Adding Comments with the Comments Panel

1. Comments are Extremely Valuable

2. An Object’s Meaning

3. Add Comments to Declarations

4. Examples of Objects’ Meanings

5. Add a Comment to a Statement

6. A Code Segment before and after adding Comments

7. A Comment-based Flowchart

(Intentionally Blank)

Comments Are Extremely Valuable

- When you design or modify code, commenting helps uncover errors in the code.

- Comments save time at every stage of life cycle of the code:

  1. In code-review,
  2. During integration testing,
  3. When you inherit someone else’s code,
  4. During maintenance and enhancements.

- With Crystal C, you can easily add comments to:

  - the code you have just implemented
  - or the legacy code that you are deciphering.

When Comments are Missing - You Lose Time

Real Code from Mozilla:

```
script   = fun->script;
minargs = fun->nargs + fun->extra;
nvars   = fun->nvars;

if( fun->flags )
{
    if( fun->flags & JSFUN_BOUND_METHOD )
        thisp = parent;
    else
        parent = NULL;
}
```

You are about to review the above code.

Can you really tell what the code does ?

For example, What does the following statement do ?

nvars = fun → nvars;

- It will take some effort and time to understand what the above code does.

When Comments are Present - You Save Time

```
script  = fun->script;                          /*  Get interpreted bytecode descriptor            */
minargs = fun->nargs + fun->extra; |            /*  minargs = minimum number of actual args + number */
                                                /*            of arg slots                         */
nvars   = fun->nvars;                           /*  Get number of local variables                  */

if( fun->flags )                                /*  if ( flags are available )                     */
{                                               /*  2{                                             */
    if( fun->flags & JSFUN_BOUND_METHOD )       /*    if ( bind this to fun-object's parent )      */
        thisp = parent;
    else
        parent = NULL;
}                                               /*  2}                                             */
```

With comments, you can
understand the code in half a minute.

There is a school of thinking that
"obvious" comments should not be added.

However, such comments are valuable
because they save a lot of time.

An Object’s Meaning

Real Code from Mozilla:

```
struct  JSFunction                                           /*< >                                                    */
{
    jsrefcount    nrefs;                                     /*  < number of referencing objects >                    */
    JSObject    * object;                                    /*  < back-pointer to GC'ed object header >              */
    JSNative      call;                                      /*  < native method pointer >                            */
    uint16        nargs;                                     /*  < minimum number of actual args >                    */
    uint16        extra;                                     /*  < number of arg slots >                              */
    uint16        nvars;                                     /*  < number of local variables >                        */
    uint8         flags;                                     /*  < bound method and other flags, see jsapi.h >        */
    uint8         spare;                                     /*  < reserved for future use >                          */
    JSAtom      * atom;                                      /*  < name for diagnostics and decompiling >             */
    JSScript    * script;                                    /*  < interpreted bytecode descriptor >                  */
    JSClass     * clasp;                                     /*  < >                                                  */
}                 ;                                          /*                                                       */
```

- In an existing file,
  if a comment is found at the end of a declaration,
  Crystal C associates it as the object’s meaning.
  (as shown above)

- **The object’s meaning is enclosed in < >.**

  The angle brackets help in separating the meaning from other objects’ meanings.
  (when more than one object is in the declaration).

- When the cursor is on an object’s name,
  Crystal C highlights its meaning (in blue color)
  and vice versa.

Objects' Meanings are used in Statements' Comments

Crystal C obtains the objects' meaning from their declarations.

```
JSScript    * script;                                /*  < interpreted bytecode descriptor >           */
uint16        nvars;                                 /*  < number of local variables >                 */
```

the meaning of script is "interpreted bytecode descriptor"

and the meaning of nvars is "number of local variables"

To assist you in commenting the statements,

it displays the object's meaning and some commonly used words in the Comments Panel.

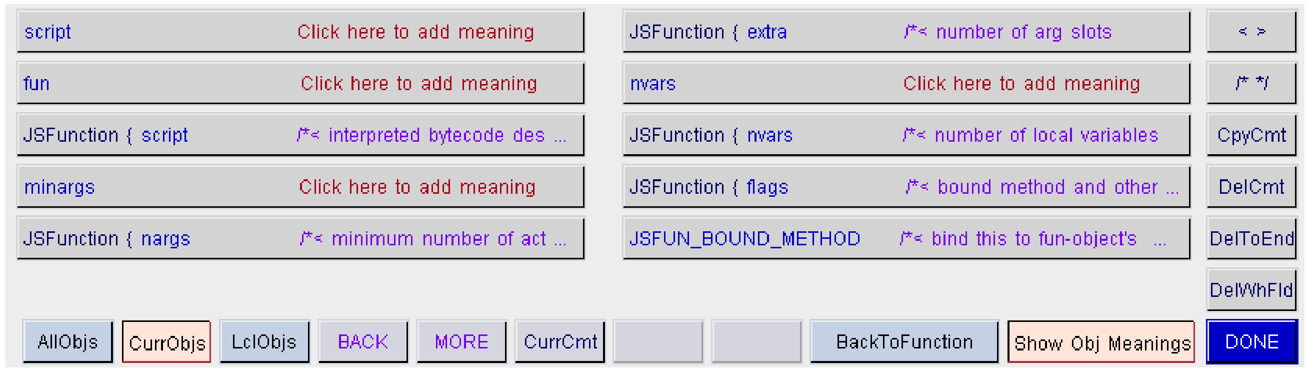
```
script   = fun->script;                              /*    Get interpreted bytecode descriptor         */
nvars    = fun->nvars;                               /*    Get number of local variables               */
```

To add the comments shown above, click "Get" and object's meaning from the Comments Panel.

To Add Comments to the Declarations - Go to the Objects-and-Meanings Panel

Before commenting the statements,

check which objects are already commented
and which objects you wish to comment:



script	Click here to add meaning	JSFunction { extra	/*< number of arg slots
fun	Click here to add meaning	nvars	Click here to add meaning
JSFunction { script	/*< interpreted bytecode des ...	JSFunction { nvars	/*< number of local variables
minargs	Click here to add meaning	JSFunction { flags	/*< bound method and other ...
JSFunction { nargs	/*< minimum number of act ...	JSFUN_BOUND_METHOD	/*< bind this to fun-object's ...

Objects-and-Meanings Panel.

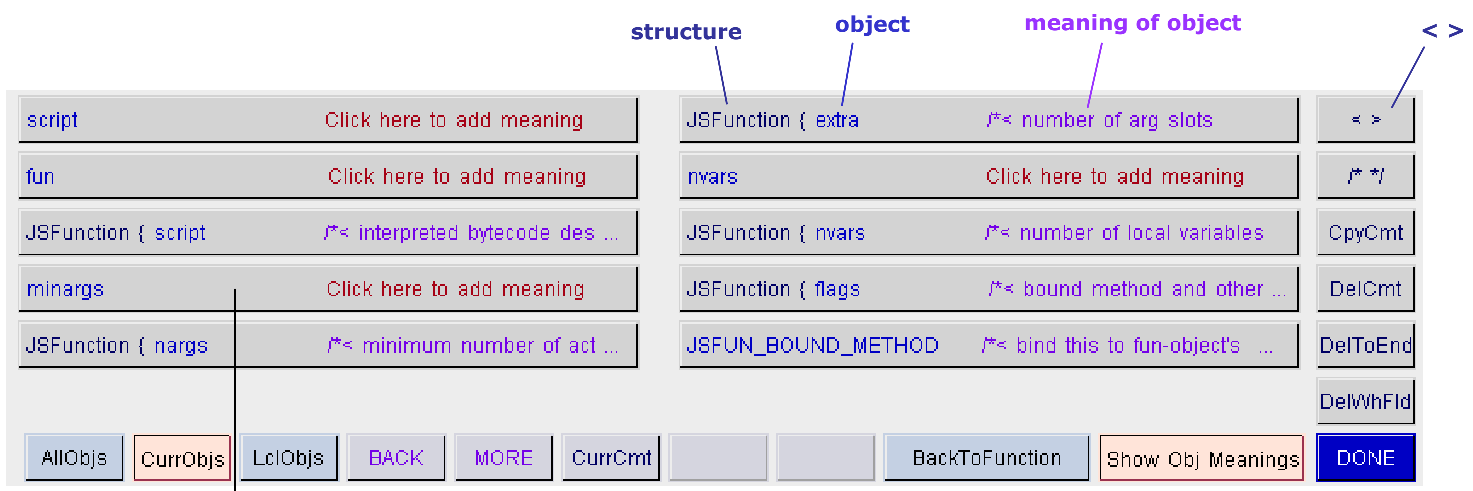
- Click the <Comment> tab.
- Click the <Show Obj Meanings> button.

You will see the **Objects-and-Meanings Panel.**

When **CurrObjs** is selected, the above panel shows objects used in the current statement and onward.

AllObjs - all objects used in the function.

The Object-and-Meanings panel - takes you to the declaration



script	Click here to add meaning	JSFunction { extra	/*< number of arg slots
fun	Click here to add meaning	nvars	Click here to add meaning
JSFunction { script	/*< interpreted bytecode des ...	JSFunction { nvars	/*< number of local variables
minargs	Click here to add meaning	JSFunction { flags	/*< bound method and other ...
JSFunction { nargs	/*< minimum number of act ...	JSFUN_BOUND_METHOD	/*< bind this to fun-object's ...

- Click a button **to go to the object's declaration.**

1. Edit/Enter Object's Meaning:

- with cursor on the object-name, click < > button.
- now edit or enter the object's meaning.
- edit or enter any nearby objects' meanings.

2. Come back to the Function:

- Click <BackToFunction> button;

  it takes you back to the function you were in.

If an object's name is clear enough,
you do not have to provide a meaning for it.

Examples of Objects' Meanings

```
struct  JSScopeProperty                              /*< scope property >                          */
{
    jsval           id;                              /*  < scope id NOTE: passed to getter and setter >  */

    JSPropertyOp    getter, setter;                  /*  < getter method >< setter method >        */

    uint32          slot;                            /*  < idx_to obj-slots element >              */

    JSSymbol      * symbols;                         /*  < ptr_to list of aliasing symbols >       */
}                     ;                              /*                                            */

struct  JSScopeProperty  * last;                     /*< ptr_to last scope property >              */
```

- The object's meaning should be precise. It will result in a natural flow when used in statements' comments. e.g.

  call getter method
  set ptr_to last scope property

- Provide any additional information after "NOTE:"as shown above.

  (Use the NOTE: button.)

- The structure JSScopeProperty has the meaning scope property.

- Always provide the meaning of the structure. It will help you when you enter the meaning of the object.

Some special words in Objects' Meanings

- ♦ When an object is a pointer, begin its meaning with "ptr_to"
- ♦ When an object is an array, begin its meaning with "arr_of"
- ♦ When an object is an index to something, begin its meaning with "idx_to" or "idx_into"

♦ Alternate meaning: Sometimes an object has two roles, i.e. two meanings.

```
jschar          * buf;                              /*  < ptr_to buf ALT: ptr_to current char >          */
```

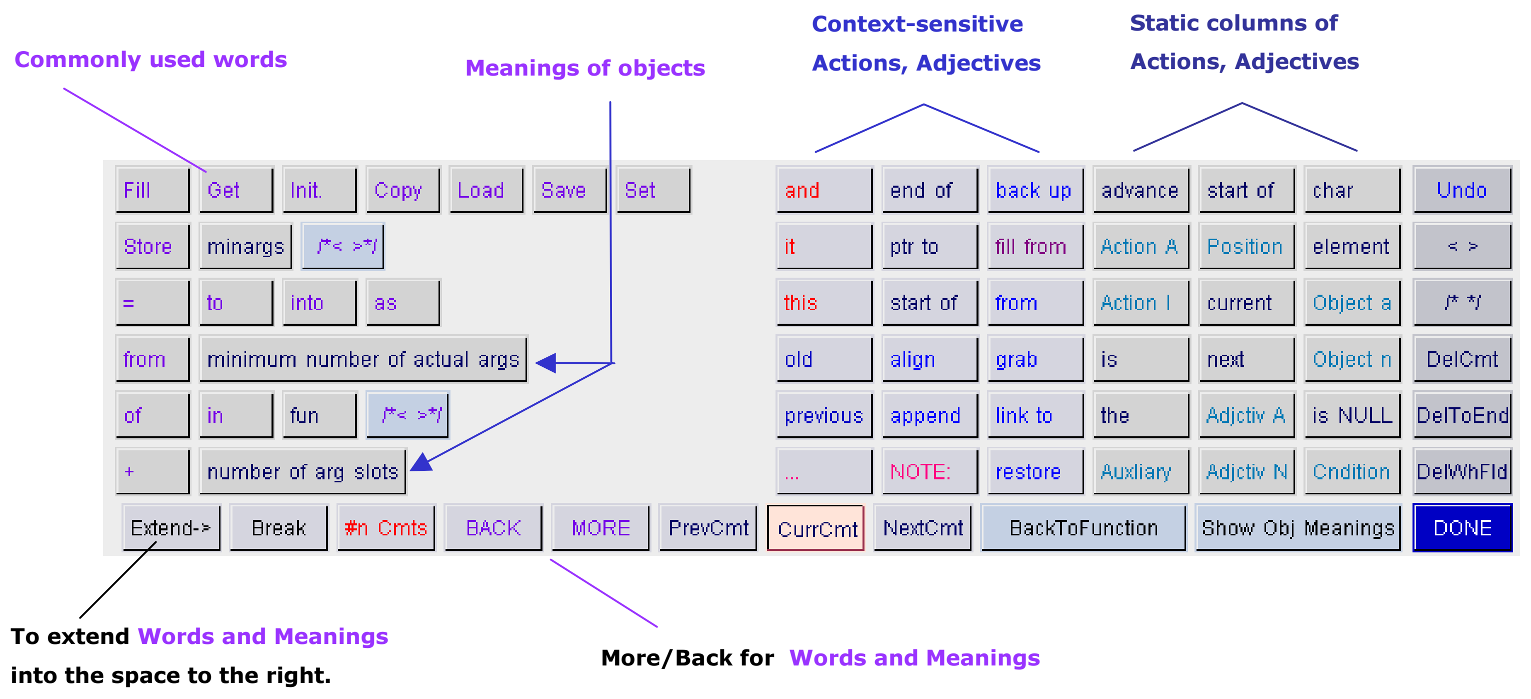
Initially buf is a pointer to the buffer. Later, as you increment it, it is a pointer to the current character.
Start each alternate meaning with "ALT:"

♦ Abbreviation:

```
jsval        arg;                                   /*  < current argument in vector ABR: current arg >     */
```

Instead of using the long form "current argument in vector" in the comments of statements,
you can provide an abbreviation by starting it with "ABR:"

Add a Comment to a Statement



- To comment a statement, deselect the <Show Obj Meanings> button.
  You will see the Comment-Phrases Panel (shown above).

- To provide the meaning to an object, click the /*< >*/ button.
  (Need not go to the Objects-and-Meaning panel.)

The Left Half + Frequently Used Words in the Right Half

- Click the /*< >*/ button to provide meaning to an object. (Need not go to the Objects-and-Meaning panel.)
- Use <Extend → > or <MORE> and <BACK> to page forward/backward.
- <#n Cmts> will display the preceding or next comments. (To help you repeat a comment.)
- <CurCmt> resets the Comments Panel to the initial state for the current statement.

The right half of the Comments Panel contains

♦ Action words	such as	advance, back up, restore
♦ Auxiliary words	such as	is, the
♦ Position words	such as	start of, end of
♦ Adjectives	such as	current, next, old, previous
♦ Objects	such as	char, element, node, tree
♦ Conditions	such as	is available, is not available, is full

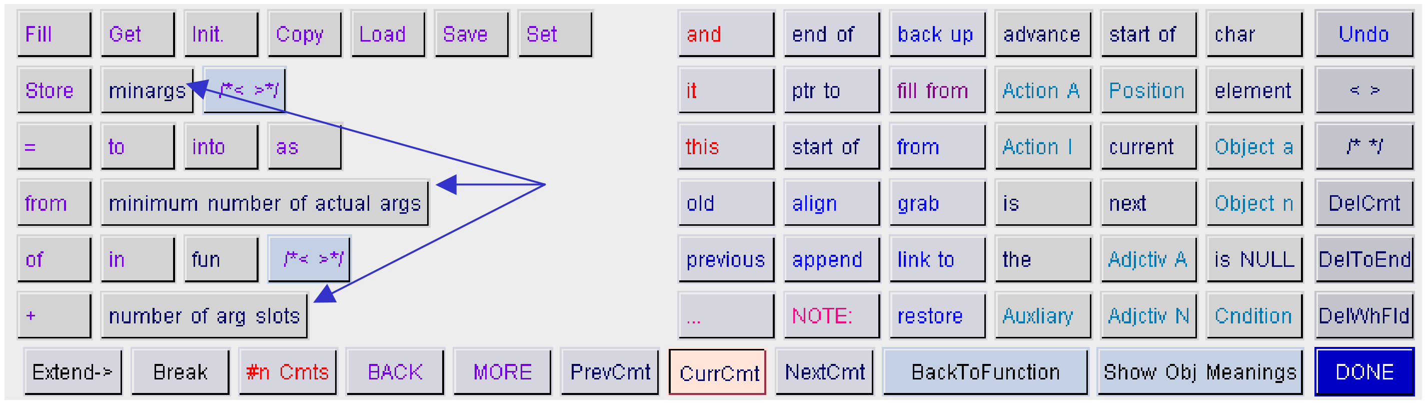
The left three columns are context-sensitive. The three right columns are static.

For more action words, click the <Action A> or <Action I> button.

For more position words, click the <Position> button.

Add a Comment for minargs = fun->nargs + fun->extra;

```
script   = fun->script;                                /*    Get interpreted bytecode descriptor             */
minargs = fun->nargs + fun->extra;     /* minargs = minimum number of actual args + number of arg slots  */
```



- Place the cursor to the right of the statement.
  - then click the buttons:

minargs

=

minimum number of actual args

+

number of arg slots

- The comment is automatically formatted.

An Overview of Adding Comments

In the preceding slides, we saw

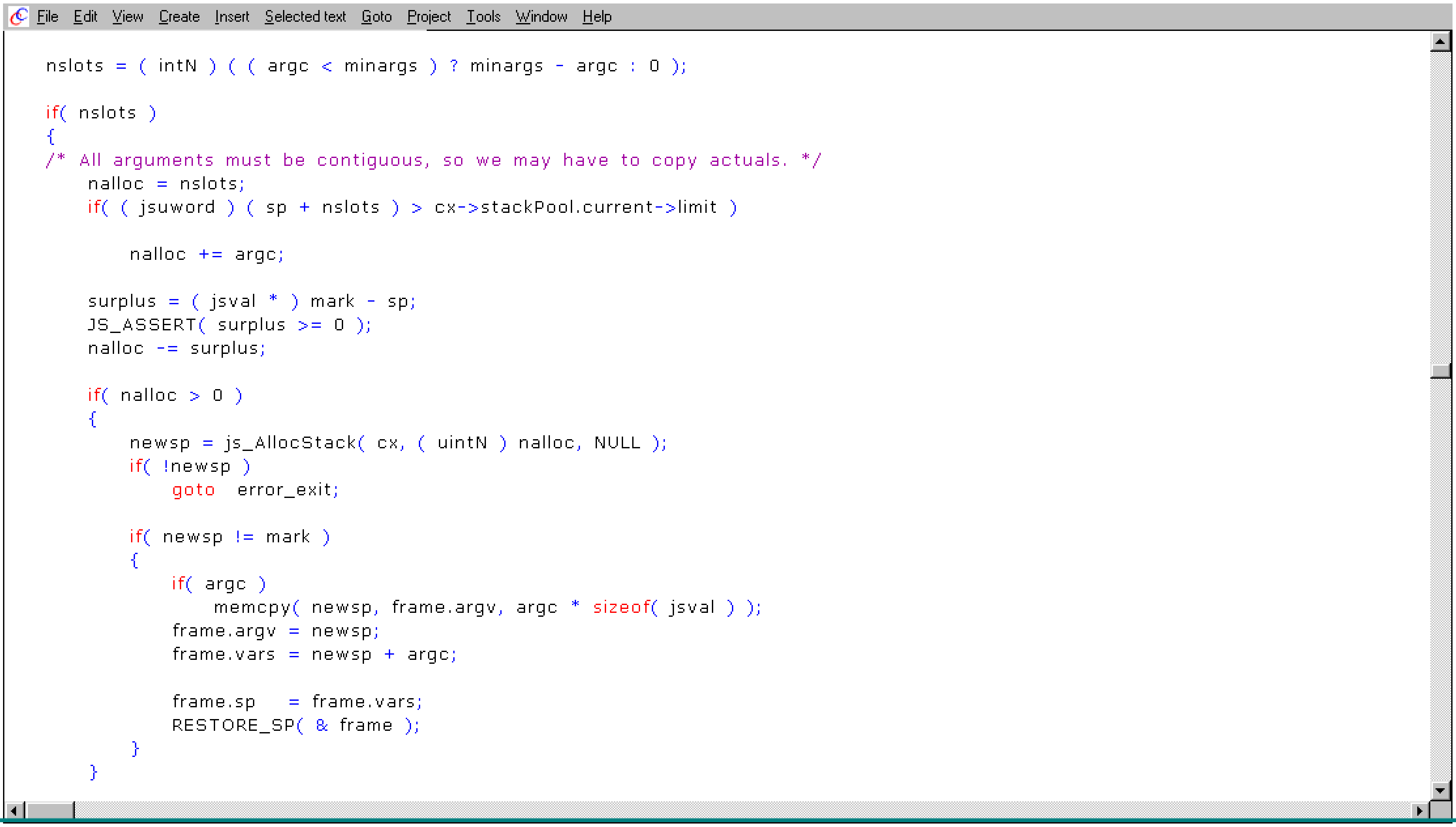
1. A declaration’s comment contains the object’s meaning.

2. Crystal C displays the objects’ meanings and other words
   to assist you in commenting the statements.

3. The Objects-and-Meanings panel helps you to
   - check which objects are already commented
   - go to the declarations of the objects you wish to comment
   - then come back to the function.

4. The Comment-Phrases panel helps you to
   - add comments to declarations and statements.

Code without Comments will Slow You Down

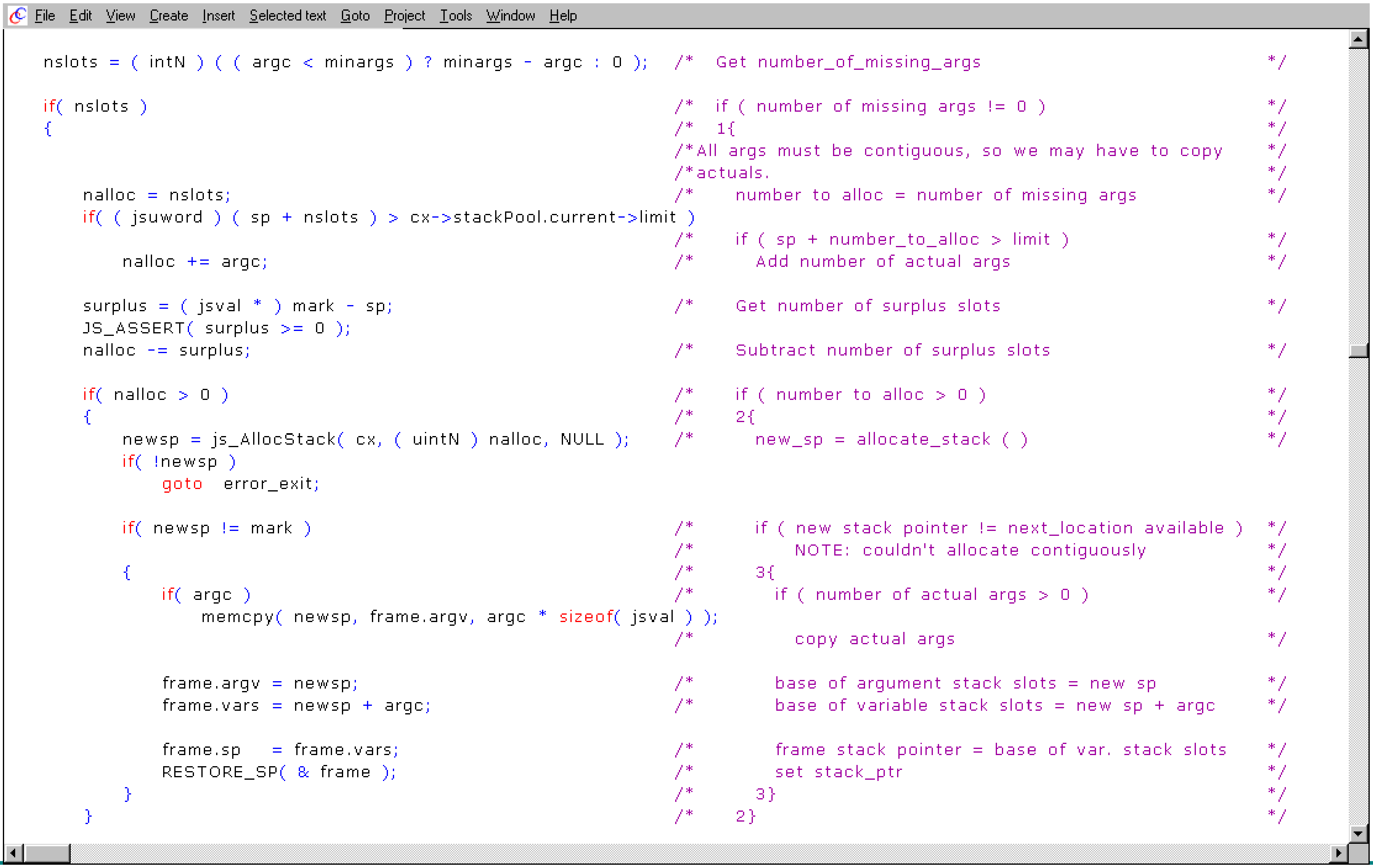


```
nslots = ( intN ) ( ( argc < minargs ) ? minargs - argc : 0 );

if( nslots )
{
/* All arguments must be contiguous, so we may have to copy actuals. */
    nalloc = nslots;
    if( ( jsuword ) ( sp + nslots ) > cx->stackPool.current->limit )

        nalloc += argc;

    surplus = ( jsval * ) mark - sp;
    JS_ASSERT( surplus >= 0 );
    nalloc -= surplus;

    if( nalloc > 0 )
    {
        newsp = js_AllocStack( cx, ( uintN ) nalloc, NULL );
        if( !newsp )
            goto  error_exit;

        if( newsp != mark )
        {
            if( argc )
                memcpy( newsp, frame.argv, argc * sizeof( jsval ) );
            frame.argv = newsp;
            frame.vars = newsp + argc;

            frame.sp    = frame.vars;
            RESTORE_SP( & frame );
        }
    }
```

How quickly can you understand the above code -

so that you can modify it or fix it?

Add Comments in less than 5 Minutes

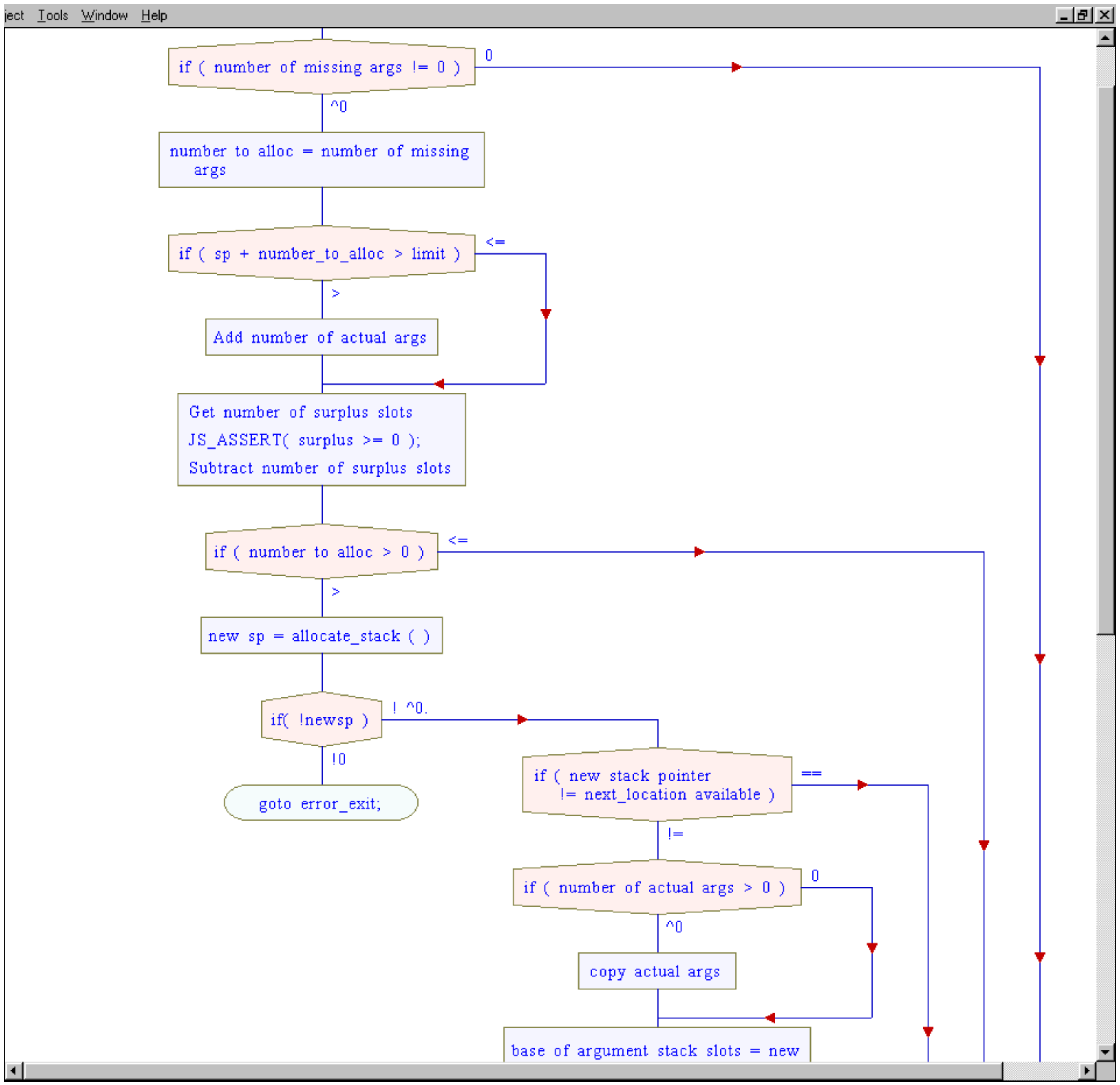

File Edit View Create Insert Selected text Goto Project Tools Window Help

```
nslots = ( intN ) ( ( argc < minargs ) ? minargs - argc : 0 );  /*  Get number_of_missing_args                          */

if( nslots )                                                     /*  if ( number of missing args != 0 )                 */
{                                                                /*  1{                                                  */
                                                                 /*All args must be contiguous, so we may have to copy */
                                                                 /*actuals.                                             */
    nalloc = nslots;                                             /*    number to alloc = number of missing args         */
    if( ( jsuword ) ( sp + nslots ) > cx->stackPool.current->limit )
                                                                 /*    if ( sp + number_to_alloc > limit )              */
        nalloc += argc;                                          /*      Add number of actual args                      */

    surplus = ( jsval * ) mark - sp;                             /*    Get number of surplus slots                      */
    JS_ASSERT( surplus >= 0 );
    nalloc -= surplus;                                           /*    Subtract number of surplus slots                 */

    if( nalloc > 0 )                                             /*    if ( number to alloc > 0 )                       */
    {                                                            /*    2{                                               */
        newsp = js_AllocStack( cx, ( uintN ) nalloc, NULL );     /*      new_sp = allocate_stack ( )                    */
        if( !newsp )
            goto  error_exit;

        if( newsp != mark )                                      /*      if ( new stack pointer != next_location available )  */
                                                                 /*        NOTE: couldn't allocate contiguously         */
        {                                                        /*      3{                                             */
            if( argc )                                           /*        if ( number of actual args > 0 )             */
                memcpy( newsp, frame.argv, argc * sizeof( jsval ) );
                                                                 /*          copy actual args                           */

            frame.argv = newsp;                                  /*        base of argument stack slots = new sp        */
            frame.vars = newsp + argc;                           /*        base of variable stack slots = new sp + argc */

            frame.sp   = frame.vars;                             /*        frame stack pointer = base of var. stack slots  */
            RESTORE_SP( & frame );                               /*        set stack_ptr                                */
        }                                                        /*      3}                                             */
    }                                                            /*    2}                                               */
```

- Context-sensitive Comments-Panel helps you compose the comments.

  You can enter the above comments in 5 minutes.

The commented code is easy to understand.


ject Tools Window Help
if (number of missing args != 0)
0
^0
number to alloc = number of missing args
if (sp + number_to_alloc > limit)
<=
>
Add number of actual args
Get number of surplus slots
JS_ASSERT(surplus >= 0);
Subtract number of surplus slots
if (number to alloc > 0)
<=
>
new sp = allocate_stack ( )
if(!newsp)
! ^0.
!0
goto error_exit;
if (new stack pointer != next_location available)
==
!=
if (number of actual args > 0)
0
^0
copy actual args
base of argument stack slots = new

A Comment-Flowchart has a wider Audience

Comment-flowcharts are understood easily by a wider audience.

You can obtain valuable input on design and test issues.