Package ‘shinybrowser’

October 14, 2022

Title Find Out Information About a User’s Web Browser in ‘Shiny’

Version 1.0.0

Description
Sometimes it's useful to know some information about your user in a ‘Shiny’ app. The available information is: browser name (such as ‘Chrome’ or ‘Safari’) and version, device type (mobile or desktop), operating system (such as ‘Windows’ or ‘Mac’ or ‘Android’) and version, and browser dimensions.

URL https://github.com/daattali/shinybrowser
    https://daattali.com/shiny/shinybrowser-demo/

BugReports https://github.com/daattali/shinybrowser/issues

Depends R (>= 3.1.0)
Imports shiny (>= 1.0.4)
Suggests shinydisconnect
License MIT + file LICENSE
Encoding UTF-8
RoxygenNote 7.1.2
NeedsCompilation no

Author Dean Attali [aut, cre] (<https://orcid.org/0000-0002-5645-3493>)
Maintainer Dean Attali <daattali@gmail.com>
Repository CRAN
Date/Publication 2022-05-18 16:40:02 UTC

R topics documented:

  detect ................................................................. 2
  get_all_info .......................................................... 3
  get_browser ............................................................. 5
  get_device .............................................................. 6
  get_os ................................................................. 8
  get_user_agent ......................................................... 9
Detect a user’s browser information

**Description**

This function must be called somewhere in a Shiny app’s UI in order to use any other `{shinybrowser}` functions.

**Usage**

detect()

**Value**

Scripts that are automatically inserted into the UI in order to use this package.

**Accuracy**

It’s important to understand there is no reliable way to detect the information in `{shinybrowser}` with 100% accuracy.

{shinybrowser} makes a best effort at identifying the most accurate information, but some browser/operating system combinations may be difficult to identify. Users can also use a variety of tools to deliberately spoof this information.

With that in mind, `{shinybrowser}` should detect the correct information in most cases.

**Supported values**

Only major browsers and operating systems are supported, which means that the RStudio Viewer may result in an "UNKNOWN" browser, and unpopular operating systems may also result in "UNKNOWN".

For a list of values that can be detected, see \`SUPPORTED_BROWSERS\`, \`SUPPORTED_DEVICES\`, and \`SUPPORTED_OPERATING_SYSTEMS\`. 
Mobile vs desktop vs tablet

{shinybrowser} attempts to detect whether a device is "mobile" or "desktop". The distinction between mobile and desktop is not always clear, so if what you actually care about is the size of the device, it might be better to use \texttt{get_width()}

Tablets return ambiguous results; some tablets self-report as mobile devices while others as desktop.

Width and height

The width and height of the browser window are only reported once, when the \texttt{detect()} function is initially called. If the user resizes the browser window, the new dimensions are not reported until the page is refreshed.

See Also

\texttt{get_all_info()}, \texttt{get_browser()}, \texttt{get_os()}, \texttt{get_device()}, \texttt{get_width()}

Examples

```r
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Your browser information:",
   verbatimTextOutput("browser_info")
  )
  server <- function(input, output, session) {
    output$browser_info <- renderPrint({
      shinybrowser::get_all_info()
    })
  }
  shinyApp(ui, server)
}
```

\texttt{get_all_info} \hspace{1cm} \textit{Get all information about user's browser}

\textbf{Description}

Get a list with all the information detected about the user’s browser.

The list is reactive, therefore it must be accessed inside a reactive context (such as an \texttt{observe} or \texttt{reactive}).

{shinybrowser} must be initialized with a call to \texttt{detect()} in the app’s ui.
Usage

get_all_info()

Value

List with all information detected about the user's browser: device, browser, os, dimensions, user_agent

Accuracy

It's important to understand there is no reliable way to detect the information in {shinybrowser} with 100% accuracy.

{shinybrowser} makes a best effort at identifying the most accurate information, but some browser/operating system combinations may be difficult to identify. Users can also use a variety of tools to deliberately spoof this information.

With that in mind, {shinybrowser} should detect the correct information in most cases.

Supported values

Only major browsers and operating systems are supported, which means that the RStudio Viewer may result in an "UNKNOWN" browser, and unpopular operating systems may also result in "UNKNOWN".

For a list of values that can be detected, see SUPPORTED_BROWSERS, SUPPORTED_DEVICES, and SUPPORTED_OPERATING_SYSTEMS.

Mobile vs desktop vs tablet

{shinybrowser} attempts to detect whether a device is "mobile" or "desktop". The distinction between mobile and desktop is not always clear, so if what you actually care about is the size of the device, it might be better to use get_width().

Tablets return ambiguous results; some tablets self-report as mobile devices while others as desktop.

Width and height

The width and height of the browser window are only reported once, when the detect() function is initially called. If the user resizes the browser window, the new dimensions are not reported until the page is refreshed.

See Also

detect(), get_browser(), get_browser_version(), get_os(), get_os_version(), get_device(), get_width(), get_height(), get_user_agent(), SUPPORTED_BROWSERS, SUPPORTED_DEVICES, SUPPORTED_OPERATING_SYSTEMS
get_browser

Examples
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Your browser information:",
    verbatimTextOutput("browser_info")
  )
  server <- function(input, output, session) {
    output$browser_info <- renderPrint({
      shinybrowser::get_all_info()
    })
  }
  shinyApp(ui, server)
}

get_browser

Get user’s browser

Description
Get the user’s browser name (such as "Chrome" or "Firefox") and version.

The value is reactive, therefore it must be accessed inside a reactive context (such as an observe or reactive).

{shinybrowser} must be initialized with a call to detect() in the app’s ui.

Usage
get_browser()

get_browser_version()

Value
User’s detected browser type
User’s detected browser version

Accuracy
It’s important to understand there is no reliable way to detect the information in {shinybrowser} with 100% accuracy.

{shinybrowser} makes a best effort at identifying the most accurate information, but some browser/operating system combinations may be difficult to identify. Users can also use a variety of tools to deliberately spoof this information.
With that in mind, {shinybrowser} should detect the correct information in most cases.

**Supported values**

Only major browsers and operating systems are supported, which means that the RStudio Viewer may result in an "UNKNOWN" browser, and unpopular operating systems may also result in "UNKNOWN".

For a list of values that can be detected, see SUPPORTED_BROWSERS, SUPPORTED_DEVICES, and SUPPORTED_OPERATING_SYSTEMS.

**See Also**

detect(), get_all_info(), is_browser_ie(), is_browser_chrome(), is_browser_firefox(), SUPPORTED_BROWSERS

**Examples**

```r
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Your browser: ",
    textOutput("browser_info")
  )
  server <- function(input, output, session) {
    output$browser_info <- renderText({
      paste(shinybrowser::get_browser(), "version", shinybrowser::get_browser_version())
    })
  }
  shinyApp(ui, server)
}
```

---

**get_device**

Get user's device (mobile or desktop)

**Description**

The value is reactive, therefore it must be accessed inside a reactive context (such as an observe or reactive).

{shinybrowser} must be initialized with a call to detect() in the app's ui.

**Usage**

get_device()
**get_device**

**Value**

User’s detected device type ("Mobile" or "Desktop")

**Accuracy**

It’s important to understand there is no reliable way to detect the information in `{shinybrowser}` with 100% accuracy.

{shinybrowser} makes a best effort at identifying the most accurate information, but some browser/operating system combinations may be difficult to identify. Users can also use a variety of tools to deliberately spoof this information.

With that in mind, `{shinybrowser}` should detect the correct information in most cases.

**Mobile vs desktop vs tablet**

{shinybrowser} attempts to detect whether a device is "mobile" or "desktop". The distinction between mobile and desktop is not always clear, so if what you actually care about is the size of the device, it might be better to use `get_width()`.

Tablets return ambiguous results; some tablets self-report as mobile devices while others as desktop.

**See Also**

`detect()`, `get_all_info()`, `is_device_mobile()`, `is_device_desktop()`, `get_width()`, `get_height()`

**Examples**

```r
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect()
  
    "Your device:",
    textOutput("device_info")
  )

  server <- function(input, output, session) {
    output$device_info <- renderText({
      shinybrowser::get_device()
    })
  }

  shinyApp(ui, server)
}
```
get_os

Get user’s operating system

Description
Get the user’s operating system (such as "Windows" or "Mac" or "Android") and version (such as "10" for Windows or "OS X" for Mac).

The value is reactive, therefore it must be accessed inside a reactive context (such as an observe or reactive).

{shinybrowser} must be initialized with a call to detect() in the app’s ui.

Usage
get_os()
get_os_version()

Value
User’s detected operating system
User’s detected operating system version

Accuracy
It’s important to understand there is no reliable way to detect the information in {shinybrowser} with 100% accuracy.

{shinybrowser} makes a best effort at identifying the most accurate information, but some browser/operating system combinations may be difficult to identify. Users can also use a variety of tools to deliberately spoof this information.

With that in mind, {shinybrowser} should detect the correct information in most cases.

Supported values
Only major browsers and operating systems are supported, which means that the RStudio Viewer may result in an "UNKNOWN" browser, and unpopular operating systems may also result in "UNKNOWN".

For a list of values that can be detected, see SUPPORTED_BROWSERS, SUPPORTED_DEVICES, and SUPPORTED_OPERATING_SYSTEMS.

See Also
detect(), get_all_info(), is_os_windows(), is_os_mac(), SUPPORTED_OPERATING_SYSTEMS
get_user_agent

Examples

if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Your operating system:",
    textOutput("os_info")
  )
  server <- function(input, output, session) {
    output$os_info <- renderText({
      paste(shinybrowser::get_os(), "version", shinybrowser::get_os_version())
    })
  }
  shinyApp(ui, server)
}

get_user_agent Get user agent string from the browser

Description

This function exposes the user agent that is reported by the browser, but it should only be used for troubleshooting purposes.

The value is reactive, therefore it must be accessed inside a reactive context (such as an observe or reactive).

{shinybrowser} must be initialized with a call to detect() in the app’s ui.

Usage

get_user_agent()

Value

User’s user-agent string

See Also

detect(), get_all_info()

Examples

if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),

  )
}
get_width

Get user’s browser dimensions (in pixels)

Description
The value is reactive, therefore it must be accessed inside a reactive context (such as an observe or reactive).

{shinybrowser} must be initialized with a call to detect() in the app’s ui.

Usage
get_width()
get_height()

Value
User’s detected browser width in pixels
User’s detected browser height in pixels

Width and height
The width and height of the browser window are only reported once, when the detect() function is initially called. If the user resizes the browser window, the new dimensions are not reported until the page is refreshed.

See Also
detect(), get_all_info()

Examples
if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    shinybrowser::detect(),
is_browser_chrome

"Your browser dimensions:",
textOutput("browser_dim")
)
server <- function(input, output, session) {
  output$browser_dim <- renderText({
    paste0(shinybrowser::get_width(), "x", shinybrowser::get_height())
  })
  shinyApp(ui, server)
}

is_browser_chrome  Is the user using Chrome?

Description

Convenience function that checks if the user’s browser is detected as Chrome. See get_browser() for details.

Usage

is_browser_chrome()

Value

Whether or not this user using Chrome

Examples

if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Are you using Chrome?",
    textOutput("result")
  )
  server <- function(input, output, session) {
    output$result <- renderText({
      shinybrowser::is_browser_chrome()
    })
  }
  shinyApp(ui, server)
}
is_browser_firefox  Is the user using Firefox?

Description
Convenience function that checks if the user’s browser is detected as Firefox. See get_browser() for details.

Usage
is_browser_firefox()

Value
Whether or not this user using Firefox

Examples
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Are you using Firefox?",
    textOutput("result")
  )
  server <- function(input, output, session) {
    output$result <- renderText({
      shinybrowser::is_browser_firefox()
    })
  }
  shinyApp(ui, server)
}

is_browser_ie  Is the user using Internet Explorer?

Description
Convenience function that checks if the user’s browser is detected as Internet Explorer. See get_browser() for details.

Usage
is_browser_ie()
is_device_desktop

Value

Whether or not this user using Internet Explorer

Examples

```r
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Are you using Internet Explorer?",
    textOutput("result")
  )
  server <- function(input, output, session) {
    output$result <- renderText({
      shinybrowser::is_browser_ie()
    })
  }
  shinyApp(ui, server)
}
```

---

is_device_desktop  Is the user on a desktop device?

Description

Convenience function that checks if the user's device is detected as desktop. See `get_device()` for details.

Usage

`is_device_desktop()`

Value

Whether or not this user is on desktop

Examples

```r
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Are you on desktop?",
    textOutput("result")
  )
  server <- function(input, output, session) {
    output$result <- renderText({
      shinybrowser::is_device_desktop()
    })
  }
  shinyApp(ui, server)
}```
is_device_mobile

Is the user on a mobile device?

Description

Convenience function that checks if the user’s device is detected as mobile. See get_device() for details.

Usage

is_device_mobile()

Value

Whether or not this user is on mobile

Examples

if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Are you on mobile?",
    textOutput("result")
  )
  server <- function(input, output, session) {
    output$result <- renderText({
      shinybrowser::is_device_mobile()
    })
  }
  shinyApp(ui, server)
}
is_os_mac

Is the user on Mac?

Description

Convenience function that checks if the user’s operating system is detected as Mac. See get_os() for details.

Usage

is_os_mac()

Value

Whether or not this user using MacOS

Examples

if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Are you on Mac?",
    textOutput("result")
  )
  server <- function(input, output, session) {
    output$result <- renderText({
      shinybrowser::is_os_mac()
    })
    shinyApp(ui, server)
  }
}

is_os_windows

Is the user on Windows?

Description

Convenience function that checks if the user’s operating system is detected as Windows. See get_os() for details.

Usage

is_os_windows()
Value

Whether or not this user using Windows

Examples

```r
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    shinybrowser::detect(),
    "Are you on Windows?",
    textOutput("result")
  )
  server <- function(input, output, session) {
    output$result <- renderText({
      shinybrowser::is_os_windows()
    })
  }
  shinyApp(ui, server)
}
```

---

**Supported Browsers**

Browsers that can be detected with `shinybrowser`

**Description**

Browsers that can be detected with `shinybrowser`

---

**Supported Devices**

Devices that can be detected with `shinybrowser`

**Description**

Devices that can be detected with `shinybrowser`

---

**Supported Operating Systems**

Operating systems that can be detected with `shinybrowser`

**Description**

Operating systems that can be detected with `shinybrowser`
Index

* datasets
  SUPPORTED_BROWSERS, 16
  SUPPORTED_DEVICES, 16
  SUPPORTED_OPERATING_SYSTEMS, 16

detect, 2
detect(), 3–10

get_all_info, 3
get_all_info(), 3, 6–10
get_browser, 5
get_browser(), 3, 4, 11, 12
get_browser_version() (get_browser), 5
get_browser_version(), 4
get_device, 6
get_device(), 3, 4, 13, 14
get_height (get_width), 10
get_height(), 4, 7
get_os, 8
get_os(), 3, 4, 15
get_os_version() (get_os), 8
get_os_version(), 4
get_user_agent, 9
get_user_agent(), 4
get_width, 10
get_width(), 3, 4, 7

is_browser_chrome, 11
is_browser_chrome(), 6
is_browser_firefox, 12
is_browser_firefox(), 6
is_browser_ie, 12
is_browser_ie(), 6
is_device.desktop, 13
is_device.desktop(), 7
is_device.mobile, 14
is_device_mobile(), 7
is_os_mac, 15
is_os_mac(), 8
is_os_windows, 15

is_os_windows(), 8
observe, 3, 5, 6, 8–10
reactive, 3, 5, 6, 8–10
SUPPORTED_BROWSERS, 2, 4, 6, 8, 16
SUPPORTED_DEVICES, 2, 4, 6, 8, 16
SUPPORTED_OPERATING_SYSTEMS, 2, 4, 6, 8, 16