

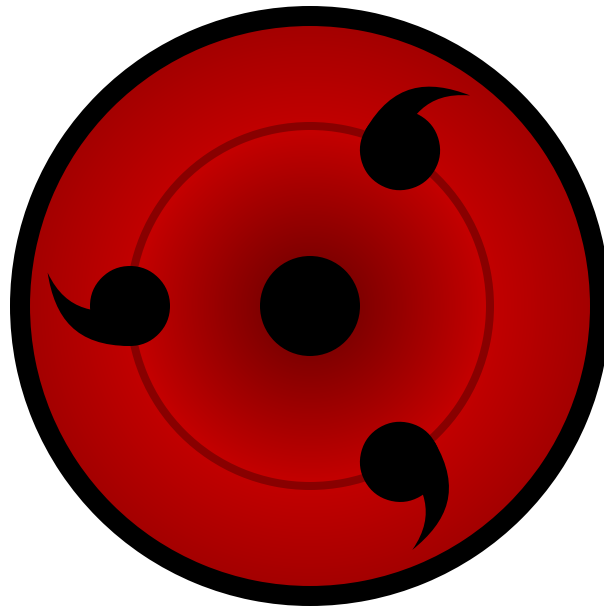
Presentation Ninja



with xaringan

Yihui Xie

2016/12/12



xaringan

/jæ.'riŋ.gæn/

Get Started

Hello World

$t = 0$

Install the **xaringan** package from **Github**:

```
devtools::install_github("yihui/xaringan")
```

Hello World

$t = 0$

Install the **xaringan** package from **Github**:

```
devtools::install_github("yihui/xaringan")
```

You are recommended to use the **RStudio IDE**, but you do not have to.

- Create a new R Markdown document from the menu File -> New File -> R Markdown -> From Template -> Ninja Presentation;¹

Hello World

$t = 0$

Install the **xaringan** package from **Github**:

```
devtools::install_github("yihui/xaringan")
```

You are recommended to use the **RStudio IDE**, but you do not have to.

- Create a new R Markdown document from the menu File -> New File -> R Markdown -> From Template -> Ninja Presentation;¹
- Click the Knit button to compile it;

Hello World

$t = 0$

Install the **xaringan** package from **Github**:

```
devtools::install_github("yihui/xaringan")
```

You are recommended to use the **RStudio IDE**, but you do not have to.

- Create a new R Markdown document from the menu File -> New File -> R Markdown -> From Template -> Ninja Presentation;¹
- Click the Knit button to compile it;
- or use the **RStudio Addin**² "Infinite Moon Reader" to live preview the slides (every time you update and save the Rmd document, the slides will be automatically reloaded in RStudio Viewer).

[1] ???????

[2] See **#2** if you do not see the template or addin in RStudio.



You only live once!

Hello Ninja

As a presentation ninja, you certainly should not be satisfied by the "Hello World" example. You need to understand more about two things:

1. The **remark.js** library;
2. The **xaringan** package;

Basically **xaringan** injected the chakra of R Markdown (minus Pandoc) into **remark.js**. The slides are rendered by remark.js in the web browser, and the Markdown source needed by remark.js is generated from R Markdown (**knitr**).

remark.js

You can see an introduction of remark.js from [its homepage](#). You should read the [remark.js Wiki](#) at least once to know how to

- create a new slide (Markdown syntax* and slide properties);
- format a slide (e.g. text alignment);
- configure the slideshow;
- and use the presentation (keyboard shortcuts).

It is important to be familiar with remark.js before you can understand the options in **xaringan**.

[*] It is different with Pandoc's Markdown! It is limited but should be enough for presentation purposes. Come on... You do not need a slide for the Table of Contents! Well, the Markdown support in remark.js [may be improved](#) in the future.

A close-up photograph of a man with light brown hair and a mustache, smiling at the camera. He is wearing a dark jacket over a light-colored button-down shirt. The background is dimly lit, showing a room with warm wall sconces, a bulletin board with papers, and other people in the distance.

I was so happy to have discovered remark.js!

Using xaringan

xaringan

Provides an R Markdown output format `xaringan::moon_reader` as a wrapper for `remark.js`, and you can use it in the YAML metadata, e.g.

```
---  
title: "A Cool Presentation"  
output:  
  xaringan::moon_reader  
    yolo: true  
    nature:  
      autoplay: 30000  
---
```

See the help page `?xaringan::moon_reader` for all possible options that you can use.

remark.js vs xaringan

Some differences between using remark.js (left) and using **xaringan** (right):

1. Start with a boilerplate HTML file;
2. Plain Markdown;
3. Write JavaScript to autoplay slides;
4. Manually configure MathJax;
5. Highlight code with `*`;
6. Edit Markdown source and refresh browser to see updated slides;

1. Start with an R Markdown document;
2. R Markdown (can embed R/other code chunks);
3. Provide an option `autoplay`;
4. MathJax just works;^{*}
5. Highlight code with `{{}}`;
6. The RStudio addin "Infinite Moon Reader" automatically refreshes slides on changes;

[*] Not really. See next page.

Math Expressions

You can write LaTeX math expressions inside a pair of dollar signs, e.g. `$\alpha+\beta$` renders $\alpha + \beta$. You can use the display style with double dollar signs:

`$$\bar{X}=\frac{1}{n}\sum_{i=1}^nX_i$$`

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$$

Limitations:

1. The source code of a LaTeX math expression must be in one line, unless it is inside a pair of double dollar signs, in which case the starting `$$` must appear in the very beginning of a line, followed immediately by a non-space character, and the ending `$$` must be at the end of a line, led by a non-space character;
2. There should not be spaces after the opening `$` or before the closing `$`.
3. Math does not work on the title slide (see [#61](#) for a workaround).

R Code

```
# a boring regression
```

```
fit = lm(dist ~ 1 + speed, data = cars)
coef(summary(fit))
```

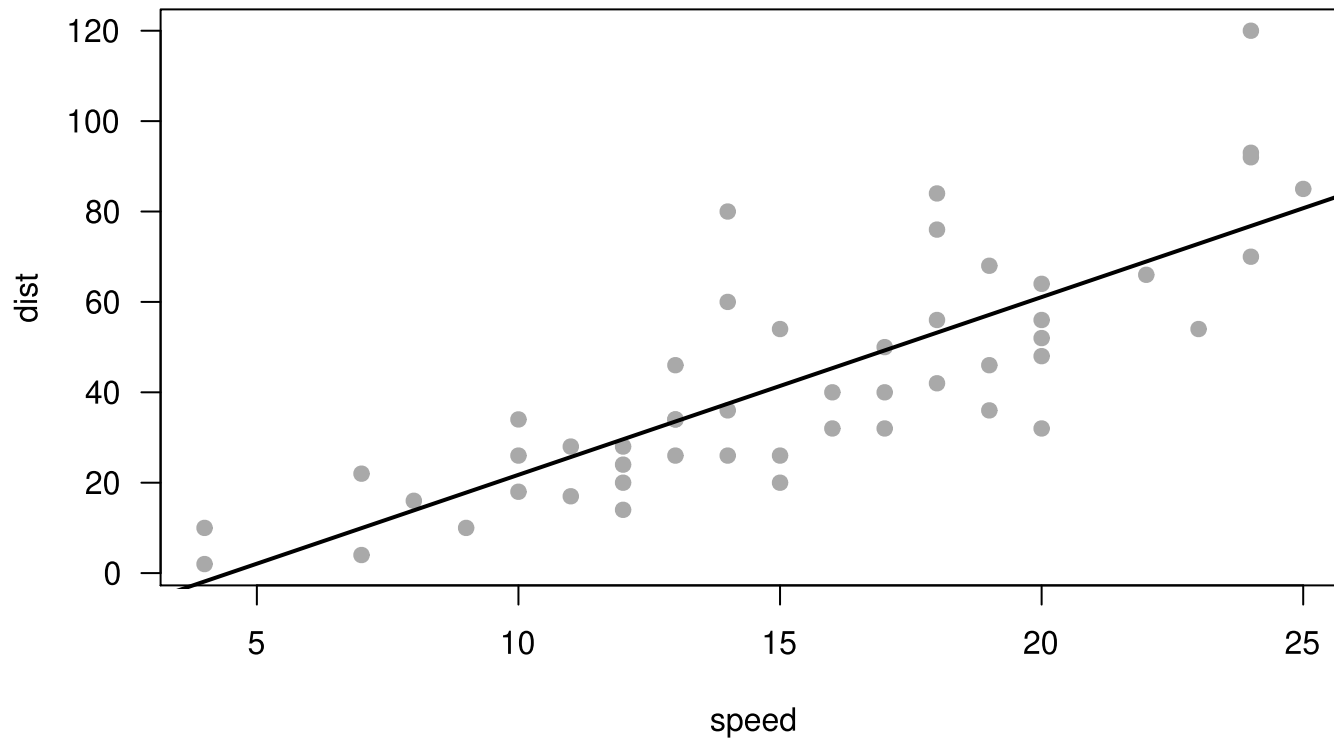
```
#               Estimate Std. Error   t value      Pr(>|t|)
# (Intercept) -17.579095   6.7584402 -2.601058 0.012318816153808781
# speed        3.932409   0.4155128  9.463990 0.0000000000001489836
```

```
dojutsu = c('????', '??', '????', '??', '????', '????')
grep('?', dojutsu, value = TRUE)
```

```
# [1] "????" "??"
```

R Plots

```
par(mar = c(4, 4, 1, .1))  
plot(cars, pch = 19, col = 'darkgray', las = 1)  
abline(fit, lwd = 2)
```



Tables

If you want to generate a table, make sure it is in the HTML format (instead of Markdown or other formats), e.g.,

```
knitr::kable(head(iris), format = 'html')
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

HTML Widgets

I have not thoroughly tested HTML widgets against **xaringan**. Some may work well, and some may not. It is a little tricky.

Similarly, the Shiny mode (`runtime: shiny`) does not work. I might get these issues fixed in the future, but these are not of high priority to me. I never turn my presentation into a Shiny app. When I need to demonstrate more complicated examples, I just launch them separately. It is convenient to share slides with other people when they are plain HTML/JS applications.

See the next page for two HTML widgets.

```
library(leaflet)
```






```
leaflet() %>% addTiles() %>% setView(-93.65, 42.0285, zoom = 17)
```



```
DT::datatable(  
  head(iris, 10),  
  fillContainer = FALSE, options = list(pageLength = 8)  
)
```

Show entries

Search:

	Sepal.Length 	Sepal.Width 	Petal.Length 	Petal.Width 	Species 
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5	3.4	1.5	0.2	setosa

Showing 1 to 8 of 10 entries

Previous

1

2

Next

Some Tips

- When you use the "Infinite Moon Reader" addin in RStudio, your R session will be blocked by default. You can click the red button on the right of the console to stop serving the slides, or use the *daemonized* mode so that it does not block your R session. To do the latter, you can set the option

```
options(servr.daemon = TRUE)
```

in your current R session, or in `~/.Rprofile` so that it is applied to all future R sessions. I do the latter by myself.

To know more about the web server, see the **servr** package.

Some Tips

- When you use the "Infinite Moon Reader" addin in RStudio, your R session will be blocked by default. You can click the red button on the right of the console to stop serving the slides, or use the *daemonized* mode so that it does not block your R session. To do the latter, you can set the option

```
options(servr.daemon = TRUE)
```

in your current R session, or in `~/.Rprofile` so that it is applied to all future R sessions. I do the latter by myself.

To know more about the web server, see the **servr** package.

- Do not forget to try the `yolo` option of `xaringan::moon_reader`.

```
output:
  xaringan::moon_reader:
    yolo: true
```


Some Tips

- Slides can be automatically played if you set the `autoplay` option under `nature`, e.g. go to the next slide every 30 seconds in a lightning talk:

```
output:  
  xaringan::moon_reader:  
    nature:  
      autoplay: 30000
```

Some Tips

- Slides can be automatically played if you set the `autoplay` option under `nature`, e.g. go to the next slide every 30 seconds in a lightning talk:

```
output:
  xaringan::moon_reader:
    nature:
      autoplay: 30000
```

- A countdown timer can be added to every page of the slides using the `countdown` option under `nature`, e.g. if you want to spend one minute on every page when you give the talk, you can set:

```
output:
  xaringan::moon_reader:
    nature:
      countdown: 60000
```

Then you will see a timer counting down from `01:00`, to `00:59`, `00:58`, ...
When the time is out, the timer will continue but the time turns red.

Some Tips

- There are several ways to build incremental slides. See [this presentation](#) for examples.
- The option `highlightLines: true` of nature will highlight code lines that start with `*`, or are wrapped in `{ { } }`, or have trailing comments `# <<`;

```
output:  
  xaringan::moon_reader:  
    nature:  
      highlightLines: true
```

See examples on the next page.

Some Tips

An example using a leading *:

```
```\r
if (TRUE) {
* message("Very important!")
}
```\r
```

Output:

```
if (TRUE) {
  message("Very important!")
}
```

This is invalid R code, so it is a plain fenced code block that is not executed.

An example using {{{}}:

```
```\{r tidy=FALSE}
if (TRUE) {
{{{ message("Very important!") }}}
}
```\r
```

Output:

```
if (TRUE) {
  message("Very important!")
}
```

```
## Very important!
```

It is valid R code so you can run it. Note that {{{}}} can wrap an R expression of multiple lines.

Some Tips

An example of using the trailing comment #<< to highlight lines:

```
```{r tidy=FALSE}  
library(ggplot2)
ggplot(mtcars) +
 aes(mpg, disp) +
 geom_point() + #<<
 geom_smooth() #<<
```
```

Output:

```
library(ggplot2)  
ggplot(mtcars) +  
  aes(mpg, disp) +  
  geom_point() +  
  geom_smooth()
```

Some Tips

- To make slides work offline, you need to download a copy of remark.js in advance, because **xaringan** uses the online version by default (see the help page `?xaringan::moon_reader`).
- You can use `xaringan::summon_remark()` to download the latest or a specified version of remark.js. By default, it is downloaded to `libs/remark-latest.min.js`.
- Then change the `chakra` option in YAML to point to this file, e.g.

```
output:  
  xaringan::moon_reader:  
    chakra: libs/remark-latest.min.js
```

- If you used Google fonts in slides (the default theme uses *Yanone Kaffeesatz*, *Droid Serif*, and *Source Code Pro*), they won't work offline unless you download or install them locally. The Heroku app **google-webfonts-helper** can help you download fonts and generate the necessary CSS.

Macros

- remark.js **allows users to define custom macros** (JS functions) that can be applied to Markdown text using the syntax `![:macroName arg1 arg2 ...]` or `![macroName](argument)`. For example, before remark.js initializes the slides, you can define a macro named `scale`:

```
remark.macros.scale = function (percentage) {  
  var url = this;  
  return '
```

Macros (continued)

- To insert macros in **xaringan** slides, you can use the option `beforeInit` under the option `nature`, e.g.,

```
output:
  xaringan::moon_reader:
    nature:
      beforeInit: "macros.js"
```

You save your `remark.js` macros in the file `macros.js`.

- The `beforeInit` option can be used to insert arbitrary JS code before `remark.create()`. Inserting macros is just one of its possible applications.

CSS

Among all options in `xaringan::moon_reader`, the most challenging but perhaps also the most rewarding one is `css`, because it allows you to customize the appearance of your slides using any CSS rules or hacks you know.

You can see the default CSS file [here](#). You can completely replace it with your own CSS files, or define new rules to override the default. See the help page `?xaringan::moon_reader` for more information.

CSS

For example, suppose you want to change the font for code from the default "Source Code Pro" to "Ubuntu Mono". You can create a CSS file named, say, `ubuntu-mono.css`:

```
@import url(https://fonts.googleapis.com/css?family=Ubuntu+Mono:400,700);  
.remark-code, .remark-inline-code { font-family: 'Ubuntu Mono'; }
```

Then set the `css` option in the YAML metadata:

```
output:  
  xaringan::moon_reader:  
    css: ["default", "ubuntu-mono.css"]
```

Here I assume `ubuntu-mono.css` is under the same directory as your Rmd.

See [yihui/xaringan#83](#) for an example of using the **Fira Code** font, which supports ligatures in program code.

Themes

Don't want to learn CSS? Okay, you can use some user-contributed themes. A theme typically consists of two CSS files `foo.css` and `foo-fonts.css`, where `foo` is the theme name. Below are some existing themes:

```
names(xaringan::list_css())
```

```
## [1] "default-fonts"      "default"            "hygge"
## [4] "metropolis-fonts"  "metropolis"         "rladies-fonts"
## [7] "rladies"
```

To use a theme, you can specify the `css` option as an array of CSS filenames (without the `.css` extensions), e.g.,

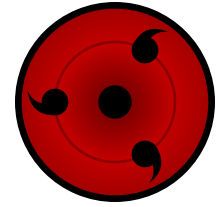
```
output:
  xaringan::moon_reader:
    css: [default, metropolis, metropolis-fonts]
```

If you want to contribute a theme to **xaringan**, please read [this blog post](#).



Naruto

Sharingan



The R package name **xaringan** was derived¹ from **Sharingan**, a dōjutsu in the Japanese anime *Naruto* with two abilities:

- the "Eye of Insight"
- the "Eye of Hypnotism"

I think a presentation is basically a way to communicate insights to the audience, and a great presentation may even "hypnotize" the audience.^{2,3}

[1] In Chinese, the pronunciation of *X* is *Sh* /ʃ/ (as in *shrimp*). Now you should have a better idea of how to pronounce my last name *Xie*.

[2] By comparison, bad presentations only put the audience to sleep.

[3] Personally I find that setting background images for slides is a killer feature of *remark.js*. It is an effective way to bring visual impact into your presentations.

Naruto terminology

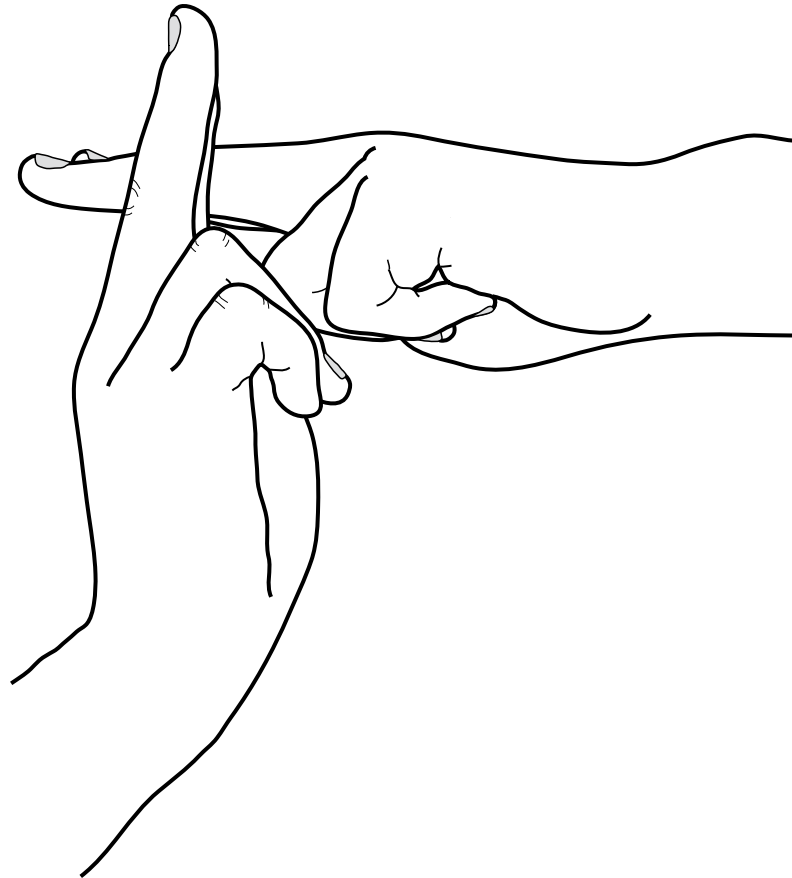
The **xaringan** package borrowed a few terms from Naruto, such as

- **Sharingan** (???; the package name)
- The **moon reader** (??; an attractive R Markdown output format)
- **Chakra** (???; the path to the remark.js library, which is the power to drive the presentation)
- **Nature transformation** (????; transform the chakra by setting different options)
- The **infinite moon reader** (????; start a local web server to continuously serve your slides)
- The **summoning technique** (download remark.js from the web)

You can click the links to know more about them if you want. The jutsu "Moon Reader" may seem a little evil, but that does not mean your slides are evil.

Hand seals (☐?)

Press h or ? to see the possible ninjutsu you can use in remark.js.



Thanks!

Slides created via the R package **xaringan**.

The chakra comes from **remark.js**, **knitr**, and **R Markdown**.