API REFERENCE

NAMESPACE html

This is the namespace in which all classes are defined.

CLASS html::Element

Each component of html document is an object of this class i.e.; it is of html::Element type.

Constructors Summary

Constructor	Description
Element ()	**Creates an Unnamed Element Object.**
Element (const std::string& name)	**Creates a named Element Object.**
Element (const std::string& name, const std::string& text)	**Creates a named Element Object with text.**
Element (const std::string& name, const std::unordered_map<std::string, std::string>& attr, const std::string& text)	**Creates a named Element Object with text and attributes.**

Member Functions Summary

Modifier and Type	Function and Description
public html::Element&	**parent_element()** Returns the parent of the element as an Element object or throws exception if not found.
public html::Element&	**first_child()** Returns the first child of the element as an Element object or throws exception if not found.
public html::Element&	**next_sibling()** Returns the next sibling of the element as an Element object or throws exception if not found.
public html::Element&	**last_child()** Returns the last child of the element as an Element object or throws exception if not found.

public html::Element&	**previous_sibling()** Returns the previous sibling of the element as an Element object or throws exception if not found.
public std::string	**element_name()** Returns the element name of the object as a string or throws exception if not found.
public std::string	**class_name()** Returns the class name of the object as a string or throws exception if not found.
public std::string	**get_attribute(const std::string& name)** Returns the value of the given attribute as a string or throws exception if not found.
public std::string	**to_html()** Returns the html content of the object as a string.
public bool	**operator == (const html::Element& E)** Compares two objects and returns true if they are equal else returns false.
public bool	**has_children ()** Returns true if the object has children else returns false.
public bool	**has_attribute (const std::string& name)** Returns true if the object has an attribute with the given name else returns false.
public bool	**has_attributes ()** Returns true if the object has attributes or returns false.
public void	**rename_element(const std::string& name)** Sets / renames the name of the Element object to given name.
public void	**set_text(const std::string& text)** Sets the text of the Element to given text.
public void	**set_attribute(const std::string& name, const std::string& value)** Adds a new attribute with given (name, value) or sets the pre-existing attribute with given value.

public void	**remove_text()** Removes the text of the Element object.
public void	**remove_attribute(const std::string& name)** Removes the attribute with the given name from the object's list of attributes.
public void	**append_child(const html::Element& E)** Appends the given Element E to the calling object's children's list.
public void	**remove_child(const html::Element& E)** Removes the given Element E from the calling object's children's list.
public void	**replace_child(const html::Element& oE,const html::Element& nE)** Replaces the given Element nE with the calling object's child oE.
public void	**insert_before(const html::Element& oE,const html::Element& nE)** Inserts the given Element nE before the calling object's child oE.
public void	**insert_after(const html::Element& oE,const html::Element& nE)** Inserts the given Element nE after the calling object's child oE.
public void	**clone(const html::Element& E,bool deep)** If deep is false clones the calling object's attributes, text to Unnamed Element E, if true clones children also.
public void	**dump(const std::string& filename)** dumps the html content of the object into a html file.

Constructors Detail

Element()
Description: creates an Unnamed Element Object.
Parameters: Takes no Parameters

Element(const std::string& name)
Description: creates a Named Element Object.
Parameters: **name-** a std::string object representing the name of the element.
Throws: **InvalidArgumentError:** if name is empty.

Element(const std::string& name, const std::string& text)
Description: creates a Named Element Object with text in it.
Parameters: **name-** a std::string object representing the name of the element. **text**- a std::string object representing the text inside the element.
Throws: **InvalidArgumentError:** if name is empty.

Element(const std::string& name, const std::unordered_map<std::string,std::string>& attr,const std::string& text)
Description: creates a Named Element Object with attributes and text in it.
Parameters: **name-** a std::string object representing the name of the element. **attr-** a std::unordered_map<std::string, std::string> object representing the attributes and their values. **text**- a std::string object representing the text inside the element.
Throws: **InvalidArgumentError:** if name is empty.

Member Functions Detail

public html::Element& parent_element()
Description: returns the parent of the element as an html::Element object or throws exception if not found.
Parameters: Takes no Parameters.
Returns: reference to parent element of the object.
Throws: **InvalidStateError**-if calling object is unnamed. **NotFoundError**- if calling object has no parent.

public html::Element& first_child()
Description: returns the first child of the element as an html::Element object or throws exception if not found.
Parameters: Takes no Parameters.
Returns: reference to first child of the object.
Throws: **InvalidStateError**-if calling object is unnamed. **NotFoundError**- if calling object has no child.

public html::Element& next_sibling()
Description: returns the next sibling of the element as an html::Element object or throws exception if not found.
Parameters: Takes no Parameters.
Returns: reference to next sibling of the object.
Throws: **InvalidStateError**-if calling object is unnamed. **NotFoundError**- if calling object has no next sibling.

public html::Element& last_child()
Description: returns the last child of the element as an html::Element object or throws exception if not found.
Parameters: Takes no Parameters.
Returns: reference to last child of the object.
Throws: **InvalidStateError**-if calling object is unnamed. **NotFoundError**- if calling object has no last child.

public html::Element& previous_sibling()
Description: returns the previous sibling of the element as an html:: Element object or throws exception if not found.
Parameters: Takes no Parameters.
Returns: reference to previous sibling of the object.
Throws: **InvalidStateError**-if calling object is unnamed. **NotFoundError**- if calling object has no previous sibling.

public std::string element_name()
Description: returns the name of the element as a std::string object or throws exception if not found.
Parameters: Takes no Parameters.
Returns: std::string object holding the name of the element.
Throws: **InvalidStateError**-if calling object is unnamed.

public std::string to_html()
Description: returns the html content of the element as a std::string object or throws exception if not valid.
Parameters: Takes no Parameters.
Returns: std::string object holding the html content of the element.
Throws: **InvalidStateError**-if calling object is unnamed.

public std::string class_name()
Description: returns the class name of the element as a std::string object or throws exception if not found.
Parameters: Takes no Parameters.
Returns: std::string object holding the class name of the element.
Throws: **InvalidStateError**-if calling object is unnamed. **NotFoundError**- if calling object has no class attribute.

public std::string get_attribute(const std::string& name)
Description: returns the value of the attribute name as a std::string object or throws exception if not found.
Parameters: **name**- a std::string object holding the name of the attribute.
Returns: std::string object holding the value of the attribute.
Throws: **InvalidStateError**-if calling object is unnamed. **NotFoundError**- if calling object has no attribute with the given name. **InvalidArgumentError:-** if name is empty.

public bool operator==(const html::Element& E)
Description: compares the calling object and E and returns true if they are equal(have same attributes, name, text, children) else returns false.
Parameters: **E**- a html::Element object
Returns: Boolean value true or false.
Throws: **InvalidStateError**-if calling object is unnamed. **InvalidArgumentError:** if E is unnamed.

public bool has_children()
Description: returns true if the calling object has children else returns false.
Parameters: Takes No parameters.
Returns: Boolean value true or false.
Throws: **InvalidStateError**-if calling object is unnamed.

public bool has_attributes()
Description: returns true if the calling object has attributes else returns false.
Parameters: Takes no Parameters.
Returns: Boolean value true or false.
Throws: **InvalidStateError**-if calling object is unnamed.

public bool has_attribute(const std::string& name)
Description: returns true if attribute with given name exists else returns false.
Parameters: **name**- a std::string object holding the name of the attribute.
Returns: Boolean value true or false.
Throws: **InvalidStateError**-if calling object is unnamed.

public void rename_element(const std::string& name)
Description: sets name of unnamed element or renames a named element.
Parameters: **name**- a std::string object holding the name of the element.
Returns: void
Throws: **InvalidModificationError:** if name is empty.

public void set_text(const std::string& text)
Description: sets text of calling object.
Parameters: **text**- a std::string object holding the text data.
Returns: void
Throws: **InvalidStateError**-if calling object is unnamed. **InvalidArgumentError-** if text is empty.

public void set_attribute(const std::string& name, const std::string& value)
Description: adds an attribute with given (name, value) or sets pre-existing attribute of the calling object.
Parameters: **name**- a std::string object holding the name of the attribute. **value-** a std::string object holding the value of the attribute.
Returns: void
Throws: **InvalidStateError**-if calling object is unnamed. **InvalidArgumentError:** if name is empty or value is empty.

public void remove_text()
Description: removes text of the element.
Parameters: Takes no parameters.
Returns: void
Throws: **InvalidStateError**-if calling object is unnamed.

public void remove_attribute(const std::string& name)
Description: removes attribute with the given name.
Parameters: **name**- a std::string object holding the name of the attribute to be removed.
Returns: void
Throws: **InvalidStateError**-if calling object is unnamed. **InvalidArgumentError-** if name is empty.

public void append_child(const html::Element& E)
Description: appends E to calling object's children list. If E already has a parent it is removed from there and appended to this object (because each object can be at one and only one location in the document)
Parameters: **E-** a html::Element object
Returns: void
Throws: **InvalidStateError**-if calling object is unnamed. **InvalidArgumentError-** if E is unnamed. **InvalidModificationError-** if E is another instance of calling object(i.e.; they are same).

public void replace_child(const html::Element& oE, const html::Element& nE)
Description: replaces calling object's child oE with the Element nE.
Parameters: **oE-** a html::Element object which is child of calling object. **nE-** a html::Element object to be replaced.
Returns: void
Throws: **InvalidStateError**-if calling object is unnamed. **InvalidArgumentError-** if oE is unnamed or nE is unnamed **InvalidModificationError-** if oE or nE is another instance of calling object (i.e.; they are same). **InvalidElementError-**if oE is not the child of calling object or has no parent.

public void remove_child(const html::Element& E)
Description: removes the child E from the calling object.
Parameters: **E-** a html::Element object
Returns: void
Throws: **InvalidStateError**-if calling object is unnamed. **InvalidArgumentError-** if E is unnamed. **InvalidModificationError-** if E is another instance of calling object (i.e.; they are same). **InvalidElementError-**if E is not the child of calling object or has no parent.

public void insert_before(const html::Element& oE, const html::Element& nE)
Description: inserts nE before the calling object's child oE.
Parameters: **oE-** a html::Element object which is child of calling object. **nE-** a html::Element object to be inserted.
Returns: void
Throws: **InvalidStateError**-if calling object is unnamed. **InvalidArgumentError-** if oE is unnamed or nE is unnamed **InvalidModificationError-** if oE or nE is another instance of calling object (i.e.; they are same). **InvalidElementError-**if oE is not the child of calling object or has no parent.

public void insert_after(const html::Element& oE, const html::Element& nE)
Description: inserts nE after the calling object's child oE.
Parameters: **oE-** a html::Element object which is child of calling object. **nE-** a html::Element object to be inserted.
Returns: void
Throws: **InvalidStateError**-if calling object is unnamed. **InvalidArgumentError-** if oE is unnamed or nE is unnamed **InvalidModificationError-** if oE or nE is another instance of calling object (i.e.; they are same). **InvalidElementError-**if oE is not the child of calling object or has no parent.

public void dump(const std::string& filename)
Description: dumps the html content of the calling object to a html file.
Parameters: **filename-** a std::string object holding the filename.
Returns: void
Throws: **InvalidStateError**-if calling object is unnamed. **InvalidArgumentError-** if filename is empty.

public void clone(const html::Element& E, bool deep)
Description: if deep is false clones the attributes, text to E else clones the children also.
Parameters: **E-** a html::Element object which is unnamed.(compulsory) **deep-** a Boolean value
Returns: void
Throws: **DataCloneError:** if calling object is unnamed. **TypeMisMatchError:** if E is named.

CLASS html::Error: public std::exception

This is the class which inherits std::exception class and overrides the what() function.

Constructors Summary

Constructor	Description
Error (const char* name, const char* message)	**creates an Error object.**

Members Summary

Member	Description
const char* name	**a read only public member variable holding the name of the error.**
const char* message	**a read only public member variable holding the message of the error.**

Member Functions Summary

Modifier and Type	Function and Description
public const char*	**what() const throw()** returns the error as a concatenation of error name and error message.

Constructors Detail

Error(const char* name, const char* message)
Description: creates an Error Object.
Parameters: **name**- a const char* representing the name of the error. **message**- a const char* representing the message of the error.

Member Functions Detail

public const char* what() const throw()
Description: returns the errors as concatenation of error name and error message.
Parameters: Takes no Parameters.