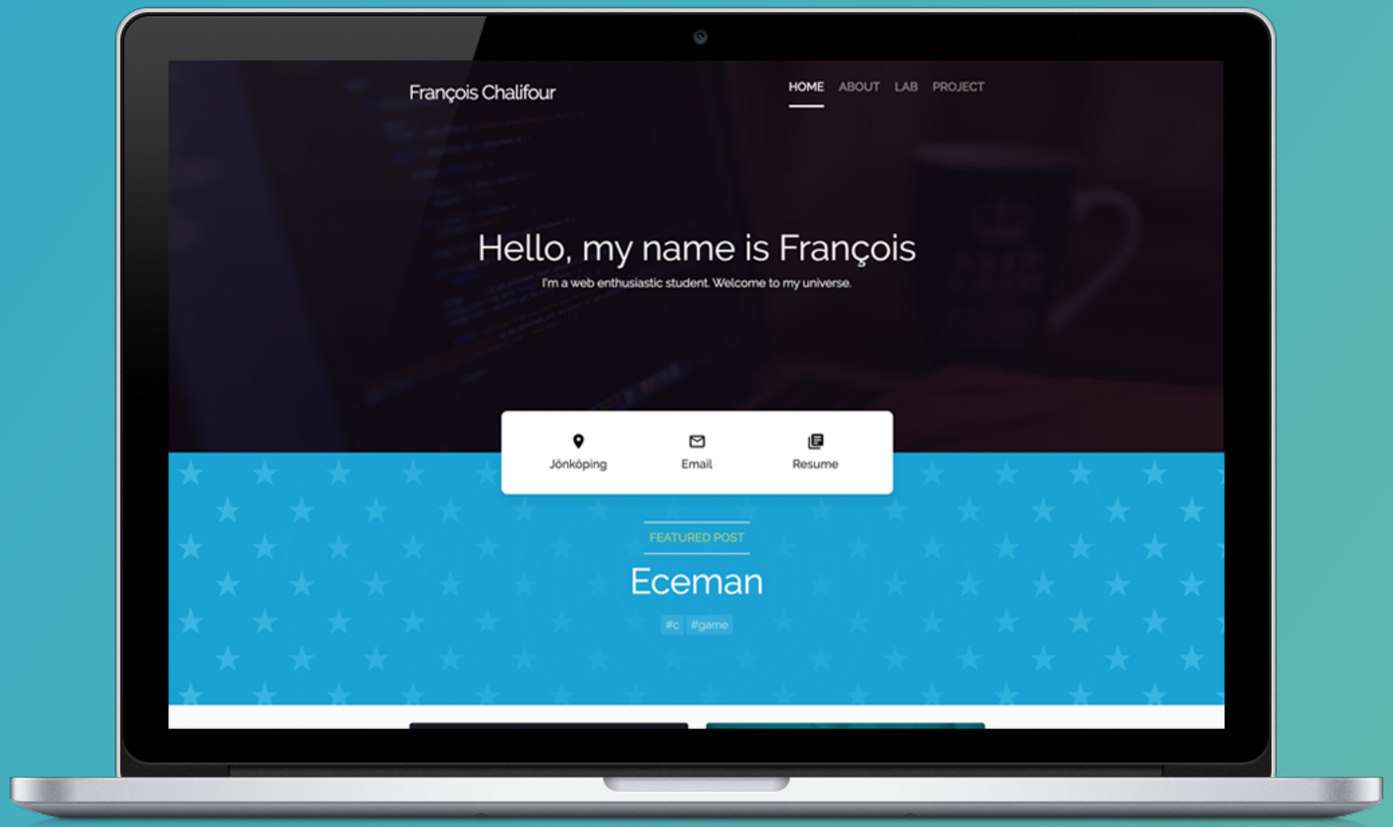
PERF MATTERS

Making the web a smooth place

Hi, I’m François

- francoischalifour.com
- github.com/francoischalifour
- twitter.com/FrancoisChlfr



WHY YOU SHOULD BE

concerned.

WHY YOU SHOULD BE *concerned.*

- **47%** users expect page load < 2s
- **40%** abandon websites that take more than 3 seconds to load
- **79%** of shoppers are less likely to buy from a slow website
- People still browse on poor mobile network connections
- Page speed is a part of Google’s ranking algorithm

"Focus on the user and all else will follow."

– Google

Content

Speed up your animations

Boost your page load

Test the performance

WHAT IS *slow?*

Depends on the user perception.

WHAT DOES THE USER

feel?

Delay	User reaction
0 - 100 ms	Instant
100 - 300 ms	Slight perceptible delay
300 - 1000 ms	Task focus, perceptible delay
1000+ ms	Mental context switch

INTRODUCING

RAIL.

Response, Animation, Idle, Load.

Response

Animation

Idle

Load

100MS

To bring something back on screen so the user feels a response **instantly**.

Response

Animation

Idle

Load

16ms

For scroll, gestures and transitions so visual changes feel **smooth and consistent**.

Response

Animation

Idle

Load

50MS

To work on the background so

the next user interaction is **responsive**.

Response

Animation

Idle

Load

1000MS

To deliver the experience so

the user’s flow is **seamless**.

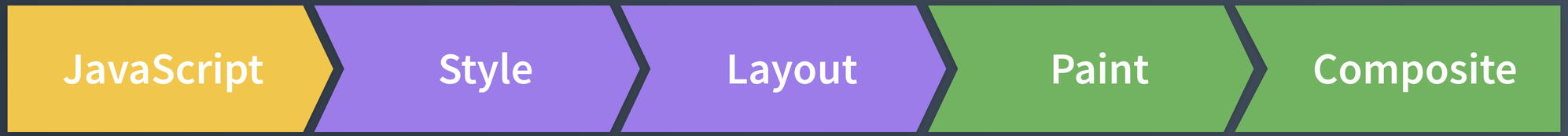
HOW TO ACHIEVE THESE

goals?

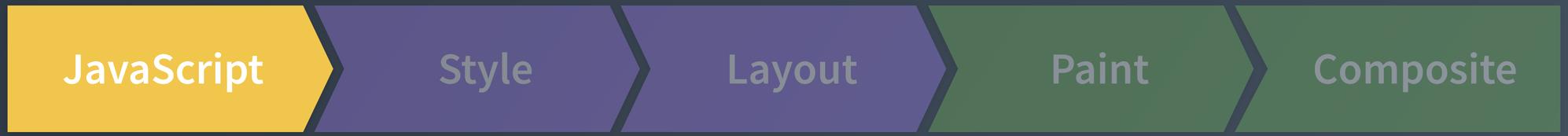
First things first.

PIXELS ARE

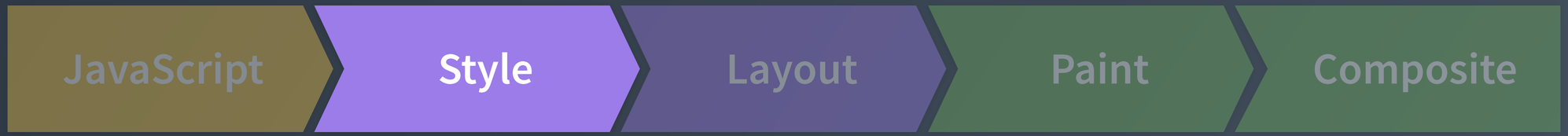
expensive.



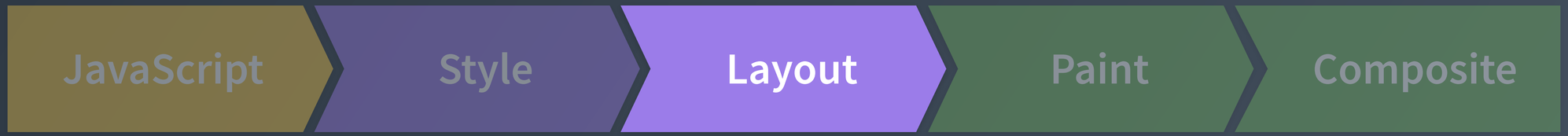
The pixel pipeline



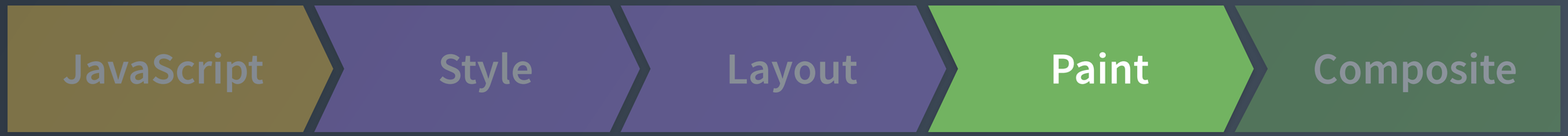
Handles work that will result in **visual changes**.



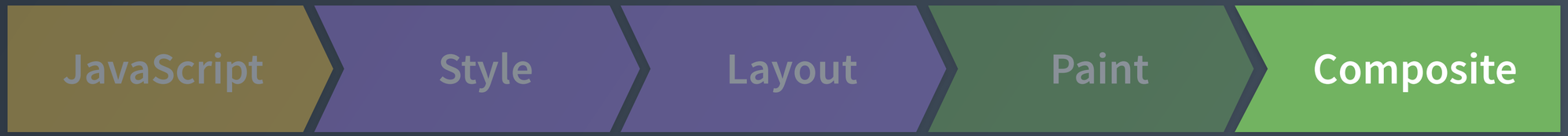
Figures out which **CSS rules** apply to which elements.



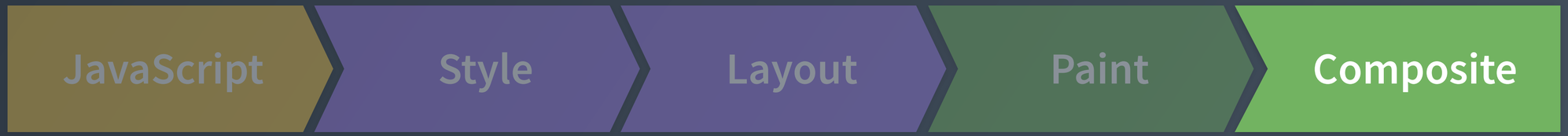
Calculates **how much space** it takes up and **where it is** on screen.



Draws out surfaces and **fills in pixels**.



Draws to the screen in **correct order**.



The pixel pipeline ✓

- ⊖ Don’t change a **layout** property
  - *Will recalculate the width, height, top and right position*
  - *Will reflow the whole page and repaint inner elements*
- ○ Avoid changing a **paint** property
  - *Will repaint the element*
- ◉ Change **composite** properties

ANIMATE A CHEAP *CSS property.*

Position	`transform: translate(npx, npx);`
Scale	`transform: scale(n);`
Rotation	`transform: rotate(ndeg);`
Skew	`transform: skew(X|Y)(ndeg);`
Matrix	`transform: matrix(3d)(...);`
Opacity	`opacity: 0...1;`

FLIP YOUR *animations.*

First, Last, Invert, Play.

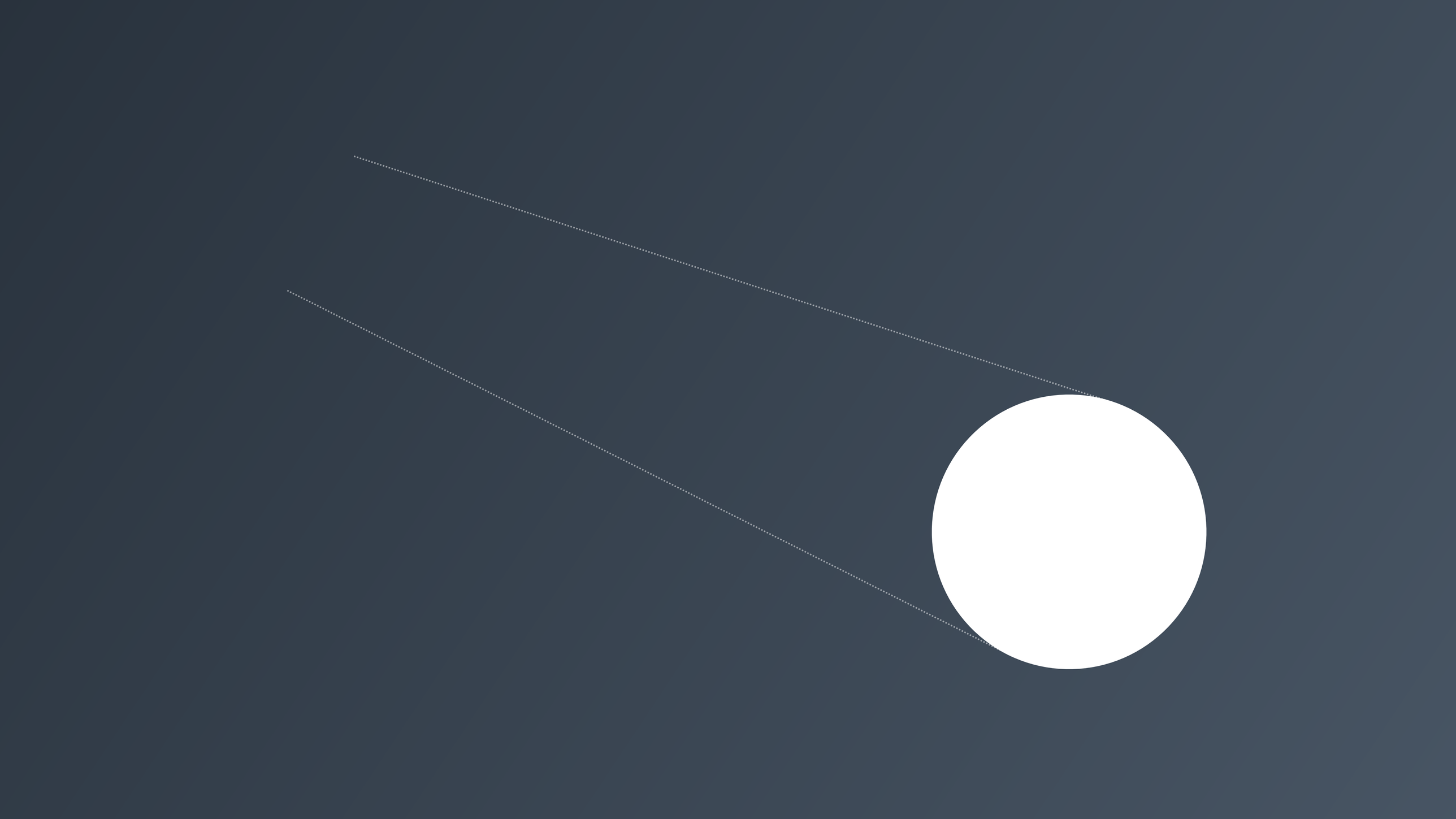
FLIP YOUR *animations.*

- **First**: the initial state of the element involved in the transition
- **Last**: the final state of the element
- **Invert**: figure out from the first and last how the element has changed
- **Play**: switch on transitions for any of the properties you changed, and then remove the inversion changes

```
circle.getBoundingClientRect()
```

```
circle.classList.add('is-final')
```

```
circle.getBoundingClientRect()
```



```
transform: translate(-900px, -322px) scale(.2);
```

```
transform: none;
```

```
// Save the element in a variable
const circle = document.getElementById('circle')

// Get its position at the initial state
const firstPosition = circle.getBoundingClientRect()

// Move it to its final state
circle.classList.add('is-final')

// Get its position at the final state
const lastPosition = circle.getBoundingClientRect()

// Compute the reverse motion
const invertTop = firstPosition.top - lastPosition.top

// Animate from the inverted position to the last
const player = circle.animate([
  { transform: `translateY(${invertTop}px)` },
  { transform: 'translateY(0)' }
], {
  duration: 700,
  easing: 'ease-in-out'
})
```

COMPRESS YOUR

data.

Images

Files

GZIP

- Use the right format (PNG, GIF, JPG)
- Compress your images
- Use thumbnails instead of HTML resizing
- Use CSS effects instead of images

Images

Files

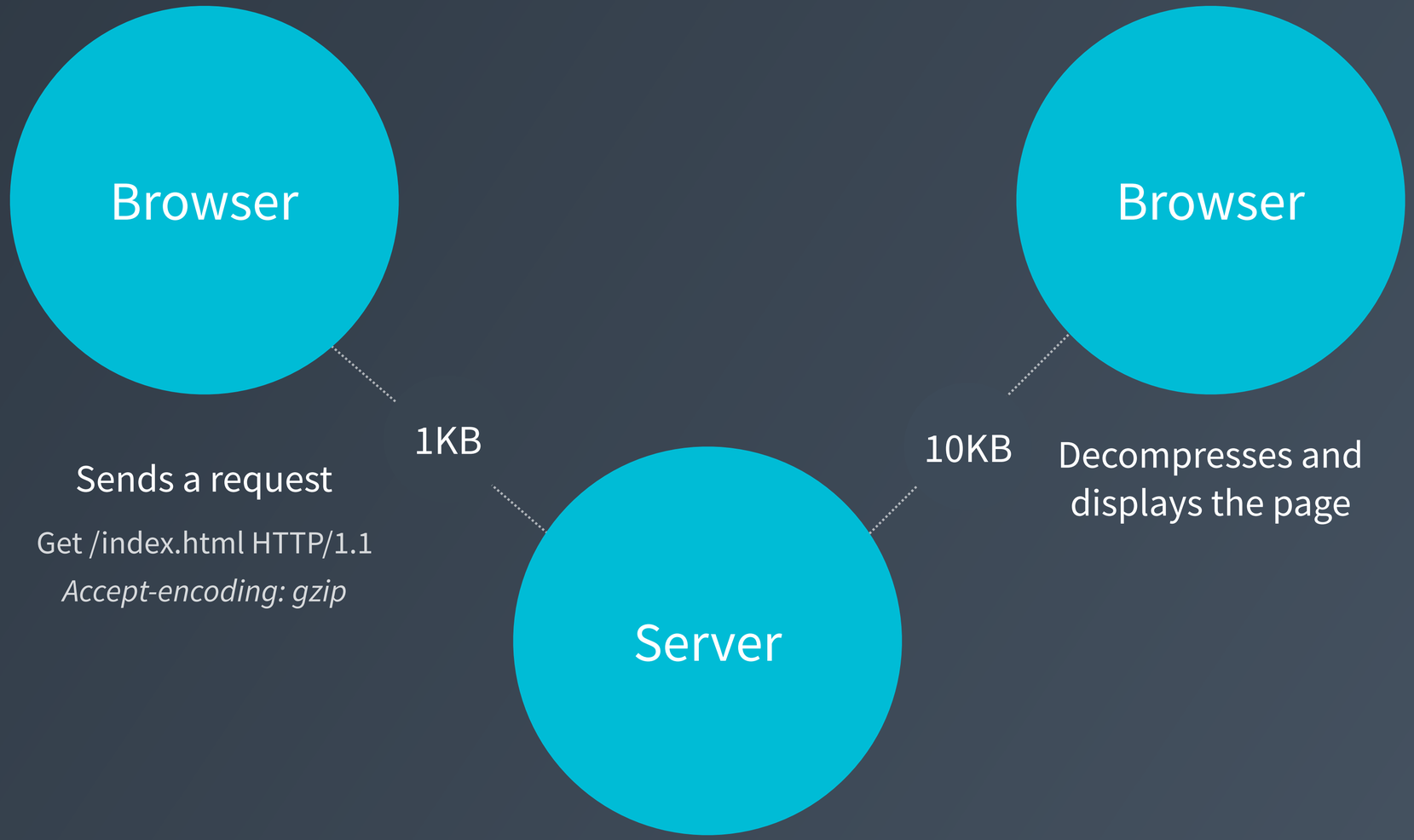
GZIP

- Lazy load
- Minify your JavaScript files
- Inline your JavaScript and CSS files
- Concatenate your files
- Load your files asynchronously
- Load only relevant scripts

Images

Files

GZIP



GO OFFLINE WITH *service workers.*

WHAT IS A

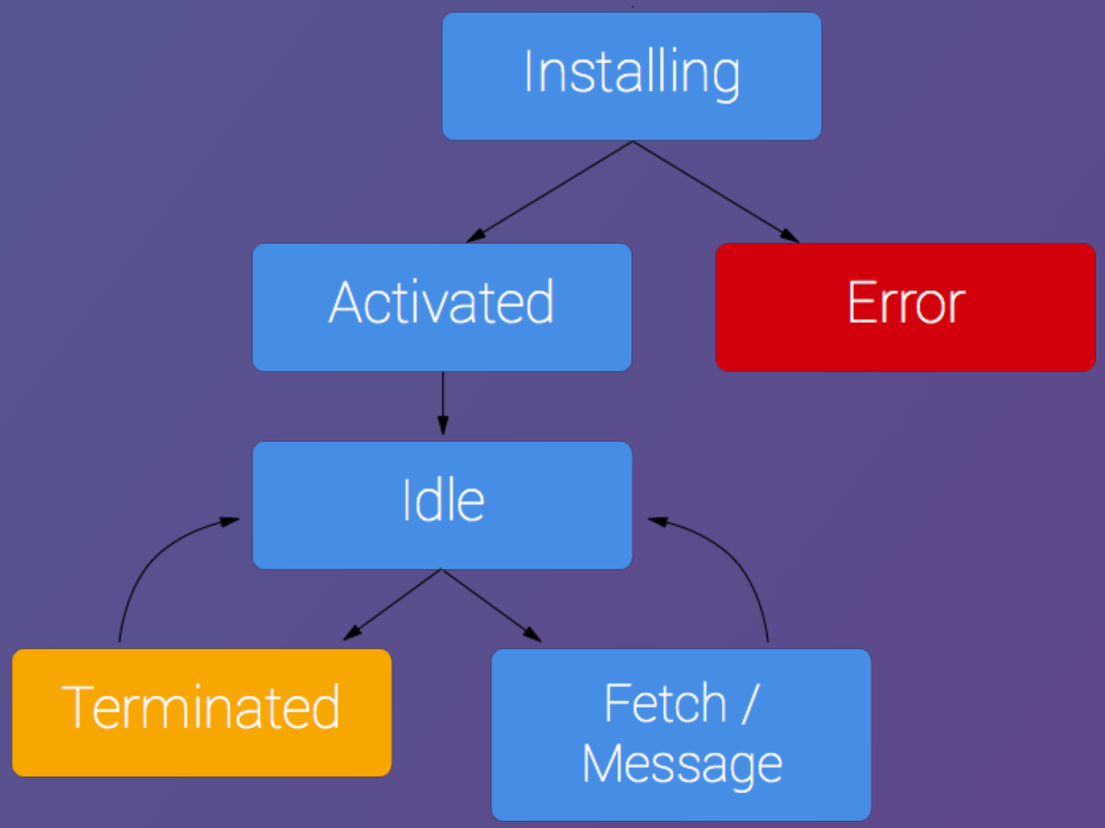
service worker?

WHAT IS A *service worker?*

- Script run by the browser in the background
- Can’t access the DOM directly
- Acts like a programmable network proxy

SERVICE WORKER

lifecycle



Register a service worker — *app.js*

```
if ('serviceWorker' in navigator) {
  navigator.serviceWorker
    .register(‘/service-worker.js')
    .then(navigator.serviceWorker.ready)
    .then(registration => {
      console.log('Service Worker has been registered with scope: ', registration.scope)
    })
    .catch(err => {
      console.error('Service Worker registration failed: ', err)
    })
}
```

Install the service worker — *service-worker.js*

```
const CACHE_NAME = 'app-cache-v1'
const CACHE_URLS = [
  '/',
  '/styles/main.css',
  '/script/main.js'
]

self.addEventListener('install', event => {
  event.waitUntil(
    caches
      .open(CACHE_NAME)
      .then(cache => {
        cache.addAll(CACHE_URLS)
      })
      .catch(err => console.error('Couldn\'t install the service worker:', err))
  )
})
```

Cache and return requests — *service-worker.js*

```
self.addEventListener('fetch', event => {
  event.respondWith(
    caches
      .open(CACHE_NAME)
      .then(cache => cache.match(event.request))
      .then(response => {
        if (response) {
          return Promise.resolve(response)
        } else {
          return fetch(event.request).then(res => {
            if (res && res.status === 200){
              caches
                .open(CACHE_NAME)
                .then(cache => {
                  cache.put(event.request, res)
                })
            }
            return res.clone()
          })
        }
      })
  )
})
```

PERFORMANCE

audit.

PageSpeed Insights

Make your web pages fast on all devices.

http://francoischalifour.com

ANALYZE

Make your web pages fast on all devices.

http://francoischalifour.com ANALYZE

Mobile | Desktop

71 / 100 Speed

! Should Fix:

Eliminate render-blocking JavaScript and CSS in above-the-fold content

▸ Show how to fix

! Consider Fixing:

Leverage browser caching

▸ Show how to fix

Enable compression

▸ Show how to fix

✓ 7 Passed Rules

▸ Show details

97 / 100 User Experience

! Consider Fixing:

Size tap targets appropriately

▸ Show how to fix

✓ 5 Passed Rules

▸ Show details

Make your web pages fast on all devices.

http://francoischalifour.com ANALYZE

Mobile Desktop

86 / 100 Suggestions Summary

! Consider Fixing:

Eliminate render-blocking JavaScript and CSS in above-the-fold content

▸ Show how to fix

Leverage browser caching

▸ Show how to fix

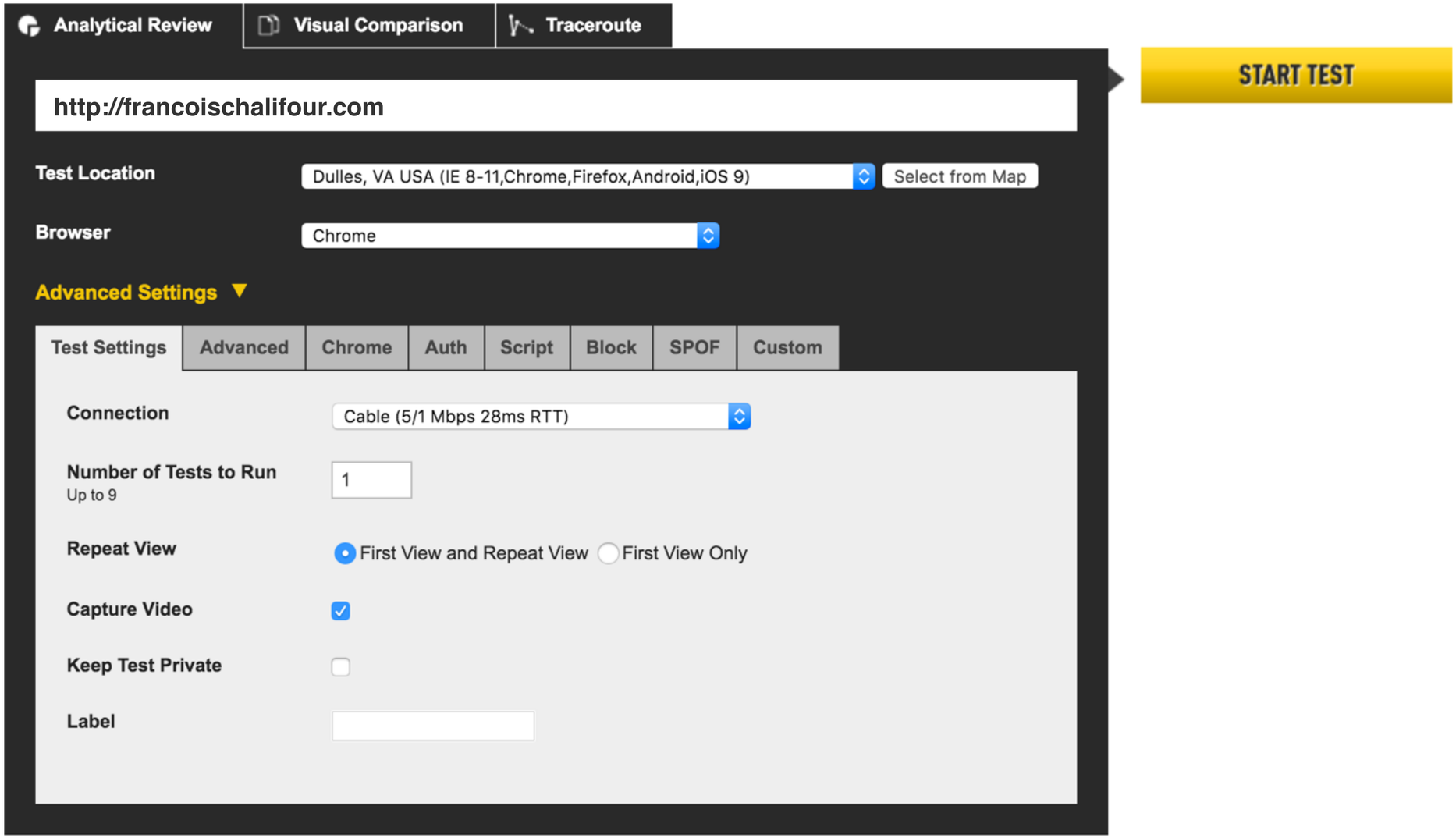
Enable compression

▸ Show how to fix

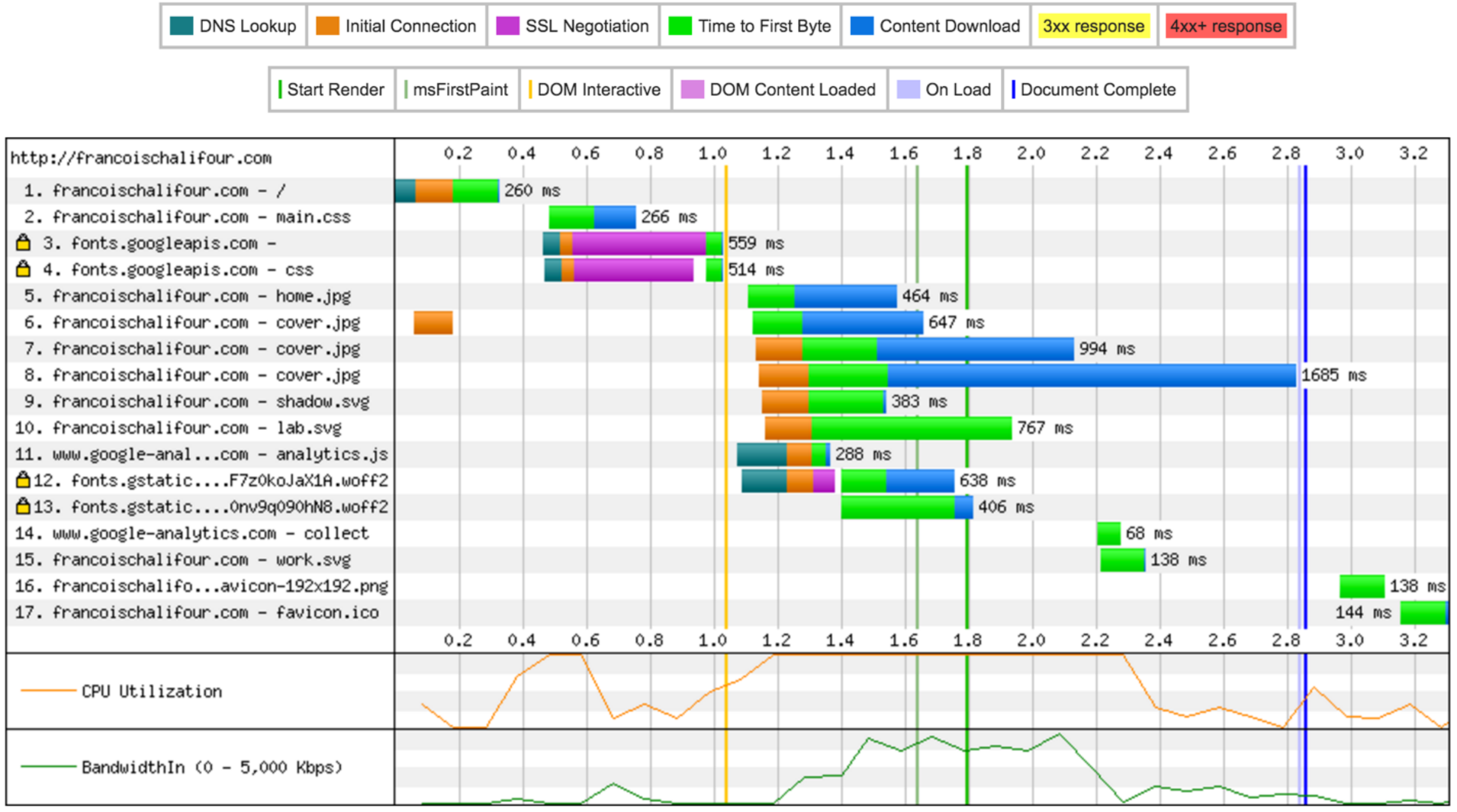
✓ 7 Passed Rules

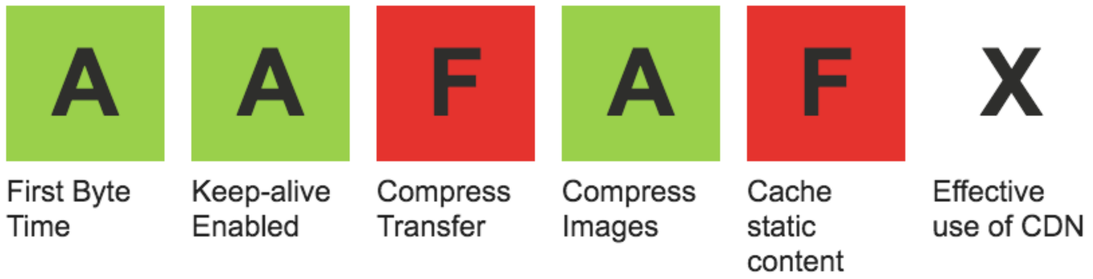
▸ Show details

WebPageTest



Waterfall View




A
A
F
A
F
X
First Byte Time
Keep-alive Enabled
Compress Transfer
Compress Images
Cache static content
Effective use of CDN

"Performance matters."

Thank you!

Sources

What Every Frontend Developer Should Know About Webpage Rendering
http://frontendbabel.info/articles/webpage-rendering-101/

Why web performance and responsive web design matter
http://gomakethings.com/why-web-performance-and-responsive-web-design-matter/

Rendering Performance
https://developers.google.com/web/fundamentals/performance/rendering/

Paul Lewis Blog
https://aerotwist.com/

Optimising for 60fps everywhere
https://engineering.gosquared.com/optimising-60fps-everywhere-in-javascript

Speeding the Console Up With the Service Worker API
https://www.clever-cloud.com/blog/engineering/2015/12/15/speeding-up-the-console-with-service-worker/

The Definitive Guide To Optimize Images For The Web
http://www.catswhocode.com/blog/the-definitive-guide-to-optimize-images-for-the-web

Ten things we know to be true
https://www.google.com/about/company/philosophy/