Design for a user

Develop for a browser

Yuliya Demchenko

yuliya.demchenko@hm.com
linkedin.com/in/demchenko

- Hyper Island, Crew 12
- Entrepreneur and Consultant
- Currently work as an Architect at H&M and FE competence community facilitator.
- First website: ~17 years ago

Rafael Youakeem

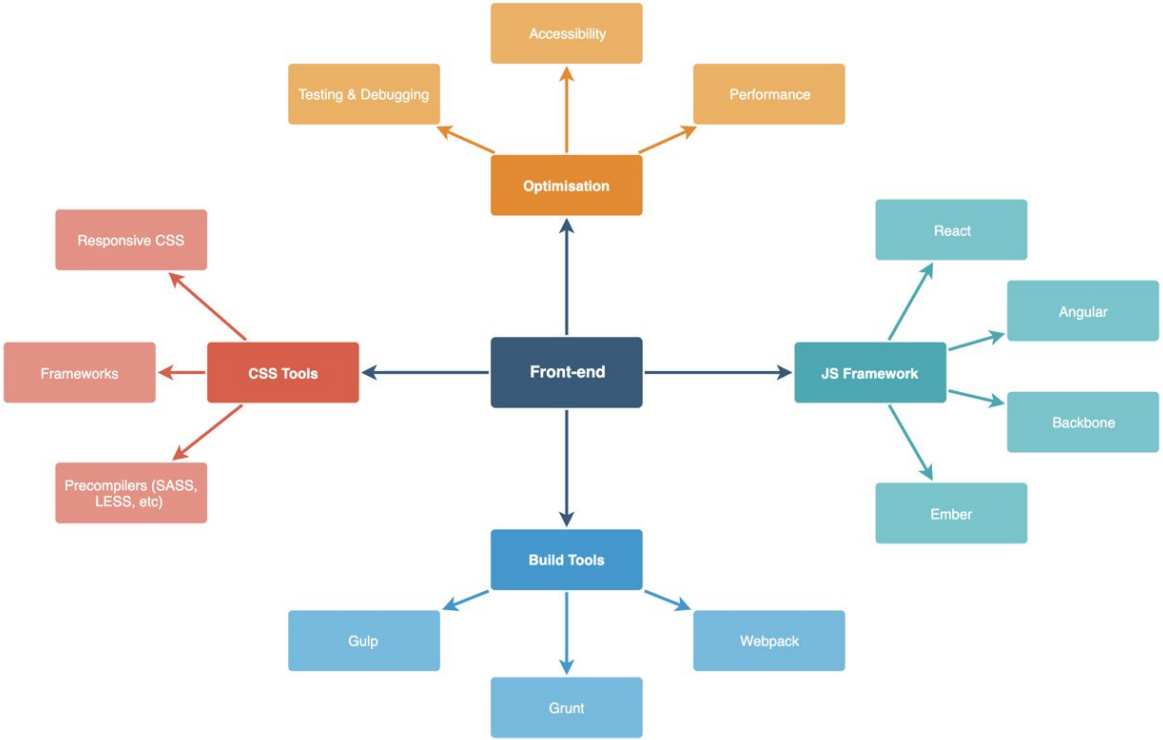
rafael.youakeem@instabox.se
linkedin.com/in/youakeem

- Technical and FE Competency Lead at Instabox
- I love learning by teaching
- Been doing FE Development for almost 12 years
- Bachelor of Law, got into development to build a MMORPG game private server

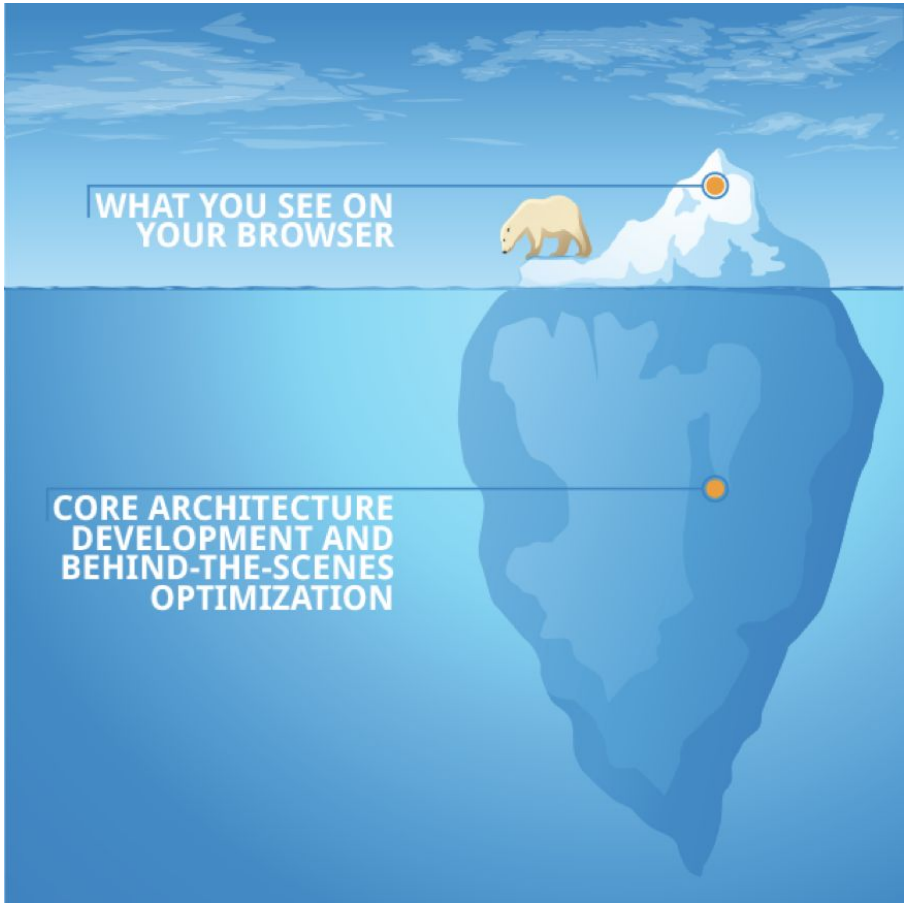
What is Front-end

HTML + CSS + JS = <3

The pillars of front-end development

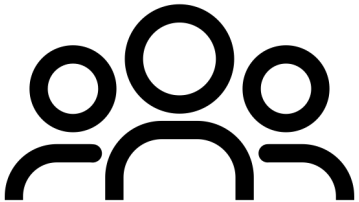


And some more...

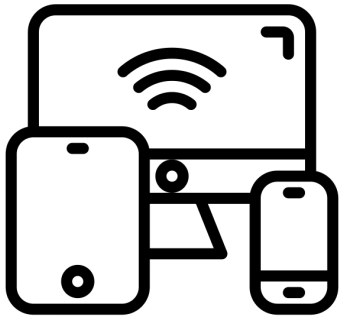

WHAT YOU SEE ON YOUR BROWSER
CORE ARCHITECTURE DEVELOPMENT AND BEHIND-THE-SCENES OPTIMIZATION

Designing for a user

Developing for a browser

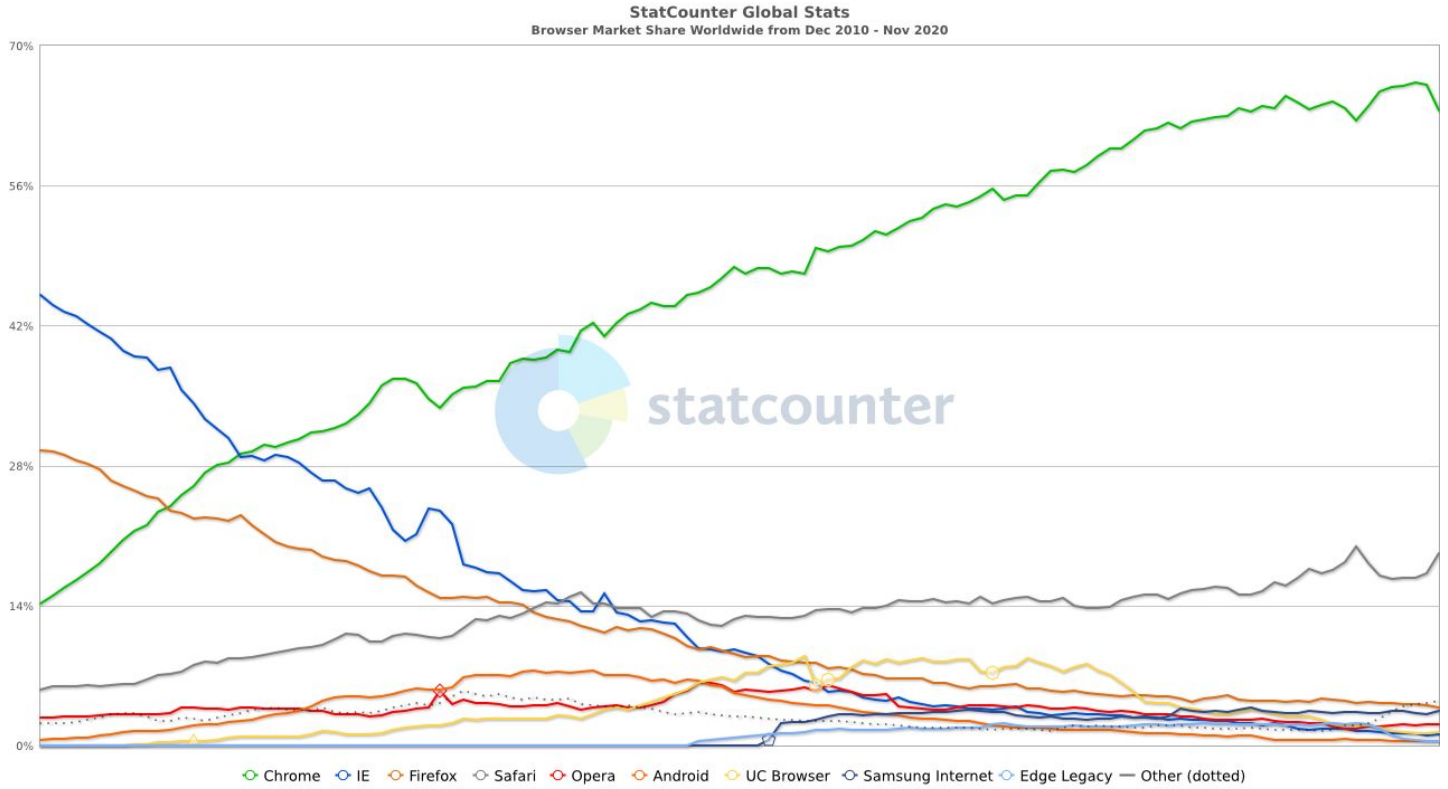


End user



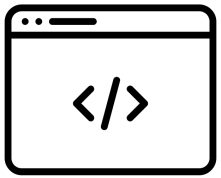
Platform

Browser Market Share 2010-2020

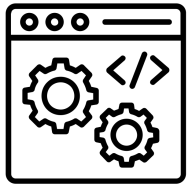


What if we build our own browser?

Browser UI

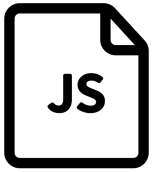


Browser Engine

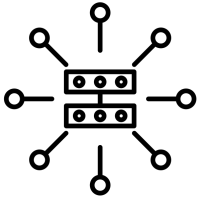


Rendering Engine

UI Back-end



Javascript interpreter



Networking

Data Storage

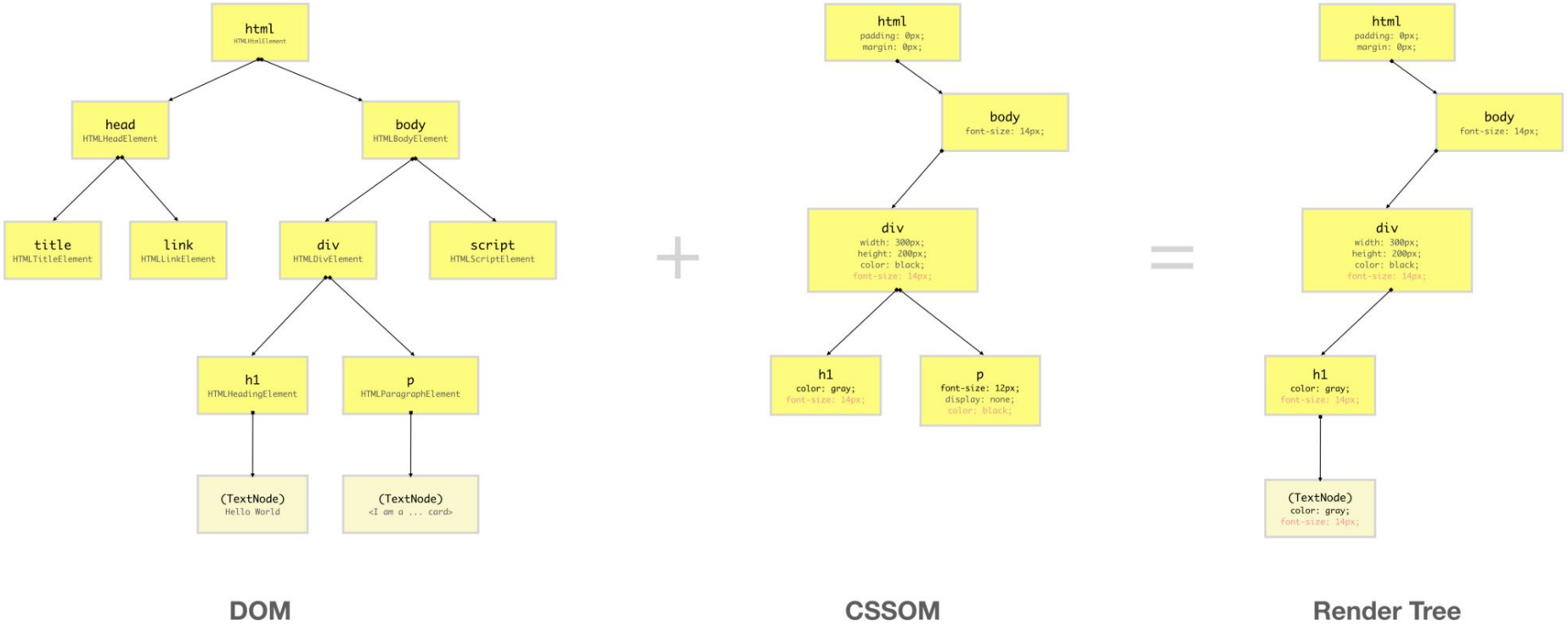
Browser Rendering Flow

The Critical Rendering Path

is the sequence of steps the browser goes through to convert the HTML, CSS, and JavaScript into pixels on the screen. Optimizing the critical render path improves render performance.The critical rendering path includes the Document Object Model (DOM), CSS Object Model (CSSOM), render tree and layout.

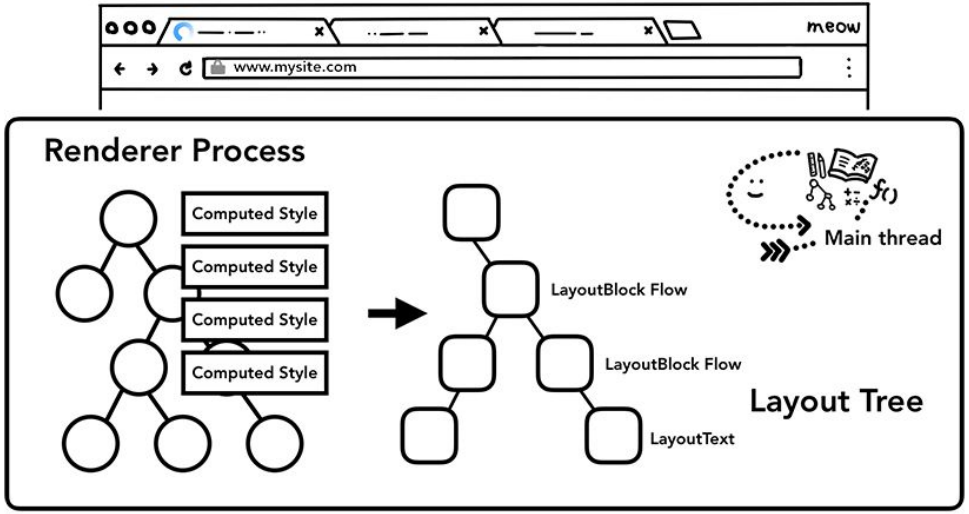
Source

Markup, HTML, DOM, CSSOM and The Render Tree



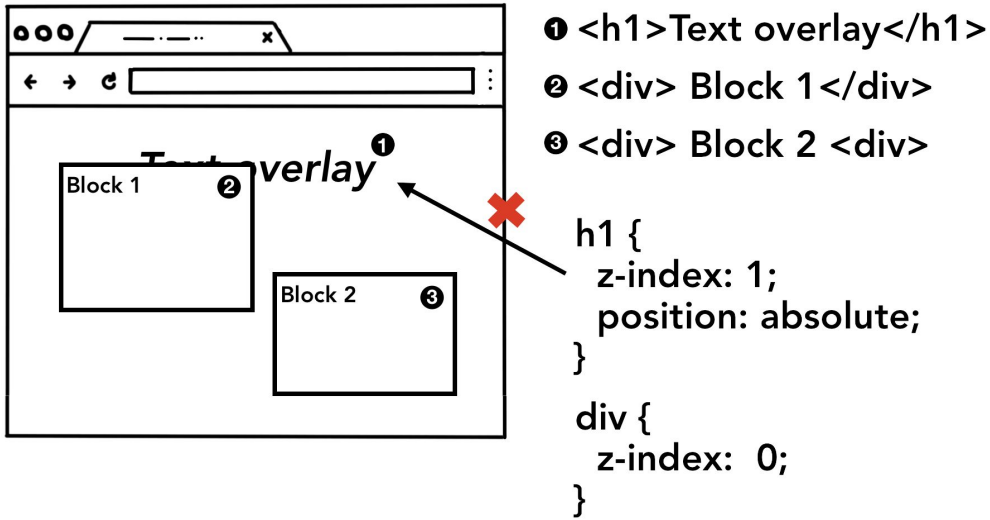
Layout

Once the render tree is built, layout becomes possible. Layout is dependent on the size of screen. The layout step determines where and how the elements are positioned on the page, determining the width and height of each element, and where they are in relation to each other.



Paint

At this paint step, the main thread walks the layout tree to create paint records. Paint record is a note of painting process like "background first, then text, then rectangle".



Compositing

Compositing is when the different layouts are being put together to build the current view.

Rendering Approach

HTML Rendering

```
<html>
<head><title>Hello World!</title></head>
<body>
  <header>
    <h1>H1 title is displyed correctly
  </header>
    <main>
      <div>
          <div>
              <div>
                  <div>
                      <div>
                          <div>
                              <div>
                                  <div>
                                      <div>
                                          <div>
                                              <div>
                                                  <div>
                                                      <div>
                                                          <div>
                                                              <div>
                                                                  <div>
                                                                      <div>
                                                                          <div>
                                                                              <p> Some content on the page that
will renderder correctly as well
</body>
</html>
```

H1 title is displyed correctly

Some content on the page that will renderder correctly as well

Browsers are more forgiving when it comes to HTML

Javascript Rendering

Default - render blocking

Async

Defer

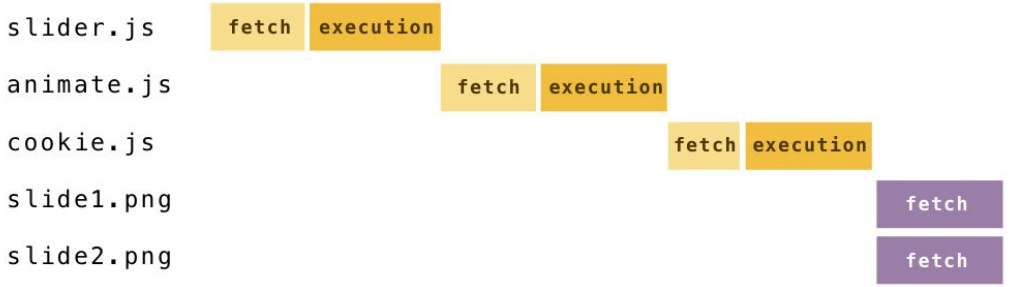
CSS is render blocking

it's required to build the render tree

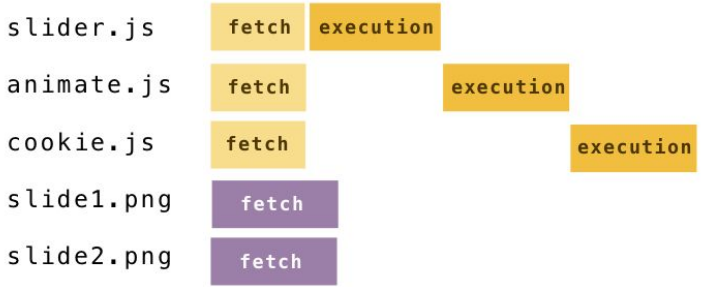
Demo!

Speculative Parsing

The HTML parser starts speculative loads for scripts, style sheets and images it finds ahead in the stream and runs the HTML tree construction algorithm speculatively.



Sequential script loading



Preloading resources with speculative parsing

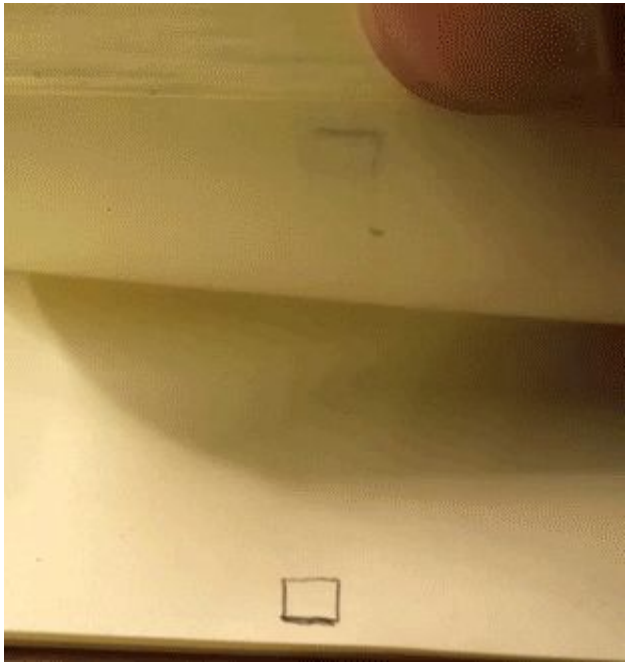
Smooth User Experience

The magic number: 16.6ms

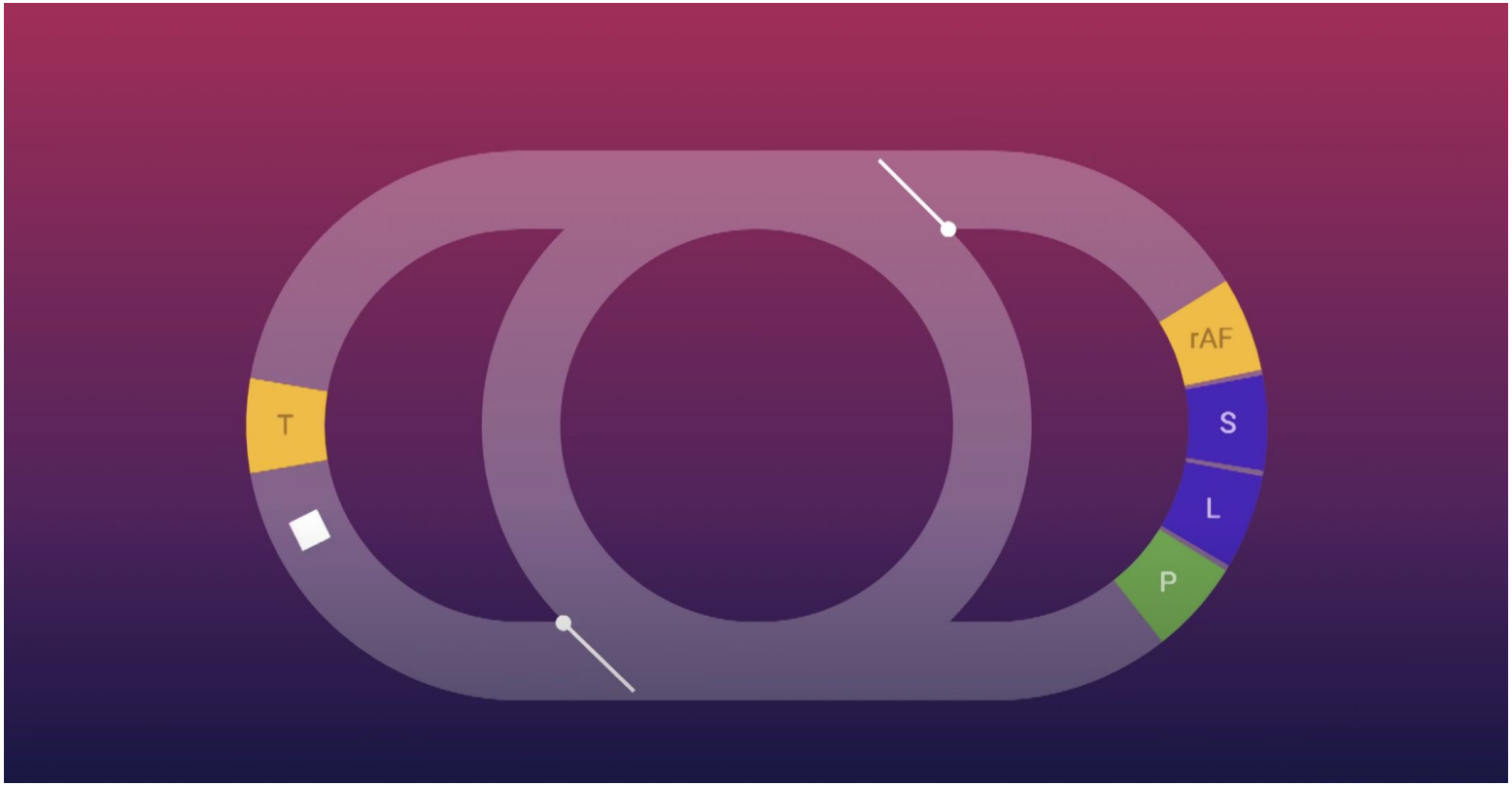
10ms

For a rate of 60 frames per second, the browser has ~16.6 milliseconds to execute scripts, recalculate styles and layout if needed, and repaint the area being updated.

Slow scripts and animating expensive CSS properties can result in jank as the browser struggles to hit a smooth frame rate.

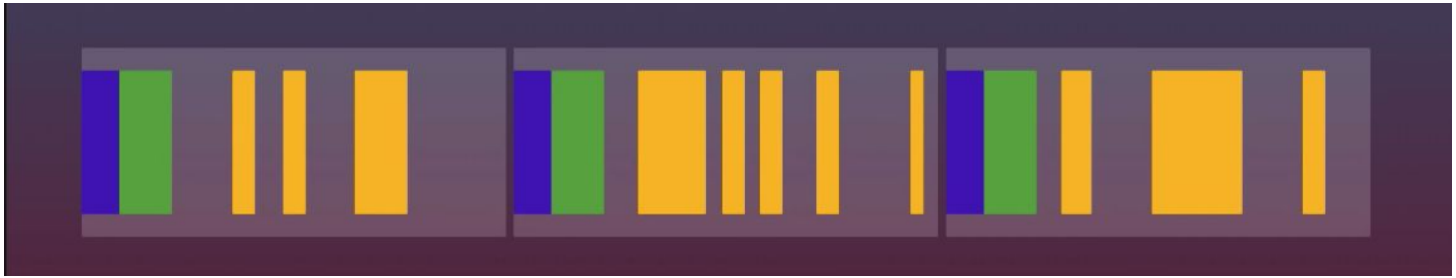


Left side. Right side.



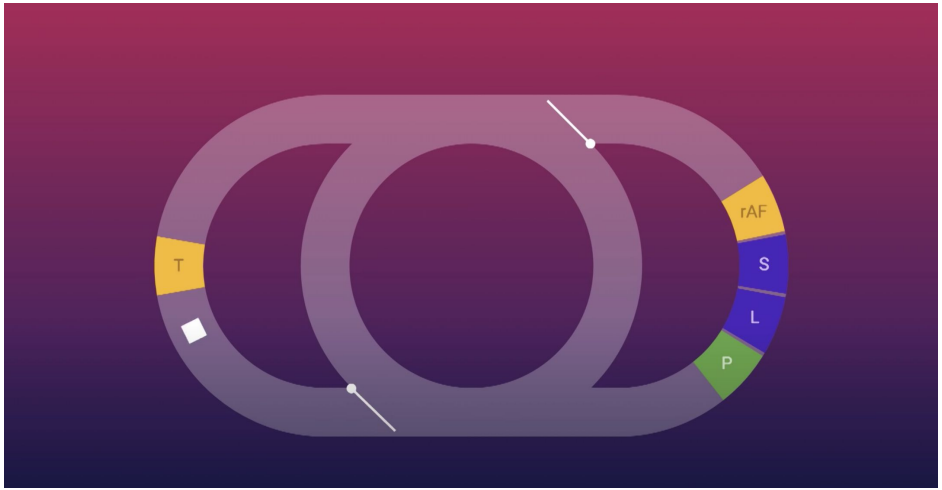
Tasks are unpredictable

Runs more than often, consuming resources



Runs anywhere within a frame (setInterval(() => {}. 1000 / 60)

requestAnimationFrame



Making Visual Changes

JS Tasks will complete before the rendering steps

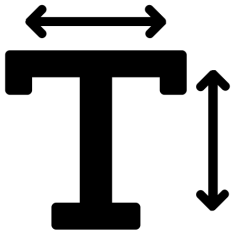
Quiz! Demo

```
//HTML
<div class="box"></div>

// CSS
.box {
    width: 300px;
    height: 300px;
}

// JS
const box = document.querySelector(".box");

box.style.backgroundColor = "red";

requestAnimationFrame(() => {
  console.log("Current background color:", box.style.backgroundColor);
});

box.style.backgroundColor = "orange";
box.style.backgroundColor = "yellow";
box.style.backgroundColor = "green";
box.style.backgroundColor = "blue";
box.style.backgroundColor = "indigo";
box.style.backgroundColor = "violet";
```

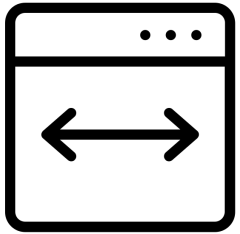
Learn from the browser - batch your changes

```
// BAD - Forces layout on each iteration
const products = querySelectorAll('.product')

products.forEach(product => {
    const productWidth = product.offsetWidth
    product.style.height = productWidth
})

// Good - Reads then writes

// Read all products widths
const productWidths = products.map(product => {
  return product.offsetWidth
})

// Update products heights
products.forEach((product, index) => {
    product.style.height = productWidths[index]
})
```

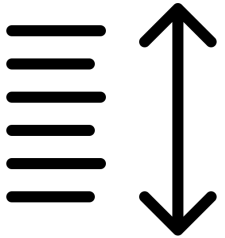
(Immediate) Triggering Layout



Font size change



Browser resize



Props like offsetHeight, clientWidth

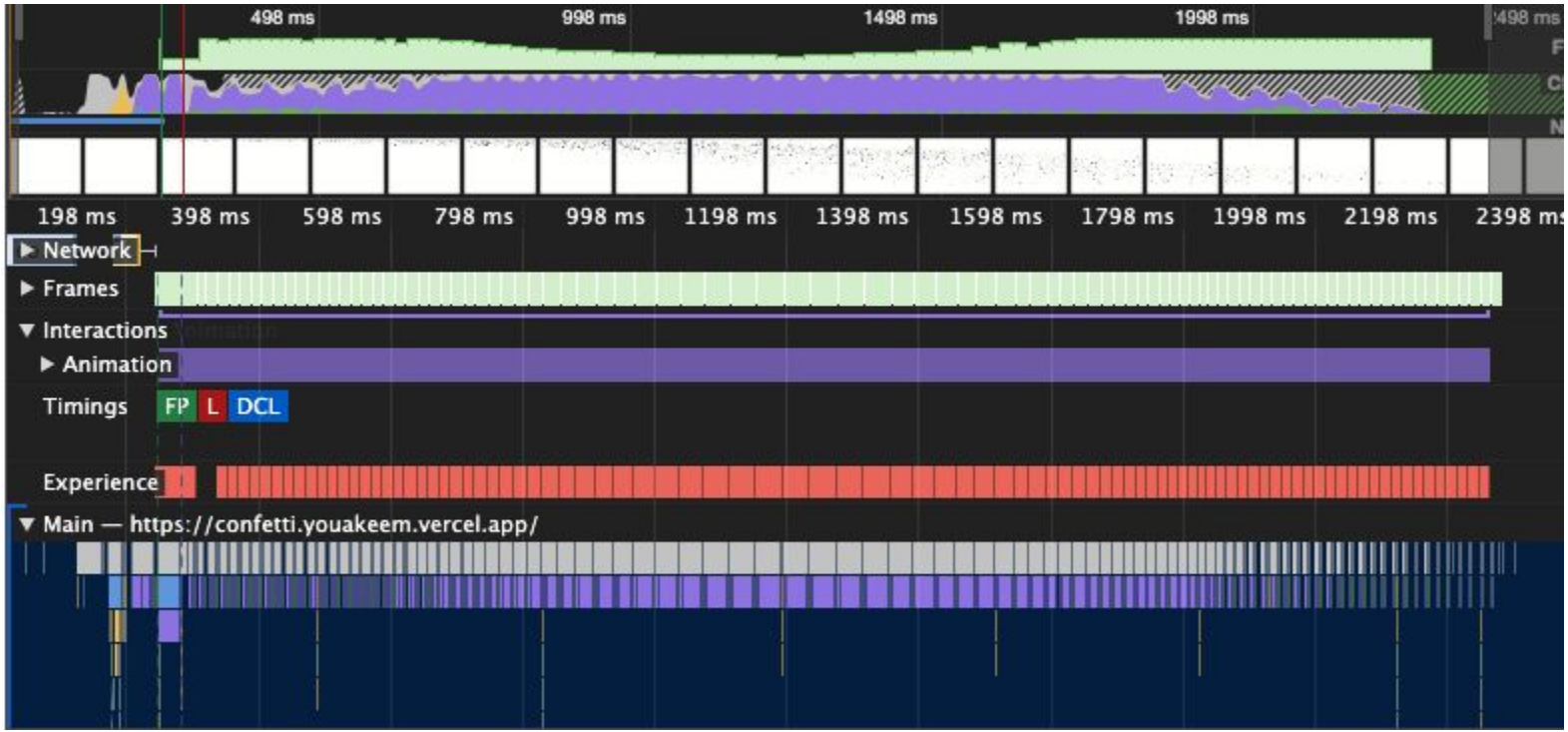
Animating the unanimateable

Demo!

Use composite only properties

- Layout and Paint are expensive and they run on the main thread blocking it.
- Composite is much cheaper and runs on its own thread keeping the main thread unblocked
- It basically means moving individual already painted layers around. They don’t affect anything else on the page.

How to profile and debug performance issues




Time to get our hands dirty!

Task: **Improve an animation (30 min)**

1. You will be split into **rooms**
2. Once in the room, agree on the **team name**
3. Go to: https://codesandbox.io/s/janky-confetti-eethd
4. **Fork** the project
5. **Make changes** in your fork
6. Join back
7. **Present your team and your project:**
    a. What was wrong in the initial animation
    b. How did you solve it

Performance takeaways

- <script> is in footer
- Inline critical CSS (could defer loading the rest of the CSS)
- We have ~10ms to do any logical operations within a frame (which provides a smooth experience and avoids janky animations)
- Use composite only properties and avoid triggering layout
- Batch style reads and writes

References and resources

- https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/
- https://medium.com/jspoint/how-the-browser-renders-a-web-page-dom-cssom-and-rendering-df10531c9969
- https://developers.google.com/web/fundamentals/performance/critical-rendering-path/constructing-the-object-model
- https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5/HTML5_Parser
- https://www.youtube.com/watch?v=SmE4OwHztCc
- https://developers.google.com/web/updates/2018/09/inside-browser-part3
- https://developers.google.com/web/fundamentals/performance/rendering/avoid-large-complex-layouts-and-layout-thrashing
- https://developers.google.com/web/fundamentals/performance/rendering/stick-to-compositor-only-properties-and-manage-layer-count
- https://csstriggers.com/
- https://gist.github.com/paulirish/5d52fb081b3570c81e3a
- https://youtu.be/cCOL7MC4Pl0?t=540

Thank You!