Rect structure

```
#include <Quickdraw.h>

typedef struct Rect {
  short       top;
  short       left;
  short       bottom;
  short       right;
} Rect ;

#define topLeft( r)     (((Point *) &(r))[0])
#define botRight( r)    (((Point *) &(r))[1])

typedef Rect *RectPtr;
```

Size	Offset	Description
2	0	Top edge of rectangle
2	2	Left edge
2	4	Bottom edge
2	6	Rightmost edge
8		

handy macros

Notes: The Rect structure defines a rectangular area of a window. Use **SetRect** to initialize a rectangle; or simply set its fields directly (to avoid the system overhead).

A valid rectangle should enclose some pixels; thus if top >= bottom or left >= right, then the system will generally treat is as an "empty rectangle" - one whose coordinates are (0,0)-(0,0) (see **EmptyRect**).

Mathematical Operations

EmptyRect	**OffsetRect**	**RectInRgn**	**UnionRect**
EqualRect	**Pt2Rect**	**RectRgn**	
InsetRect	**PtInRect**	**SectRect**	
MapRect	**PinRect**	**SetRect**	

Drawing Operations

EraseRect	**FillRect**	**FrameRect**	**InvertRect**

Other Fns Using a Rect parameter

ClipRect	**FrameOval**	**NewControl**	**ShieldCursor**
CopyBits	**FrameRoundRect**	**NewDialog**	**StdArc**
CopyMask	**GrowWindow**	**NewWindow**	**StdBits**
DragWindow	**InvalRect**	**OpenPicture**	**StdOval**
DrawPicture	**InvertArc**	**PaintOval**	**StdRRect**
EraseArc	**InvertOval**	**PaintRect**	**TENew**
EraseOval	**InvertRoundRect**	**PaintRoundRect**	**TEUpdate**
EraseRoundRect	**LNew**	**PlotIcon**	**TextBox**
FillArc	**LRect**	**PrOpenPage**	**ValidRect**
FillOval	**MapPoly**	**PtToAngle**	
FillRoundRect	**MapPt**	**ScalePt**	
FrameArc	**MapRgn**	**ScrollRect**	

Note: Always pass the *address* of a Rect to functions. All examples in this Guide use: **FooFn**(&r,...) to make this clear, and function prototypes correctly refer to the parameter as a **Rect ***. Of course, Apple documentation (with its Pascal bias) assumes you will mentally convert "Rect" to "&Rect".