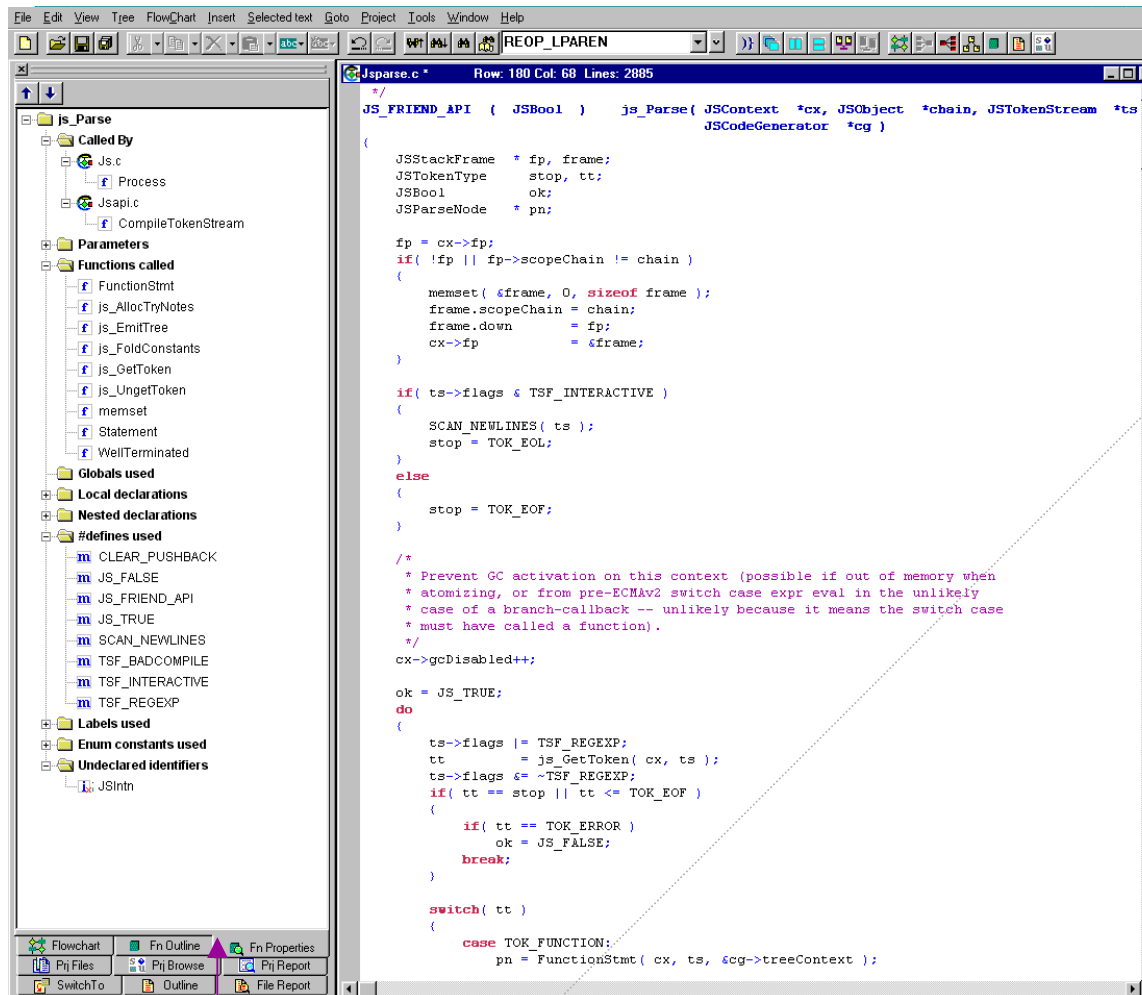
As we saw above, in about twenty minutes, you can clearly understand a 350-line function. Without the flowcharts, it will take more than one hour.

### Flowcharts are Valuable in many situations

1. after designing or modifying code – view the whole function;  
make sure all conditions are covered.
2. during code-reviews, the whole team saves a lot of time
3. during integration testing, you can easily understand other team members' code.
4. when you inherit legacy code.
5. when you are a new member of the team.

You can export a flowchart as a .bmp or .jpg file for documentation.

## Function Properties - Called By, Functions Called, Globals Used etc.



◆ Place the cursor in a function.

Click the "Fn Properties" tab in the browse-window.

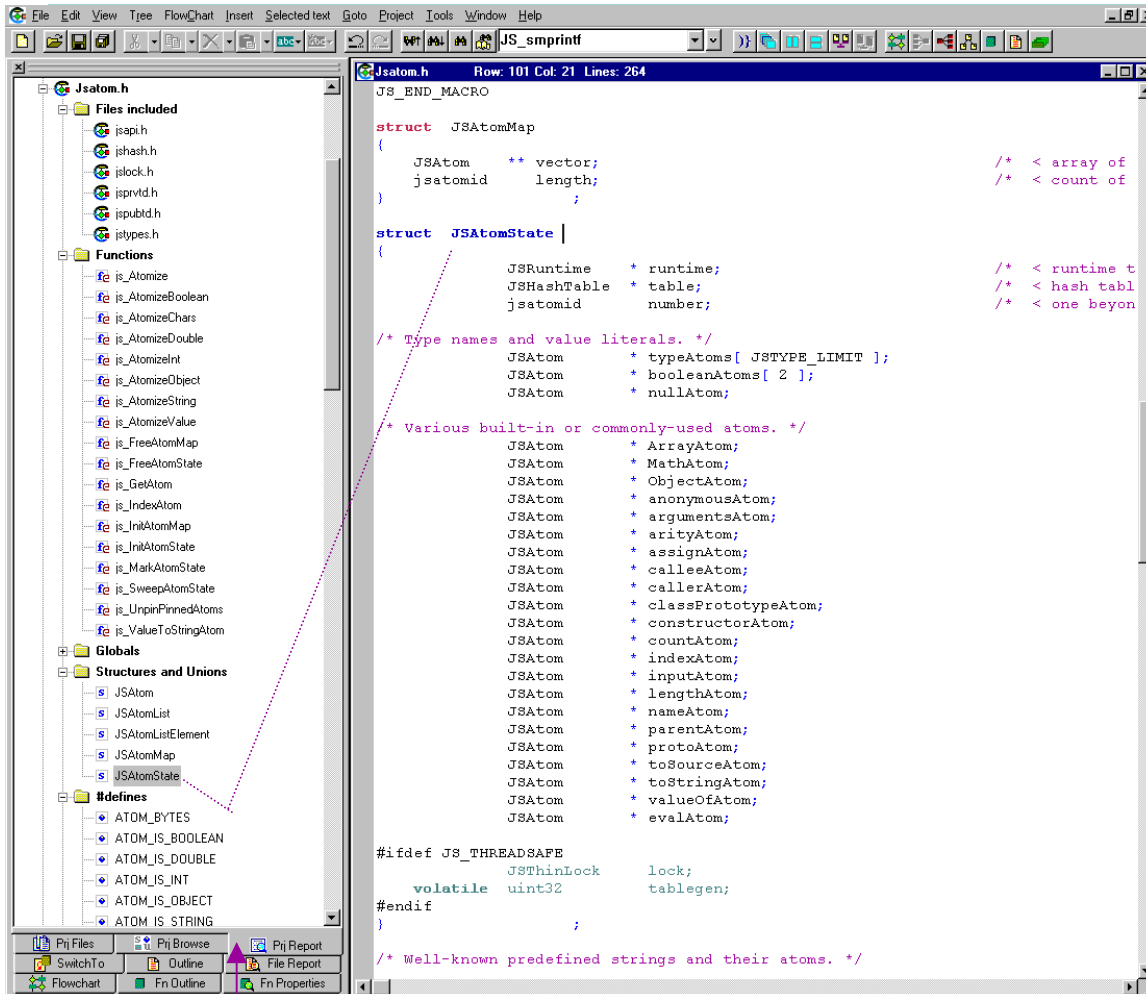
◆ Double-click on a node.

Also, try right-click on a node.



## Overview of Globals, Functions, Structs etc.

from each Source File



Project Report

Get an overview of source files:

- a file-by-file report of all the globals in the project;
- all the functions in the project,
- all the structs in the project, etc.

To go to the definition of a global or a function :

double-click on an entry in the overview report.

## Software Metrics

Crystal REVS for C Documentation - js\_REVS\_07-14-04.spr - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit RealGuide

Address [C:\MOZILLA\SRV\js\\_REVS\\_07-14-04\html\docs\js\\_REVS\\_07-14-04.spr.html](C:\MOZILLA\SRV\js_REVS_07-14-04\html\docs\js_REVS_07-14-04.spr.html) Go Links

Search Web ...attempting to retrieve buttons from Yahoo!

**js\_REVS\_07-14-04.spr**

[Overview](#)  
[Metrics](#)  
[Documentation](#)  
[Structs/Unions](#)  
[All Files](#)  
[Cross Reference](#)

**Metrics**

[Whole Project](#)

File-wise

by Volume

[Lines](#)  
[Control-Flow](#)  
[Declarations](#)  
[Identifiers](#)

by Complexity

[Halstead's](#)  
[McCabe's](#)

Function-wise

by Volume

[Lines](#)  
[Control-Flow](#)  
[Declarations](#)  
[Identifiers](#)

by Complexity

[Halstead's](#)  
[McCabe's](#)

**McCabe's Complexity Metrics : js\_REVS\_07-14-04.spr**

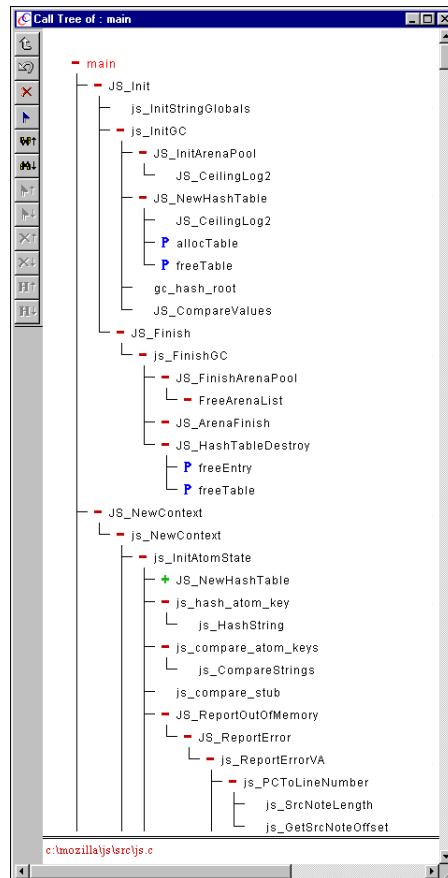
File	Lines LoC	Case LBLs	Funs	Complexities			Densities			Essential ed(G)
				Cyclomatic v(G)	Modified Cyclomatic	Design iv(G)	Essential ev(G)	Cyclomatic vd(G)	Design id(G)	
<a href="#">Js.c</a>	515		31	123	124	49	90	0.24	0.40	0.73
<a href="#">Jsaddr.c</a>	12		4	4	4		4	0.33		1.00
<a href="#">Jsaddr.h</a>	7		4	4	4		4	0.57		1.00
<a href="#">Jsapi.c</a>	1338		178	366	368	109	301	0.28	0.30	0.82
<a href="#">Jsapi.h</a>	365									
<a href="#">Jsarena.c</a>	235		13	47	47	20	41	0.20	0.43	0.87
<a href="#">Jsarena.h</a>	64									
<a href="#">Jsarray.c</a>	678		32	235	236	54	201	0.35	0.23	0.85
<a href="#">Jsarray.h</a>	13									
<a href="#">Jsatom.c</a>	393		29	101	101	33	93	0.26	0.33	0.92
<a href="#">Jsatom.h</a>	133									
<a href="#">Jsbit.h</a>	12									
<a href="#">Jsbool.c</a>	118		8	28	28	7	24	0.24	0.25	0.86
<a href="#">Jsbool.h</a>	7									
<a href="#">Jsclist.h</a>	23									
<a href="#">Jsctxt.c</a>	223		9	51	51	31	28	0.23	0.61	0.55
<a href="#">Jsctxt.h</a>	118									
<a href="#">Jscompat.h</a>	11									
<a href="#">Jsconfig.h</a>	57									

My Computer

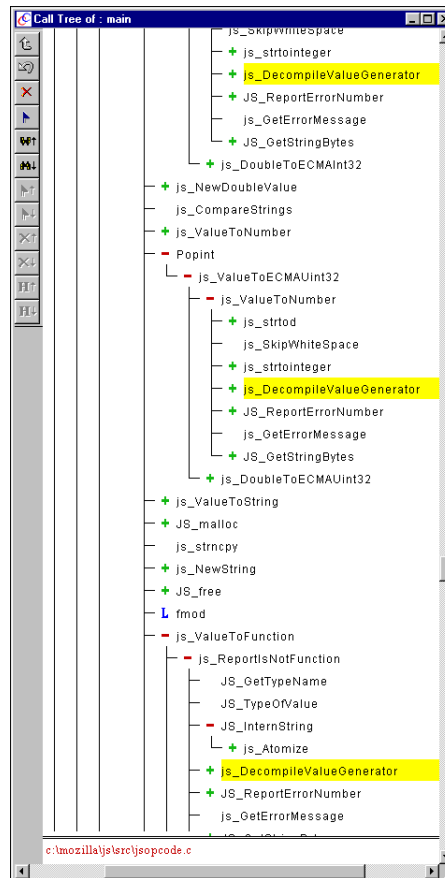
Start Crystal REVS for C [PR...] Crystal REVS for C D... 6:18 PM

You can also create javadoc-like HTML Reports.

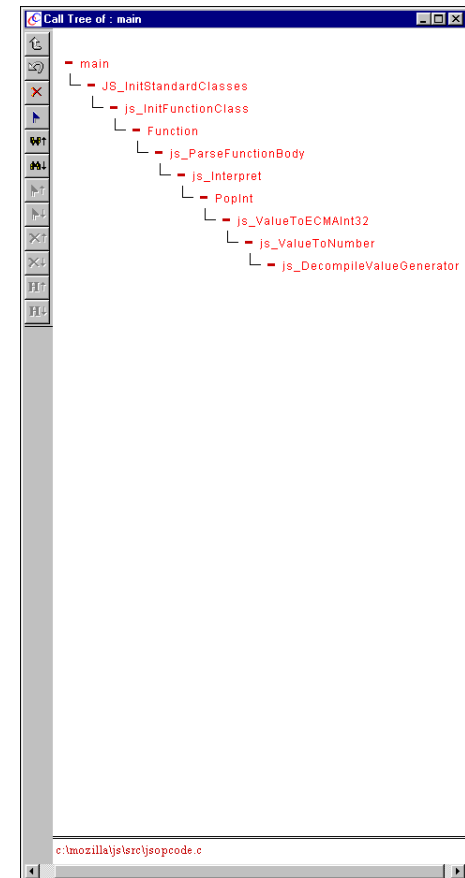
## Call-Trees, Caller-Trees, File-Trees



1. Call Tree of `main()`



2. The result of a node-search for `js_DecompileValueGenerator()` in the tree.



3. **View just the call-path:**  
Crystal C hides all nodes except those in the call-path of `js_DecompileValueGenerator()`

If you are not using Crystal C/C++,  
you are spending a lot more time than you need to.