



# Advancing FOCUS

Glen R. Goodwin



@areinet



arei.net



github.com/arei



# Advancing FOCUS

Glen R. Goodwin



@areinet



arei.net



github.com/arei



# Advancing FOCUS

Glen R. Goodwin



@areinet



arei.net



github.com/arei









**STUPENDOUS**

**AMAZING**



**TOTALLY RIVETING**

# Focus Subsystem

# Focus Subsystem

# Focus Subsystem



# Focus Subsystem

# Focus Subsystem

- Does it work well?

- Does it work well?
- Does it meet the needs of the developers?



- Does it work well?
- Does it meet the needs of the developers?
- Could it do more to be better?

# Focus



**First Name**

**Last Name**

**Mailing Address**

**City**

**State/Province**

**Postal Code**

**Country**



**First Name**

**Last Name**

**Mailing Address**

**City**

**State/Province**

**Postal Code**

**Country**



[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)  
[Wikipedia store](#)

[Interaction](#)  
[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact page](#)

[Tools](#)  
[What links here](#)  
[Related changes](#)  
[Upload file](#)  
[Special pages](#)  
[Permanent link](#)  
[Page information](#)  
[Wikidata item](#)  
[Cite this page](#)

[Print/export](#)

👤 Not logged in   [Talk](#)   [Contributions](#)   [Create account](#)   [Log in](#)

Article

**Talk**

Read

[Edit](#)

[View history](#)

🔍

# Focus (computing)

From Wikipedia, the free encyclopedia

In [computing](#), the **focus** indicates the component of the [graphical user interface](#) which is selected to receive input. Text entered at the keyboard or pasted from a [clipboard](#) is sent to the component which has the focus. Moving the focus away from a specific user interface element is known as a **blur** event in relation to this element. Typically, the focus is withdrawn from an element by giving another element the focus. This means that focus and blur events typically both occur virtually simultaneously, but in relation to different user interface elements, one that gets the focus and one that gets blurred.

The concept is similar to a [cursor](#) in a text-based environment. However, when considering a graphical interface, there is also a [mouse](#) pointer involved. Moving the mouse will typically move the mouse pointer without changing the focus. The focus can usually be changed by clicking on a component that can receive focus with the mouse. Many desktops also allow the focus to be changed with the keyboard. By convention, the Tab ⇧ key is used to move the focus to the next focusable component and ⇧ Shift + Tab ⇧ to the previous one. When graphical interfaces were first introduced, many computers did not have mice, so this alternative was necessary. This feature makes it easier for people that have a hard time using a mouse to use the user interface. In certain circumstances, the [arrow keys](#) can also be used to move focus.

## Contents [\[hide\]](#)

- Window focus
  - Click to focus
  - Focus follows pointer
  - Sloppy focus
- Focus models used by X11 window managers
- Intra-window component focus
- See also
- References
- Notes





BitTorrent OSx

Downloaded: 12.00 MB at 1.37 KB/s - 1 seed in 2 peers  
Sibelius 3.roast  
7.8 MB at 0.11 MB/s - 0 seeds in 0 peers  
Sibelius Xtras.sitx  
1.9 MB at 0.11 MB/s - 0 seeds in 0 peers

Name: 4  
Max Slots: 0  
Max UL KIB/s: 0  
Ratio: 1.0  
Autostop: ☐  
Log: ☐

Transfer	
122 MIB	6.3 KIB/s
57.2 MIB	1.6 KIB/s
0 B	0 B/s
7.8 MIB	0 B/s
0 B	0 B/s
0 B	0 B/s
0 B	0 B/s

Loading "Test" (Reaktor5)...

FileMP3.org

Buddy List

didonotfalter@mac.com

henryfaber@njs.netlab.cz

Lorie Clay

joebonka@hotmail.com

Tara Bethune-Leamen

Offline

Chat with Brian H.

i'm not sure i really understand how to get to the meat once you see it there

3:20 PM

i just got some shots of Max's kiddie pools!

7:25 PM

hey did you hear - Steve Jobs called the father of the kid who was killed over the ipod in brooklyn

and they have PHP5, and the zend optimizer for PHP!

gosh, dreamhost is

you



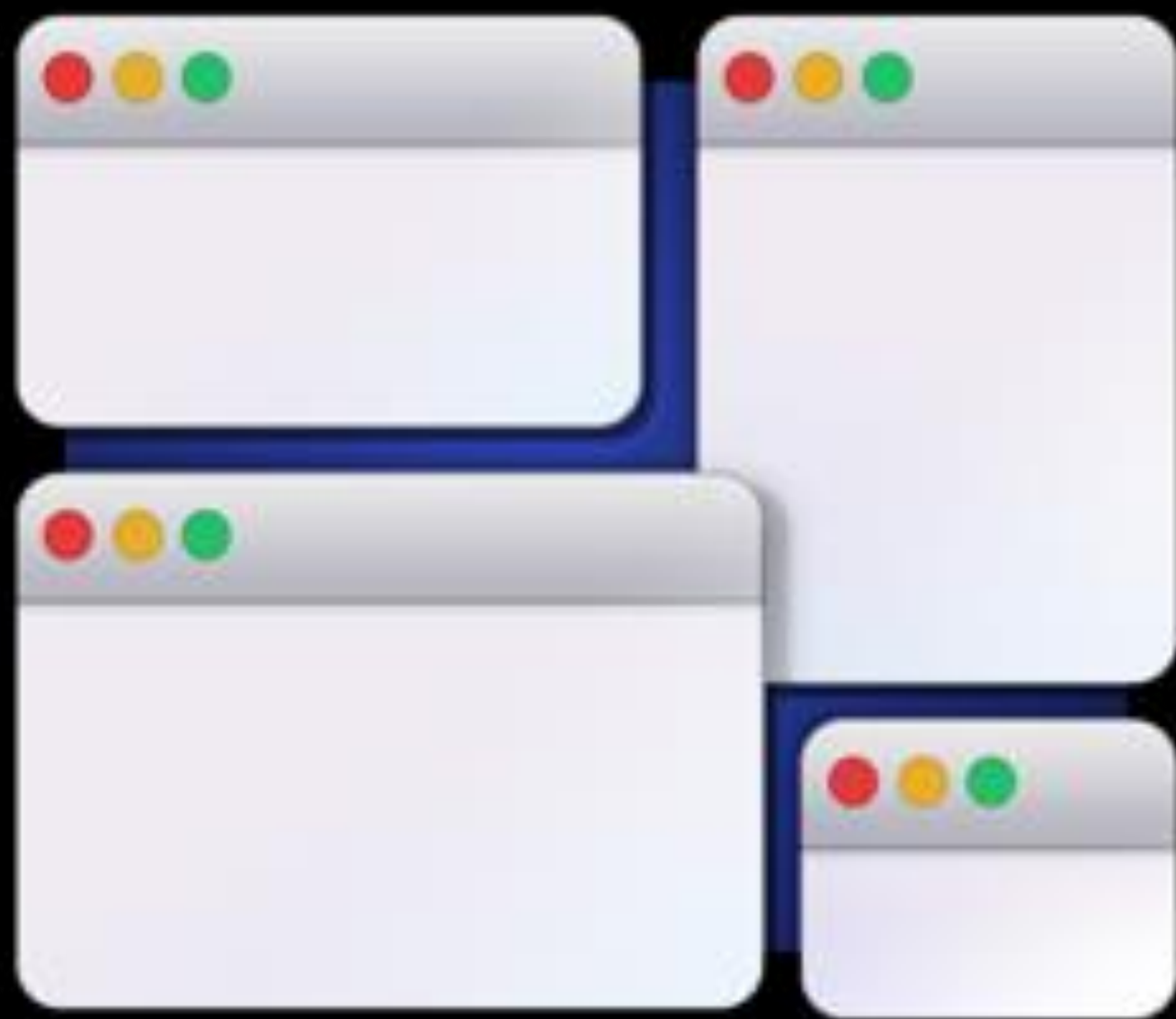


ANY

Shift

Ctrl





S

E

N

D

H

E

L

P

P

L

E

A

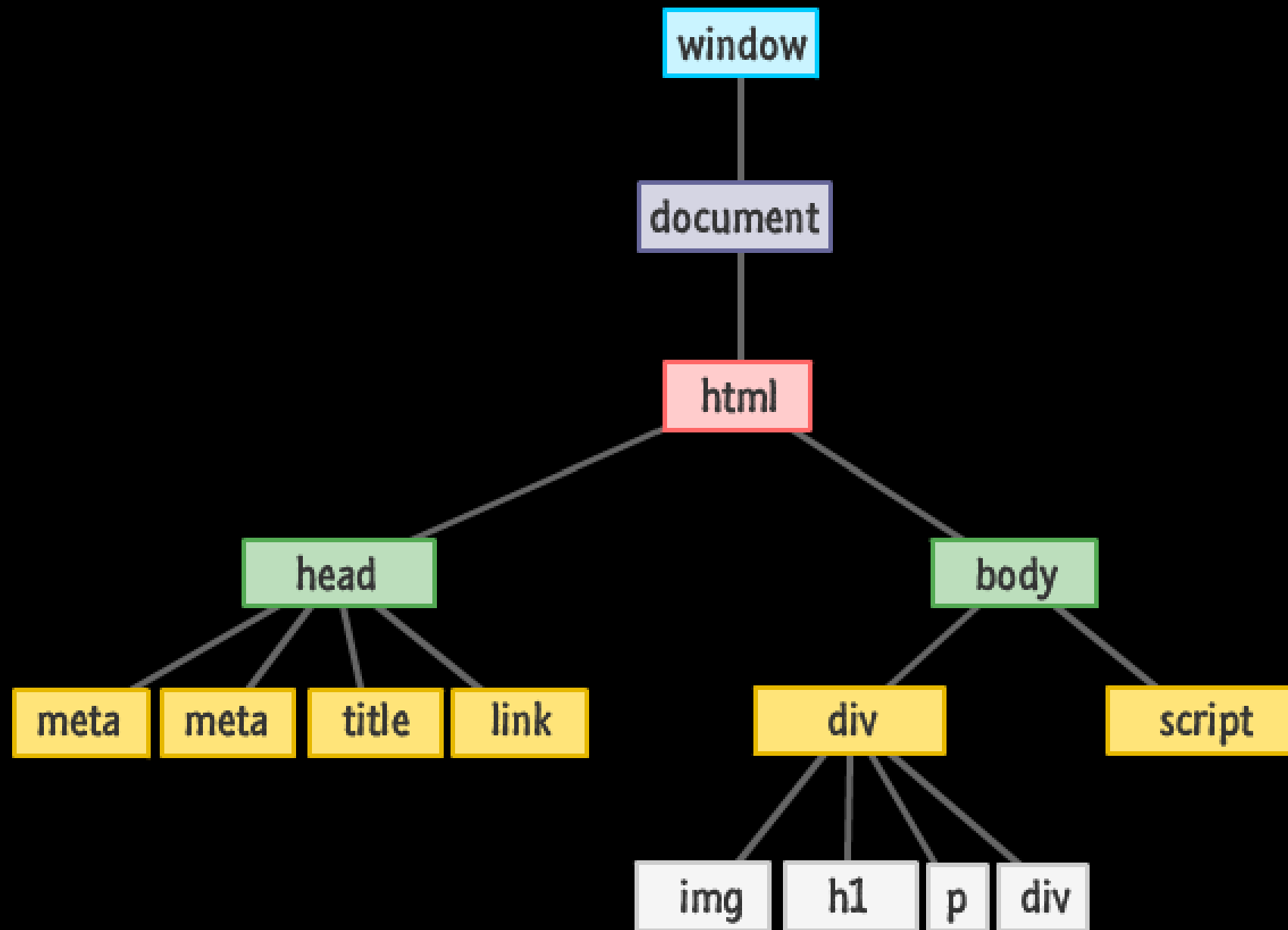
S

E

X



You have too many tabs open.



document.**activeElement**

**Keyboard**



**Operating System**



**Application (Browser)**



**Tab/Window**



**Document**



**Element**

## Properties

**document.activeElement**

## Methods

**element.focus()**

**element.blur()**

## Events

**focus**

**blur**

**focusin**

**focusout**

## Properties

**document.activeElement**

## Methods

**element.focus()**

**element.blur()**

## Events

**focus**

**blur**

**focusin**

**focusout**



# Document Object Model (DOM) Level 1 Specification

Version 1.0

W3C Recommendation 1 October, 1998

## This version

<http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001>  
<http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/DOM.ps>  
<http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/DOM.pdf>  
<http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/DOM.tgz>  
<http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/DOM.zip>  
<http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/DOM.txt>

## Latest version

<http://www.w3.org/TR/REC-DOM-Level-1>

## Previous versions

<http://www.w3.org/TR/1998/PR-DOM-Level-1-19980818>  
<http://www.w3.org/TR/1998/WD-DOM-19980720>  
<http://www.w3.org/TR/1998/WD-DOM-19980416>  
<http://www.w3.org/TR/WD-DOM-19980318>  
<http://www.w3.org/TR/WD-DOM-971209>  
<http://www.w3.org/TR/WD-DOM-971009>

## WG Chair

Lauren Wood, *SoftQuad, Inc.*

## Editors

Vidur Apparao, *Netscape*  
Steve Byrne, *Sun*  
Mike Champion, *ArborText*  
Scott Isaacs, *Microsoft*



## Interface *HTMLSelectElement*

The select element allows the selection of an option. The contained options can be directly accessed through the select element as a collection. See the [SELECT element definition](#) in HTML 4.0.

### IDL Definition

```
interface HTMLSelectElement : HTMLElement {
  readonly attribute DOMString      type;
          attribute long             selectedIndex;
          attribute DOMString       value;
  readonly attribute long            length;
  readonly attribute HTMLFormElement form;
  readonly attribute HTMLCollection options;
          attribute boolean         disabled;
          attribute boolean         multiple;
          attribute DOMString       name;
          attribute long            size;
          attribute long            tabIndex;

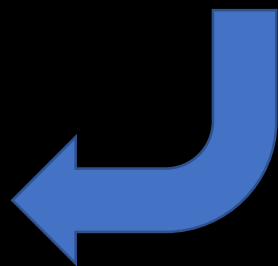
  void      add(in HTMLElement element,
               in HTMLElement before);
  void      remove(in long index);
  void      blur();
  void      focus();
};
```

### Attributes

- type**  
The type of control created.
- selectedIndex**  
The ordinal index of the selected option. The value -1 is returned if no element is selected. If multiple options are selected, the index of the first selected option is returned.
- value**  
The current form control value.
- length**  
The number of options in this SELECT.



**Not really Glen...**



**But pretend it is and  
he's as dashing as  
international  
heart-throb  
Brian Posehn...**

## Username

|

---

Username must:

- begin with a letter.
- be lowercase.
- only contain letters A-Z or numbers 0-9.



```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);

  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {

      ?????????????????

    }
  });
});
```



[zephjs.com](https://zephjs.com)



[zephjs.com](https://zephjs.com)

```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);

  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {

      ?????????????????

    }
  });
});
```



```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);

  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {

      ?????????????????

    }
  });
});
```

```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);

  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {

      ?????????????

    }
  });
});
```

```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
```

```
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);
```

```
  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {
```

```
      ?????????????
```

```
    }
  });
});
```

```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);
```

```
    onEventAt("input","keyup",(element,content,event)=>{
      if (event.keyCode===13) {

        ?????????????

      }
    });
  });
});
```

```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);

  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {

      ?????????????????

    }
  });
});
```

```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);

  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {
```

????????????????

```
    }
  });
});
```

## Properties

**document.activeElement**

## Methods

**element.focus()**

**element.blur()**

## Events

**focus**

**blur**

**focusin**

**focusout**

```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);

  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {
```

????????????????

```
    }
  });
});
```



```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);

  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {
```

????????????????

```
    }
  });
});
```

```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);

  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {
```

????????????

```
    }
  });
});
```

```
import {ZephComponents,html,onEventAt} from "ZephJS";
ZephComponents.define("my-user-field",()=>{
  html(`
    <div class="wrapper">
      <input class="field" type="text"></input>
    </div>
  `);

  onEventAt("input","keyup",(element,content,event)=>{
    if (event.keyCode===13) {
```

????????????????

```
    }
  });
});
```

html																col	table												
head	span													fieldset	form	body	h1	section	colgroup	tr									
title	a													meter	select	aside	h2	header	caption	td									
meta	rt	dfn	em	i	small	ins	hr	p	div	blockquote	legend	optgroup	address	h3	nav	menu	th												
base	rp	abbr	time	b	strong	del	br	figcaption	ol	dl	label	option	datalist	h4	article	command	tbody												
link	noscript	q	var	sub	mark	kbd	wbr	figure	ul	dt	input	output	keygen	h5	footer	summary	thead												
style	script	cite	samp	sup	ruby	bdo	code	pre	li	dd	textarea	button	progress	h6	hgroup	details	tfoot												
																	img	area	map	embed	object	param	source	iframe	canvas	track*	audio	video	device*

- Root element
- Text-level semantics
- Forms
- Tabular data
- Metadata and scripting
- Grouping content
- Document sections
- Interactive elements
- Embedding content

# Periodic Table of the Elements

[illegible]

- Root element
- Text-level semantics
- Forms
- Tabular data
- Metadata and scripting
- Grouping content
- Document sections
- Interactive elements
- Embedding content



# Periodic Table of the Elements

html

head

title

meta

base

link

style

span

a

rt

dfn

em

i

small

ins

hr

p

div

blockquote

rp

abbr

time

b

strong

del

br

figcaption

ol

dl

label

option

datalist

q

var

sub

mark

kbd

wbr

figure

ul

dt

input

output

keygen

cite

samp

sup

ruby

bdo

code

pre

i

dd

textarea

button

progress

fieldset

form

body

h1

section

h2

header

h3

nav

h4

article

h5

h5

hgroup

aside

h2

h3

h4

h5

h5

tfoot

caption

td

th

tbody

thead

tfoot

col

table

colgroup

tr

td

th

tbody

thead

tfoot

img

area

map

embed

object

param

source

iframe

canvas

track\*

audio

video

device\*

- Root element
- Text-level semantics
- Forms
- Tabular data
- Metadata and scripting
- Grouping content
- Document sections
- Interactive elements
- Embedding content



# Periodic Table of the Elements

The image displays a periodic table of HTML elements, organized into groups and color-coded by function. The elements are arranged in rows and columns, with some elements highlighted by red circles.

**Highlighted Elements (Red Circles):**

- `a` (Link)
- `select` (Form)
- `input` (Form)
- `link` (Link)

**Color Coding:**

- Green:** `html`, `head`, `title`, `meta`, `base`, `link`, `style`, `script`, `code`, `pre`, `li`, `dd`, `textarea`, `button`, `progress`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `hgroup`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `hgroup`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `hgroup`.
- Blue:** `span`, `div`, `p`, `hr`, `br`, `img`, `area`, `map`, `embed`, `object`, `param`, `source`, `iframe`, `canvas`, `track`, `audio`, `video`, `device`.
- Yellow:** `rt`, `dfn`, `em`, `i`, `small`, `ins`, `del`, `sub`, `sup`, `ruby`, `bdo`, `code`, `pre`, `li`, `dd`, `textarea`, `button`, `progress`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `hgroup`.
- Orange:** `col`, `table`, `tr`, `td`, `th`, `tbody`, `thead`, `tfoot`, `thead`, `tfoot`.
- Purple:** `img`, `area`, `map`, `embed`, `object`, `param`, `source`, `iframe`, `canvas`, `track`, `audio`, `video`, `device`.

- Root element
- Text-level semantics
- Forms
- Tabular data
- Metadata and scripting
- Grouping content
- Document sections
- Interactive elements
- Embedding content

[illegible]

- Root element
- Metadata and scripting
- Embedding content

- Text-level semantics
- Grouping content

- Forms
- Document sections

- Tabular data
- Interactive elements



# Periodic Table of the Elements

The image displays a periodic table of HTML elements, categorized by color and grouped by function. The elements are arranged in rows and columns, with some elements circled in red.

**Elements circled in red:**

- `a` (Link)
- `link` (Link)
- `input` (Form)
- `select` (Form)
- `textarea` (Form)
- `button` (Form)

**Other elements visible:**

- `html` (Structure)
- `head` (Structure)
- `title` (Structure)
- `meta` (Structure)
- `base` (Structure)
- `style` (Structure)
- `script` (Structure)
- `span` (Text)
- `div` (Text)
- `p` (Text)
- `br` (Text)
- `hr` (Text)
- `small` (Text)
- `em` (Text)
- `strong` (Text)
- `del` (Text)
- `sub` (Text)
- `sup` (Text)
- `code` (Text)
- `pre` (Text)
- `li` (Text)
- `ul`
- `ol`
- `dl`
- `dt`
- `dd`
- `figure`
- `figcaption`
- `img` (Image)
- `area` (Image)
- `map` (Image)
- `embed` (Image)
- `object` (Image)
- `param` (Image)
- `source` (Image)
- `iframe` (Image)
- `canvas` (Image)
- `track` (Image)
- `audio` (Image)
- `video` (Image)
- `device` (Image)

- Root element
- Text-level semantics
- Forms
- Tabular data
- Metadata and scripting
- Grouping content
- Document sections
- Interactive elements
- Embedding content

# Periodic Table of the Elements

- Root element
- Text-level semantics
- Forms
- Tabular data
- Metadata and scripting
- Grouping content
- Document sections
- Interactive elements
- Embedding content



# Periodic Table of the Elements

The periodic table of HTML elements is organized into groups based on their function and structure. The elements are arranged in a grid, with some elements circled in red. The categories are:

- Root element
- Metadata and scripting
- Text-level semantics
- Grouping content
- Forms
- Document sections
- Tabular data
- Interactive elements

The elements are arranged in a grid, with some elements circled in red. The categories are:

- Root element
- Metadata and scripting
- Text-level semantics
- Grouping content
- Forms
- Document sections
- Tabular data
- Interactive elements

- Root element
- Metadata and scripting
- Embedding content

- Text-level semantics
- Grouping content

- Forms
- Document sections

- Tabular data
- Interactive elements

**EXCEPT!**

# Periodic Table of the Elements

[illegible]



# Periodic Table of the Elements

The diagram illustrates the structure of HTML elements, categorized by color and function. A large red question mark is overlaid on the grid, suggesting a focus on understanding or identifying these elements.

**Legend:**

- Root element (Green)
- Metadata and scripting (Blue)
- Embedding content (Purple)
- Text-level semantics (Yellow)
- Grouping content (Orange)
- Forms (Light Green)
- Document sections (Light Blue)
- Tabular data (Brown)
- Interactive elements (Grey)

**Grid of HTML Elements:**

Category	Element
Root element	html
Metadata and scripting	head, title, meta, base, link, style, script
Text-level semantics	span, a, rt, dfn, em, i, small, ins, hr, p, div, rp, abbr, time, b, strong, del, br, figcaption, ol, q, var, sub, mark, kbd, wbr, cite, samp, sup, ruby, bdo, code
Grouping content	div, p, hr, p, div, br, figcaption, ol, dd
Forms	fieldset, form, meter, select, legend, optgroup, label, option, input, output, keygen, textarea, button, progress
Document sections	body, h1, section, h2, header, h3, nav, h4, article, h5, footer, h6, hgroup
Tabular data	col, table, colgroup, tr, td, caption, th, tbody, tfoot
Interactive elements	menu, command, summary, details
Embedding content	img, area, map, embed, object, param, source, iframe, canvas, track*, audio, video, device*



# Periodic Table of the Elements

The image displays a periodic table of HTML elements, categorized by color and function. A large red question mark is overlaid on the table, suggesting a quiz or a question about the elements.

**Legend:**

- Root element (Green)
- Metadata and scripting (Blue)
- Text-level semantics (Yellow)
- Grouping content (Orange)
- Forms (Light Green)
- Document sections (Light Blue)
- Embedding content (Purple)
- Interactive elements (Light Grey)
- Tabular data (Light Orange)

**Elements shown:**

- Root element: `html`
- Metadata and scripting: `head`, `title`, `meta`, `base`, `link`, `style`, `noscript`, `script`
- Text-level semantics: `span`, `a`, `rt`, `dfn`, `em`, `i`, `small`, `ins`, `hr`, `p`, `div`, `rp`, `abbr`, `time`, `b`, `strong`, `del`, `br`, `figcaption`, `ol`

- Root element
- Metadata and scripting
- Embedding content

- Text-level semantics
- Grouping content

- Forms
- Document sections

- Tabular data
- Interactive elements

# Periodic Table of the Elements

The diagram illustrates the structure of HTML elements, categorized by color and function. A large red question mark is overlaid on the grid, suggesting a focus on understanding or questioning the structure.

**Legend:**

- Root element (Green)
- Metadata and scripting (Blue)
- Embedding content (Purple)
- Text-level semantics (Yellow)
- Grouping content (Orange)
- Forms (Light Green)
- Document sections (Light Blue)
- Interactive elements (Grey)

**Grid Content:**

Category	Element
Root element	html
Metadata and scripting	head, title, meta, base, link, style, script
Text-level semantics	span, a, rt, dfn, em, i, small, ins, hr, p, div, rp, abbr, time, b, strong, del, br, figcaption, ol, q, var, sub, mark, kbd, wbr, cite, samp, sup, ruby, bdo, code
Grouping content	col, table, colgroup, tr, td, th, tbody, tfoot, thead
Forms	fieldset, form, meter, select, optgroup, label, option, data-list, input, output, keygen, textarea, button, progress
Document sections	body, h1, section, h2, header, h3, nav, h4, article, h5, footer, h6, hgroup
Interactive elements	command, summary, details
Embedding content	img, area, map, embed, object, param, source, iframe, canvas, track*, audio, video, device*



[illegible]



## § 6.4 Focus

### § 6.4.1 Introduction

*This section is non-normative.*

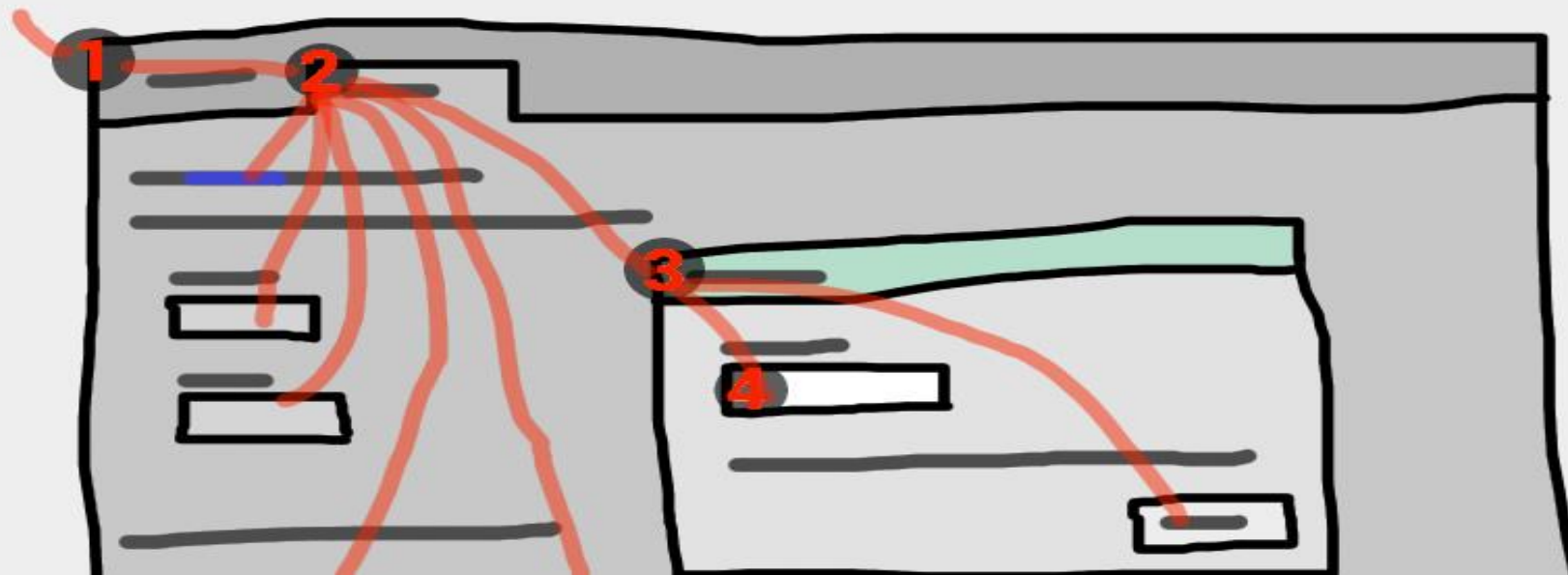
An HTML user interface typically consists of multiple interactive widgets, such as form controls, scrollable regions, links, dialog boxes, browser tabs, and so forth. These widgets form a hierarchy, with some (e.g. browser tabs, dialog boxes) containing others (e.g. links, form controls).

When interacting with an interface using a keyboard, key input is channeled from the system, through the hierarchy of interactive widgets, to an active widget, which is said to be [focused](#).

#### Example

Consider an HTML application running in a browser tab running in a graphical environment. Suppose this application had a page with some text controls and links, and was currently showing a modal dialog, which itself had a text control and a button.

The hierarchy of focusable widgets, in this scenario, would include the browser window, which would have, amongst its children, the browser tab containing the HTML application. The tab itself would have as its children the various links and text controls, as well as the dialog. The dialog itself would have as its children the text control and the button.



- Elements that have their tabindex focus flag set, that are not actually disabled, that are not expressly inert, and that are either being rendered or being used as relevant canvas fallback content.
- The shapes of area elements in an image map associated with an img element that is being rendered and is not expressly inert.
- The user-agent provided subwidgets of elements that are being rendered and are not actually disabled or expressly inert
- The scrollable regions of elements that are being rendered and are not expressly inert.
- The viewport of a Document that is in a browsing context and is not inert.
- Any other element or part of an element, especially to aid with accessibility or to better match platform conventions.

- Elements that have their tabindex focus flag set, that are not actually disabled, that are not expressly inert, and that are either being rendered or being used as relevant canvas fallback content.
- The shapes of area elements in an image map associated with an img element that is being rendered and is not expressly inert.
- The user-agent provided subwidgets of elements that are being rendered and are not actually disabled or expressly inert
- The scrollable regions of elements that are being rendered and are not expressly inert.
- The viewport of a Document that is in a browsing context and is not inert.
- Any other element or part of an element, especially to aid with accessibility or to better match platform conventions.

- Elements that have their tabindex focus flag set, that are not actually disabled, that are not expressly inert, and that are either being rendered or being used as relevant canvas fallback content.
- The shapes of area elements in an image map associated with an img element that is being rendered and is not expressly inert.
- The user-agent provided subwidgets of elements that are being rendered and are not actually disabled or expressly inert
- The scrollable regions of elements that are being rendered and are not expressly inert.
- The viewport of a Document that is in a browsing context and is not inert.
- Any other element or part of an element, especially to aid with accessibility or to better match platform conventions.

- Elements that have their tabindex focus flag set, that are not actually disabled, that are not expressly inert, and that are either being rendered or being used as relevant canvas fallback content.
- The shapes of area elements in an image map associated with an img element that is being rendered and is not expressly inert.
- The user-agent provided subwidgets of elements that are being rendered and are not actually disabled or expressly inert
- The scrollable regions of elements that are being rendered and are not expressly inert.
- The viewport of a Document that is in a browsing context and is not inert.
- Any other element or part of an element, especially to aid with accessibility or to better match platform conventions.





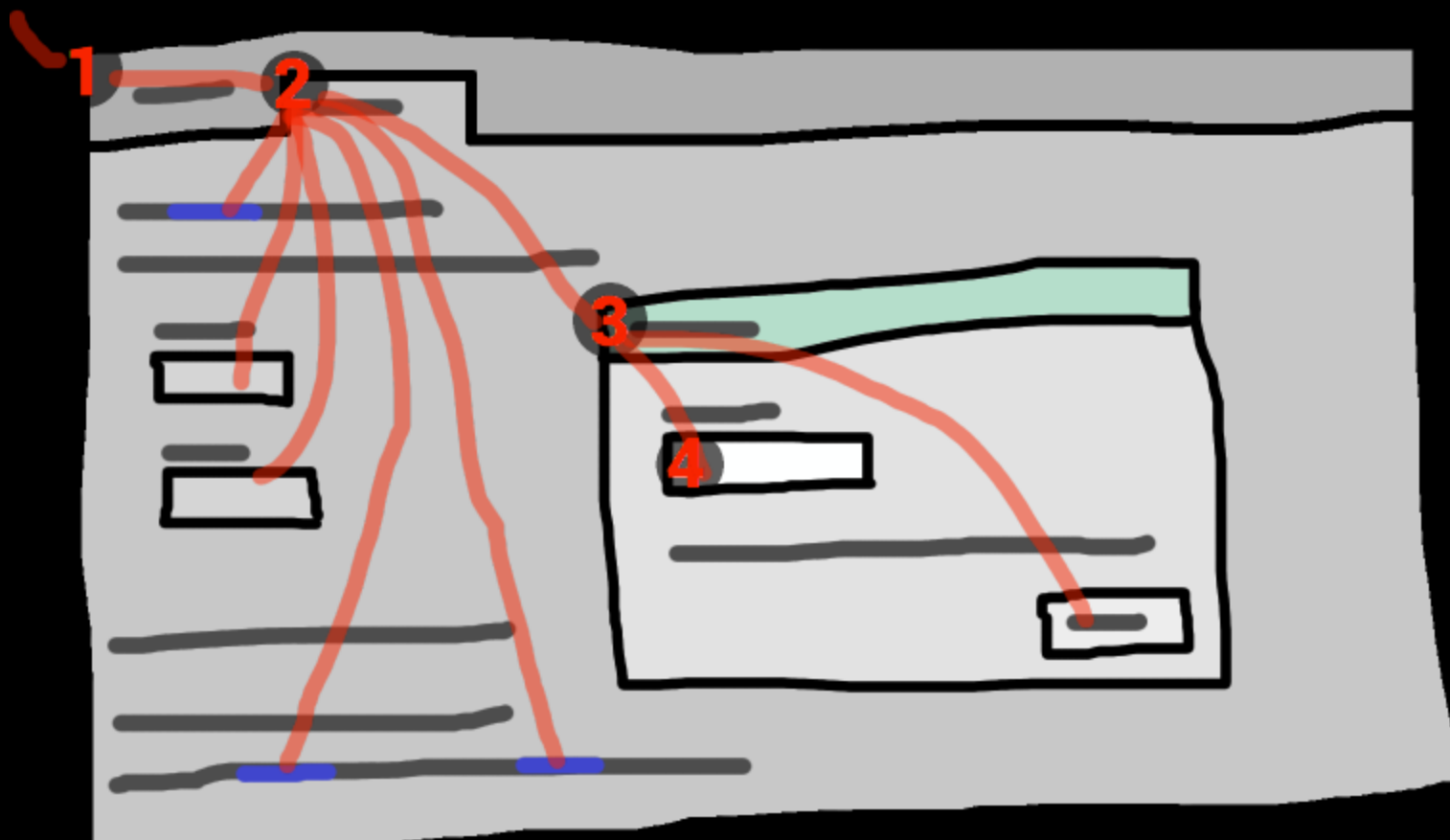
**1** No way to automatically  
advance (or reverse) focus

**1** No way to automatically advance (or reverse) focus

**2** No way to determine what is focusable

**1** No way to automatically advance (or reverse) focus

**2** No way to determine what is focusable













✓ **.isFocusable(e)**

- ✓ **.isFocusable(e)**
- ✓ **.forward()**
- ✓ **.backward()**

- ✓ **.isFocusable(e)**
- ✓ **.forward()**
- ✓ **.backward()**

- ✓ `.isFocusable(e)`
- ✓ `.forward()`
- ✓ `.backward()`
- ✓ `.trap()`

- ✓ `.isFocusable(e)`
- ✓ `.forward()`
- ✓ `.backward()`
- ✓ `.trap()`
- ✓ `.order()`

- ✓ `.isFocusable(e)`
- ✓ `.forward()`
- ✓ `.backward()`
- ✓ `.trap()`
- ✓ `.order()`

- ✓ **.isFocusable(e)**
- ✓ **.forward()**
- ✓ **.backward()**
- ✓ **.trap()**
- ✓ **.order()**
- ✓ **.previousActiveElement**
- ✓ **.history**

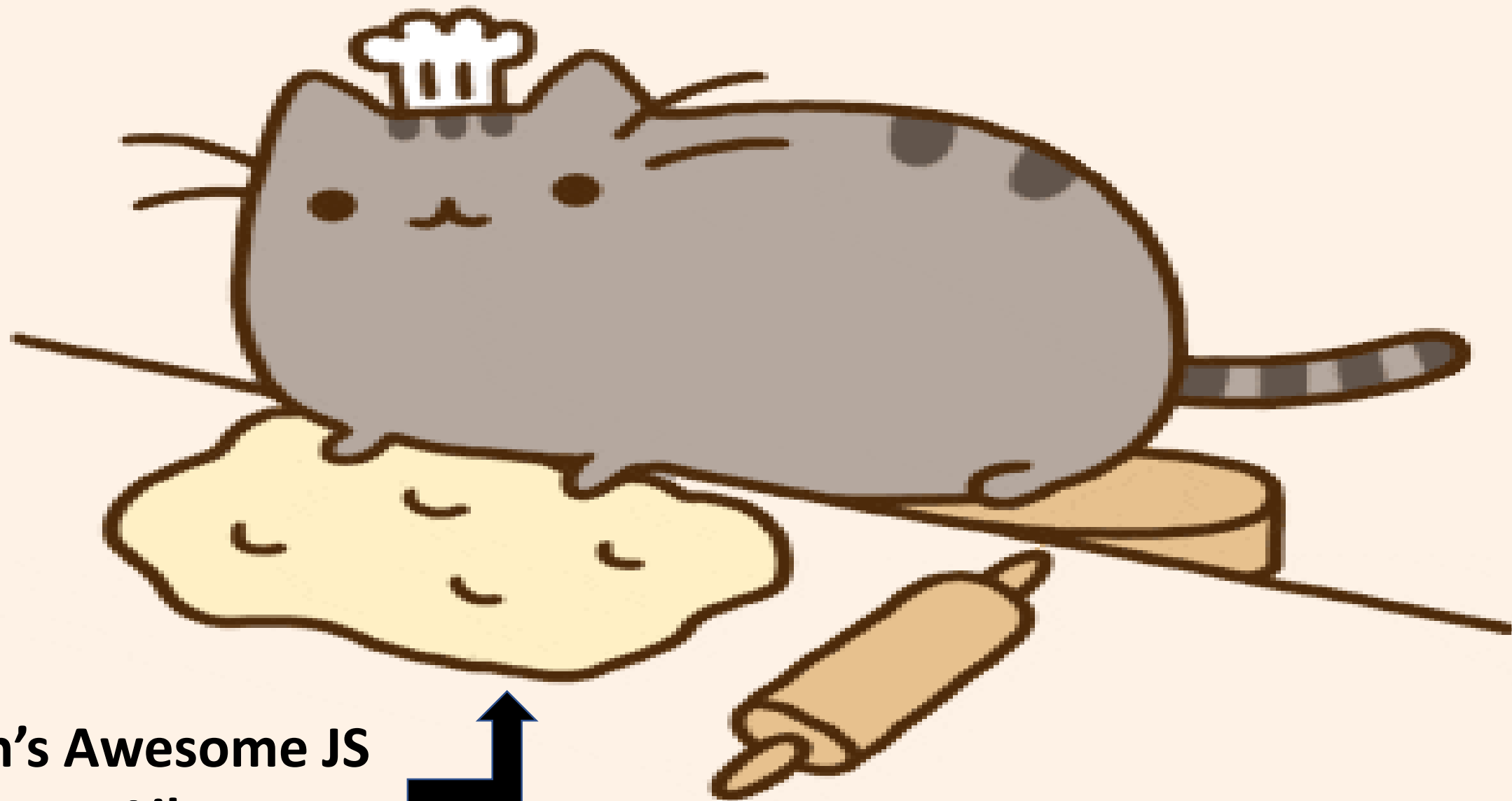


- ✓ **.isFocusable(e)**
- ✓ **.forward()**
- ✓ **.backward()**
- ✓ **.trap()**
- ✓ **.order()**
- ✓ **.previousActiveElement**
- ✓ **.history**
- ✓ **.autofocus(e)**

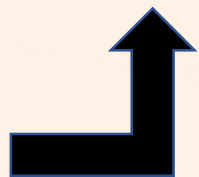
- ✓ `.isFocusable(e)`
- ✓ `.forward()`
- ✓ `.backward()`
- ✓ `.trap()`
- ✓ `.order()`
- ✓ `.previousActiveElement`
- ✓ `.history`
- ✓ `.autofocus(e)`
- ✓ `???`

- ✓ `.isFocusable(e)`
- ✓ `.forward()`
- ✓ `.backward()`
- ✓ `.trap()`
- ✓ `.order()`
- ✓ `.previousActiveElement`
- ✓ `.history`
- ✓ `.autofocus(e)`
- ✓ `???`





**Glen's Awesome JS  
Focus Library**



📖 README.md

# tabbable

build passing

---

**SEEKING CO-MAINTAINERS!** Continued development of this project is going to require the work of one or more dedicated co-maintainers (or forkers). If you're interested, please comment in [this issue](#).

---

Returns an array of all\* tabbable DOM nodes within a containing node. (\* "all" has some necessary caveats, which you'll learn about by reading below.)

The following are considered tabbable:

- `<button>` s
- `<input>` s
- `<select>` s
- `<textarea>` s
- `<a>` s with `href` or `xlink:href` attributes
- `<audio>` s and `<video>` s with `controls` attributes

**window.focusManager**







W3C

W3C



W3C



W3C



W3C







**Still not Glen...**

**But let's pretend  
Glen is this cool  
and popular and  
super well-known...**



how to propose changes to the HTML standard



- All
- News
- Videos
- Images
- Shopping
- More
- Settings
- Tools

About 191,000,000 results (0.94 seconds)

### 7 W3C Recommendation Track Process

<https://www.w3.org/2004/02/Process-20040205/tr.html>

Feb 5, 2004 - 7.7.1 **Proposal** to Rescind a Recommendation; 7.7.2 Publication of a ... Note: W3C Recommendations are similar to the **standards** ... A Working Group MUST document **changes** (both substantive and minor) between steps.

[Recommendation Track ...](#) · [Advancing a Technical ...](#) · [Modifying a W3C ...](#)

You visited this page on 7/16/19.

People also ask

What is html6?

What is the latest HTML standard?

What is HTML structure?

What is w3c Candidate Recommendation?

Feedback

### HTML 5 - Wikipedia

<https://en.wikipedia.org/wiki/HTML5>

HTML 5 is a software solution stack that defines the properties and behaviors of web page content by implementing a markup-based pattern to it. HTML 5 is the fifth and current major version of HTML, and





**Web Platform  
Incubator  
Community  
Group**



**Web Platform  
Incubator  
Community  
Group**



**WHATWG**





**WHATWG**













**JS  
DAILY**







# The Focus Traversal API Proposal

# The **Focus Traversal API** Proposal

focusManager.isFocusable(e)  
focusManager.forward()  
focusManager.backward()  
focusManager.next(e)  
focusManager.previous(e)  
focusManager.hasFocus(e)  
focusManager.focus(e)  
focusManager.orderedElements()  
focusManager.currentlyFocused  
focusManager.previouslyFocused  
focusManager.history

Coming Soon...

focusManager.trap(e)  
focusManager.order(a)  
focusManager.autofocus(e)

# The **Focus Traversal API** Proposal

focusManager.isFocusable(e)  
focusManager.forward()  
focusManager.backward()  
focusManager.next(e)  
focusManager.previous(e)  
focusManager.hasFocus(e)  
focusManager.focus(e)  
focusManager.orderedElements()  
focusManager.currentlyFocused  
focusManager.previouslyFocused  
focusManager.history

Coming Soon...

focusManager.trap(e)  
focusManager.order(a)  
focusManager.autofocus(e)

# The **Focus Traversal API** Proposal

focusManager.isFocusable(e)  
focusManager.forward()  
focusManager.backward()  
focusManager.next(e)  
focusManager.previous(e)  
focusManager.hasFocus(e)  
focusManager.focus(e)  
focusManager.orderedElements()  
focusManager.currentlyFocused  
focusManager.previouslyFocused  
focusManager.history

Coming Soon...

focusManager.trap(e)  
focusManager.order(a)  
focusManager.autofocus(e)

# The **Focus Traversal API** Proposal

`focusManager.isFocusable(e)`  
`focusManager.forward()`  
`focusManager.backward()`  
`focusManager.next(e)`  
`focusManager.previous(e)`  
`focusManager.hasFocus(e)`  
`focusManager.focus(e)`  
`focusManager.orderedElements()`  
`focusManager.currentlyFocused`  
`focusManager.previouslyFocused`  
`focusManager.history`

Coming Soon...

`focusManager.trap(e)`  
`focusManager.order(a)`  
`focusManager.autofocus(e)`

# The **Focus Traversal API** Proposal

---

[github.com/awesomeeng/FocusTraversalAPI](https://github.com/awesomeeng/FocusTraversalAPI)

`npm install focus-traversal-api-polyfill`

# The **Focus Traversal API** Proposal

---

[github.com/awesomeeng/FocusTraversalAPI](https://github.com/awesomeeng/FocusTraversalAPI)

```
npm install focus-traversal-api-polyfill
```



# The **Focus Traversal API** Proposal

---

[github.com/awesomeeng/FocusTraversalAPI](https://github.com/awesomeeng/FocusTraversalAPI)

```
npm install focus-traversal-api-polyfill
```

A man with a neutral expression, wearing a blue and red basketball jersey and a black headband, is shown from the chest up. The background is a blurred crowd in a stadium.

**Perhaps I could be  
of some assistance?**

===== TRIVIAL TO BE TRIVIAL =====

1. Full English music stand  
2. Jordan Lumber  
3. Ball and Taper  
4. Pin and Nail





**Let's make Browser Focus work for us... Spread the word about the Focus Traversal API Proposal!**

<https://github.com/awesomeeng/FocusTraversalAPI>

#FocusTraversalAPI #JavaScript #Web

**@AREINET**



**Let's make Browser Focus work for us... Spread the word about the Focus Traversal API Proposal!**

<https://github.com/awesomeeng/FocusTraversalAPI>

#FocusTraversalAPI #JavaScript #Web

**@AREINET**







Focus Traversal API Explainer

<https://github.com/awesomeeng/FocusTraversalAPI/blob/master/EXPLAINER.md>

Github Repo

<https://github.com/awesomeeng/FocusTraversalAPI>

Github Get Involved Page

[https://github.com/awesomeeng/FocusTraversalAPI/blob/master/GET\\_INVOLVED.md](https://github.com/awesomeeng/FocusTraversalAPI/blob/master/GET_INVOLVED.md)

WICG W3C Discourse Discussion

<https://discourse.wicg.io/t/proposal-focus-traversal-api/3427>

WHATWG Github Issue

<https://github.com/whatwg/html/issues/4784>

ZephJS Library

<https://zephjs.com>