

# A Formalization of Web Components

Achim D. Brucker

Michael Herzberg

February 6, 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 DOM with shadow trees provide the technical basis for defining web components, the DOM standard 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 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.

**Keywords:** Web Components, DOM



# Contents

- 1 Introduction** **7**
  
- 2 Web Components** **11**
  - 2.1 DOM Components (Core\_DOM\_Components) . . . . . 11
  - 2.2 Shadow Root Components (Shadow\_DOM\_Components) . . . . . 23
  
- 3 Example** **29**
  - 3.1 Testing fancy\_tabs (fancy\_tabs) . . . . . 29



# 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 [7] 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.

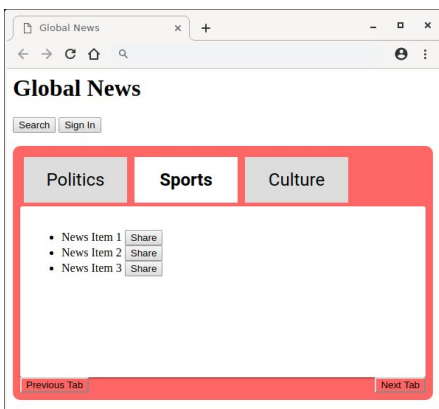
It is important to understand that slotting does *not change* the structure of the DOM (i. e., the underlying pointer graph): instead, slotting is implemented using special element attributes such as “slot,” which control the final rendering. The DOM standard specifies methods that inspect the effect 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 [2, 3], 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 [4].

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.

## 1 Introduction

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 [5, 6]). The structure follows the theory dependencies (see Figure 1.2).

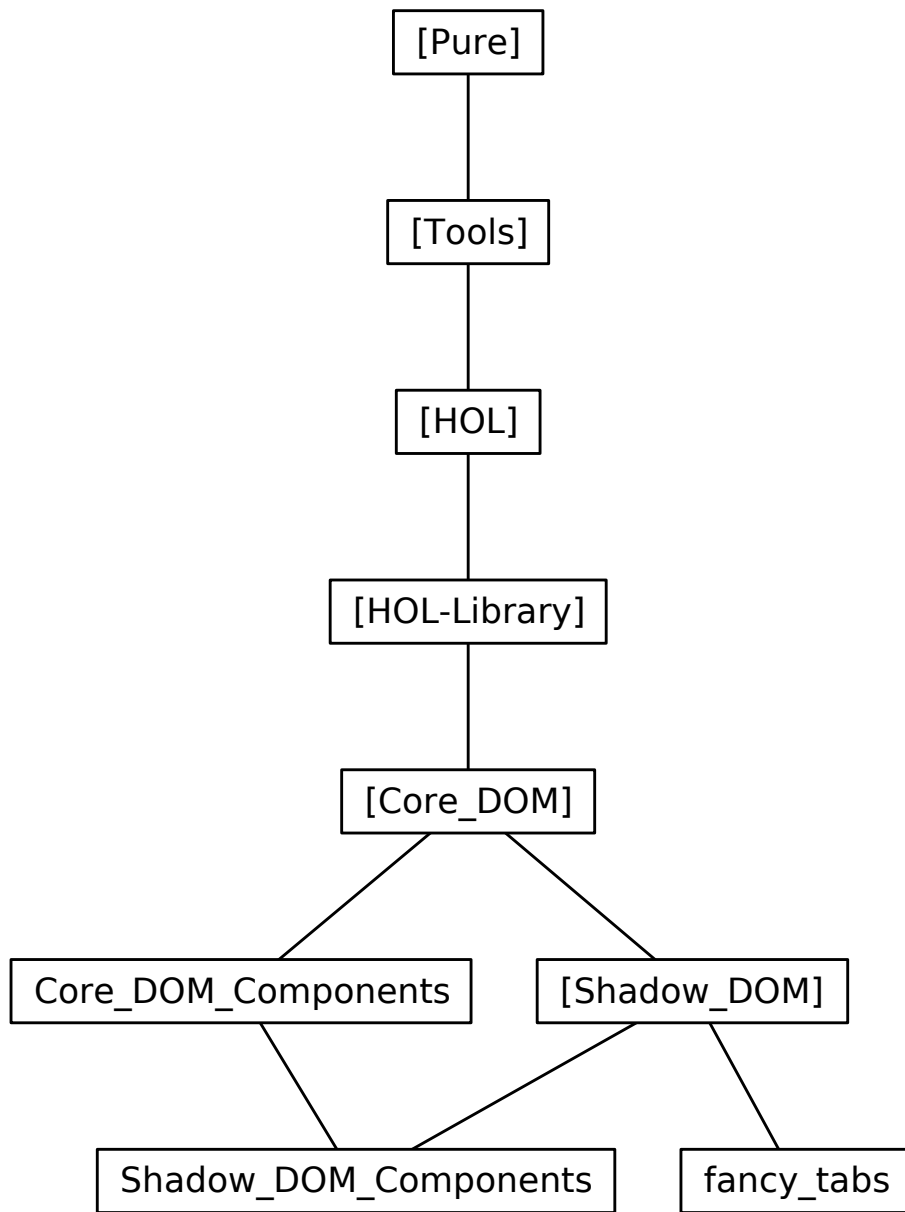


Figure 1.2: The Dependency Graph of the Isabelle Theories.



# 2 Web Components

## 2.1 DOM Components (Core\_DOM\_Components)

```
theory Core_DOM_Components
  imports Core_DOM.Core_DOM
begin

locale l_get_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_component :: "(>) object_ptr ⇒ (>, (>) object_ptr list) dom_prog"
  where
    "a_get_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_component ptr|r) ∩ fset (object_ptr_kinds h).
           set |h ⊢ a_get_component ptr|r) in
      let arg_components' =
          (∪ ptr ∈ (∪ ptr ∈ S_arg. set |h ⊢ a_get_component ptr|r) ∩ fset (object_ptr_kinds h').
           set |h' ⊢ a_get_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_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_component ptr|r) ∩ fset (object_ptr_kinds h).
           set |h ⊢ a_get_component ptr|r) in
      let arg_components' =
          (∪ ptr ∈ (∪ ptr ∈ S_arg. set |h ⊢ a_get_component ptr|r) ∩ fset (object_ptr_kinds h').
           set |h' ⊢ a_get_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') -
```

$(\bigcup ptr \in S_{arg}. set \mid h \vdash a\_get\_component \ ptr \mid_r). preserved (get\_M \ outside\_ptr \ id) \ h \ h')$ "

**lemma** "a\_is\_strongly\_dom\_component\_safe  $S_{arg} \ S_{result} \ h \ h'$   $\implies$  a\_is\_weakly\_dom\_component\_safe  $S_{arg} \ S_{result} \ h \ h'$ "

*<proof>*

**definition** is\_document\_component :: "(\_) object\_ptr list  $\Rightarrow$  bool"

where

"is\_document\_component c = is\_document\_ptr\_kind (hd c)"

**definition** is\_disconnected\_component :: "(\_) object\_ptr list  $\Rightarrow$  bool"

where

"is\_disconnected\_component c = is\_node\_ptr\_kind (hd c)"

end

**global interpretation** l\_get\_component<sub>Core\_DOM\_defs</sub> get\_root\_node get\_root\_node\_locs to\_tree\_order

defines get\_component = a\_get\_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\_component\_defs =

fixes get\_component :: "(\_) object\_ptr  $\Rightarrow$  (\_, (\_) object\_ptr list) dom\_prog"

fixes is\_strongly\_dom\_component\_safe ::

"(\_) object\_ptr set  $\Rightarrow$  (\_) object\_ptr set  $\Rightarrow$  (\_) heap  $\Rightarrow$  (\_) heap  $\Rightarrow$  bool"

fixes is\_weakly\_dom\_component\_safe ::

"(\_) object\_ptr set  $\Rightarrow$  (\_) object\_ptr set  $\Rightarrow$  (\_) heap  $\Rightarrow$  (\_) heap  $\Rightarrow$  bool"

**locale** l\_get\_component<sub>Core\_DOM</sub> =

l\_to\_tree\_order\_wf +

l\_get\_component\_defs +

l\_get\_component<sub>Core\_DOM\_defs</sub> +

l\_get\_ancestors +

l\_get\_ancestors\_wf +

l\_get\_root\_node +

l\_get\_root\_node\_wf<sub>Core\_DOM</sub> +

l\_get\_parent +

l\_get\_parent\_wf +

l\_get\_element\_by +

l\_to\_tree\_order\_wf\_get\_root\_node\_wf +

assumes get\_component\_impl: "get\_component = a\_get\_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\_component\_def = a\_get\_component\_def[folded get\_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  $\vdash$  ok (get\_component ptr)"

shows "ptr  $\in$  object\_ptr\_kinds h"

*<proof>*

**lemma** get\_component\_ok:

assumes "heap\_is\_wellformed h" and "type\_wf h" and "known\_ptrs h"

assumes "ptr  $\in$  object\_ptr\_kinds h"

shows "h  $\vdash$  ok (get\_component ptr)"

*<proof>*

lemma `get_dom_component_ptr`:  
 assumes "heap\_is\_wellformed h" and "type\_wf h" and "known\_ptrs h"  
 assumes "h ⊢ get\_component ptr →<sub>r</sub> c"  
 shows "ptr ∈ set c"  
*<proof>*

lemma `get_component_parent_inside`:  
 assumes "heap\_is\_wellformed h" and "type\_wf h" and "known\_ptrs h"  
 assumes "h ⊢ get\_component ptr →<sub>r</sub> c"  
 assumes "cast node\_ptr ∈ set c"  
 assumes "h ⊢ get\_parent node\_ptr →<sub>r</sub> Some parent"  
 shows "parent ∈ set c"  
*<proof>*

lemma `get_component_subset`:  
 assumes "heap\_is\_wellformed h" and "type\_wf h" and "known\_ptrs h"  
 assumes "h ⊢ get\_component ptr →<sub>r</sub> c"  
 assumes "ptr' ∈ set c"  
 shows "h ⊢ get\_component ptr' →<sub>r</sub> c"  
*<proof>*

lemma `get_component_to_tree_order_subset`:  
 assumes "heap\_is\_wellformed h" and "type\_wf h" and "known\_ptrs h"  
 assumes "h ⊢ to\_tree\_order ptr →<sub>r</sub> nodes"  
 assumes "h ⊢ get\_component ptr →<sub>r</sub> c"  
 shows "set nodes ⊆ set c"  
*<proof>*

lemma `get_component_to_tree_order`:  
 assumes "heap\_is\_wellformed h" and "type\_wf h" and "known\_ptrs h"  
 assumes "h ⊢ get\_component ptr →<sub>r</sub> c"  
 assumes "h ⊢ to\_tree\_order ptr' →<sub>r</sub> to"  
 assumes "ptr ∈ set to"  
 shows "h ⊢ get\_component ptr' →<sub>r</sub> c"  
*<proof>*

lemma `get_component_root_node_same`:  
 assumes "heap\_is\_wellformed h" and "type\_wf h" and "known\_ptrs h"  
 assumes "h ⊢ get\_component ptr →<sub>r</sub> c"  
 assumes "h ⊢ get\_root\_node ptr →<sub>r</sub> root\_ptr"  
 assumes "x ∈ set c"  
 shows "h ⊢ get\_root\_node x →<sub>r</sub> 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\_component ptr →<sub>r</sub> c"  
 assumes "h ⊢ get\_component ptr' →<sub>r</sub> c'"  
 shows "set c ∩ set c' = {} ∨ c = c'"  
*<proof>*

lemma `get_component_separates_tree_order`:  
 assumes "heap\_is\_wellformed h" and "type\_wf h" and "known\_ptrs h"  
 assumes "h ⊢ get\_component ptr →<sub>r</sub> c"  
 assumes "h ⊢ to\_tree\_order ptr →<sub>r</sub> to"  
 assumes "h ⊢ get\_component ptr' →<sub>r</sub> c'"  
 assumes "ptr' ∉ set c"  
 shows "set to ∩ set c' = {}"  
*<proof>*

```

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

interpretation i_get_component?: l_get_componentCore_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_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_componentCore_DOM_axioms [instances]

```

### 2.1.1 get\_child\_nodes

```

locale l_get_component_get_child_nodesCore_DOM =
  l_get_componentCore_DOM +
  l_get_element_byCore_DOM
begin

lemma get_dom_component_get_child_nodes:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ get_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_component:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h ⊢ get_component ptr →r c"
  assumes "h ⊢ get_child_nodes ptr →r children"
  shows "cast ' set children ⊆ set c"
⟨proof⟩

lemma get_child_nodes_strongly_dom_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_dom_component_safe {ptr} (cast ' set children) h h'"
⟨proof⟩
end

interpretation i_get_component_get_child_nodes?: l_get_component_get_child_nodesCore_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_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_component_get_child_nodesCore_DOM_axioms [instances]

```

### 2.1.2 get\_parent

```

locale l_get_component_get_parentCore_DOM =
  l_get_componentCore_DOM +
  l_get_parentCore_DOM +
  l_get_element_byCore_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_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_component_get_parent?: l_get_component_get_parentCore_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_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_component_get_parentCore_DOM_axioms [instances]

```

### 2.1.3 get\_root\_node

```

locale l_get_component_get_root_nodeCore_DOM =
  l_get_componentCore_DOM +
  l_get_root_nodeCore_DOM +
  l_get_element_byCore_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_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_component_get_root_node?: l_get_component_get_root_nodeCore_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_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_component_get_root_nodeCore_DOM_axioms [instances]
```

### 2.1.4 get\_element\_by\_id

```
locale l_get_component_get_element_by_idCore_DOM =
  l_get_componentCore_DOM +
  l_first_in_tree_orderCore_DOM +
  l_to_tree_orderCore_DOM +
  l_get_element_byCore_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_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_component_get_element_by_id?: l_get_component_get_element_by_idCore_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_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_component_get_element_by_idCore_DOM_axioms [instances]
```

### 2.1.5 get\_elements\_by\_class\_name

```
locale l_get_component_get_elements_by_class_nameCore_DOM =
  l_get_componentCore_DOM +
  l_first_in_tree_orderCore_DOM +
  l_to_tree_orderCore_DOM +
  l_get_element_byCore_DOM
begin

lemma get_elements_by_class_name_result_in_tree_order:
  assumes "h ⊢ get_elements_by_class_name ptr name →r results"
  assumes "h ⊢ to_tree_order ptr →r to"
  assumes "result ∈ set results"
  shows "cast result ∈ set to"
⟨proof⟩

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_component ptr →r c"
  assumes "h ⊢ get_elements_by_class_name ptr' name →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_component_get_elements_by_class_name?: l_get_component_get_elements_by_class_nameCore_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_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_component_get_elements_by_class_nameCore_DOM_axioms [instances]

```

### 2.1.6 get\_elements\_by\_tag\_name

```

locale l_get_component_get_elements_by_tag_nameCore_DOM =
  l_get_componentCore_DOM +
  l_first_in_tree_orderCore_DOM +
  l_to_tree_orderCore_DOM +
  l_get_element_byCore_DOM
begin

```

```

lemma get_elements_by_tag_name_result_in_tree_order:
  assumes "h ⊢ get_elements_by_tag_name ptr name →r results"
  assumes "h ⊢ to_tree_order ptr →r to"
  assumes "result ∈ set results"
  shows "cast result ∈ set to"
⟨proof⟩

```

```

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_component ptr →r c"
  assumes "h ⊢ get_elements_by_tag_name ptr' name →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_component_get_elements_by_tag_name?: l_get_component_get_elements_by_tag_nameCore_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_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_component_get_elements_by_tag_nameCore_DOM_axioms [instances]

```

### 2.1.7 remove\_child

```

lemma remove_child_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 ptr and child where
  "heap_is_wellformed h" and "type_wf h" and "known_ptrs h" and
  "h  $\vdash$  remove_child ptr child  $\rightarrow_h$  h'" and
  " $\neg$  is_strongly_dom_component_safe { ptr, cast child} {} h h'"
<proof>

```

### 2.1.8 adopt\_node

```

lemma adopt_node_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 child where
  "heap_is_wellformed h" and "type_wf h" and "known_ptrs h" and
  "h  $\vdash$  adopt_node document_ptr child  $\rightarrow_h$  h'" and
  " $\neg$  is_strongly_dom_component_safe {cast document_ptr, cast child} {} h h'"
<proof>

```

### 2.1.9 create\_element

```

lemma create_element_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 document_ptr and new_element_ptr and tag where
  "heap_is_wellformed h" and "type_wf h" and "known_ptrs h" and
  "h  $\vdash$  create_element document_ptr tag  $\rightarrow_r$  new_element_ptr  $\rightarrow_h$  h'" and
  " $\neg$  is_strongly_dom_component_safe {cast document_ptr} {cast new_element_ptr} h h'"
<proof>

```

```

locale l_get_component_create_elementCore_DOM =
  l_get_componentCore_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_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_elementCore_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_nodesCore_DOM type_wf get_disconnected_nodes get_disconnected_nodes_locs +
  l_set_disconnected_nodesCore_DOM type_wf set_disconnected_nodes set_disconnected_nodes_locs +
  l_set_tag_nameCore_DOM type_wf set_tag_name set_tag_name_locs +
  l_new_element_get_child_nodesCore_DOM type_wf known_ptr get_child_nodes get_child_nodes_locs +
  l_new_element_get_disconnected_nodes get_disconnected_nodes get_disconnected_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_set_tag_name_get_disconnected_nodes type_wf set_tag_name set_tag_name_locs
  get_disconnected_nodes get_disconnected_nodes_locs +
  l_set_disconnected_nodes type_wf set_disconnected_nodes set_disconnected_nodes_locs +
  l_set_disconnected_nodes_get_child_nodes set_disconnected_nodes set_disconnected_nodes_locs
  get_child_nodes get_child_nodes_locs +
  l_set_disconnected_nodes_get_disconnected_nodes type_wf get_disconnected_nodes
  get_disconnected_nodes_locs set_disconnected_nodes set_disconnected_nodes_locs +
  l_new_element type_wf +
  l_create_element_wfCore_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_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 set"
  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_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 →h h'"
  shows "is_weakly_dom_component_safe {cast document_ptr} {cast result} h h'"
⟨proof⟩
end

interpretation i_get_component_create_element?: l_get_component_create_elementCore_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_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_component_create_elementCore_DOM_axioms [instances]

```

## 2.1.10 create\_character\_data

```

lemma create_character_data_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 document_ptr and new_character_data_ptr and val where
      "heap_is_wellformed h" and "type_wf h" and "known_ptrs h" and
      "h ⊢ create_character_data document_ptr val →r new_character_data_ptr →h h'" and
      "¬ is_strongly_dom_component_safe {cast document_ptr} {cast new_character_data_ptr} h h'"
⟨proof⟩

```

```

locale l_get_component_create_character_dataCore_DOM =
  l_get_componentCore_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_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_dataCore_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_nodesCore_DOM type_wf get_disconnected_nodes get_disconnected_nodes_locs +
  l_set_disconnected_nodesCore_DOM type_wf set_disconnected_nodes set_disconnected_nodes_locs +
  l_set_valCore_DOM type_wf set_val set_val_locs +
  l_create_character_data_wfCore_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_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_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_component_create_character_data?: l_get_component_create_character_dataCore_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_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_component_create_character_dataCore_DOM_axioms [instances]
```

### 2.1.11 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_dom_component_safe {} {cast new_document_ptr} h h'"
⟨proof⟩

locale l_get_component_create_documentCore_DOM =
  l_get_componentCore_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_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_documentCore_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_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  $\Rightarrow$  (_) object_ptr set  $\Rightarrow$  (_) heap  $\Rightarrow$  (_) heap  $\Rightarrow$  bool"
and get_root_node :: "(_) object_ptr  $\Rightarrow$  (()) heap, exception, (()) object_ptr) prog"
and get_root_node_locs :: "(()) heap  $\Rightarrow$  (()) heap  $\Rightarrow$  bool) set"
and get_ancestors :: "(_) object_ptr  $\Rightarrow$  (()) heap, exception, (()) object_ptr list) prog"
and get_ancestors_locs :: "(()) heap  $\Rightarrow$  (()) heap  $\Rightarrow$  bool) set"
and get_element_by_id ::
"(_) object_ptr  $\Rightarrow$  char list  $\Rightarrow$  (()) heap, exception, (()) element_ptr option) prog"
and get_elements_by_class_name ::
"(_) object_ptr  $\Rightarrow$  char list  $\Rightarrow$  (()) heap, exception, (()) element_ptr list) prog"
and get_elements_by_tag_name ::
"(_) object_ptr  $\Rightarrow$  char list  $\Rightarrow$  (()) heap, exception, (()) element_ptr list) prog"
and create_document :: "(()) heap, exception, (()) document_ptr) prog"
begin

```

```

lemma create_document_is_weakly_component_safe_step:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h  $\vdash$  create_document  $\rightarrow_h$  h'"
  assumes "ptr  $\neq$  cast |h  $\vdash$  create_document|r"
  shows "preserved (get_MObject ptr getter) h h'"
  <proof>

```

```

lemma create_document_is_weakly_component_safe:
  assumes "heap_is_wellformed h" and "type_wf h" and "known_ptrs h"
  assumes "h  $\vdash$  create_document  $\rightarrow_r$  result"
  assumes "h  $\vdash$  create_document  $\rightarrow_h$  h'"
  shows "is_weakly_dom_component_safe {} {cast result} h h'"
  <proof>
end

```

```

interpretation i_get_component_create_document?: l_get_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_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_component_create_document_Core_DOM_axioms [instances]

```

### 2.1.12 insert\_before

```

lemma insert_before_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 ptr and child and ref where
  "heap_is_wellformed h" and "type_wf h" and "known_ptrs h" and
  "h  $\vdash$  insert_before ptr child ref  $\rightarrow_h$  h'" and
  " $\neg$  is_strongly_dom_component_safe ({ptr, cast child}  $\cup$  (cast ' set_option ref)) {} h h'"
  <proof>

```

```

lemma append_child_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 ptr and child where
  "heap_is_wellformed h" and "type_wf h" and "known_ptrs h" and
  "h  $\vdash$  append_child ptr child  $\rightarrow_h$  h'" and
  " $\neg$  is_strongly_dom_component_safe {ptr, cast child} {} h h'"
  <proof>

```

### 2.1.13 get\_owner\_document

```

lemma get_owner_document_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 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_strongly_dom_component_safe {ptr} {cast owner_document} h h'"
⟨proof⟩

end

```

## 2.2 Shadow Root Components (Shadow\_DOM\_Components)

```
theory Shadow_DOM_Components
```

```
  imports
```

```
    Shadow_DOM.Shadow_DOM
```

```
    Core_DOM_Components
```

```
begin
```

### 2.2.1 get\_component

```

global_interpretation l_get_component_Core_DOM_defs get_root_node get_root_node_locs to_tree_order
  defines get_component = a_get_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_component?: l_get_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_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_component_Core_DOM_axioms [instances]
```

### 2.2.2 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_dom_component_safe {cast host} {cast new_shadow_root_ptr} h h'"
⟨proof⟩

```

```
locale l_get_dom_component_attach_shadow_root_Core_DOM =
```

```

  l_get_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_component_safe is_weakly_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_modeShadow_DOM type_wf set_mode set_mode_locs +
l_set_shadow_rootShadow_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_component_safe ::
    "(_) object_ptr set ⇒ (>) object_ptr set ⇒ (>) heap ⇒ (>) heap ⇒ bool"
  and is_weakly_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_MObject ptr getter) h h'"
⟨proof⟩
end

interpretation i_get_dom_component_attach_shadow_root?: l_get_dom_component_attach_shadow_rootCore_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_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_rootCore_DOM_axioms [instances]

```

### 2.2.3 get\_shadow\_root

```

lemma get_shadow_root_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
    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⟩

locale l_get_shadow_root_componentShadow_DOM =
  l_get_shadow_root +
  l_heap_is_wellformedShadow_DOM +
  l_get_componentCore_DOM +
  l_get_root_nodeCore_DOM +
  l_get_root_node_wfCore_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_component (cast host)|r ∩ set |h ⊢ get_component (cast shadow_root_ptr)|r = {}"
⟨proof⟩
end

interpretation i_get_shadow_root_component?: l_get_shadow_root_componentShadow_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_wellformedCore_DOM get_host get_host_locs
  get_disconnected_document get_disconnected_document_locs known_ptrs to_tree_order get_parent
  get_parent_locs get_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_componentShadow_DOM_axioms [instances]

```

### 2.2.4 get\_host

```

lemma get_host_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
    shadow_root_ptr and host and h' where
      "heap_is_wellformed h" and "type_wf h" and "known_ptrs h" and
      "h ⊢ get_host shadow_root_ptr →r host →h h'" and
      "¬ is_weakly_dom_component_safe {cast shadow_root_ptr} {cast host} h h'"
⟨proof⟩

locale l_get_host_componentShadow_DOM =
  l_heap_is_wellformedShadow_DOM +
  l_get_host +
  l_get_componentCore_DOM +
  l_get_shadow_root_componentShadow_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_component (cast host)|r ∩ set |h ⊢ get_component (cast shadow_root_ptr)|r = {}"
⟨proof⟩
end

```

```

interpretation i_get_host_component?: l_get_host_componentShadow_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_wellformedCore_DOM get_host get_host_locs get_disconnected_document
  get_disconnected_document_locs known_ptrs to_tree_order get_parent get_parent_locs get_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_componentShadow_DOM_axioms [instances]

```

## 2.2.5 get\_root\_node\_si

```

locale l_get_component_get_root_node_siShadow_DOM =
  l_get_root_node_si_wfShadow_DOM +
  l_get_componentCore_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_component ptr'|r = set |h ⊢ get_component root|r ∨
  set |h ⊢ get_component ptr'|r ∩ set |h ⊢ get_component root|r = {}"
  ⟨proof⟩
end

```

```

interpretation i_get_component_get_root_node_si?: l_get_component_get_root_node_siShadow_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_wellformedCore_DOM get_disconnected_document
  get_disconnected_document_locs to_tree_order get_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_component_get_root_node_siShadow_DOM_axioms [instances]

```

## 2.2.6 get\_assigned\_nodes

```

lemma assigned_nodes_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 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_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

```



## 2 Web Components

```

  assumes "node_ptr ∈ set nodes"
  shows "set |h ⊢ get_component (cast element_ptr)|r ∩ set |h ⊢ get_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_componentShadow_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_wellformedCore_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_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
  adopt_node adopt_node_locs set_child_nodes set_child_nodes_locs
⟨proof⟩

```

```

declare l_assigned_nodes_componentShadow_DOM_axioms [instances]

```

### get\_owner\_document

```

locale l_get_owner_document_componentShadow_DOM =
  l_get_owner_document_wfShadow_DOM +
  l_get_componentCore_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_component ptr|r ∩ set |h ⊢ get_component (cast owner_document)|r = {}"
⟨proof⟩

```

end

```

interpretation i_get_owner_document_component?: l_get_owner_document_componentShadow_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 get_root_node_si
  get_root_node_si_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_wellformedCore_DOM get_disconnected_document get_disconnected_document_locs
  known_ptrs get_ancestors_si get_ancestors_si_locs to_tree_order get_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_owner_document_componentShadow_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

# 3 Example

## 3.1 Testing fancy\_tabs (fancy\_tabs)

This theory contains the test cases for fancy\_tabs.

```
theory fancy_tabs
imports
  "Shadow_DOM.Shadow_DOM"
begin

definition fancy_tabs_heap :: "heapfinal" where
  "fancy_tabs_heap = create_heap [(cast (document_ptr.Ref 1), cast (create_document_obj html (Some (cast
(element_ptr.Ref 1))) [])),
  (cast (element_ptr.Ref 1), cast (create_element_obj 'html' [cast (element_ptr.Ref 2), cast (element_ptr.Ref
4)] fmempty None)),
  (cast (element_ptr.Ref 2), cast (create_element_obj 'head' [cast (element_ptr.Ref 3)] fmempty None)),
  (cast (element_ptr.Ref 3), cast (create_element_obj 'title' [cast (character_data_ptr.Ref 1)] fmempty
None)),
  (cast (character_data_ptr.Ref 1), cast (create_character_data_obj 'Global%20News')),
  (cast (element_ptr.Ref 4), cast (create_element_obj 'body' [cast (element_ptr.Ref 5), cast (element_ptr.Ref
6), cast (element_ptr.Ref 7), cast (element_ptr.Ref 8), cast (element_ptr.Ref 9), cast (element_ptr.Ref
10), cast (element_ptr.Ref 30)] fmempty None)),
  (cast (element_ptr.Ref 5), cast (create_element_obj 'h1' [cast (character_data_ptr.Ref 2)] fmempty
None)),
  (cast (character_data_ptr.Ref 2), cast (create_character_data_obj 'Global%20News')),
  (cast (element_ptr.Ref 6), cast (create_element_obj 'button' [cast (character_data_ptr.Ref 3)] fmempty
None)),
  (cast (character_data_ptr.Ref 3), cast (create_character_data_obj 'Search')),
  (cast (element_ptr.Ref 7), cast (create_element_obj 'button' [cast (character_data_ptr.Ref 4)] fmempty
None)),
  (cast (character_data_ptr.Ref 4), cast (create_character_data_obj 'Sign%20In')),
  (cast (element_ptr.Ref 8), cast (create_element_obj 'br' [] fmempty None)),
  (cast (element_ptr.Ref 9), cast (create_element_obj 'br' [] fmempty None)),
  (cast (element_ptr.Ref 10), cast (create_element_obj 'fancy-tabs' [cast (element_ptr.Ref 11), cast
(element_ptr.Ref 12), cast (element_ptr.Ref 13), cast (element_ptr.Ref 14), cast (element_ptr.Ref 15), cast
(element_ptr.Ref 22)] (fmap_of_list [('', 'background'), ('', '')]) (Some (cast (shadow_root_ptr.Ref 1))))),
  (cast (element_ptr.Ref 11), cast (create_element_obj 'button' [cast (character_data_ptr.Ref 5)] (fmap_of_list
[('', 'slot'), ('', 'title')]) None)),
  (cast (character_data_ptr.Ref 5), cast (create_character_data_obj 'Politics')),
  (cast (element_ptr.Ref 12), cast (create_element_obj 'button' [cast (character_data_ptr.Ref 6)] (fmap_of_list
[('', 'slot'), ('', 'title'), ('', 'selected'), ('', '')]) None)),
  (cast (character_data_ptr.Ref 6), cast (create_character_data_obj 'Sports')),
  (cast (element_ptr.Ref 13), cast (create_element_obj 'button' [cast (character_data_ptr.Ref 7)] (fmap_of_list
[('', 'slot'), ('', 'title')]) None)),
  (cast (character_data_ptr.Ref 7), cast (create_character_data_obj 'Culture')),
  (cast (element_ptr.Ref 14), cast (create_element_obj 'section' [cast (character_data_ptr.Ref 8)] fmempty
None)),
  (cast (character_data_ptr.Ref 8), cast (create_character_data_obj 'content%20panel%201')),
  (cast (element_ptr.Ref 15), cast (create_element_obj 'ul' [cast (element_ptr.Ref 16), cast (element_ptr.Ref
18), cast (element_ptr.Ref 20)] fmempty None)),
  (cast (element_ptr.Ref 16), cast (create_element_obj 'li' [cast (character_data_ptr.Ref 9), cast (element_ptr
17)] fmempty None)),
  (cast (character_data_ptr.Ref 9), cast (create_character_data_obj 'News%20Item%201')),
  (cast (element_ptr.Ref 17), cast (create_element_obj 'button' [cast (character_data_ptr.Ref 10)] fmempty
None)),
  (cast (character_data_ptr.Ref 10), cast (create_character_data_obj 'Share')),"
```

### 3 Example

```
(cast (element_ptr.Ref 18), cast (create_element_obj 'li' [cast (character_data_ptr.Ref 11), cast
(element_ptr.Ref 19)] fmempty None)),
  (cast (character_data_ptr.Ref 11), cast (create_character_data_obj 'News%20Item%202')),
  (cast (element_ptr.Ref 19), cast (create_element_obj 'button' [cast (character_data_ptr.Ref 12)] fmempty
None)),
  (cast (character_data_ptr.Ref 12), cast (create_character_data_obj 'Share')),
  (cast (element_ptr.Ref 20), cast (create_element_obj 'li' [cast (character_data_ptr.Ref 13), cast
(element_ptr.Ref 21)] fmempty None)),
  (cast (character_data_ptr.Ref 13), cast (create_character_data_obj 'News%20Item%203')),
  (cast (element_ptr.Ref 21), cast (create_element_obj 'button' [cast (character_data_ptr.Ref 14)] fmempty
None)),
  (cast (character_data_ptr.Ref 14), cast (create_character_data_obj 'Share')),
  (cast (element_ptr.Ref 22), cast (create_element_obj 'section' [cast (character_data_ptr.Ref 15)]
fmempty None)),
  (cast (character_data_ptr.Ref 15), cast (create_character_data_obj 'content%20panel%203')),
  (cast (shadow_root_ptr.Ref 1), cast (create_shadow_root_obj Open [cast (element_ptr.Ref 23), cast (element_ptr.
24), cast (element_ptr.Ref 26), cast (element_ptr.Ref 28), cast (element_ptr.Ref 29)])),
  (cast (element_ptr.Ref 23), cast (create_element_obj 'style' [cast (character_data_ptr.Ref 16)] fmempty
None)),
  (cast (character_data_ptr.Ref 16), cast (create_character_data_obj '---shortened---')),
  (cast (element_ptr.Ref 24), cast (create_element_obj 'div' [cast (element_ptr.Ref 25)] (fmap_of_list
[('id', 'tabs')] None)),
  (cast (element_ptr.Ref 25), cast (create_element_obj 'slot' [] (fmap_of_list [('id', 'tabsSlot'),
('name', 'title')] None)),
  (cast (element_ptr.Ref 26), cast (create_element_obj 'div' [cast (element_ptr.Ref 27)] (fmap_of_list
[('id', 'panels')] None)),
  (cast (element_ptr.Ref 27), cast (create_element_obj 'slot' [] (fmap_of_list [('id', 'panelsSlot')]
None)),
  (cast (element_ptr.Ref 28), cast (create_element_obj 'button' [cast (character_data_ptr.Ref 17)] fmempty
None)),
  (cast (character_data_ptr.Ref 17), cast (create_character_data_obj 'Previous%20Tab')),
  (cast (element_ptr.Ref 29), cast (create_element_obj 'button' [cast (character_data_ptr.Ref 18)] (fmap_of_list
[('style', 'float:right')] None)),
  (cast (character_data_ptr.Ref 18), cast (create_character_data_obj 'Next%20Tab')),
  (cast (element_ptr.Ref 30), cast (create_element_obj 'script' [] fmempty None))]"

definition fancy_tabs_document :: "(unit, unit, unit, unit, unit, unit) object_ptr option" where "fancy_tabs_docum
= Some (cast (document_ptr.Ref 1))"

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](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 (WPDAl) track at WWW 2018*, 2018. URL <https://www.brucker.ch/bibliography/abstract/brucker.ea-fdom-2018>.
- [4] 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](http://www.isa-afp.org/entries/Shadow_DOM.html), Formal proof development.
- [5] 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>.
- [6] M. Herzberg. *Formal Foundations for Provably Safe Web Components*. PhD thesis, The University of Sheffield, 2020.
- [7] WHATWG. DOM – living standard, Feb. 2019. URL <https://dom.spec.whatwg.org/commit-snapshots/7fa83673430f767d329406d0aed901f296332216/>. Last Updated 11 February 2019.