Die Schnittstelle der ML5

ml5::application

```
class application : public object {
   public:
      window & get_win () const;

      template <typename ...A> int run (A && ...args);

   protected:
      virtual unique_ptr <window> make_window () const = 0;

      virtual void on_exit  ();
      virtual void on_init  ();
      virtual void on_timer (timer_event const &);
};
```

ml5::collection

```
using collection = container <shared_ptr <object>>;
```

ml5::container<>

```
template <typename T> class container : public object {
   public:
      virtual void   add      (T value) = 0;
      virtual void   clear    () = 0;
      virtual bool   contains (T const & value) const = 0;
      virtual void   remove   (T const & value) = 0;
      virtual size_t size     () const = 0;

      virtual unique_ptr <iterator <T>> make_iterator () const = 0;

      util::iterator_adapter <T> begin () const;
      util::iterator_adapter <T> end   () const;

      bool           empty        () const;
      iterator <T> * new_iterator () const;
};

ostream & operator << (ostream & lhs, container <T> const & rhs);
```

ml5::event

```
class event : public object {
   protected:
      template <typename E> E const & get_event () const;
};
```

ml5::key_event

```
class key_event final : public event {
   public:
      int get_key_code () const;
};
```

ml5::menu_event

```
class menu_event final : public event {
   public:
      string const & get_item           () const;
      string const & get_title          () const;
      string         get_title_and_item () const;
};
```

ml5::mouse_event

```
class mouse_event final : public event {
   public:
      int     get_button         () const;
      wxPoint get_position       () const;
      int     get_wheel_rotation () const;
      bool    is_left_button     () const;
      bool    is_right_button    () const;
};
```

ml5::paint_event

```
class paint_event final : public event {
   public:
      using context_t = /* ... */;

      context_t & get_context () const;
      bool        has_context () const;
};
```

ml5::size_event

```
class size_event final : public event {
   public:
      wxSize get_size () const;
};
```

ml5::timer_event

```
class timer_event final : public event {
};
```

ml5::integer

```
using integer = value <int>;
```

ml5::iterator<>

```
template <typename T> class iterator : public object {
   public:
      virtual T &  get_current () const = 0;
      virtual bool is_at_end   () const = 0;
      virtual void to_next     () = 0;

      bool not_at_end () const;
};
```

ml5::object

```
class object {
   public:
      using minfo_cont_t = /* ... */;

      metainfo_base const & get_metainfo        () const;
      minfo_cont_t const &  get_metainfo_bases  () const;
      metainfo_base const & get_metainfo_static ();

      string const & get_class_name () const;

      bool is_a (string const & name) const;
      bool is_a (type_info const & type) const;

      template <typename C> bool is_a () const;
};
```

ml5::string

```
using string = value <std::string>;
```

ml5::value<>

```
template <typename T> class value final : public object {
   public:
      value (T);

      value & operator = (T);

      operator T &       ();
      operator T const & () const;

      friend bool operator == (value const & lhs, value const & rhs);
      friend bool operator <  (value const & lhs, value const & rhs);
      friend bool operator != (value const & lhs, value const & rhs);
      friend bool operator <= (value const & lhs, value const & rhs);
      friend bool operator >  (value const & lhs, value const & rhs);
      friend bool operator >= (value const & lhs, value const & rhs);
};

value <T> make_value (T v);

istream & operator >> (istream & lhs, value <T> & rhs);
ostream & operator << (ostream & lhs, value <T> const & rhs);
```

ml5::vector<>

```
template <typename T> class vector final : public container <T> {
   public:
      template <typename U> vector (initializer_list <U> const values);

      T &       operator [] (size_t idx);
      T const & operator [] (size_t idx) const;

      void   add      (T value) override;
      void   clear    () override;
      bool   contains (T const & value) const override;
      void   remove   (T const & value) override;
      size_t size     () const override;

      unique_ptr <iterator <T>> make_iterator () const override;

      T &       at   (size_t idx);
      T const & at   (size_t idx) const;
      void      swap (vector & other);
};

void swap (vector <T> & lhs, vector <T> & rhs);
```

ml5::vector_iterator<>

```
template <typename T> class vector_iterator final : public iterator <T> {
   public:
      T &  get_current () const override;
      bool is_at_end   () const override;
      void to_next     () override;
};
```

ml5::window

```
class window : public object {
   public:
      using mitem_cont_t = vector <tuple <string, string>>;

      window (string title);

      void          add_menu         (string const &       title,
                                      mitem_cont_t const & items) const;
      void          cursor_off       () const;
      void          cursor_on        () const;
      application & get_app          () const;
      int           get_height       () const;
      wxSize        get_size         () const;
      int           get_width        () const;
      void          refresh          () const;
      void          restart_timer    (milliseconds interval) const;
      void          set_status_text  (string const & text) const;
      void          show_message_box (string const & text);
      void          start_timer      (milliseconds interval) const;
      void          stop_timer       () const;

   protected:
      virtual void on_exit                    ();
      virtual void on_init                    ();
      virtual void on_key                     (key_event const &);
      virtual void on_menu                    (menu_event const &);
      virtual void on_mouse_double_down       (mouse_event const &);
      virtual void on_mouse_down              (mouse_event const &);
      virtual void on_mouse_left_double_down  (mouse_event const &);
      virtual void on_mouse_left_down         (mouse_event const &);
      virtual void on_mouse_left_up           (mouse_event const &);
      virtual void on_mouse_move              (mouse_event const &);
      virtual void on_mouse_right_double_down (mouse_event const &);
      virtual void on_mouse_right_down        (mouse_event const &);
      virtual void on_mouse_right_up          (mouse_event const &);
      virtual void on_mouse_up                (mouse_event const &);
      virtual void on_mouse_wheel             (mouse_event const &);
      virtual void on_paint                   (paint_event const &);
      virtual void on_size                    (size_event const &);
      virtual void on_timer                   (timer_event const &);
};
```