

# Summary of Lesson 14

## Functions , introduced in this lesson

```
// atom buffer (array of atom)
t_atombuf *atombuf_new (long argc, t_atom *argv);

// deallocate atom
void atombuf_free (t_atombuf *ab);

// debug atom
void postatom (t_atom *item);

// popup window
void ouchstring (char *fmtstring, void *items...);

// get max version
short maxversion (void);

// open dialog
short open_dialog (char *filename, short *path,
OSType *dstType, SFTypelist typelist, short
numtypes);

// open dialog prompt
short open_promptset (char *prompt);

// search from max search path
short locatefile (char *filename, short *path, short *binary);

// for getting fqname
short path_topathname(short path, char *file, char *name);
```

---

### ex1.c

```
#include "ext.h"

// loadmess

typedef struct {
    t_object b_ob;
    t_atom *array_of_atoms;
    int number_of_atoms;
    void *out;
} t_ex1;

void ex1_loadbang(t_ex1 *x);

void *ex1_new(t_symbol *s, short argc, t_atom *argv);
```

```

void ex1_free(t_ex1 *x);

void *ex1_class;

void main()
{
    setup((t_messlist **)&ex1_class, (method)ex1_new, (method)ex1_free,
(short)sizeof(t_ex1), 0L, A_GIMME, 0L);
    addmess((method)ex1_loadbang, "loadbang", A_CANT, 0L);
}

void ex1_loadbang(t_ex1 *x)
{
    outlet_list(x->out, 0L , x->number_of_atoms, x->array_of_atoms);
}

void *ex1_new(t_symbol *s, short argc, t_atom *argv)
{
    t_ex1 *x;
    int i;
    x = (t_ex1*)newobject(ex1_class);
    x->array_of_atoms = (t_atom*)malloc(sizeof(t_atom) * argc);
    if(!x->array_of_atoms)
    {
        error("ex1:couldn't allocate memory");
        return NULL;
    }

    x->number_of_atoms = argc;
    for(i = 0; i<x->number_of_atoms;i++)
    {
        x->array_of_atoms[i].a_type = argv[i].a_type;
        switch(x->array_of_atoms[i].a_type)
        {
            case A_LONG:
                x->array_of_atoms[i].a_w.w_long = argv[i].a_w.w_long;
                break;
            case A_FLOAT:
                x->array_of_atoms[i].a_w.w_float = argv[i].a_w.w_float;
                break;
            case A_SYM:
                x->array_of_atoms[i].a_w.w_sym = argv[i].a_w.w_sym;
                break;
        }
    }
    x->out = outlet_new(x,0L);

    return x;
}

```

```

void ex1_free(t_ex1 *x)
{
    if(x->array_of_atoms);
        free(x->array_of_atoms);
}

```

---

## ex2.c

```

#include "ext.h"

// loadmess implemented with atom_buf

typedef struct {
    t_object b_ob;
    t_atombuf *ab;
    void *out;
} t_ex2;

void ex2_loadbang(t_ex2 *x);

void *ex2_new(t_symbol *s, short argc, t_atom *argv);
void ex2_free(t_ex2 *x);

void *ex2_class;

void main()
{
    setup((t_messlist **)&ex2_class, (method)ex2_new, (method)ex2_free,
(short)sizeof(t_ex2), 0L, A_GIMME, 0L);
    addmess((method)ex2_loadbang, "loadbang", A_CANT, 0L);
}

void ex2_loadbang(t_ex2 *x)
{
    outlet_list(x->out, 0L , x->ab->a_argc, x->ab->a_argv);
}

void *ex2_new(t_symbol *s, short argc, t_atom *argv)
{
    t_ex2 *x;
    x = (t_ex2*)newobject(ex2_class);
    x->ab = atombuf_new(argc,argv);
    if(!x->ab)
    {
        error("ex2:couldn't allocate memory");
        return NULL;
    }
}

```

```

    x->out = outlet_new(x,0L);
    return x;
}

void ex2_free(t_ex2 *x)
{
    if(x->ab)
        atombuf_free(x->ab);
}

```

---

### ex3.c

```

#include "ext.h"

// debug atoms

typedef struct {
    t_object b_ob;
    t_atombuf *ab;
    void *out;
} t_ex3;

void ex3_loadbang(t_ex3 *x);

void *ex3_new(t_symbol *s, short argc, t_atom *argv);
void ex3_free(t_ex3 *x);

void *ex3_class;

void main()
{
    setup((t_messlist **)&ex3_class, (method)ex3_new, (method)ex3_free,
    (short)sizeof(t_ex3), 0L, A_GIMME, 0L);
    addmess((method)ex3_loadbang, "loadbang", A_CANT, 0L);
}

void ex3_loadbang(t_ex3 *x)
{
    int i;
    for(i = 0; i < x->ab->a_argc; i++)
        postatom(x->ab->a_argv+i);
}

void *ex3_new(t_symbol *s, short argc, t_atom *argv)
{
    t_ex3 *x;
    x = (t_ex3*)newobject(ex3_class);
    x->ab = atombuf_new(argc, argv);
    if(!x->ab)

```

```

    {
        error("ex3:couldn't allocate memory");
        return NULL;
    }

    x->out = outlet_new(x,0L);
    return x;
}

void ex3_free(t_ex3 *x)
{
    if(x->ab)
        atombuf_free(x->ab);
}

```

---

#### ex4.c

```

#include "ext.h"

// ouchstring

typedef struct {
    t_object b_ob;
    t_atombuf *ab;
} t_ex4;

void ex4_bang(t_ex4 *x);
void *ex4_new(void);

void *ex4_class;

void main()
{
    setup((t_messlist **)&ex4_class, (method)ex4_new, 0L, (short)sizeof(t_ex4),
    0L, 0L);
    addbang((method)ex4_bang);
}

void ex4_bang(t_ex4 *x)
{
    ouchstring("I am going to crash... bye bye my friends");
}

void *ex4_new(void)
{
    t_ex4 *x;
    x = (t_ex4*)newobject(ex4_class);
    return x;
}

```

---

**ex5.c**

```
#include "ext.h"

// get version

typedef struct {
    t_object b_ob;
    t_atombuf *ab;
} t_ex5;

void ex5_bang(t_ex5 *x);
void *ex5_new(void);

void *ex5_class;

void main()
{
    setup((t_messlist **)&ex5_class, (method)ex5_new, 0L, (short)sizeof(t_ex5),
    0L, 0L);
    addbang((method)ex5_bang);
}

void ex5_bang(t_ex5 *x)
{
    post("Maxversion is %x",maxversion());
}

void *ex5_new(void)
{
    t_ex5 *x;
    x = (t_ex5*)newobject(ex5_class);
    return x;
}
```

---

**ex6.c**

```
#include "ext.h"

// open dialog

typedef struct {
    t_object b_ob;
    t_atombuf *ab;
} t_ex6;

void ex6_bang(t_ex6 *x);
```

```

void *ex6_new(void);

void *ex6_class;

void main()
{
    setup((t_messlist **)&ex6_class, (method)ex6_new, 0L, (short)sizeof(t_ex6),
    0L, 0L);
    addbang((method)ex6_bang);
}

void ex6_bang(t_ex6 *x)
{
    char filename[256];
    short path;
    long dstType;
    open_dialog(filename,&path,&dstType,0L,0L);
    post("you choosed file:%s",filename);
}

void *ex6_new(void)
{
    t_ex6 *x;
    x = (t_ex6*)newobject(ex6_class);
    return x;
}

```

---

## ex7.c

```

#include "ext.h"

// get Fully Qualified Name

typedef struct {
    t_object b_ob;
    t_atombuf *ab;
} t_ex7;

void ex7_bang(t_ex7 *x);
void *ex7_new(void);

void *ex7_class;

void main()
{
    setup((t_messlist **)&ex7_class, (method)ex7_new, 0L, (short)sizeof(t_ex7),
    0L, 0L);
    addbang((method)ex7_bang);
}

```

```

}

void ex7_bang(t_ex7 *x)
{
    char filename[256];
    char fqn[256];
    short path;
    long dstType;

    open_promptset("Please choose your file...");
    if(open_dialog(filename,&path,&dstType,0L,0L))
    {
        post("User canceled");
        return;
    }
    path_topathname(path, filename, fqn);
    post("fqn: %s",fqn);
}

```

```

void *ex7_new(void)
{
    t_ex7 *x;
    x = (t_ex7*)newobject(ex7_class);
    return x;
}

```

---

## ex8.c

```

#include "ext.h"

// find a file from the search path

typedef struct {
    t_object b_ob;
    t_atombuf *ab;
} t_ex8;

void ex8_locate(t_ex8 *x,t_symbol *filename);
void *ex8_new(void);

void *ex8_class;

void main()
{
    setup((t_messlist **)&ex8_class, (method)ex8_new, 0L, (short)sizeof(t_ex8),
    0L, 0L);
    addmess((method)ex8_locate,"locate",A_SYM,0L);
}

void ex8_locate(t_ex8 *x,t_symbol *filename)

```



```
{
    short id;
    char fqn[256];
    short binary;
    if(locatefile(filename->s_name,&id,&binary))
    {
        error("file not found");
        return;
    }
    path_topathname(id, filename->s_name, fqn);

    post("fqn : %s",fqn);
}
```

```
void *ex8_new(void)
{
    t_ex8 *x;
    x = (t_ex8*)newobject(ex8_class);
    return x;
}
```