A Formalization of Safely Composable Web Components

Achim D. Brucker Michael Herzberg

September 9, 2026

*Department of Computer Science, University of Exeter, Exeter, UK
a.brucker@exeter.ac.uk

† Department of Computer Science, The University of Sheffield, Sheffield, UK
msherzberg1@sheffield.ac.uk

Abstract

While the (safely composable) DOM with shadow trees provide the technical basis for defining web components, it does neither defines the concept of web components nor specifies the safety properties that web components should guarantee. Consequently, the standard also does not discuss how or even if the methods for modifying the DOM respect component boundaries. In AFP entry, we present a formally verified model of *safely composable* web components and define safety properties which ensure that different web components can only interact with each other using well-defined interfaces. Moreover, our verification of the application programming interface (API) of the DOM revealed numerous invariants that implementations of the DOM API need to preserve to ensure the integrity of components.

In comparison to the strict standard compliance formalization of Web Components in the AFP entry "DOM Components", the notion of components in this entry (based on "SC DOM" and "Shadow SC DOM") provides much stronger safety guarantees.

Keywords: Web Components, DOM

Contents

1 Introduction **7**

2 Safely Composable Web Components **11**
2.1 Core SC DOM Components (Core_DOM_DOM_Components) 11
2.2 Core SC DOM Components II (Core_DOM_SC_DOM_Components) 23
2.3 Scope Components (Core_DOM_SC_DOM_Components) 23
2.4 Shadow SC DOM Components (Shadow_DOM_DOM_Components) 35
2.5 Shadow root components (Shadow_DOM_DOM_Components) 35
2.6 Shadow SC DOM Components II (Shadow_DOM_SC_DOM_Components) 40
2.7 Shadow root scope components (Shadow_DOM_SC_DOM_Components) 40

1 Introduction

The trend towards ever more complex client-side web applications is unstoppable. Compared to traditional software development, client-side web development lacks a well-established component model which allows easily and safely reusing implementations. The Document Object Model (DOM) essentially defines a tree-like data structure (the *node tree*) for representing documents in general and HTML documents in particular.

Shadow trees are a recent addition to the DOM standard [10] to enable web developers to partition the node tree into "sub-trees." The vision of shadow trees is to enable web developers to provide a library of re-usable and customizable widgets. For example, let us consider a multi-tab view called *Fancy Tab*, which is a simplified version of [1].

The left-hand side of Figure 1.1 shows the rendered output of the widget in use while the right-hand side shows the HTML source code snippet. It provides a custom HTML tag `<fancy-tabs>` using an HTML template that developers can use to include the widget. Its children will be rendered inside the widget, more precisely, inside its *slots* (elements of type `slot`). It has a slot called "title" and a default slot, which receives all children that do not specify a "slot" attribute.

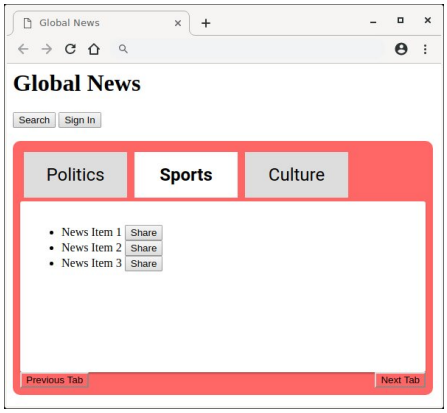
It is important to understand that slotting does *not change* the structure of the DOM (i. e., the underlying pointer graph): instead, slotting is implemented using special element attributes such as "slot," which control the final rendering. The DOM standard specifies methods that inspect the effect of these attributes such as `assigned_slot`, but the majority of DOM methods do not consider the semantics of these attributes and therefore do not traverse into shadow trees.

This provides an important boundary for client-side code. For example, a JavaScript program coming from the widget developer that changes the style attributes of the "Previous Tab" and "Next Tab" buttons in the lower corners of the widget will not affect buttons belonging to other parts coming from outside, i. e., the application of the widget consumer. Similarly, a JavaScript program that changes the styles of buttons outside of Fancy Tab, such as the navigation buttons, will not have any effect on them, even in the case of duplicate identifiers.

Sadly, the DOM standard neither defines the concept of web components nor specifies the safety properties that they should guarantee, not even informally. Consequently, the standard also does not discuss how or even if the methods for modifying the node tree respect component boundaries. Thus, shadow roots are only the very first step in defining a safe web component model.

Earlier [3, 4], we presented a formalization of the "flat" DOM (called Core DOM) without any support for shadow trees or components. We then extended this formalisation with support for shadow trees and slots [7].

In this AFP entries, we use the basis provided by our earlier work for defining a *formally verified model of web components* in general and, in particular, the notion of *weak* and *strong component safety*. For all methods that query, modify, or transform the DOM, we formally analyze their level of component safety. In more detail,



(a) User view

```
<fancy-tabs>
  <button slot="title">Politics</button>
  <button slot="title" selected>Sports</button>
  <button slot="title">Culture</button>
  <section>content panel 1</section>
  <ul>
    <li>News Item 1 <button>Share</button></li>
    <li>News Item 2 <button>Share</button></li>
    <li>News Item 3 <button>Share</button></li>
  </ul>
  <section>content panel 3</section>
</fancy-tabs>
```

(b) Consumer view

Figure 1.1: A simple example: a fancy tab component.

the contribution of this AFP entry is four-fold:

1. We provide a formal model of web components and their safety guarantees to web developers, enabling a compositional development of web applications,
2. for each method, we formally verify that it is either weakly or strongly component safe, or we provide a proof showing that it is not component safe,
3. we fill the gaps in the standard by explicitly formalizing invariants that are left out in the standard. These invariants are required to ensure that methods in the standard preserve a valid node tree. Finally,
4. we present a formal model of the DOM with shadow roots including the methods for querying, modifying, and transforming DOM instances with shadow roots.

Overall, our work gives web developers the guarantee that their code will respect the component boundaries as long as they abstain from or are careful when using certain DOM methods such as `appendChild` or `ownerDocument`.

The rest of this document is automatically generated from the formalization in Isabelle/HOL, i.e., all content is checked by Isabelle (we refer readers interested in a more high-level presentation of the work to [8, 9]. The structure follows the theory dependencies (see Figure 1.2).

Important Note: This document describes the formalization of the *Safely Composable Web Components* (based on the SC DOM), which deviated in one important aspect from the official DOM standard: in the SC DOM, the shadow root is a sub-class of the document class (instead of a base class). This modification results in a stronger notion of web components that provide improved safety properties for the composition of web components. While the SC DOM still passes the compliance test suite as provided by the authors of the DOM standard, its data model is different. We refer readers interested in a formalisation of the standard compliant DOM to the AFP entries “Core_DOM” [2], “Shadow_DOM” [6], and “COM_Components” [5].

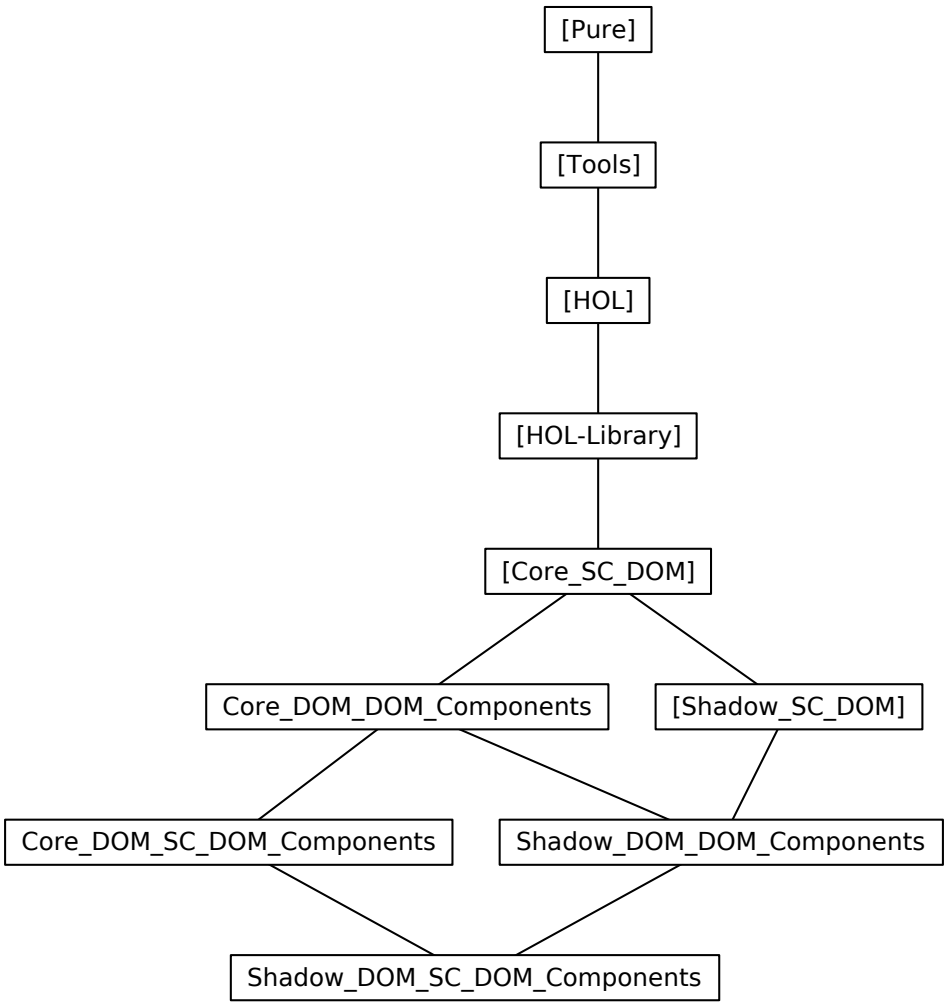


Figure 1.2: The Dependency Graph of the Isabelle Theories.

2 Safely Composable Web Components

2.1 Core SC DOM Components (Core_DOM_DOM_Components)

theory *Core_DOM_DOM_Components*
  imports *Core_SC_DOM.Core_DOM*
begin

2.1.1 Components

locale *l_get_dom_component*$_{Core_DOM}$*_defs =*
  l_get_root_node_defs get_root_node get_root_node_locs +
  l_to_tree_order_defs to_tree_order
  for *get_root_node :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr) prog"*
    and *get_root_node_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"*
    and *to_tree_order :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"*
begin

definition *a_get_dom_component :: "(_) object_ptr ⇒ (_, (_) object_ptr list) dom_prog"*
  where
    "a_get_dom_component ptr = do {
      root ← get_root_node ptr;
      to_tree_order root
    }"

definition *a_is_strongly_dom_component_safe ::*
  "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"
  where
    "a_is_strongly_dom_component_safe S_{arg} S_{result} *h h' = (*
      let removed_pointers = fset (object_ptr_kinds h) - fset (object_ptr_kinds h') in
      let added_pointers = fset (object_ptr_kinds h') - fset (object_ptr_kinds h) in
      let arg_components =
        (⋃ptr ∈ (⋃ptr ∈ S_{arg}*. set |h ⊢ a_get_dom_component ptr|*$_r$*) ∩*
        fset (object_ptr_kinds h). set |h ⊢ a_get_dom_component ptr|$_r$*) in*
      let arg_components' =
        (⋃ptr ∈ (⋃ptr ∈ S_{arg}*. set |h ⊢ a_get_dom_component ptr|*$_r$*) ∩*
        fset (object_ptr_kinds h'). set |h' ⊢ a_get_dom_component ptr|$_r$*) in*
      removed_pointers ⊆ arg_components ∧
      added_pointers ⊆ arg_components' ∧
      S_{result} *⊆ arg_components' ∧*
      (∀ outside_ptr ∈ fset (object_ptr_kinds h) ∩ fset (object_ptr_kinds h') -
        (⋃ptr ∈ S_{arg}*. set |h ⊢ a_get_dom_component ptr|*$_r$*). preserved (get_M outside_ptr id) h h'))"*

definition *a_is_weakly_dom_component_safe ::*
  "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"
  where
    "a_is_weakly_dom_component_safe S_{arg} S_{result} *h h' = (*
      let removed_pointers = fset (object_ptr_kinds h) - fset (object_ptr_kinds h') in
      let added_pointers = fset (object_ptr_kinds h') - fset (object_ptr_kinds h) in
      let arg_components =
        (⋃ptr ∈ (⋃ptr ∈ S_{arg}*. set |h ⊢ a_get_dom_component ptr|*$_r$*) ∩*
        fset (object_ptr_kinds h). set |h ⊢ a_get_dom_component ptr|$_r$*) in*
      let arg_components' =
        (⋃ptr ∈ (⋃ptr ∈ S_{arg}*. set |h ⊢ a_get_dom_component ptr|*$_r$*) ∩*
        fset (object_ptr_kinds h'). set |h' ⊢ a_get_dom_component ptr|$_r$*) in*
      removed_pointers ⊆ arg_components ∧
      S_{result} *⊆ arg_components' ∪ added_pointers ∧*

```
      (∀ outside_ptr ∈ fset (object_ptr_kinds h) ∩ fset (object_ptr_kinds h') -
        (⋃ptr ∈ S_arg. set |h ⊢ a_get_dom_component ptr|_r). preserved (get_M outside_ptr id) h h'))"

lemma "a_is_strongly_dom_component_safe S_arg S_result h h' ⟹ a_is_weakly_dom_component_safe S_arg S_result
h h'"
  ⟨proof⟩

definition is_document_component :: "(_) object_ptr list ⇒ bool"
  where
    "is_document_component c = is_document_ptr_kind (hd c)"

definition is_disconnected_component :: "(_) object_ptr list ⇒ bool"
  where
    "is_disconnected_component c = is_node_ptr_kind (hd c)"
end

global_interpretation l_get_dom_component_Core_DOM_defs get_root_node get_root_node_locs to_tree_order
  defines get_dom_component = a_get_dom_component
    and is_strongly_dom_component_safe = a_is_strongly_dom_component_safe
    and is_weakly_dom_component_safe = a_is_weakly_dom_component_safe
  ⟨proof⟩


locale l_get_dom_component_defs =
  fixes get_dom_component :: "(_) object_ptr ⇒ (_, (_) object_ptr list) dom_prog"
  fixes is_strongly_dom_component_safe ::
    "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"
  fixes is_weakly_dom_component_safe ::
    "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"

locale l_get_dom_component_Core_DOM =
  l_to_tree_order_wf +
  l_get_dom_component_defs +
  l_get_dom_component_Core_DOM_defs +
  l_get_ancestors +
  l_get_ancestors_wf +
  l_get_root_node +
  l_get_root_node_wf_Core_DOM +
  l_get_parent +
  l_get_parent_wf +
  l_get_element_by +
  l_to_tree_order_wf_get_root_node_wf +

  assumes get_dom_component_impl: "get_dom_component = a_get_dom_component"
  assumes is_strongly_dom_component_safe_impl:
    "is_strongly_dom_component_safe = a_is_strongly_dom_component_safe"
  assumes is_weakly_dom_component_safe_impl:
    "is_weakly_dom_component_safe = a_is_weakly_dom_component_safe"
begin
lemmas get_dom_component_def = a_get_dom_component_def[folded get_dom_component_impl]
lemmas is_strongly_dom_component_safe_def =
  a_is_strongly_dom_component_safe_def[folded is_strongly_dom_component_safe_impl]
lemmas is_weakly_dom_component_safe_def =
  a_is_weakly_dom_component_safe_def[folded is_weakly_dom_component_safe_impl]

lemma get_dom_component_ptr_in_heap:
  assumes "h ⊢ ok (get_dom_component ptr)"
  shows "ptr |∈| object_ptr_kinds h"
  ⟨proof⟩

lemma get_dom_component_ok:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "ptr |∈| object_ptr_kinds h"
```

shows *"h ⊢ ok (get_dom_component ptr)"*
⟨*proof*⟩

lemma *get_dom_component_ptr:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →r c"*
shows *"ptr ∈ set c"*
⟨*proof*⟩

lemma *get_dom_component_parent_inside:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →r c"*
assumes *"cast node_ptr ∈ set c"*
assumes *"h ⊢ get_parent node_ptr →r Some parent"*
shows *"parent ∈ set c"*
⟨*proof*⟩

lemma *get_dom_component_subset:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →r c"*
assumes *"ptr' ∈ set c"*
shows *"h ⊢ get_dom_component ptr' →r c"*
⟨*proof*⟩

lemma *get_dom_component_to_tree_order_subset:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ to_tree_order ptr →r nodes"*
assumes *"h ⊢ get_dom_component ptr →r c"*
shows *"set nodes ⊆ set c"*
⟨*proof*⟩

lemma *get_dom_component_to_tree_order:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →r c"*
assumes *"h ⊢ to_tree_order ptr' →r to"*
assumes *"ptr ∈ set to"*
shows *"h ⊢ get_dom_component ptr' →r c"*
⟨*proof*⟩

lemma *get_dom_component_root_node_same:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →r c"*
assumes *"h ⊢ get_root_node ptr →r root_ptr"*
assumes *"x ∈ set c"*
shows *"h ⊢ get_root_node x →r root_ptr"*
⟨*proof*⟩

lemma *get_dom_component_no_overlap:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →r c"*
assumes *"h ⊢ get_dom_component ptr' →r c'"*
shows *"set c ∩ set c' = {} ∨ c = c'"*
⟨*proof*⟩

lemma *get_dom_component_separates_tree_order:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →r c"*
assumes *"h ⊢ to_tree_order ptr →r to"*
assumes *"h ⊢ get_dom_component ptr' →r c'"*
assumes *"ptr' ∉ set c"*
shows *"set to ∩ set c' = {}"*
⟨*proof*⟩

lemma *get_dom_component_separates_tree_order_general:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →r c"*
assumes *"h ⊢ to_tree_order ptr'' →r to''"*
assumes *"ptr'' ∈ set c"*
assumes *"h ⊢ get_dom_component ptr' →r c'"*
assumes *"ptr' ∉ set c"*
shows *"set to'' ∩ set c' = {}"*
⟨*proof*⟩
end

interpretation *i_get_dom_component?: l_get_dom_component*$_{Core_DOM}$
heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs to_tree_order get_parent
get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id get_elements_by_class_name
get_elements_by_tag_name
⟨*proof*⟩
declare *l_get_dom_component*$_{Core_DOM}$*_axioms [instances]*

get_child_nodes

locale *l_get_dom_component_get_child_nodes*$_{Core_DOM}$ *=*
l_get_dom_component$_{Core_DOM}$ *+*
l_get_element_by$_{Core_DOM}$
begin

lemma *get_child_nodes_is_strongly_dom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →r c"*
assumes *"h ⊢ get_child_nodes ptr' →r children"*
assumes *"child ∈ set children"*
shows *"cast child ∈ set c ⟷ ptr' ∈ set c"*
⟨*proof*⟩

lemma *get_child_nodes_get_dom_component:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →r c"*
assumes *"h ⊢ get_child_nodes ptr →r children"*
shows *"cast ‘ set children ⊆ set c"*
⟨*proof*⟩

lemma
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_child_nodes ptr →r children"*
assumes *"h ⊢ get_child_nodes ptr →h h'"*
shows *"is_strongly_dom_component_safe {ptr} (cast ‘ set children) h h'"*
⟨*proof*⟩
end

interpretation *i_get_dom_component_get_child_nodes?: l_get_dom_component_get_child_nodes*$_{Core_DOM}$
heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs to_tree_order get_parent
get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id get_elements_by_class_name
get_elements_by_tag_name first_in_tree_order get_attribute get_attribute_locs
⟨*proof*⟩
declare *l_get_dom_component_get_child_nodes*$_{Core_DOM}$*_axioms [instances]*

get_parent

locale *l_get_dom_component_get_parent*$_{Core_DOM}$ =
  l_get_dom_component$_{Core_DOM}$ +
  l_get_parent$_{Core_DOM}$ +
  l_get_element_by$_{Core_DOM}$
begin

lemma *get_parent_is_strongly_dom_component_safe_step:*
  assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
  assumes *"h ⊢ get_dom_component ptr →$_r$ c"*
  assumes *"h ⊢ get_parent ptr' →$_r$ Some parent"*
  shows *"parent ∈ set c ⟷ cast ptr' ∈ set c"*
  ⟨*proof*⟩

lemma *get_parent_is_strongly_dom_component_safe:*
  assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
  assumes *"h ⊢ get_parent node_ptr →$_r$ Some parent"*
  assumes *"h ⊢ get_parent node_ptr →$_h$ h'"*
  shows *"is_strongly_dom_component_safe {cast node_ptr} {parent} h h'"*
⟨*proof*⟩
end

interpretation *i_get_dom_component_get_parent?: l_get_dom_component_get_parent*$_{Core_DOM}$
  heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs to_tree_order get_parent
  get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
  is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
  get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id get_elements_by_class_name
  get_elements_by_tag_name first_in_tree_order get_attribute get_attribute_locs
  ⟨*proof*⟩
declare *l_get_dom_component_get_parent*$_{Core_DOM}$*_axioms [instances]*

get_root_node

locale *l_get_dom_component_get_root_node*$_{Core_DOM}$ =
  l_get_dom_component$_{Core_DOM}$ +
  l_get_root_node$_{Core_DOM}$ +
  l_get_element_by$_{Core_DOM}$
begin

lemma *get_root_node_is_strongly_dom_component_safe_step:*
  assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
  assumes *"h ⊢ get_dom_component ptr →$_r$ c"*
  assumes *"h ⊢ get_root_node ptr' →$_r$ root"*
  shows *"root ∈ set c ⟷ ptr' ∈ set c"*
⟨*proof*⟩

lemma *get_root_node_is_strongly_dom_component_safe:*
  assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
  assumes *"h ⊢ get_root_node ptr →$_r$ root"*
  assumes *"h ⊢ get_root_node ptr →$_h$ h'"*
  shows *"is_strongly_dom_component_safe {ptr} {root} h h'"*
⟨*proof*⟩
end

interpretation *i_get_dom_component_get_root_node?: l_get_dom_component_get_root_node*$_{Core_DOM}$
  heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs to_tree_order get_parent
  get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
  is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
  get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id get_elements_by_class_name
  get_elements_by_tag_name first_in_tree_order get_attribute get_attribute_locs
  ⟨*proof*⟩

declare *l_get_dom_component_get_root_node$_{Core_DOM}$_axioms [instances]*

get_element_by_id

locale *l_get_dom_component_get_element_by_id$_{Core_DOM}$ =*
l_get_dom_component$_{Core_DOM}$ +
l_first_in_tree_order$_{Core_DOM}$ +
l_to_tree_order$_{Core_DOM}$ +
l_get_element_by$_{Core_DOM}$
begin

lemma *get_element_by_id_is_strongly_dom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →$_r$ c"*
assumes *"h ⊢ get_element_by_id ptr' idd →$_r$ Some result"*
shows *"cast result ∈ set c ⟷ ptr' ∈ set c"*
⟨*proof*⟩

lemma *get_element_by_id_is_strongly_dom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_element_by_id ptr idd →$_r$ Some result"*
assumes *"h ⊢ get_element_by_id ptr idd →$_h$ h'"*
shows *"is_strongly_dom_component_safe {ptr} {cast result} h h'"*
⟨*proof*⟩
end

interpretation *i_get_dom_component_get_element_by_id?: l_get_dom_component_get_element_by_id$_{Core_DOM}$*
heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs to_tree_order get_parent
get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id get_elements_by_class_name
get_elements_by_tag_name first_in_tree_order get_attribute get_attribute_locs
⟨*proof*⟩
declare *l_get_dom_component_get_element_by_id$_{Core_DOM}$_axioms [instances]*

get_elements_by_class_name

locale *l_get_dom_component_get_elements_by_class_name$_{Core_DOM}$ =*
l_get_dom_component$_{Core_DOM}$ +
l_first_in_tree_order$_{Core_DOM}$ +
l_to_tree_order$_{Core_DOM}$ +
l_get_element_by$_{Core_DOM}$
begin

lemma *get_elements_by_class_name_is_strongly_dom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →$_r$ c"*
assumes *"h ⊢ get_elements_by_class_name ptr' idd →$_r$ results"*
assumes *"result ∈ set results"*
shows *"cast result ∈ set c ⟷ ptr' ∈ set c"*
⟨*proof*⟩

lemma *get_elements_by_class_name_pure [simp]:*
"pure (get_elements_by_class_name ptr name) h"
⟨*proof*⟩

lemma *get_elements_by_class_name_is_strongly_dom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_elements_by_class_name ptr name →$_r$ results"*
assumes *"h ⊢ get_elements_by_class_name ptr name →$_h$ h'"*
shows *"is_strongly_dom_component_safe {ptr} (cast ` set results) h h'"*
⟨*proof*⟩

end

interpretation *i_get_dom_component_get_elements_by_class_name?:*
l_get_dom_component_get_elements_by_class_name$_{Core_DOM}$
heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs to_tree_order get_parent
get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id get_elements_by_class_name
get_elements_by_tag_name first_in_tree_order get_attribute get_attribute_locs
⟨*proof*⟩
declare *l_get_dom_component_get_elements_by_class_name*$_{Core_DOM}$*_axioms [instances]*

get_elements_by_tag_name

locale *l_get_dom_component_get_elements_by_tag_name*$_{Core_DOM}$ =
l_get_dom_component$_{Core_DOM}$ +
l_first_in_tree_order$_{Core_DOM}$ +
l_to_tree_order$_{Core_DOM}$ +
l_get_element_by$_{Core_DOM}$
begin

lemma *get_elements_by_tag_name_is_strongly_dom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_dom_component ptr →$_r$ c"*
assumes *"h ⊢ get_elements_by_tag_name ptr' idd →$_r$ results"*
assumes *"result ∈ set results"*
shows *"cast result ∈ set c ⟷ ptr' ∈ set c"*
⟨*proof*⟩

lemma *get_elements_by_tag_name_is_strongly_dom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_elements_by_tag_name ptr name →$_r$ results"*
assumes *"h ⊢ get_elements_by_tag_name ptr name →$_h$ h'"*
shows *"is_strongly_dom_component_safe {ptr} (cast ` set results) h h'"*
⟨*proof*⟩
end

interpretation *i_get_dom_component_get_elements_by_tag_name?:*
l_get_dom_component_get_elements_by_tag_name$_{Core_DOM}$
heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs to_tree_order get_parent
get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id get_elements_by_class_name
get_elements_by_tag_name first_in_tree_order get_attribute get_attribute_locs
⟨*proof*⟩
declare *l_get_dom_component_get_elements_by_tag_name*$_{Core_DOM}$*_axioms [instances]*

remove_child

lemma *remove_child_unsafe: "¬(∀ (h*
:: ('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}) heap
) h' ptr child. heap_is_wellformed h ⟶ type_wf h ⟶ known_ptrs h ⟶
h ⊢ remove_child ptr child →$_h$ h' ⟶ is_weakly_dom_component_safe {ptr, cast child} {} h h')"
⟨*proof*⟩

adopt_node

lemma *adopt_node_unsafe: "¬(∀ (h*
:: ('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},

'CharacterData::{equal,linorder}, 'Document::{equal,linorder}) heap
) h' document_ptr child. heap_is_wellformed h ⟶ type_wf h ⟶ known_ptrs h ⟶ h ⊢ adopt_node document_ptr child →h h' ⟶ is_weakly_dom_component_safe {cast document_ptr, cast child} {} h h')"
⟨*proof*⟩

create_element

lemma *create_element_not_strongly_dom_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder}, 'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder}, 'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder}, 'CharacterData::{equal,linorder}, 'Document::{equal,linorder}) heap" **and**
h' **and** *document_ptr* **and** *new_element_ptr* **and** *tag* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ create_element document_ptr tag →r new_element_ptr →h h'" **and**
"¬ is_strongly_dom_component_safe {cast document_ptr} {cast new_element_ptr} h h'"
⟨*proof*⟩

locale *l_get_dom_component_create_element*$_{Core_DOM}$ =
l_get_dom_component$_{Core_DOM}$ *heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs to_tree_order get_parent get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name +*
l_create_element$_{Core_DOM}$ *get_disconnected_nodes get_disconnected_nodes_locs set_disconnected_nodes set_disconnected_nodes_locs set_tag_name set_tag_name_locs type_wf create_element known_ptr +*
l_get_disconnected_nodes$_{Core_DOM}$ *type_wf get_disconnected_nodes get_disconnected_nodes_locs +*
l_set_disconnected_nodes$_{Core_DOM}$ *type_wf set_disconnected_nodes set_disconnected_nodes_locs +*
l_set_tag_name$_{Core_DOM}$ *type_wf set_tag_name set_tag_name_locs +*
l_new_element_get_disconnected_nodes get_disconnected_nodes get_disconnected_nodes_locs +
l_new_element_get_child_nodes type_wf known_ptr get_child_nodes get_child_nodes_locs +
l_set_tag_name_get_child_nodes type_wf set_tag_name set_tag_name_locs known_ptr get_child_nodes get_child_nodes_locs +
l_create_element_wf$_{Core_DOM}$ *known_ptr known_ptrs type_wf get_child_nodes get_child_nodes_locs get_disconnected_nodes get_disconnected_nodes_locs heap_is_wellformed parent_child_rel set_tag_name set_tag_name_locs set_disconnected_nodes set_disconnected_nodes_locs create_element*
for *known_ptr :: "(_::linorder) object_ptr ⇒ bool"*
and *heap_is_wellformed :: "(_) heap ⇒ bool"*
and *parent_child_rel :: "(_) heap ⇒ ((_) object_ptr × (_) object_ptr) set"*
and *type_wf :: "(_) heap ⇒ bool"*
and *known_ptrs :: "(_) heap ⇒ bool"*
and *to_tree_order :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"*
and *get_parent :: "(_) node_ptr ⇒ ((_) heap, exception, (_) object_ptr option) prog"*
and *get_parent_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"*
and *get_child_nodes :: "(_) object_ptr ⇒ ((_) heap, exception, (_) node_ptr list) prog"*
and *get_child_nodes_locs :: "(_) object_ptr ⇒ ((_) heap ⇒ (_) heap ⇒ bool) set"*
and *get_dom_component :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"*
and *is_strongly_dom_component_safe :: "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"*
and *is_weakly_dom_component_safe :: "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"*
and *get_root_node :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr) prog"*
and *get_root_node_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"*
and *get_ancestors :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"*
and *get_ancestors_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"*
and *get_element_by_id :: "(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr option) prog"*
and *get_elements_by_class_name :: "(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr list) prog"*
and *get_elements_by_tag_name :: "(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr*

```
list) prog"
    and get_disconnected_nodes :: "(_) document_ptr ⇒ ((_) heap, exception, (_) node_ptr list) prog"
    and get_disconnected_nodes_locs :: "(_) document_ptr ⇒ ((_) heap ⇒ (_) heap ⇒ bool) set"
    and set_disconnected_nodes :: "(_) document_ptr ⇒ (_) node_ptr list ⇒ ((_) heap, exception, unit)
prog"
    and set_disconnected_nodes_locs :: "(_) document_ptr ⇒ ((_) heap, exception, unit) prog set"
    and set_tag_name :: "(_) element_ptr ⇒ char list ⇒ ((_) heap, exception, unit) prog"
    and set_tag_name_locs :: "(_) element_ptr ⇒ ((_) heap, exception, unit) prog set"
    and create_element :: "(_) document_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr) prog"
begin

lemma create_element_is_weakly_dom_component_safe_step:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ create_element document_ptr tag →_h h'"
  assumes "ptr ∉ set |h ⊢ get_dom_component (cast document_ptr)|_r"
  assumes "ptr ≠ cast |h ⊢ create_element document_ptr tag|_r"
  shows "preserved (get_M ptr getter) h h'"
⟨proof⟩


lemma create_element_is_weakly_dom_component_safe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ create_element document_ptr tag →_r result"
  assumes "h ⊢ create_element document_ptr tag →_h h'"
  shows "is_weakly_dom_component_safe {cast document_ptr} {cast result} h h'"
⟨proof⟩
end

interpretation i_get_dom_component_create_element?: l_get_dom_component_create_element_Core_DOM
  known_ptr heap_is_wellformed parent_child_rel type_wf known_ptrs to_tree_order get_parent
  get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
  is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
  get_element_by_id get_elements_by_class_name get_elements_by_tag_name get_disconnected_nodes
  get_disconnected_nodes_locs set_disconnected_nodes set_disconnected_nodes_locs set_tag_name
  set_tag_name_locs create_element
  ⟨proof⟩
declare l_get_dom_component_create_element_Core_DOM_axioms [instances]
```

create_character_data

```
lemma create_character_data_not_strongly_dom_component_safe:
  obtains
    h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder},
'element_ptr::{equal,linorder}, 'character_data_ptr::{equal,linorder},
'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}) heap" and
    h' and document_ptr and create_character_datanew_character_data_ptr and tag where
    "heap_is_wellformed h" and "type_wf h" and "known_ptrs h" and
    "h ⊢ create_character_data document_ptr tag →_r create_character_datanew_character_data_ptr →_h h'"
and
    "¬ is_strongly_dom_component_safe {cast document_ptr} {cast create_character_datanew_character_data_ptr}
h h'"
⟨proof⟩

locale l_get_dom_component_create_character_data_Core_DOM =
  l_get_dom_component_Core_DOM heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
  to_tree_order get_parent get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component
  is_strongly_dom_component_safe is_weakly_dom_component_safe get_root_node get_root_node_locs
  get_ancestors get_ancestors_locs get_disconnected_nodes get_disconnected_nodes_locs
  get_element_by_id get_elements_by_class_name get_elements_by_tag_name +
  l_create_character_data_Core_DOM get_disconnected_nodes get_disconnected_nodes_locs set_disconnected_nodes
  set_disconnected_nodes_locs set_val set_val_locs type_wf create_character_data known_ptr +
```

```
  l_get_disconnected_nodes_Core_DOM type_wf get_disconnected_nodes get_disconnected_nodes_locs +
  l_set_disconnected_nodes_Core_DOM type_wf set_disconnected_nodes set_disconnected_nodes_locs +
  l_set_val_Core_DOM type_wf set_val set_val_locs +
  l_create_character_data_wf_Core_DOM known_ptr type_wf get_child_nodes get_child_nodes_locs
  get_disconnected_nodes get_disconnected_nodes_locs heap_is_wellformed parent_child_rel set_val
  set_val_locs set_disconnected_nodes set_disconnected_nodes_locs
  create_character_data known_ptrs
  for known_ptr :: "(_::linorder) object_ptr ⇒ bool"
    and heap_is_wellformed :: "(_) heap ⇒ bool"
    and parent_child_rel :: "(_) heap ⇒ ((_) object_ptr × (_) object_ptr) set"
    and type_wf :: "(_) heap ⇒ bool"
    and known_ptrs :: "(_) heap ⇒ bool"
    and to_tree_order :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"
    and get_parent :: "(_) node_ptr ⇒ ((_) heap, exception, (_) object_ptr option) prog"
    and get_parent_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"
    and get_child_nodes :: "(_) object_ptr ⇒ ((_) heap, exception, (_) node_ptr list) prog"
    and get_child_nodes_locs :: "(_) object_ptr ⇒ ((_) heap ⇒ (_) heap ⇒ bool) set"
    and get_dom_component :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"
    and is_strongly_dom_component_safe :: "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_)
heap ⇒ bool"
    and is_weakly_dom_component_safe :: "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap
⇒ bool"
    and get_root_node :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr) prog"
    and get_root_node_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"
    and get_ancestors :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"
    and get_ancestors_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"
    and get_element_by_id :: "(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr option)
prog"
    and get_elements_by_class_name :: "(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr
list) prog"
    and get_elements_by_tag_name :: "(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr
list) prog"
    and set_val :: "(_) character_data_ptr ⇒ char list ⇒ ((_) heap, exception, unit) prog"
    and set_val_locs :: "(_) character_data_ptr ⇒ ((_) heap, exception, unit) prog set"
    and get_disconnected_nodes :: "(_) document_ptr ⇒ ((_) heap, exception, (_) node_ptr list) prog"
    and get_disconnected_nodes_locs :: "(_) document_ptr ⇒ ((_) heap ⇒ (_) heap ⇒ bool) set"
    and set_disconnected_nodes :: "(_) document_ptr ⇒ (_) node_ptr list ⇒ ((_) heap, exception, unit)
prog"
    and set_disconnected_nodes_locs :: "(_) document_ptr ⇒ ((_) heap, exception, unit) prog set"
    and create_character_data :: "(_) document_ptr ⇒ char list ⇒ ((_) heap, exception, (_) character_data_ptr)
prog"
begin

lemma create_character_data_is_weakly_dom_component_safe_step:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ create_character_data document_ptr text →_h h'"
  assumes "ptr ∉ set |h ⊢ get_dom_component (cast document_ptr)|_r"
  assumes "ptr ≠ cast |h ⊢ create_character_data document_ptr text|_r"
  shows "preserved (get_M ptr getter) h h'"
⟨proof⟩

lemma create_character_data_is_weakly_dom_component_safe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ create_character_data document_ptr text →_r result"
  assumes "h ⊢ create_character_data document_ptr text →_h h'"
  shows "is_weakly_dom_component_safe {cast document_ptr} {cast result} h h'"
⟨proof⟩

end

interpretation i_get_dom_component_create_character_data?: l_get_dom_component_create_character_data_Core_DOM
  known_ptr heap_is_wellformed parent_child_rel type_wf known_ptrs to_tree_order get_parent
  get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
```

is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_element_by_id get_elements_by_class_name get_elements_by_tag_name set_val set_val_locs
get_disconnected_nodes get_disconnected_nodes_locs set_disconnected_nodes set_disconnected_nodes_locs
create_character_data
⟨*proof*⟩
declare *l_get_dom_component_create_character_data*$_{Core_DOM}$*_axioms [instances]*

create_document

lemma *create_document_unsafe: "¬(∀ (h*
:: ('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}) heap
) h' new_document_ptr. heap_is_wellformed h ⟶ type_wf h ⟶ known_ptrs h ⟶
h ⊢ create_document →$_r$ new_document_ptr ⟶ h ⊢ create_document →$_h$ h' ⟶
is_strongly_dom_component_safe {} {cast new_document_ptr} h h')"
⟨*proof*⟩

locale *l_get_dom_component_create_document*$_{Core_DOM}$ =
l_get_dom_component$_{Core_DOM}$ *heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs*
to_tree_order get_parent get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component
is_strongly_dom_component_safe is_weakly_dom_component_safe get_root_node get_root_node_locs
get_ancestors get_ancestors_locs get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id
get_elements_by_class_name get_elements_by_tag_name +
l_create_document$_{Core_DOM}$ *create_document*
for *known_ptr :: "(_::linorder) object_ptr ⇒ bool"*
and *heap_is_wellformed :: "(_) heap ⇒ bool"*
and *parent_child_rel :: "(_) heap ⇒ ((_) object_ptr × (_) object_ptr) set"*
and *type_wf :: "(_) heap ⇒ bool"*
and *known_ptrs :: "(_) heap ⇒ bool"*
and *to_tree_order :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"*
and *get_parent :: "(_) node_ptr ⇒ ((_) heap, exception, (_) object_ptr option) prog"*
and *get_parent_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"*
and *get_child_nodes :: "(_) object_ptr ⇒ ((_) heap, exception, (_) node_ptr list) prog"*
and *get_child_nodes_locs :: "(_) object_ptr ⇒ ((_) heap ⇒ (_) heap ⇒ bool) set"*
and *get_dom_component :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"*
and *is_strongly_dom_component_safe ::*
"(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"
and *is_weakly_dom_component_safe ::*
"(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"
and *get_root_node :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr) prog"*
and *get_root_node_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"*
and *get_ancestors :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"*
and *get_ancestors_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"*
and *get_element_by_id ::*
"(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr option) prog"
and *get_elements_by_class_name ::*
"(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr list) prog"
and *get_elements_by_tag_name ::*
"(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr list) prog"
and *create_document :: "((_) heap, exception, (_) document_ptr) prog"*
begin

lemma *create_document_is_weakly_dom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ create_document →$_h$ h'"*
assumes *"ptr ≠ cast |h ⊢ create_document|$_r$"*
shows *"preserved (get_M*$_{Object}$ *ptr getter) h h'"*
⟨*proof*⟩

lemma *create_document_is_weakly_dom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*

assumes *"h ⊢ create_document →r result"*
assumes *"h ⊢ create_document →h h'"*
shows *"is_weakly_dom_component_safe {} {cast result} h h'"*
⟨*proof*⟩
end

interpretation *i_get_dom_component_create_document?: l_get_dom_component_create_document*$_{Core_DOM}$
known_ptr heap_is_wellformed parent_child_rel type_wf known_ptrs to_tree_order get_parent
get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_element_by_id get_elements_by_class_name get_elements_by_tag_name create_document
⟨*proof*⟩
declare *l_get_dom_component_create_document*$_{Core_DOM}$*_axioms [instances]*

insert_before

lemma *insert_before_unsafe: "¬(∀(h*
:: ('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}) heap
) h' ptr child. heap_is_wellformed h ⟶ type_wf h ⟶ known_ptrs h ⟶
h ⊢ insert_before ptr child None →h h' ⟶ is_weakly_dom_component_safe {ptr, cast child} {} h h')"
⟨*proof*⟩

lemma *insert_before_unsafe2: "¬(∀(h*
:: ('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}) heap
) h' ptr child ref. heap_is_wellformed h ⟶ type_wf h ⟶ known_ptrs h ⟶
h ⊢ insert_before ptr child (Some ref) →h h' ⟶
is_weakly_dom_component_safe {ptr, cast child, cast ref} {} h h')"
⟨*proof*⟩

lemma *append_child_unsafe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}) heap" **and**
h' **and** *ptr* **and** *child* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ append_child ptr child →h h'" **and**
"¬ is_weakly_dom_component_safe {ptr, cast child} {} h h'"
⟨*proof*⟩

get_owner_document

lemma *get_owner_document_unsafe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}) heap" **and**
h' **and** *ptr* **and** *owner_document* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ get_owner_document ptr →r owner_document →h h'" **and**
"¬is_weakly_dom_component_safe {ptr} {cast owner_document} h h'"
⟨*proof*⟩

end

2.2 Core SC DOM Components II (Core_DOM_SC_DOM_Components)

theory *Core_DOM_SC_DOM_Components*
imports
Core_DOM_DOM_Components
begin
declare *[[smt_timeout=2400]]*

2.3 Scope Components (Core_DOM_SC_DOM_Components)

2.3.1 Definition

locale *l_get_scdom_component*$_{Core_DOM}$*_defs =*
l_get_disconnected_nodes_defs get_disconnected_nodes get_disconnected_nodes_locs +
l_get_owner_document_defs get_owner_document +
l_to_tree_order_defs to_tree_order
for *get_owner_document :: "(_::linorder) object_ptr ⇒ ((_) heap, exception, (_) document_ptr) prog"*
and *get_disconnected_nodes :: "(_) document_ptr ⇒ ((_) heap, exception, (_) node_ptr list) prog"*
and *get_disconnected_nodes_locs :: "(_) document_ptr ⇒ ((_) heap ⇒ (_) heap ⇒ bool) set"*
and *to_tree_order :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"*
begin
definition *a_get_scdom_component :: "(_) object_ptr ⇒ (_, (_) object_ptr list) dom_prog"*
where
"a_get_scdom_component ptr = do {
document ← get_owner_document ptr;
disc_nodes ← get_disconnected_nodes document;
tree_order ← to_tree_order (cast document);
disconnected_tree_orders ← map_M (to_tree_order ∘ cast) disc_nodes;
return (tree_order @ (concat disconnected_tree_orders))
}"

definition *a_is_strongly_scdom_component_safe ::*
"(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"
where
"a_is_strongly_scdom_component_safe S_{arg} S_{result} *h h' = (*
let removed_pointers = fset (object_ptr_kinds h) - fset (object_ptr_kinds h') in
let added_pointers = fset (object_ptr_kinds h') - fset (object_ptr_kinds h) in
let arg_components =
(⋃ptr ∈ (⋃ptr ∈ S_{arg}*. set |h ⊢ a_get_scdom_component ptr|*$_r$*) ∩*
fset (object_ptr_kinds h). set |h ⊢ a_get_scdom_component ptr|$_r$*) in*
let arg_components' =
(⋃ptr ∈ (⋃ptr ∈ S_{arg}*. set |h ⊢ a_get_scdom_component ptr|*$_r$*) ∩*
fset (object_ptr_kinds h'). set |h' ⊢ a_get_scdom_component ptr|$_r$*) in*
removed_pointers ⊆ arg_components ∧
added_pointers ⊆ arg_components' ∧
S_{result} *⊆ arg_components' ∧*
(∀outside_ptr ∈ fset (object_ptr_kinds h) ∩ fset (object_ptr_kinds h') -
(⋃ptr ∈ S_{arg}*. set |h ⊢ a_get_scdom_component ptr|*$_r$*). preserved (get_M outside_ptr id) h h'))"*

definition *a_is_weakly_scdom_component_safe ::*
"(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"
where
"a_is_weakly_scdom_component_safe S_{arg} S_{result} *h h' = (*
let removed_pointers = fset (object_ptr_kinds h) - fset (object_ptr_kinds h') in
let added_pointers = fset (object_ptr_kinds h') - fset (object_ptr_kinds h) in
let arg_components =
(⋃ptr ∈ (⋃ptr ∈ S_{arg}*. set |h ⊢ a_get_scdom_component ptr|*$_r$*) ∩*
fset (object_ptr_kinds h). set |h ⊢ a_get_scdom_component ptr|$_r$*) in*
let arg_components' =
(⋃ptr ∈ (⋃ptr ∈ S_{arg}*. set |h ⊢ a_get_scdom_component ptr|*$_r$*) ∩*
fset (object_ptr_kinds h'). set |h' ⊢ a_get_scdom_component ptr|$_r$*) in*

```
      removed_pointers ⊆ arg_components ∧
      S_result ⊆ arg_components' ∪ added_pointers  ∧
      (∀ outside_ptr ∈ fset (object_ptr_kinds h) ∩ fset (object_ptr_kinds h') -
        (⋃ptr ∈ S_arg. set |h ⊢ a_get_scdom_component ptr|_r). preserved (get_M outside_ptr id) h h'))"
end

global_interpretation l_get_scdom_component_Core_DOM_defs get_owner_document get_disconnected_nodes
  get_disconnected_nodes_locs to_tree_order
  defines get_scdom_component = "l_get_scdom_component_Core_DOM_defs.a_get_scdom_component
get_owner_document get_disconnected_nodes to_tree_order"
    and is_strongly_scdom_component_safe = a_is_strongly_scdom_component_safe
    and is_weakly_scdom_component_safe = a_is_weakly_scdom_component_safe
  ⟨proof⟩

locale l_get_scdom_component_defs =
  fixes get_scdom_component :: "(_) object_ptr ⇒ (_, (_) object_ptr list) dom_prog"
  fixes is_strongly_scdom_component_safe ::
    "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"
  fixes is_weakly_scdom_component_safe ::
    "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap ⇒ bool"


locale l_get_scdom_component_Core_DOM =
  l_get_scdom_component_defs +
  l_get_scdom_component_Core_DOM_defs +
  assumes get_scdom_component_impl: "get_scdom_component = a_get_scdom_component"
  assumes is_strongly_scdom_component_safe_impl:
    "is_strongly_scdom_component_safe = a_is_strongly_scdom_component_safe"
  assumes is_weakly_scdom_component_safe_impl:
    "is_weakly_scdom_component_safe = a_is_weakly_scdom_component_safe"
begin
lemmas get_scdom_component_def = a_get_scdom_component_def[folded get_scdom_component_impl]
lemmas is_strongly_scdom_component_safe_def =
  a_is_strongly_scdom_component_safe_def[folded is_strongly_scdom_component_safe_impl]
lemmas is_weakly_scdom_component_safe_def =
  a_is_weakly_scdom_component_safe_def[folded is_weakly_scdom_component_safe_impl]
end

interpretation i_get_scdom_component?: l_get_scdom_component_Core_DOM
  get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe
  get_owner_document get_disconnected_nodes get_disconnected_nodes_locs to_tree_order
  ⟨proof⟩
declare l_get_scdom_component_Core_DOM_axioms [instances]


locale l_get_dom_component_get_scdom_component_Core_DOM =
  l_get_scdom_component_Core_DOM +
  l_get_dom_component_Core_DOM +
  l_heap_is_wellformed +
  l_get_owner_document +
  l_get_owner_document_wf +
  l_get_disconnected_nodes +
  l_to_tree_order +
  l_known_ptr +
  l_known_ptrs +
  l_get_owner_document_wf_get_root_node_wf +
  assumes known_ptr_impl: "known_ptr = DocumentClass.known_ptr"
begin

lemma known_ptr_node_or_document: "known_ptr ptr ⟹ is_node_ptr_kind ptr ∨ is_document_ptr_kind ptr"
  ⟨proof⟩

lemma get_scdom_component_ptr_in_heap2:
```

assumes *"h ⊢ ok (get_scdom_component ptr)"*
shows *"ptr |∈| object_ptr_kinds h"*
⟨*proof*⟩

lemma *get_scdom_component_subset_get_dom_component:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"h ⊢ get_dom_component ptr →r c"*
shows *"set c ⊆ set sc"*
⟨*proof*⟩

lemma *get_scdom_component_ptrs_same_owner_document:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"ptr' ∈ set sc"*
assumes *"h ⊢ get_owner_document ptr →r owner_document"*
shows *"h ⊢ get_owner_document ptr' →r owner_document"*
⟨*proof*⟩

lemma *get_scdom_component_ptrs_same_scope_component:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"ptr' ∈ set sc"*
shows *"h ⊢ get_scdom_component ptr' →r sc"*
⟨*proof*⟩

lemma *get_scdom_component_ok:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"ptr |∈| object_ptr_kinds h"*
shows *"h ⊢ ok (get_scdom_component ptr)"*
⟨*proof*⟩

lemma *get_scdom_component_ptr_in_heap:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"ptr' ∈ set sc"*
shows *"ptr' |∈| object_ptr_kinds h"*
⟨*proof*⟩

lemma *get_scdom_component_contains_get_dom_component:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"ptr' ∈ set sc"*
obtains *c* **where** *"h ⊢ get_dom_component ptr' →r c"* **and** *"set c ⊆ set sc"*
⟨*proof*⟩

lemma *get_scdom_component_owner_document_same:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"ptr' ∈ set sc"*
obtains *owner_document* **where** *"h ⊢ get_owner_document ptr' →r owner_document"* **and** *"cast owner_document ∈ set sc"*
⟨*proof*⟩

lemma *get_scdom_component_different_owner_documents:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_owner_document ptr →r owner_document"*
assumes *"h ⊢ get_owner_document ptr' →r owner_document'"*
assumes *"owner_document ≠ owner_document'"*
shows *"set |h ⊢ get_scdom_component ptr|r ∩ set |h ⊢ get_scdom_component ptr'|r = {}"*
⟨*proof*⟩

lemma *get_scdom_component_ptr:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r c"*
shows *"ptr ∈ set c"*
⟨*proof*⟩
end

locale *l_get_dom_component_get_scdom_component = l_get_owner_document_defs + l_heap_is_wellformed_defs + l_type_wf + l_known_ptrs + l_get_scdom_component_defs + l_get_dom_component_defs +*
assumes *get_scdom_component_subset_get_dom_component:*
"heap_is_wellformed h ⟹ type_wf h ⟹ known_ptrs h ⟹ h ⊢ get_scdom_component ptr →r sc ⟹ h ⊢ get_dom_component ptr →r c ⟹ set c ⊆ set sc"
assumes *get_scdom_component_ptrs_same_scope_component:*
"heap_is_wellformed h ⟹ type_wf h ⟹ known_ptrs h ⟹ h ⊢ get_scdom_component ptr →r sc ⟹ ptr' ∈ set sc ⟹ h ⊢ get_scdom_component ptr' →r sc"
assumes *get_scdom_component_ptrs_same_owner_document:*
"heap_is_wellformed h ⟹ type_wf h ⟹ known_ptrs h ⟹ h ⊢ get_scdom_component ptr →r sc ⟹ ptr' ∈ set sc ⟹ h ⊢ get_owner_document ptr →r owner_document ⟹ h ⊢ get_owner_document ptr' →r owner_documen
assumes *get_scdom_component_ok:*
"heap_is_wellformed h ⟹ type_wf h ⟹ known_ptrs h ⟹ ptr |∈| object_ptr_kinds h ⟹ h ⊢ ok (get_scdom_component ptr)"
assumes *get_scdom_component_ptr_in_heap:*
"heap_is_wellformed h ⟹ type_wf h ⟹ known_ptrs h ⟹ h ⊢ get_scdom_component ptr →r sc ⟹ ptr' ∈ set sc ⟹ ptr' |∈| object_ptr_kinds h"
assumes *get_scdom_component_contains_get_dom_component:*
"(⋀c. h ⊢ get_dom_component ptr' →r c ⟹ set c ⊆ set sc ⟹ thesis) ⟹ heap_is_wellformed h ⟹ type_wf h ⟹ known_ptrs h ⟹ h ⊢ get_scdom_component ptr →r sc ⟹ ptr' ∈ set sc ⟹ thesis"
assumes *get_scdom_component_owner_document_same:*
"(⋀owner_document. h ⊢ get_owner_document ptr' →r owner_document ⟹ cast owner_document ∈ set sc ⟹ thesis) ⟹
heap_is_wellformed h ⟹ type_wf h ⟹ known_ptrs h ⟹ h ⊢ get_scdom_component ptr →r sc ⟹ ptr' ∈ set sc ⟹ thesis"
assumes *get_scdom_component_different_owner_documents:*
"heap_is_wellformed h ⟹ type_wf h ⟹ known_ptrs h ⟹ h ⊢ get_owner_document ptr →r owner_document ⟹ h ⊢ get_owner_document ptr' →r owner_document' ⟹ owner_document ≠ owner_document' ⟹ set |h ⊢ get_scdom_component ptr|r ∩ set |h ⊢ get_scdom_component ptr'|r = {}"

interpretation *i_get_dom_component_get_scdom_component?: l_get_dom_component_get_scdom_component*$_{Core_DOM}$
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_owner_document get_disconnected_nodes get_disconnected_nodes_locs to_tree_order heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs get_parent get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name
⟨*proof*⟩
declare *l_get_dom_component_get_scdom_component*$_{Core_DOM}$*_axioms [instances]*

lemma *get_dom_component_get_scdom_component_is_l_get_dom_component_get_scdom_component [instances]:*
"l_get_dom_component_get_scdom_component get_owner_document heap_is_wellformed type_wf known_ptr known_ptrs get_scdom_component get_dom_component"
⟨*proof*⟩

get_child_nodes

locale *l_get_scdom_component_get_child_nodes*$_{Core_DOM}$ *=*
l_get_dom_component_get_scdom_component +
l_get_scdom_component$_{Core_DOM}$ *+*
l_get_dom_component_get_child_nodes$_{Core_DOM}$
begin
lemma *get_child_nodes_is_strongly_scdom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"h ⊢ get_child_nodes ptr' →r children"*

assumes *"child ∈ set children"*
shows *"cast child ∈ set sc ⟷ ptr' ∈ set sc"*
⟨*proof*⟩

lemma *get_child_nodes_is_strongly_scdom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_child_nodes ptr →r children"*
assumes *"h ⊢ get_child_nodes ptr →h h'"*
shows *"is_strongly_scdom_component_safe {ptr} (cast ` set children) h h'"*
⟨*proof*⟩
end

interpretation *i_get_scdom_component_get_child_nodes?: l_get_scdom_component_get_child_nodes*$_{Core_DOM}$
get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe
get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe get_disconnected_nodes
get_disconnected_nodes_locs to_tree_order get_parent get_parent_locs get_child_nodes
get_child_nodes_locs get_root_node get_root_node_locs get_ancestors get_ancestors_locs get_element_by_id
get_elements_by_class_name get_elements_by_tag_name first_in_tree_order get_attribute get_attribute_locs
⟨*proof*⟩
declare *l_get_scdom_component_get_child_nodes*$_{Core_DOM}$*_axioms [instances]*

get_parent

locale *l_get_scdom_component_get_parent*$_{Core_DOM}$ *=*
l_get_dom_component_get_scdom_component +
l_get_scdom_component$_{Core_DOM}$ *+*
l_get_dom_component_get_parent$_{Core_DOM}$
begin

lemma *get_parent_is_strongly_scdom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"h ⊢ get_parent ptr' →r Some parent"*
shows *"parent ∈ set sc ⟷ cast ptr' ∈ set sc"*
⟨*proof*⟩

lemma *get_parent_is_strongly_scdom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_parent node_ptr →r Some parent"*
assumes *"h ⊢ get_parent node_ptr →h h'"*
shows *"is_strongly_scdom_component_safe {cast node_ptr} {parent} h h'"*
⟨*proof*⟩
end

interpretation *i_get_scdom_component_get_parent?: l_get_scdom_component_get_parent*$_{Core_DOM}$
get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_dom_component
is_strongly_dom_component_safe is_weakly_dom_component_safe get_disconnected_nodes
get_disconnected_nodes_locs to_tree_order get_parent get_parent_locs get_child_nodes get_child_nodes_locs
get_root_node get_root_node_locs get_ancestors get_ancestors_locs get_element_by_id
get_elements_by_class_name get_elements_by_tag_name first_in_tree_order get_attribute get_attribute_locs
⟨*proof*⟩
declare *l_get_scdom_component_get_parent*$_{Core_DOM}$*_axioms [instances]*

get_root_node

locale *l_get_scdom_component_get_root_node*$_{Core_DOM}$ *=*
l_get_dom_component_get_scdom_component +
l_get_scdom_component$_{Core_DOM}$ *+*
l_get_dom_component_get_root_node$_{Core_DOM}$
begin

lemma *get_root_node_is_strongly_scdom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"h ⊢ get_root_node ptr' →r root"*
shows *"root ∈ set sc ⟷ ptr' ∈ set sc"*
⟨*proof*⟩

lemma *get_root_node_is_strongly_scdom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_root_node ptr →r root"*
assumes *"h ⊢ get_root_node ptr →h h'"*
shows *"is_strongly_scdom_component_safe {ptr} {root} h h'"*
⟨*proof*⟩
end

interpretation *i_get_scdom_component_get_root_node?: l_get_scdom_component_get_root_node*$_{Core_DOM}$
get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_dom_component
is_strongly_dom_component_safe is_weakly_dom_component_safe get_disconnected_nodes
get_disconnected_nodes_locs to_tree_order get_parent get_parent_locs get_child_nodes
get_child_nodes_locs get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_element_by_id get_elements_by_class_name get_elements_by_tag_name first_in_tree_order
get_attribute get_attribute_locs
⟨*proof*⟩
declare *l_get_scdom_component_get_root_node*$_{Core_DOM}$*_axioms [instances]*

get_element_by_id

locale *l_get_scdom_component_get_element_by_id*$_{Core_DOM}$ *=*
l_get_dom_component_get_scdom_component +
l_get_scdom_component$_{Core_DOM}$ *+*
l_get_dom_component_get_element_by_id$_{Core_DOM}$
begin

lemma *get_element_by_id_is_strongly_scdom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"h ⊢ get_element_by_id ptr' idd →r Some result"*
shows *"cast result ∈ set sc ⟷ ptr' ∈ set sc"*
⟨*proof*⟩

lemma *get_element_by_id_is_strongly_scdom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_element_by_id ptr idd →r Some result"*
assumes *"h ⊢ get_element_by_id ptr idd →h h'"*
shows *"is_strongly_scdom_component_safe {ptr} {cast result} h h'"*
⟨*proof*⟩
end

interpretation *i_get_scdom_component_get_element_by_id?: l_get_scdom_component_get_element_by_id*$_{Core_DOM}$
get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_dom_component
is_strongly_dom_component_safe is_weakly_dom_component_safe get_disconnected_nodes
get_disconnected_nodes_locs to_tree_order get_parent get_parent_locs get_child_nodes
get_child_nodes_locs get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_element_by_id get_elements_by_class_name get_elements_by_tag_name first_in_tree_order
get_attribute get_attribute_locs
⟨*proof*⟩
declare *l_get_scdom_component_get_element_by_id*$_{Core_DOM}$*_axioms [instances]*

get_elements_by_class_name

locale *l_get_scdom_component_get_elements_by_class_name*$_{Core_DOM}$ *=*

```
  l_get_dom_component_get_scdom_component +
  l_get_scdom_component_Core_DOM +
  l_get_dom_component_get_elements_by_class_name_Core_DOM
begin

lemma get_elements_by_class_name_is_strongly_scdom_component_safe_step:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ get_scdom_component ptr →_r sc"
  assumes "h ⊢ get_elements_by_class_name ptr' idd →_r results"
  assumes "result ∈ set results"
  shows "cast result ∈ set sc ⟷ ptr' ∈ set sc"
  ⟨proof⟩


lemma get_elements_by_class_name_is_strongly_scdom_component_safe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ get_elements_by_class_name ptr idd →_r results"
  assumes "h ⊢ get_elements_by_class_name ptr idd →_h h'"
  shows "is_strongly_scdom_component_safe {ptr} (cast ` set results) h h'"
⟨proof⟩
end

interpretation i_get_scdom_component_get_elements_by_class_name?: l_get_scdom_component_get_elements_by_class_name_
  get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
  get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_dom_component
  is_strongly_dom_component_safe is_weakly_dom_component_safe get_disconnected_nodes
  get_disconnected_nodes_locs to_tree_order get_parent get_parent_locs get_child_nodes get_child_nodes_locs
  get_root_node get_root_node_locs get_ancestors get_ancestors_locs get_element_by_id
  get_elements_by_class_name get_elements_by_tag_name first_in_tree_order get_attribute get_attribute_locs
  ⟨proof⟩
declare l_get_scdom_component_get_element_by_id_Core_DOM_axioms [instances]
```

get_elements_by_tag_name

```
locale l_get_scdom_component_get_elements_by_tag_name_Core_DOM =
  l_get_dom_component_get_scdom_component +
  l_get_scdom_component_Core_DOM +
  l_get_dom_component_get_elements_by_tag_name_Core_DOM
begin

lemma get_elements_by_tag_name_is_strongly_scdom_component_safe_step:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ get_scdom_component ptr →_r sc"
  assumes "h ⊢ get_elements_by_tag_name ptr' idd →_r results"
  assumes "result ∈ set results"
  shows "cast result ∈ set sc ⟷ ptr' ∈ set sc"
  ⟨proof⟩


lemma get_elements_by_tag_name_is_strongly_scdom_component_safe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ get_elements_by_tag_name ptr idd →_r results"
  assumes "h ⊢ get_elements_by_tag_name ptr idd →_h h'"
  shows "is_strongly_scdom_component_safe {ptr} (cast ` set results) h h'"
⟨proof⟩
end

interpretation i_get_scdom_component_get_elements_by_tag_name?:
  l_get_scdom_component_get_elements_by_tag_name_Core_DOM
  get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
  get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe
  get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe
  get_disconnected_nodes get_disconnected_nodes_locs to_tree_order get_parent get_parent_locs
```

```
  get_child_nodes get_child_nodes_locs get_root_node get_root_node_locs get_ancestors
  get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name
  first_in_tree_order get_attribute get_attribute_locs
  ⟨proof⟩
declare l_get_scdom_component_get_element_by_id_Core_DOM_axioms [instances]
```

remove_child

```
locale l_get_scdom_component_remove_child_Core_DOM =
  l_get_dom_component_get_scdom_component +
  l_get_dom_component_Core_DOM +
  l_get_scdom_component_Core_DOM +
  l_remove_child_Core_DOM +
  l_set_child_nodes_Core_DOM +
  l_set_disconnected_nodes_Core_DOM +
  l_get_owner_document_wf +
  l_remove_child_wf2_Core_DOM get_child_nodes get_child_nodes_locs set_child_nodes set_child_nodes_locs
  get_parent get_parent_locs get_owner_document get_disconnected_nodes get_disconnected_nodes_locs
  set_disconnected_nodes set_disconnected_nodes_locs remove_child remove_child_locs remove type_wf
  known_ptr known_ptrs heap_is_wellformed parent_child_rel
begin
lemma remove_child_is_component_unsafe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ remove_child ptr' child →_h h'"
  assumes "ptr ∉ set |h ⊢ get_dom_component ptr'|_r"
  assumes "ptr ∉ set |h ⊢ get_dom_component (cast |h ⊢ get_owner_document (cast_node_ptr2object_ptr child)|_r)|_r"

  shows "preserved (get_M ptr getter) h h'"
⟨proof⟩

lemma remove_child_is_strongly_dom_component_safe_step:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ remove_child ptr' child →_h h'"
  assumes "ptr ∉ set |h ⊢ get_scdom_component ptr'|_r"

  assumes "ptr ∉ set |h ⊢ get_scdom_component (cast child)|_r"
  shows "preserved (get_M ptr getter) h h'"
⟨proof⟩



lemma remove_child_is_strongly_dom_component_safe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ remove_child ptr child →_h h'"
  shows "is_strongly_scdom_component_safe {ptr, cast child} {} h h'"
⟨proof⟩
end

interpretation i_get_scdom_component_remove_child?: l_get_scdom_component_remove_child_Core_DOM
  get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
  get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_dom_component
  is_strongly_dom_component_safe is_weakly_dom_component_safe to_tree_order get_parent
  get_parent_locs get_child_nodes get_child_nodes_locs get_root_node get_root_node_locs get_ancestors
  get_ancestors_locs get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id
  get_elements_by_class_name get_elements_by_tag_name set_child_nodes set_child_nodes_locs
  set_disconnected_nodes set_disconnected_nodes_locs remove_child remove_child_locs remove
  ⟨proof⟩
declare l_get_scdom_component_remove_child_Core_DOM_axioms [instances]
```

adopt_node

```
locale l_get_scdom_component_adopt_node_Core_DOM =
  l_get_dom_component_get_scdom_component +
```

```
  l_adopt_node_Core_DOM +
  l_remove_child_Core_DOM +
  l_set_child_nodes_Core_DOM +
  l_set_disconnected_nodes_Core_DOM +
  l_adopt_node_wf +
  l_get_dom_component_Core_DOM +
  l_get_owner_document_wf +
  l_get_scdom_component_Core_DOM +
  l_adopt_node_wf_Core_DOM +
  l_heap_is_wellformed_Core_DOM
begin
lemma adopt_node_is_component_unsafe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ adopt_node document_ptr node_ptr →_h h'"
  assumes "ptr ∉ set |h ⊢ get_dom_component (cast document_ptr)|_r"
  assumes "ptr ∉ set |h ⊢ get_dom_component (cast node_ptr)|_r"
  assumes "ptr ∉ set |h ⊢ get_dom_component (cast |h ⊢ get_owner_document (cast_node_ptr2object_ptr node_ptr)|_r)|_r"
  shows "preserved (get_M ptr getter) h h'"
⟨proof⟩

lemma adopt_node_is_strongly_dom_component_safe_step:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ adopt_node document_ptr node_ptr →_h h'"
  assumes "ptr ∉ set |h ⊢ get_scdom_component (cast document_ptr)|_r"
  assumes "ptr ∉ set |h ⊢ get_scdom_component (cast node_ptr)|_r"
  shows "preserved (get_M ptr getter) h h'"
⟨proof⟩

lemma adopt_node_is_strongly_scdom_component_safe:
  assumes "heap_is_wellformed h" and type_wf: "type_wf h" and known_ptrs: "known_ptrs h"
  assumes "h ⊢ adopt_node document_ptr child →_h h'"
  shows "is_strongly_scdom_component_safe {cast document_ptr, cast child} {} h h'"
⟨proof⟩
end

interpretation i_get_scdom_component_adopt_node?: l_get_scdom_component_adopt_node_Core_DOM
  get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
  get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_dom_component
  is_strongly_dom_component_safe is_weakly_dom_component_safe get_parent get_parent_locs remove_child
  remove_child_locs get_disconnected_nodes get_disconnected_nodes_locs set_disconnected_nodes
  set_disconnected_nodes_locs adopt_node adopt_node_locs get_child_nodes get_child_nodes_locs
  set_child_nodes set_child_nodes_locs remove to_tree_order get_root_node get_root_node_locs
  get_ancestors get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name
  ⟨proof⟩
declare l_get_scdom_component_adopt_node_Core_DOM_axioms [instances]
```

create_element

```
locale l_get_scdom_component_create_element_Core_DOM =
  l_get_dom_component_get_scdom_component +
  l_get_dom_component_create_element_Core_DOM +
  l_create_element_wf_Core_DOM +
  l_get_scdom_component_Core_DOM +
  l_to_tree_order_Core_DOM +
  l_get_owner_document_Core_DOM
begin

lemma create_element_is_strongly_scdom_component_safe_step:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ create_element document_ptr tag →_h h'"
  assumes "ptr ∉ set |h ⊢ get_scdom_component (cast document_ptr)|_r"
  assumes "ptr ≠ cast |h ⊢ create_element document_ptr tag|_r"
  shows "preserved (get_M ptr getter) h h'"
```

⟨*proof*⟩

lemma *create_element_is_strongly_scdom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ create_element document_ptr tag →$_r$ result"*
assumes *"h ⊢ create_element document_ptr tag →$_h$ h'"*
shows *"is_strongly_scdom_component_safe {cast document_ptr} {cast result} h h'"*
⟨*proof*⟩
end

interpretation *i_get_scdom_component_remove_child?: l_get_scdom_component_remove_child$_{Core_DOM}$*
get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs get_scdom_component
is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe to_tree_order get_parent get_parent_locs get_child_nodes get_child_nodes_locs
get_root_node get_root_node_locs get_ancestors get_ancestors_locs get_disconnected_nodes get_disconnected_nodes_l
get_element_by_id get_elements_by_class_name get_elements_by_tag_name set_child_nodes set_child_nodes_locs
set_disconnected_nodes set_disconnected_nodes_locs remove_child remove_child_locs remove
⟨*proof*⟩
declare *l_get_scdom_component_remove_child$_{Core_DOM}$_axioms [instances]*

create_character_data

locale *l_get_scdom_component_create_character_data$_{Core_DOM}$ =*
l_get_dom_component_get_scdom_component +
l_get_dom_component_create_character_data$_{Core_DOM}$ +
l_get_scdom_component$_{Core_DOM}$ +
l_create_character_data_wf$_{Core_DOM}$ +
l_get_scdom_component$_{Core_DOM}$ +
l_to_tree_order$_{Core_DOM}$ +
l_get_owner_document$_{Core_DOM}$
begin

lemma *create_character_data_is_strongly_dom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ create_character_data document_ptr text →$_h$ h'"*
assumes *"ptr ∉ set |h ⊢ get_scdom_component (cast document_ptr)|$_r$"*
assumes *"ptr ≠ cast |h ⊢ create_character_data document_ptr text|$_r$"*
shows *"preserved (get_M ptr getter) h h'"*
⟨*proof*⟩

lemma *create_character_data_is_strongly_dom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ create_character_data document_ptr text →$_r$ result"*
assumes *"h ⊢ create_character_data document_ptr text →$_h$ h'"*
shows *"is_strongly_scdom_component_safe {cast document_ptr} {cast result} h h'"*
⟨*proof*⟩
end

interpretation *i_get_scdom_component_create_character_data?: l_get_scdom_component_create_character_data$_{Core_DOM}$*
get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs get_scdom_component
is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe to_tree_order get_parent get_parent_locs get_child_nodes get_child_nodes_locs
get_root_node get_root_node_locs get_ancestors get_ancestors_locs get_element_by_id get_elements_by_class_name
get_elements_by_tag_name set_val set_val_locs get_disconnected_nodes get_disconnected_nodes_locs
set_disconnected_nodes set_disconnected_nodes_locs create_character_data
⟨*proof*⟩
declare *l_get_scdom_component_create_character_data$_{Core_DOM}$_axioms [instances]*

create_document

lemma *create_document_not_strongly_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},

```
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}) heap" and
    h' and new_document_ptr where
    "heap_is_wellformed h" and "type_wf h" and "known_ptrs h" and
    "h ⊢ create_document →_r new_document_ptr →_h h'" and
    "¬ is_strongly_scdom_component_safe {} {cast new_document_ptr} h h'"
⟨proof⟩

locale l_get_scdom_component_create_document_Core_DOM =
  l_get_dom_component_get_scdom_component +
  l_get_dom_component_create_document_Core_DOM +
  l_get_scdom_component_Core_DOM
begin

lemma create_document_is_weakly_scdom_component_safe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ create_document →_r result"
  assumes "h ⊢ create_document →_h h'"
  shows "is_weakly_scdom_component_safe {} {cast result} h h'"
⟨proof⟩
end

interpretation i_get_scdom_component_create_document?: l_get_scdom_component_create_document_Core_DOM
  get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
  get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_dom_component
  is_strongly_dom_component_safe is_weakly_dom_component_safe to_tree_order get_parent get_parent_locs
  get_child_nodes get_child_nodes_locs get_root_node get_root_node_locs get_ancestors get_ancestors_locs
  get_element_by_id get_elements_by_class_name get_elements_by_tag_name create_document
  get_disconnected_nodes get_disconnected_nodes_locs
  ⟨proof⟩
declare l_get_scdom_component_create_document_Core_DOM_axioms [instances]
```

insert_before

```
locale l_get_dom_component_insert_before_Core_DOM =
  l_get_dom_component_Core_DOM +
  l_set_child_nodes_Core_DOM +
  l_set_disconnected_nodes_Core_DOM +
  l_remove_child_Core_DOM +
  l_adopt_node_Core_DOM +
  l_insert_before_Core_DOM +
  l_append_child_Core_DOM +
  l_get_owner_document_wf +
  l_get_dom_component_get_scdom_component +
  l_get_scdom_component_Core_DOM +
  l_insert_before_wf_Core_DOM +
  l_set_child_nodes_get_disconnected_nodes +
  l_remove_child +
  l_get_root_node_wf +
  l_set_disconnected_nodes_get_disconnected_nodes_wf +
  l_set_disconnected_nodes_get_ancestors +
  l_get_ancestors_wf +
  l_get_owner_document +
  l_heap_is_wellformed_Core_DOM
begin
lemma insert_before_is_component_unsafe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ insert_before ptr' child ref →_h h'"
  assumes "ptr ∉ set |h ⊢ get_dom_component ptr'|_r"
  assumes "ptr ∉ set |h ⊢ get_dom_component (cast child)|_r"
  assumes "ptr ∉ set |h ⊢ get_dom_component (cast |h ⊢ get_owner_document ptr'|_r)|_r"
  assumes "ptr ∉ set |h ⊢ get_dom_component (cast |h ⊢ get_owner_document (cast child)|_r)|_r"
```

shows *"preserved (get_M ptr getter) h h'"*
⟨*proof*⟩

lemma *insert_before_is_strongly_dom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ insert_before ptr' child ref →_h h'"*
assumes *"ptr ∉ set |h ⊢ get_scdom_component ptr'|_r"*
assumes *"ptr ∉ set |h ⊢ get_scdom_component (cast child)|_r"*
shows *"preserved (get_M ptr getter) h h'"*
⟨*proof*⟩

lemma *insert_before_is_strongly_dom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ insert_before ptr node child →_h h'"*
shows *"is_strongly_scdom_component_safe ({ptr, cast node} ∪ (case child of Some ref ⇒ {cast ref} | None ⇒ {}
)) {} h h'"*
⟨*proof*⟩

lemma *append_child_is_strongly_dom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ append_child ptr' child →_h h'"*
assumes *"ptr ∉ set |h ⊢ get_scdom_component ptr'|_r"*
assumes *"ptr ∉ set |h ⊢ get_scdom_component (cast child)|_r"*
shows *"preserved (get_M ptr getter) h h'"*
⟨*proof*⟩

lemma *append_child_is_strongly_dom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ append_child ptr child →_h h'"*
shows *"is_strongly_scdom_component_safe {ptr, cast child} {} h h'"*
⟨*proof*⟩
end

interpretation *i_get_dom_component_insert_before?: l_get_dom_component_insert_before$_{Core_DOM}$*
heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs to_tree_order get_parent
get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id get_elements_by_class_name
get_elements_by_tag_name set_child_nodes set_child_nodes_locs set_disconnected_nodes set_disconnected_nodes_locs
get_owner_document remove_child remove_child_locs remove adopt_node adopt_node_locs insert_before
insert_before_locs append_child get_scdom_component is_strongly_scdom_component_safe
is_weakly_scdom_component_safe
⟨*proof*⟩
declare *l_get_dom_component_insert_before$_{Core_DOM}$_axioms [instances]*

get_owner_document

locale *l_get_owner_document_scope_component$_{Core_DOM}$ =*
l_get_scdom_component$_{Core_DOM}$ +
l_get_owner_document_wf$_{Core_DOM}$ +
l_get_dom_component$_{Core_DOM}$ +
l_get_dom_component_get_scdom_component +
l_get_owner_document_wf_get_root_node_wf
begin
lemma *get_owner_document_is_strongly_scdom_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →_r sc"*
assumes *"h ⊢ get_owner_document ptr' →_r owner_document"*
shows *"cast owner_document ∈ set sc ⟷ ptr' ∈ set sc"*
⟨*proof*⟩

lemma *get_owner_document_is_strongly_scdom_component_safe:*

assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_owner_document ptr →r owner_document"*
assumes *"h ⊢ get_owner_document ptr →h h'"*
shows *"is_strongly_scdom_component_safe {ptr} {cast owner_document} h h'"*
⟨*proof*⟩
end

interpretation *i_get_owner_document_scope_component?: l_get_owner_document_scope_component*$_{Core_DOM}$
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe
get_owner_document get_disconnected_nodes get_disconnected_nodes_locs to_tree_order known_ptr
known_ptrs type_wf heap_is_wellformed parent_child_rel get_child_nodes get_child_nodes_locs
get_parent get_parent_locs get_ancestors get_ancestors_locs get_root_node get_root_node_locs
get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe get_element_by_id
get_elements_by_class_name get_elements_by_tag_name
⟨*proof*⟩
declare *l_get_owner_document_scope_component*$_{Core_DOM}$*_axioms [instances]*

end

2.4 Shadow SC DOM Components (Shadow_DOM_DOM_Components)

theory *Shadow_DOM_DOM_Components*
imports
Shadow_SC_DOM.Shadow_DOM
Core_DOM_DOM_Components
begin

declare *[[smt_timeout = 1200]]*

2.5 Shadow root components (Shadow_DOM_DOM_Components)

2.5.1 get_component

global_interpretation *l_get_dom_component*$_{Core_DOM}$*_defs get_root_node get_root_node_locs to_tree_order*
defines *get_dom_component = a_get_dom_component*
and *is_strongly_dom_component_safe = a_is_strongly_dom_component_safe*
and *is_weakly_dom_component_safe = a_is_weakly_dom_component_safe*
⟨*proof*⟩

interpretation *i_get_dom_component?: l_get_dom_component*$_{Core_DOM}$
heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs to_tree_order get_parent
get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id get_elements_by_class_name
get_elements_by_tag_name
⟨*proof*⟩
declare *l_get_dom_component*$_{Core_DOM}$*_axioms [instances]*

2.5.2 attach_shadow_root

locale *l_get_dom_component_attach_shadow_root*$_{Core_DOM}$ =
l_get_dom_component$_{Core_DOM}$ *heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs*
to_tree_order get_parent get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component
is_strongly_dom_component_safe is_weakly_dom_component_safe get_root_node get_root_node_locs
get_ancestors get_ancestors_locs get_disconnected_nodes get_disconnected_nodes_locs get_element_by_id
get_elements_by_class_name get_elements_by_tag_name +
l_attach_shadow_root$_{Shadow_DOM}$ *known_ptr set_shadow_root set_shadow_root_locs set_mode set_mode_locs*
attach_shadow_root type_wf get_tag_name get_tag_name_locs get_shadow_root get_shadow_root_locs +
l_set_mode$_{Shadow_DOM}$ *type_wf set_mode set_mode_locs +*
l_set_shadow_root$_{Shadow_DOM}$ *type_wf set_shadow_root set_shadow_root_locs*

```
  for known_ptr :: "(_::linorder) object_ptr ⇒ bool"
    and heap_is_wellformed :: "(_) heap ⇒ bool"
    and parent_child_rel :: "(_) heap ⇒ ((_) object_ptr × (_) object_ptr) set"
    and type_wf :: "(_) heap ⇒ bool"
    and known_ptrs :: "(_) heap ⇒ bool"
    and to_tree_order :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"
    and get_parent :: "(_) node_ptr ⇒ ((_) heap, exception, (_) object_ptr option) prog"
    and get_parent_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"
    and get_child_nodes :: "(_) object_ptr ⇒ ((_) heap, exception, (_) node_ptr list) prog"
    and get_child_nodes_locs :: "(_) object_ptr ⇒ ((_) heap ⇒ (_) heap ⇒ bool) set"
    and get_dom_component :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"
    and get_root_node :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr) prog"
    and get_root_node_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"
    and get_ancestors :: "(_) object_ptr ⇒ ((_) heap, exception, (_) object_ptr list) prog"
    and get_ancestors_locs :: "((_) heap ⇒ (_) heap ⇒ bool) set"
    and get_element_by_id :: "(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr option)
prog"
    and get_elements_by_class_name :: "(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr
list) prog"
    and get_elements_by_tag_name :: "(_) object_ptr ⇒ char list ⇒ ((_) heap, exception, (_) element_ptr
list) prog"
    and set_shadow_root :: "(_) element_ptr ⇒ (_) shadow_root_ptr option ⇒ ((_) heap, exception, unit)
prog"
    and set_shadow_root_locs :: "(_) element_ptr ⇒ ((_) heap, exception, unit) prog set"
    and set_mode :: "(_) shadow_root_ptr ⇒ shadow_root_mode ⇒ ((_) heap, exception, unit) prog"
    and set_mode_locs :: "(_) shadow_root_ptr ⇒ ((_) heap, exception, unit) prog set"
    and attach_shadow_root :: "(_) element_ptr ⇒ shadow_root_mode ⇒ ((_) heap, exception, (_) shadow_root_ptr)
prog"
    and get_disconnected_nodes :: "(_) document_ptr ⇒ ((_) heap, exception, (_) node_ptr list) prog"
    and get_disconnected_nodes_locs :: "(_) document_ptr ⇒ ((_) heap ⇒ (_) heap ⇒ bool) set"
    and is_strongly_dom_component_safe :: "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_)
heap ⇒ bool"
    and is_weakly_dom_component_safe :: "(_) object_ptr set ⇒ (_) object_ptr set ⇒ (_) heap ⇒ (_) heap
⇒ bool"
    and get_tag_name :: "(_) element_ptr ⇒ ((_) heap, exception, char list) prog"
    and get_tag_name_locs :: "(_) element_ptr ⇒ ((_) heap ⇒ (_) heap ⇒ bool) set"
    and get_shadow_root :: "(_) element_ptr ⇒ ((_) heap, exception, (_) shadow_root_ptr option) prog"
    and get_shadow_root_locs :: "(_) element_ptr ⇒ ((_) heap ⇒ (_) heap ⇒ bool) set"
begin

lemma attach_shadow_root_is_weakly_dom_component_safe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ attach_shadow_root element_ptr shadow_root_mode →_h h'"
  assumes "ptr ≠ cast |h ⊢ attach_shadow_root element_ptr shadow_root_mode|_r"
  assumes "ptr ∉ set |h ⊢ get_dom_component (cast element_ptr)|_r"
  shows "preserved (get_M_Object ptr getter) h h'"
⟨proof⟩
end

interpretation i_get_dom_component_attach_shadow_root?: l_get_dom_component_attach_shadow_root_Core_DOM
  known_ptr heap_is_wellformed parent_child_rel type_wf known_ptrs to_tree_order get_parent
  get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component get_root_node get_root_node_locs
  get_ancestors get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name
  set_shadow_root set_shadow_root_locs set_mode set_mode_locs attach_shadow_root get_disconnected_nodes
  get_disconnected_nodes_locs is_strongly_dom_component_safe is_weakly_dom_component_safe get_tag_name
  get_tag_name_locs get_shadow_root get_shadow_root_locs
  ⟨proof⟩
declare l_get_dom_component_attach_shadow_root_Core_DOM_axioms [instances]
```

2.5.3 get_shadow_root

```
locale l_get_shadow_root_component_Shadow_DOM =
  l_get_shadow_root +
```

```
  l_heap_is_wellformed_Shadow_DOM +
  l_get_dom_component_Core_DOM +
  l_get_root_node_Core_DOM +
  l_get_root_node_wf_Core_DOM +
  l_remove_shadow_root_get_child_nodes
begin
lemma get_shadow_root_is_component_unsafe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ get_shadow_root host →_r Some shadow_root_ptr"
  shows "set |h ⊢ get_dom_component (cast host)|_r ∩ set |h ⊢ get_dom_component (cast shadow_root_ptr)|_r
= {}"
⟨proof⟩
end

interpretation i_get_shadow_root_component?: l_get_shadow_root_component_Shadow_DOM
  type_wf get_shadow_root get_shadow_root_locs get_child_nodes get_child_nodes_locs
  get_disconnected_nodes get_disconnected_nodes_locs get_tag_name get_tag_name_locs known_ptr
  heap_is_wellformed parent_child_rel heap_is_wellformed_Core_DOM get_host get_host_locs
  get_disconnected_document get_disconnected_document_locs known_ptrs to_tree_order get_parent
  get_parent_locs get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe
  get_root_node get_root_node_locs get_ancestors get_ancestors_locs get_element_by_id
  get_elements_by_class_name get_elements_by_tag_name remove_shadow_root remove_shadow_root_locs
  ⟨proof⟩
declare l_get_shadow_root_component_Shadow_DOM_axioms [instances]
```

2.5.4 get_host

```
locale l_get_host_component_Shadow_DOM =
  l_heap_is_wellformed_Shadow_DOM +
  l_get_host +
  l_get_dom_component_Core_DOM +
  l_get_shadow_root_component_Shadow_DOM
begin
lemma get_host_is_component_unsafe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ get_host shadow_root_ptr →_r host"
  shows "set |h ⊢ get_dom_component (cast host)|_r ∩ set |h ⊢ get_dom_component (cast shadow_root_ptr)|_r
= {}"
⟨proof⟩
end

interpretation i_get_host_component?: l_get_host_component_Shadow_DOM
  get_child_nodes get_child_nodes_locs get_disconnected_nodes get_disconnected_nodes_locs
  get_shadow_root get_shadow_root_locs get_tag_name get_tag_name_locs known_ptr type_wf heap_is_wellformed
  parent_child_rel heap_is_wellformed_Core_DOM get_host get_host_locs get_disconnected_document
  get_disconnected_document_locs known_ptrs to_tree_order get_parent get_parent_locs get_dom_component
  is_strongly_dom_component_safe is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors
  get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name remove_shadow_root
  remove_shadow_root_locs
  ⟨proof⟩
declare l_get_host_component_Shadow_DOM_axioms [instances]
```

2.5.5 get_root_node_si

```
locale l_get_dom_component_get_root_node_si_Shadow_DOM =
  l_get_root_node_si_wf_Shadow_DOM +
  l_get_dom_component_Core_DOM
begin

lemma get_root_node_si_is_component_unsafe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ get_root_node_si ptr' →_r root"
  shows "set |h ⊢ get_dom_component ptr'|_r = set |h ⊢ get_dom_component root|_r ∨
```

```
set |h ⊢ get_dom_component ptr'|_r ∩ set |h ⊢ get_dom_component root|_r = {}"
```
⟨*proof*⟩
end

interpretation *i_get_dom_component_get_root_node_si?: l_get_dom_component_get_root_node_si*$_{Shadow_DOM}$
type_wf known_ptr known_ptrs get_parent get_parent_locs get_child_nodes get_child_nodes_locs
get_host get_host_locs get_ancestors_si get_ancestors_si_locs get_root_node_si get_root_node_si_locs
get_disconnected_nodes get_disconnected_nodes_locs get_shadow_root get_shadow_root_locs get_tag_name
get_tag_name_locs heap_is_wellformed parent_child_rel heap_is_wellformed$_{Core_DOM}$ *get_disconnected_document*
get_disconnected_document_locs to_tree_order get_dom_component is_strongly_dom_component_safe
is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs
get_element_by_id get_elements_by_class_name get_elements_by_tag_name
⟨*proof*⟩
declare *l_get_dom_component_get_root_node_si*$_{Shadow_DOM}$*_axioms [instances]*

2.5.6 get_assigned_nodes

lemma *get_shadow_root_not_weakly_dom_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}, 'Shadowroot::{equal,linorder}) heap" **and**
element_ptr **and** *shadow_root_ptr_opt* **and** *h'* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ get_shadow_root element_ptr →_r shadow_root_ptr_opt →_h h'" **and**
"¬ is_weakly_dom_component_safe {cast element_ptr} (cast ‘ set_option shadow_root_ptr_opt) h h'"
⟨*proof*⟩

lemma *assigned_nodes_not_weakly_dom_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder}, 'CharacterData::{equal,linorder},
'Document::{equal,linorder}, 'Shadowroot::{equal,linorder}) heap" **and**
node_ptr **and** *nodes* **and** *h'* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ assigned_nodes node_ptr →_r nodes →_h h'" **and**
"¬ is_weakly_dom_component_safe {cast node_ptr} (cast ‘ set nodes) h h'"
⟨*proof*⟩

lemma *get_composed_root_node_not_weakly_dom_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}, 'Shadowroot::{equal,linorder}) heap" **and**
ptr **and** *root* **and** *h'* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ get_root_node_si ptr →_r root →_h h'" **and**
"¬ is_weakly_dom_component_safe {ptr} {root} h h'"
⟨*proof*⟩

lemma *assigned_slot_not_weakly_dom_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder}, 'CharacterData::{equal,linorder},
'Document::{equal,linorder}, 'Shadowroot::{equal,linorder}) heap" **and**
node_ptr **and** *slot_opt* **and** *h'* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ assigned_slot node_ptr →_r slot_opt →_h h'" **and**
"¬ is_weakly_dom_component_safe {cast node_ptr} (cast ‘ set_option slot_opt) h h'"

⟨*proof*⟩

locale *l_assigned_nodes_component*$_{Shadow_DOM}$ =
l_get_tag_name +
l_get_child_nodes +
l_heap_is_wellformed$_{Shadow_DOM}$ +
l_find_slot$_{Shadow_DOM}$ +
l_assigned_nodes$_{Shadow_DOM}$ +
l_assigned_nodes_wf$_{Shadow_DOM}$ +
l_get_dom_component$_{Core_DOM}$ +
l_adopt_node$_{Shadow_DOM}$ +
l_remove_child$_{Core_DOM}$ +
l_remove_child_wf2 +
l_insert_before_wf +
l_insert_before_wf2 +
l_append_child$_{Core_DOM}$ +
l_append_child_wf$_{Shadow_DOM}$ +
l_set_disconnected_nodes_get_tag_name +
l_set_shadow_root_get_child_nodes +
l_set_child_nodes_get_tag_name +
l_get_shadow_root_component$_{Shadow_DOM}$ +
l_remove_shadow_root$_{Shadow_DOM}$ +
l_remove_shadow_root_get_tag_name +
l_set_disconnected_nodes_get_shadow_root +
l_set_child_nodes_get_shadow_root +
l_remove_shadow_root_wf$_{Shadow_DOM}$
begin
lemma *find_slot_is_component_unsafe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ find_slot open_flag node_ptr →$_r$ Some slot"*
shows *"set |h ⊢ get_dom_component (cast node_ptr)|$_r$ ∩ set |h ⊢ get_dom_component (cast slot)|$_r$ = {}"*
⟨*proof*⟩

lemma *assigned_nodes_is_component_unsafe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ assigned_nodes element_ptr →$_r$ nodes"*
assumes *"node_ptr ∈ set nodes"*
shows *"set |h ⊢ get_dom_component (cast element_ptr)|$_r$ ∩ set |h ⊢ get_dom_component (cast node_ptr)|$_r$ = {}"*
⟨*proof*⟩

lemma *flatten_dom_assigned_nodes_become_children:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ flatten_dom →$_h$ h'"*
assumes *"h ⊢ assigned_nodes slot →$_r$ nodes"*
assumes *"nodes ≠ []"*
shows *"h' ⊢ get_child_nodes (cast slot) →$_r$ nodes"*
⟨*proof*⟩
end

interpretation *i_assigned_nodes_component?: l_assigned_nodes_component*$_{Shadow_DOM}$
type_wf get_tag_name get_tag_name_locs known_ptr get_child_nodes get_child_nodes_locs
get_disconnected_nodes get_disconnected_nodes_locs get_shadow_root get_shadow_root_locs
heap_is_wellformed parent_child_rel heap_is_wellformed$_{Core_DOM}$ *get_host get_host_locs get_disconnected_document*
get_disconnected_document_locs get_parent get_parent_locs get_mode get_mode_locs get_attribute
get_attribute_locs first_in_tree_order find_slot assigned_slot known_ptrs to_tree_order assigned_nodes
assigned_nodes_flatten flatten_dom get_root_node get_root_node_locs remove insert_before insert_before_locs
append_child remove_shadow_root remove_shadow_root_locs set_shadow_root set_shadow_root_locs remove_child
remove_child_locs get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe get_ancestors
get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name get_owner_document
set_disconnected_nodes set_disconnected_nodes_locs get_ancestors_di get_ancestors_di_locs
adopt_node adopt_node_locs adopt_node$_{Core_DOM}$ *adopt_node_locs*$_{Core_DOM}$ *set_child_nodes set_child_nodes_locs*

⟨*proof*⟩
declare *l_assigned_nodes_component*$_{Shadow_DOM}$*_axioms [instances]*

get_owner_document

locale *l_get_owner_document_component*$_{Shadow_DOM}$ =
l_get_owner_document_wf$_{Shadow_DOM}$ +
l_get_dom_component$_{Core_DOM}$
begin
lemma *get_owner_document_is_component_unsafe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_owner_document ptr →$_r$ owner_document"*
assumes *"¬is_document_ptr_kind |h ⊢ get_root_node ptr|$_r$"*
shows *"set |h ⊢ get_dom_component ptr|$_r$ ∩ set |h ⊢ get_dom_component (cast owner_document)|$_r$ = {}"*
⟨*proof*⟩

end

interpretation *i_get_owner_document_component?: l_get_owner_document_component*$_{Shadow_DOM}$
type_wf get_disconnected_nodes get_disconnected_nodes_locs known_ptr get_child_nodes get_child_nodes_locs
DocumentClass.known_ptr get_parent get_parent_locs DocumentClass.type_wf get_root_node get_root_node_locs
CD.a_get_owner_document get_host get_host_locs get_owner_document get_shadow_root get_shadow_root_locs
get_tag_name get_tag_name_locs heap_is_wellformed parent_child_rel heap_is_wellformed$_{Core_DOM}$
get_disconnected_document get_disconnected_document_locs known_ptrs get_ancestors get_ancestors_locs to_tree_orde
get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe get_element_by_id
get_elements_by_class_name get_elements_by_tag_name
⟨*proof*⟩
declare *l_get_owner_document_component*$_{Shadow_DOM}$*_axioms [instances]*

definition *is_shadow_root_component :: "(_) object_ptr list ⇒ bool"*
where
"is_shadow_root_component c = is_shadow_root_ptr_kind (hd c)"

end

2.6 Shadow SC DOM Components II (Shadow_DOM_SC_DOM_Components)

theory *Shadow_DOM_SC_DOM_Components*
imports
Core_DOM_SC_DOM_Components
Shadow_DOM_DOM_Components
begin

2.7 Shadow root scope components (Shadow_DOM_SC_DOM_Components)

2.7.1 get_scope_component

global_interpretation *l_get_scdom_component*$_{Core_DOM}$*_defs get_owner_document get_disconnected_nodes*
get_disconnected_nodes_locs to_tree_order
defines *get_scdom_component = a_get_scdom_component*
and *is_strongly_scdom_component_safe = a_is_strongly_scdom_component_safe*
and *is_weakly_scdom_component_safe = a_is_weakly_scdom_component_safe*
⟨*proof*⟩
interpretation *i_get_scdom_component?: l_get_scdom_component*$_{Core_DOM}$
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe
get_owner_document get_disconnected_nodes get_disconnected_nodes_locs to_tree_order
⟨*proof*⟩
declare *l_get_scdom_component*$_{Core_DOM}$*_axioms [instances]*

get_component

locale *l_get_dom_component_get_scdom_component*$_{Shadow_DOM}$ =
  l_get_scdom_component$_{Core_DOM}$ +
  l_get_dom_component$_{Core_DOM}$ +
  l_heap_is_wellformed +
  l_get_owner_document +
  l_get_owner_document_wf +
  l_get_disconnected_nodes +
  l_to_tree_order +
  l_known_ptr +
  l_known_ptrs +
  l_get_owner_document_wf_get_root_node_wf +
  assumes *known_ptr_impl: "known_ptr = ShadowRootClass.known_ptr"*
begin

lemma *known_ptr_node_or_document: "known_ptr ptr ⟹ is_node_ptr_kind ptr ∨ is_document_ptr_kind ptr"*
  ⟨*proof*⟩

lemma *get_scdom_component_subset_get_dom_component:*
  assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
  assumes *"h ⊢ get_scdom_component ptr →r sc"*
  assumes *"h ⊢ get_dom_component ptr →r c"*
  shows *"set c ⊆ set sc"*
⟨*proof*⟩

lemma *get_scdom_component_ptrs_same_owner_document:*
  assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
  assumes *"h ⊢ get_scdom_component ptr →r sc"*
  assumes *"ptr' ∈ set sc"*
  assumes *"h ⊢ get_owner_document ptr →r owner_document"*
  shows *"h ⊢ get_owner_document ptr' →r owner_document"*
⟨*proof*⟩

lemma *get_scdom_component_ptrs_same_scope_component:*
  assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
  assumes *"h ⊢ get_scdom_component ptr →r sc"*
  assumes *"ptr' ∈ set sc"*
  shows *"h ⊢ get_scdom_component ptr' →r sc"*
⟨*proof*⟩

lemma *get_scdom_component_ok:*
  assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
  assumes *"ptr |∈| object_ptr_kinds h"*
  shows *"h ⊢ ok (get_scdom_component ptr)"*
  ⟨*proof*⟩

lemma *get_scdom_component_ptr_in_heap:*
  assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
  assumes *"h ⊢ get_scdom_component ptr →r sc"*
  assumes *"ptr' ∈ set sc"*
  shows *"ptr' |∈| object_ptr_kinds h"*
  ⟨*proof*⟩

lemma *get_scdom_component_contains_get_dom_component:*
  assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
  assumes *"h ⊢ get_scdom_component ptr →r sc"*
  assumes *"ptr' ∈ set sc"*
  obtains *c* **where** *"h ⊢ get_dom_component ptr' →r c"* **and** *"set c ⊆ set sc"*
⟨*proof*⟩

lemma *get_scdom_component_owner_document_same:*

assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"ptr' ∈ set sc"*
obtains *owner_document* **where** *"h ⊢ get_owner_document ptr' →r owner_document"* **and** *"cast owner_document ∈ set sc"*
⟨*proof*⟩

lemma *get_scdom_component_different_owner_documents:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_owner_document ptr →r owner_document"*
assumes *"h ⊢ get_owner_document ptr' →r owner_document'"*
assumes *"owner_document ≠ owner_document'"*
shows *"set |h ⊢ get_scdom_component ptr|r ∩ set |h ⊢ get_scdom_component ptr'|r = {}"*
⟨*proof*⟩
end

interpretation *i_get_dom_component_get_scdom_component?: l_get_dom_component_get_scdom_component*$_{Shadow_DOM}$
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_owner_document get_disconnected_nodes get_disconnected_nodes_locs to_tree_order heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs get_parent get_parent_locs get_child_nodes get_child_nodes_locs get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name
⟨*proof*⟩
declare *l_get_dom_component_get_scdom_component*$_{Shadow_DOM}$*_axioms [instances]*

lemma *get_dom_component_get_scdom_component_is_l_get_dom_component_get_scdom_component [instances]:*
"l_get_dom_component_get_scdom_component get_owner_document heap_is_wellformed type_wf known_ptr known_ptrs get_scdom_component get_dom_component"
⟨*proof*⟩

2.7.2 attach_shadow_root

lemma *attach_shadow_root_not_strongly_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder}, 'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder}, 'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder}, 'CharacterData::{equal,linorder}, 'Document::{equal,linorder}, 'ShadowRoot::{equal,linorder}) heap" **and**
h' **and** *host* **and** *new_shadow_root_ptr* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ attach_shadow_root host m →r new_shadow_root_ptr →h h'" **and**
"¬ is_strongly_scdom_component_safe {cast host} {cast new_shadow_root_ptr} h h'"
⟨*proof*⟩

locale *l_get_scdom_component_attach_shadow_root*$_{Core_DOM}$ =
l_get_dom_component_get_scdom_component +
l_get_dom_component_attach_shadow_root$_{Core_DOM}$ *+*
l_get_dom_component_get_scdom_component$_{Shadow_DOM}$
begin
lemma *attach_shadow_root_is_weakly_component_safe_step:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ attach_shadow_root element_ptr shadow_root_mode →h h'"*
assumes *"ptr ≠ cast |h ⊢ attach_shadow_root element_ptr shadow_root_mode|r"*
assumes *"ptr ∉ set |h ⊢ get_scdom_component (cast element_ptr)|r"*
shows *"preserved (get_M*$_{Object}$ *ptr getter) h h'"*
⟨*proof*⟩

lemma *attach_shadow_root_is_weakly_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ attach_shadow_root element_ptr shadow_root_mode →r result"*
assumes *"h ⊢ attach_shadow_root element_ptr shadow_root_mode →h h'"*
shows *"is_weakly_scdom_component_safe {cast element_ptr} {cast result} h h'"*

⟨*proof*⟩
end

interpretation *i_get_scdom_component_attach_shadow_root?: l_get_scdom_component_attach_shadow_root*$_{Core_DOM}$
get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe
get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe to_tree_order
get_parent get_parent_locs get_child_nodes get_child_nodes_locs get_root_node get_root_node_locs
get_ancestors get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name
set_shadow_root set_shadow_root_locs set_mode set_mode_locs attach_shadow_root get_disconnected_nodes
get_disconnected_nodes_locs get_tag_name get_tag_name_locs get_shadow_root get_shadow_root_locs
⟨*proof*⟩
declare *l_get_scdom_component_attach_shadow_root*$_{Core_DOM}$*_axioms [instances]*

2.7.3 get_shadow_root

lemma *get_shadow_root_not_weakly_scdom_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder}, 'CharacterData::{equal,linorder},
'Document::{equal,linorder}, 'Shadowroot::{equal,linorder}) heap" **and**
element_ptr **and** *shadow_root_ptr_opt* **and** *h'* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ get_shadow_root_safe element_ptr →$_r$ *shadow_root_ptr_opt →*$_h$ *h'"* **and**
"¬ is_weakly_scdom_component_safe {cast element_ptr} (cast ‘ set_option shadow_root_ptr_opt) h h'"
⟨*proof*⟩

locale *l_get_shadow_root_scope_component*$_{Shadow_DOM}$ =
l_get_dom_component_get_scdom_component +
l_get_shadow_root_component$_{Shadow_DOM}$ *+*
l_create_document +
l_get_owner_document_wf$_{Shadow_DOM}$ *+*
l_heap_is_wellformed$_{Shadow_DOM}$ *+*
l_get_mode +
l_get_scdom_component$_{Core_DOM}$ *+*
l_get_shadow_root_safe$_{Shadow_DOM}$ *+*
assumes *known_ptrs_impl: "known_ptrs = a_known_ptrs"*
begin
lemma *get_shadow_root_components_disjunct:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →*$_r$ *sc"*
assumes *"h ⊢ get_shadow_root host →*$_r$ *Some shadow_root_ptr"*
shows *"set |h ⊢ get_scdom_component (cast host)|*$_r$ *∩ set | h ⊢ get_scdom_component (cast shadow_root_ptr)|*$_r$
= {}"
⟨*proof*⟩

lemma *get_shadow_root_is_strongly_scdom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_shadow_root_safe element_ptr →*$_r$ *shadow_root_ptr_opt →*$_h$ *h'"*
assumes *"∀ shadow_root_ptr ∈ fset (shadow_root_ptr_kinds h). h ⊢ get_mode shadow_root_ptr →*$_r$ *Closed"*
shows *"is_strongly_scdom_component_safe {cast element_ptr} (cast ‘ set_option shadow_root_ptr_opt) h h'"*
⟨*proof*⟩
end

interpretation *i_get_shadow_root_scope_component?: l_get_shadow_root_scope_component*$_{Shadow_DOM}$
get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr known_ptrs
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe get_dom_component
is_strongly_dom_component_safe is_weakly_dom_component_safe get_shadow_root get_shadow_root_locs
get_child_nodes get_child_nodes_locs get_disconnected_nodes get_disconnected_nodes_locs get_tag_name
get_tag_name_locs heap_is_wellformed$_{Core_DOM}$ *get_host get_host_locs get_disconnected_document*
get_disconnected_document_locs to_tree_order get_parent get_parent_locs get_root_node get_root_node_locs
get_ancestors get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name

remove_shadow_root remove_shadow_root_locs create_document DocumentClass.known_ptr DocumentClass.type_wf
CD.a_get_owner_document get_mode get_mode_locs get_shadow_root_safe get_shadow_root_safe_locs
⟨*proof*⟩
declare *l_get_shadow_root_scope_component*$_{Shadow_DOM}$*_axioms [instances]*

2.7.4 get_host

lemma *get_host_not_weakly_scdom_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}, 'Shadowroot::{equal,linorder}) heap" **and**
shadow_root_ptr **and** *element_ptr* **and** *h'* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ get_host shadow_root_ptr →r element_ptr →h h'" **and**
"¬ is_weakly_scdom_component_safe {cast shadow_root_ptr} {cast element_ptr} h h'"
⟨*proof*⟩

locale *l_get_host_scope_component*$_{Shadow_DOM}$ =
l_get_shadow_root_scope_component$_{Shadow_DOM}$ +
l_get_host_component$_{Shadow_DOM}$
begin
lemma *get_host_components_disjunct:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_scdom_component ptr →r sc"*
assumes *"h ⊢ get_host shadow_root_ptr →r host"*
shows *"set |h ⊢ get_scdom_component (cast host)|r ∩ set | h ⊢ get_scdom_component (cast shadow_root_ptr)|r*
= {}"
⟨*proof*⟩
end

interpretation *i_get_host_scope_component?: l_get_host_scope_component*$_{Shadow_DOM}$
get_owner_document heap_is_wellformed parent_child_rel type_wf known_ptr
known_ptrs get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe
get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe get_shadow_root
get_shadow_root_locs get_child_nodes get_child_nodes_locs get_disconnected_nodes get_disconnected_nodes_locs
get_tag_name get_tag_name_locs heap_is_wellformed$_{Core_DOM}$ *get_host get_host_locs get_disconnected_document*
get_disconnected_document_locs to_tree_order get_parent get_parent_locs get_root_node get_root_node_locs
get_ancestors get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name
remove_shadow_root remove_shadow_root_locs create_document DocumentClass.known_ptr DocumentClass.type_wf
CD.a_get_owner_document get_mode get_mode_locs get_shadow_root_safe get_shadow_root_safe_locs
⟨*proof*⟩
declare *l_get_host_scope_component*$_{Shadow_DOM}$*_axioms [instances]*

2.7.5 get_root_node_si

lemma *get_composed_root_node_not_weakly_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder},
'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder},
'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder},
'CharacterData::{equal,linorder}, 'Document::{equal,linorder}, 'Shadowroot::{equal,linorder}) heap" **and**
ptr **and** *root* **and** *h'* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ get_root_node_si ptr →r root →h h'" **and**
"¬ is_weakly_scdom_component_safe {ptr} {root} h h'"
⟨*proof*⟩

locale *l_get_scdom_component_get_root_node_si*$_{Shadow_DOM}$ =
l_get_dom_component_get_root_node_si$_{Shadow_DOM}$ +
l_get_dom_component_get_scdom_component
begin

lemma *get_root_node_si_is_component_unsafe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ get_root_node_si ptr' →r root"*
shows *"set |h ⊢ get_scdom_component ptr'|r = set |h ⊢ get_scdom_component root|r ∨ set |h ⊢ get_scdom_component ptr'|r ∩ set |h ⊢ get_scdom_component root|r = {}"*
⟨*proof*⟩
end

interpretation *i_get_scdom_component_get_root_node_si?: l_get_scdom_component_get_root_node_si*$_{Shadow_DOM}$
type_wf known_ptr known_ptrs get_parent get_parent_locs get_child_nodes get_child_nodes_locs get_host get_host_locs get_ancestors_si get_ancestors_si_locs get_root_node_si get_root_node_si_locs get_disconnected_node get_disconnected_nodes_locs get_shadow_root get_shadow_root_locs get_tag_name get_tag_name_locs heap_is_wellforme parent_child_rel heap_is_wellformed$_{Core_DOM}$ *get_disconnected_document get_disconnected_document_locs to_tree_order get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe get_root_node get_root_node_locs get_ancestors get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name get_owner_document get_scdom_component is_strongly_scdom_component_safe*
⟨*proof*⟩
declare *l_get_scdom_component_get_root_node_si*$_{Shadow_DOM}$*_axioms [instances]*

2.7.6 get_assigned_nodes

lemma *assigned_nodes_not_weakly_scdom_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder}, 'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder}, 'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder}, 'CharacterData::{equal,linorder}, 'Document::{equal,linorder}, 'Shadowroot::{equal,linorder}) heap" **and**
node_ptr **and** *nodes* **and** *h'* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ assigned_nodes node_ptr →r nodes →h h'" **and**
"¬ is_weakly_scdom_component_safe {cast node_ptr} (cast ' set nodes) h h'"
⟨*proof*⟩

lemma *assigned_slot_not_weakly_component_safe:*
obtains
h :: "('object_ptr::{equal,linorder}, 'node_ptr::{equal,linorder}, 'element_ptr::{equal,linorder}, 'character_data_ptr::{equal,linorder}, 'document_ptr::{equal,linorder}, 'shadow_root_ptr::{equal,linorder}, 'Object::{equal,linorder}, 'Node::{equal,linorder}, 'Element::{equal,linorder}, 'CharacterData::{equal,linorder}, 'Document::{equal,linorder}, 'Shadowroot::{equal,linorder}) heap" **and**
node_ptr **and** *slot_opt* **and** *h'* **where**
"heap_is_wellformed h" **and** *"type_wf h"* **and** *"known_ptrs h"* **and**
"h ⊢ assigned_slot node_ptr →r slot_opt →h h'" **and**
"¬ is_weakly_scdom_component_safe {cast node_ptr} (cast ' set_option slot_opt) h h'"
⟨*proof*⟩

locale *l_assigned_nodes_scope_component*$_{Shadow_DOM}$ =
l_assigned_nodes_component$_{Shadow_DOM}$ +
l_get_shadow_root_scope_component$_{Shadow_DOM}$ +
l_find_slot$_{Shadow_DOM}$
begin

lemma *find_slot_is_component_unsafe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ find_slot open_flag node_ptr →r Some slot"*
shows *"set |h ⊢ get_scdom_component (cast node_ptr)|r ∩ set |h ⊢ get_scdom_component (cast slot)|r = {}"*
⟨*proof*⟩

lemma *assigned_nodes_is_component_unsafe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*

assumes *"h ⊢ assigned_nodes element_ptr →r nodes"*
assumes *"node_ptr ∈ set nodes"*
shows *"set |h ⊢ get_scdom_component (cast element_ptr)|r ∩ set |h ⊢ get_scdom_component (cast node_ptr)|r = {}"*
⟨*proof*⟩

lemma *assigned_slot_is_strongly_scdom_component_safe:*
assumes *"heap_is_wellformed h"* **and** *"type_wf h"* **and** *"known_ptrs h"*
assumes *"h ⊢ assigned_slot element_ptr →r slot_opt →h h'"*
assumes *"∀shadow_root_ptr ∈ fset (shadow_root_ptr_kinds h). h ⊢ get_mode shadow_root_ptr →r Closed"*
shows *"is_strongly_scdom_component_safe {cast element_ptr} (cast ‘ set_option slot_opt) h h'"*
⟨*proof*⟩
end

interpretation *i_assigned_nodes_scope_component?: l_assigned_nodes_scope_component*$_{Shadow_DOM}$
type_wf get_tag_name get_tag_name_locs known_ptr get_child_nodes get_child_nodes_locs
get_disconnected_nodes get_disconnected_nodes_locs get_shadow_root get_shadow_root_locs
heap_is_wellformed parent_child_rel heap_is_wellformed$_{Core_DOM}$ *get_host get_host_locs*
get_disconnected_document get_disconnected_document_locs get_parent get_parent_locs
get_mode get_mode_locs get_attribute get_attribute_locs first_in_tree_order find_slot
assigned_slot known_ptrs to_tree_order assigned_nodes assigned_nodes_flatten flatten_dom
get_root_node get_root_node_locs remove insert_before insert_before_locs append_child
remove_shadow_root remove_shadow_root_locs set_shadow_root set_shadow_root_locs remove_child
remove_child_locs get_dom_component is_strongly_dom_component_safe is_weakly_dom_component_safe
get_ancestors get_ancestors_locs get_element_by_id get_elements_by_class_name get_elements_by_tag_name
get_owner_document set_disconnected_nodes set_disconnected_nodes_locs get_ancestors_di get_ancestors_di_locs
adopt_node adopt_node_locs adopt_node$_{Core_DOM}$ *adopt_node_locs*$_{Core_DOM}$ *set_child_nodes set_child_nodes_locs*
get_scdom_component is_strongly_scdom_component_safe is_weakly_scdom_component_safe create_document
DocumentClass.known_ptr DocumentClass.type_wf CD.a_get_owner_document get_shadow_root_safe
get_shadow_root_safe_locs
⟨*proof*⟩
declare *l_assigned_nodes_scope_component*$_{Shadow_DOM}$*_axioms [instances]*
end

Bibliography

[1] E. Bidelman. Shadow dom v1: Self-contained web components, May 2017. URL https://developers.google.com/web/fundamentals/getting-started/primers/shadowdom.

[2] A. D. Brucker and M. Herzberg. The core DOM. *Archive of Formal Proofs*, dec 2018. ISSN 2150-914x. URL https://www.brucker.ch/bibliography/abstract/brucker.ea-afp-core-dom-2018-a. http://www.isa-afp.org/entries/Core_DOM.html, Formal proof development.

[3] A. D. Brucker and M. Herzberg. A formal semantics of the core DOM in Isabelle/HOL. In *Proceedings of the Web Programming, Design, Analysis, And Implementation (WPDAI) track at WWW 2018*, 2018. URL https://www.brucker.ch/bibliography/abstract/brucker.ea-fdom-2018.

[4] A. D. Brucker and M. Herzberg. The safely composable DOM. *Archive of Formal Proofs*, Sept. 2020. ISSN 2150-914x. URL http://www.brucker.ch/bibliography/abstract/brucker.ea-afp-core-sc-dom-2020. http://www.isa-afp.org/entries/Core_SC_DOM.html, Formal proof development.

[5] A. D. Brucker and M. Herzberg. A formalization of web components. *Archive of Formal Proofs*, Sept. 2020. ISSN 2150-914x. URL http://www.brucker.ch/bibliography/abstract/brucker.ea-afp-dom-components-2020. http://www.isa-afp.org/entries/DOM_Components.html, Formal proof development.

[6] A. D. Brucker and M. Herzberg. Shadow dom: A formal model of the document object model with shadow roots. *Archive of Formal Proofs*, Sept. 2020. ISSN 2150-914x. URL http://www.brucker.ch/bibliography/abstract/brucker.ea-afp-shadow-dom-2020. http://www.isa-afp.org/entries/Shadow_DOM.html, Formal proof development.

[7] A. D. Brucker and M. Herzberg. Shadow sc dom: A formal model of the safelty composable document object model with shadow roots. *Archive of Formal Proofs*, Sept. 2020. ISSN 2150-914x. URL http://www.brucker.ch/bibliography/abstract/brucker.ea-afp-shadow-sc-dom-2020. http://www.isa-afp.org/entries/Shadow_SC_DOM.html, Formal proof development.

[8] A. D. Brucker and M. Herzberg. A formally verified model of web components. In S.-S. Jongmans and F. Arbab, editors, *Formal Aspects of Component Software (FACS)*, number 12018 in Lecture Notes in Computer Science. Springer-Verlag, Heidelberg, 2020. ISBN 3-540-25109-X. doi: 10.1007/978-3-030-40914-2_3. URL http://www.brucker.ch/bibliography/abstract/brucker.ea-web-components-2019.

[9] M. Herzberg. *Formal Foundations for Provably Safe Web Components*. PhD thesis, The University of Sheffield, 2020.

[10] WHATWG. DOM – living standard, Feb. 2019. URL https://dom.spec.whatwg.org/commit-snapshots/7fa83673430f767d329406d0aed901f296332216/. Last Updated 11 February 2019.