builder.io

Qwik

Resumable frameworks

Frontend Masters

I am Miško Hevery

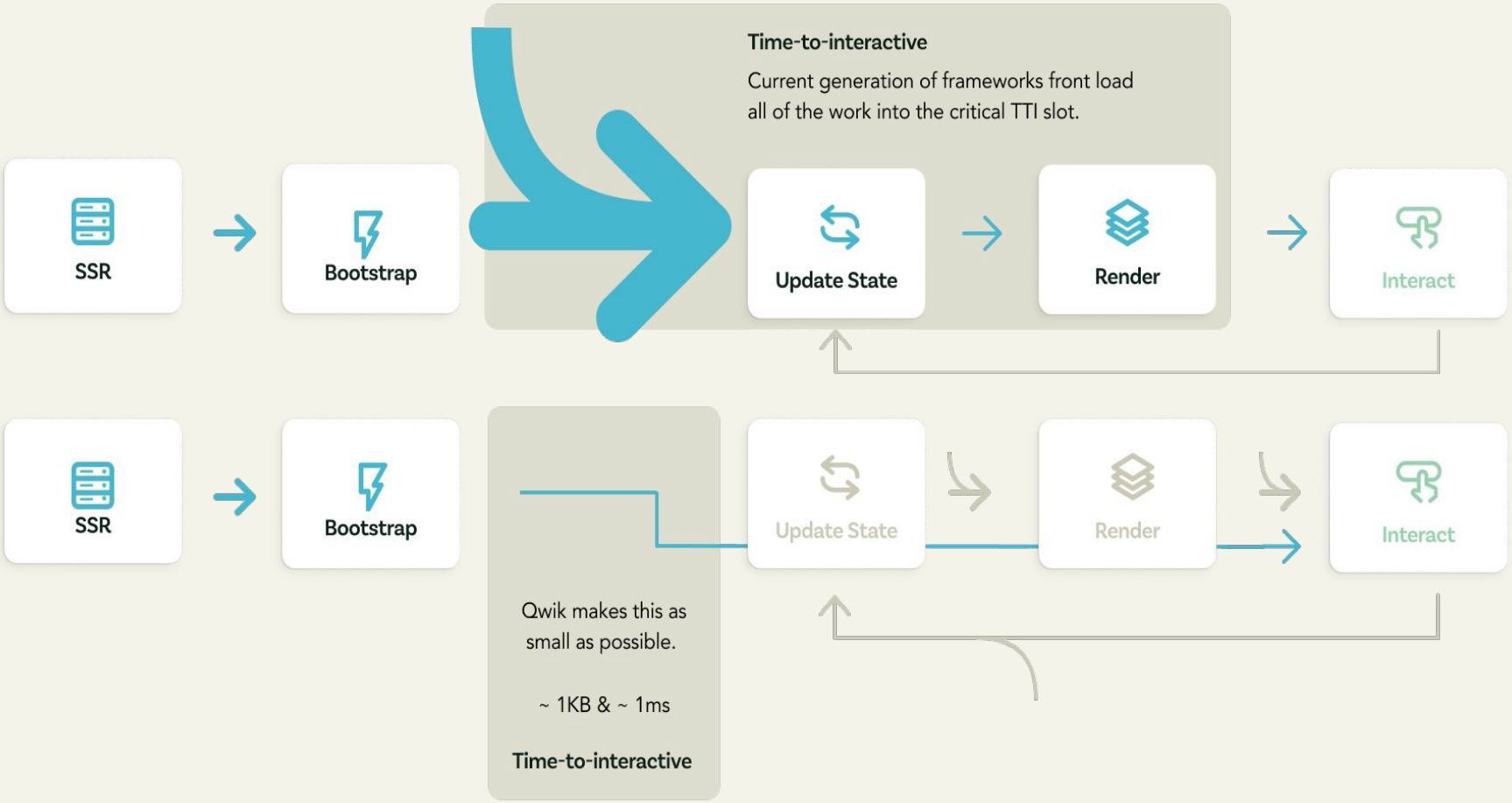
CTO at Builder.io

- Visual CMS
- Qwik
- AngularJS and Angular
- Zone.js
- Clean code talks
- cocreated Karma

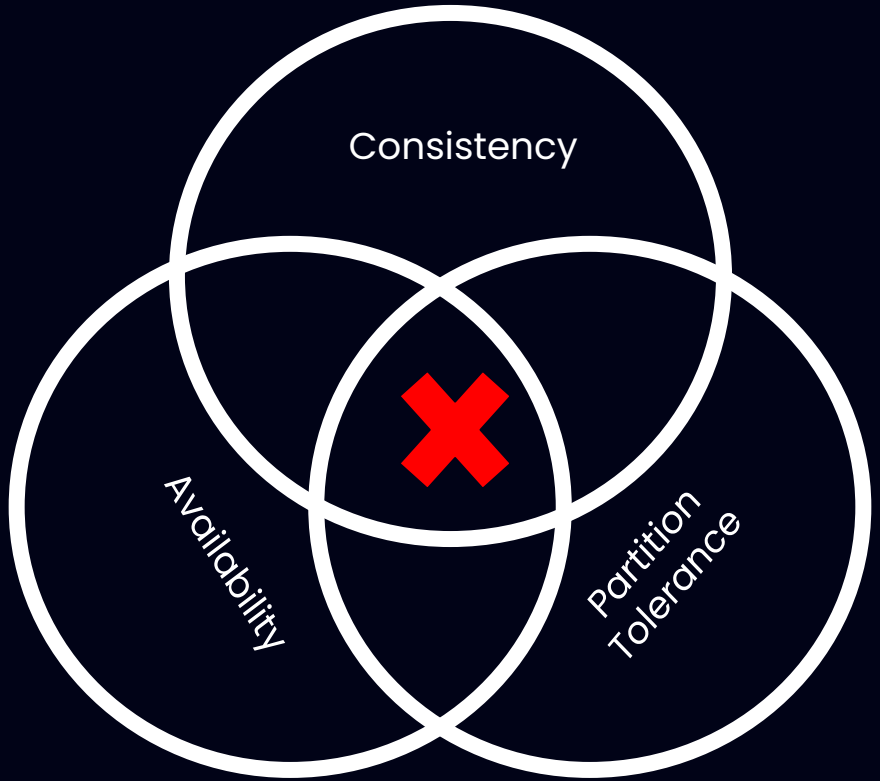
Agenda

1. Understanding Resumability
    a. Building your own resumable framework
2. Lazy loading
    a. Optimizer and $
3. Serialization of Events
4. Serialization of Data
    a. Tree shaking of data
5. Reactivity
    a. Proxies
    b. Signals
    c. Value passing capturing

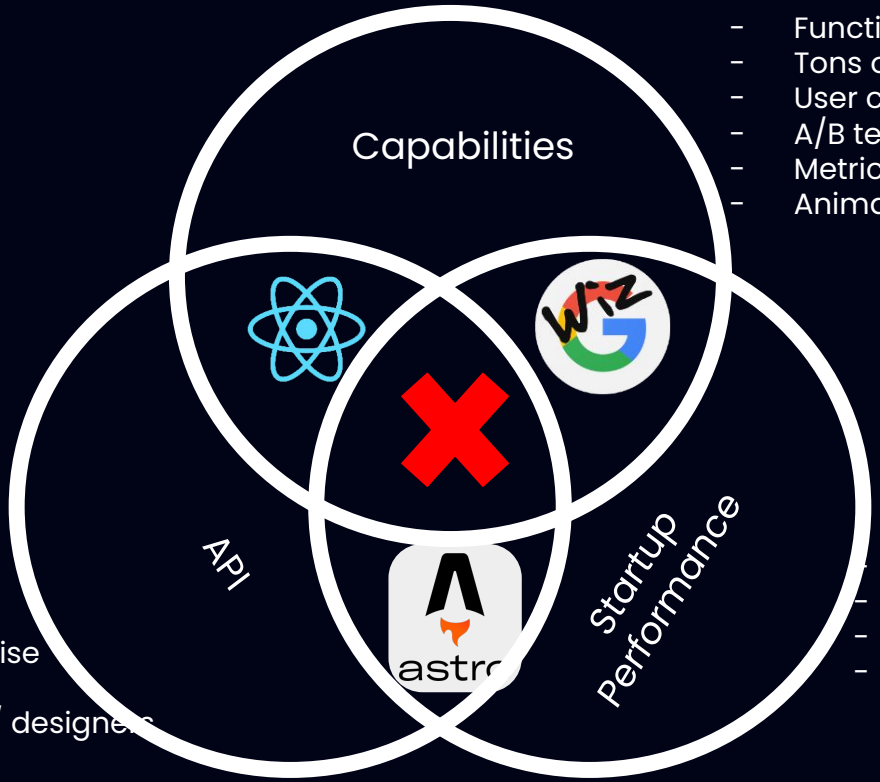
1. Serialization of Reactivity graph
2. Out of order rendering
3. Prefetching
    a. production build
    b. collecting statistics
2. React
3. Streaming
4. MicroFrontEnd Architecture
5. Personalization


Time-to-interactive
Current generation of frameworks front load all of the work into the critical TTI slot.
SSR
Bootstrap
Update State
Render
Interact
SSR
Bootstrap
Update State
Render
Interact
Qwik makes this as small as possible.
~ 1KB & ~ 1ms
Time-to-interactive

CAP theorem



frontend CAP theorem

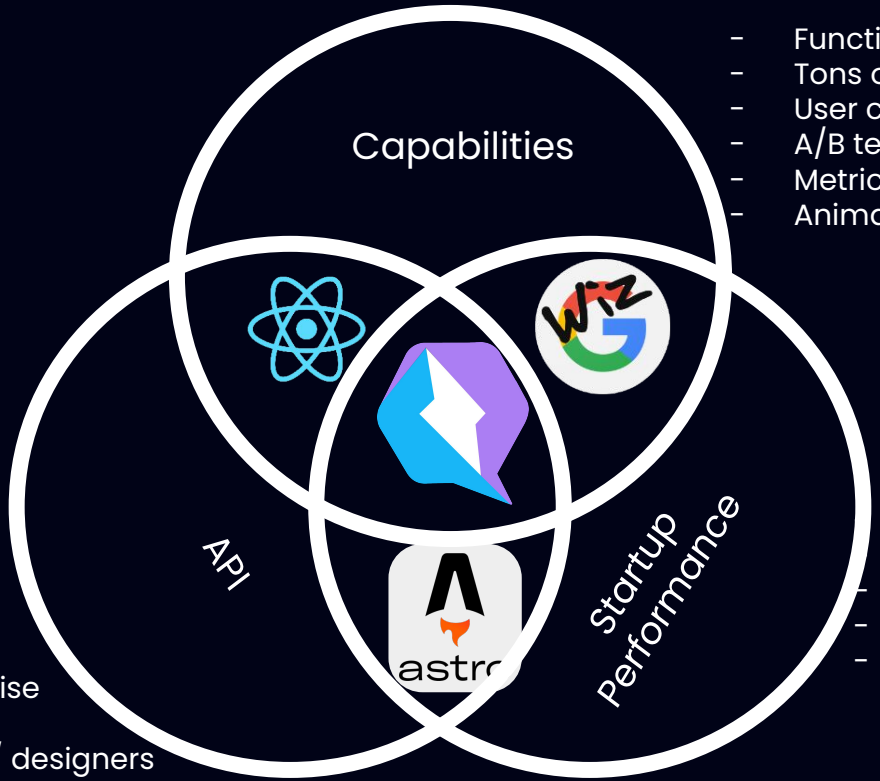


- Functionality
- Tons of interactivity
- User customizations
- A/B testing
- Metrics
- Animations

- DX
- Easy to build
- Low time to to market
- Low lock-in
- Leverage team expertise
- Fast iteration
- All-hands: Marketing / designers

- 100/100 lighthouse
- Instant time to interactive
- Fast in slow devices
- Fast in slow networks
- SEO

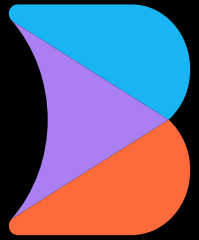
Frontend new paradigm



- Functionality
- Tons of interactivity
- User customizations
- A/B testing
- Metrics
- Animations

- DX
- Easy to build
- Microfrontends
- Low time to to market
- Low lock-in
- Leverage team expertise
- Fast iteration
- All-hands: Marketing / designers

100/100 lighthouse
Instant time to interactive
- Fast in slow devices
- Fast in slow networks
- SEO



- Drag and drop development
- A/B testing
- User personalization
- Using your stack (React / Vue…)
- Edge optimized

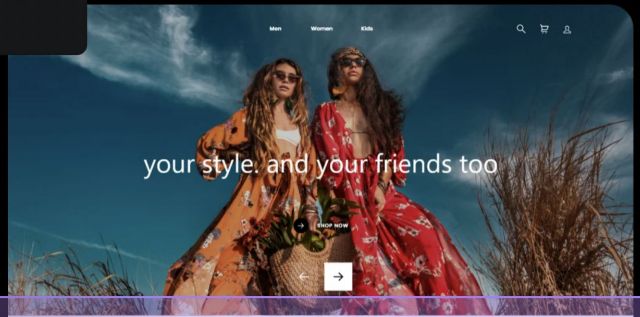
- First-party code
- Constant scalability
- HTML-first, zero JS
- Highly interactive
- MPA & SPA

- Third-party code
- Analytics and third party code
- Execute in web worker

- Write code once generate idoimatic code which runs on any framework
- Transpiler

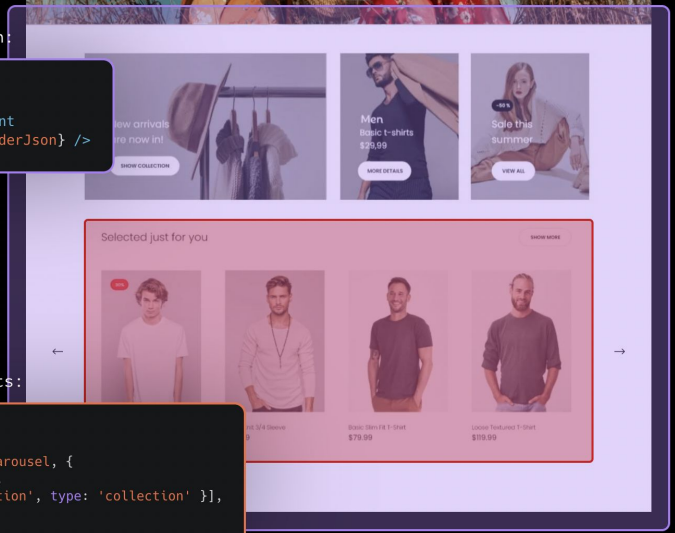
Your Site/App Code:

```
export function HomePage() {
  return <>...</>
}
```



Builder Section:

```
<BuilderComponent
  content={builderJson} />
```



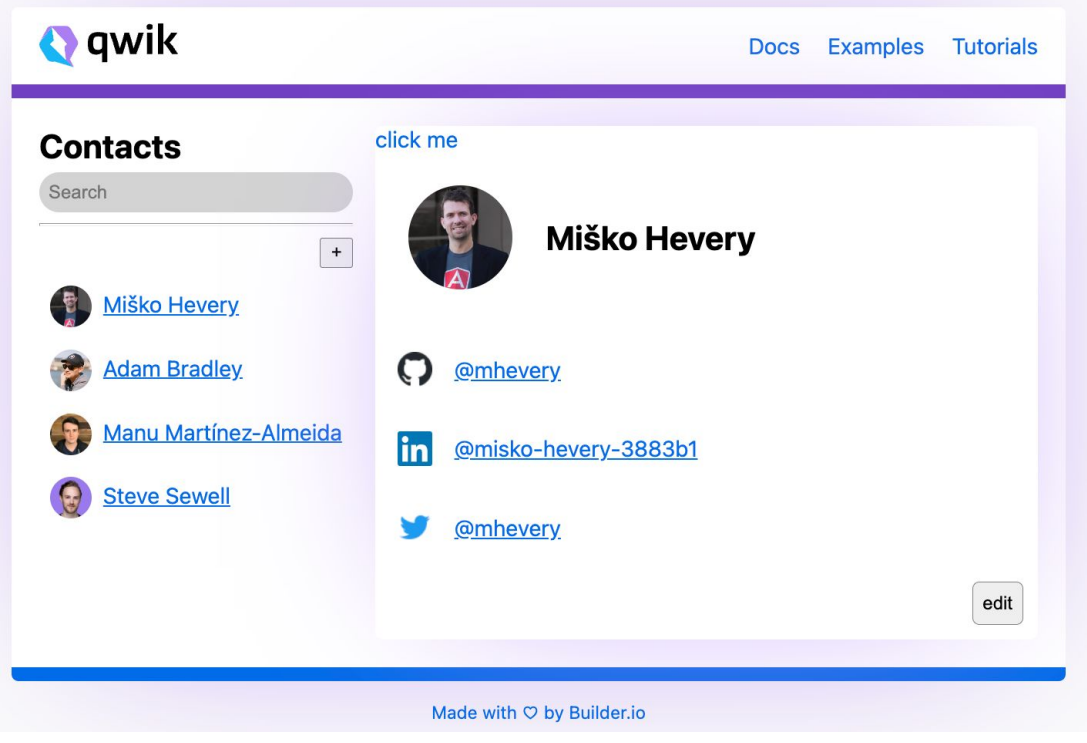
Register your components:

```
registerComponent(ProductCarousel, {
  name: 'Product Carousel',
  inputs: [{ name: 'collection', type: 'collection' }],
})
```

http://localhost:5173/

Agenda

1. Installing Qwik
2. Qwik-City File Layout Overview
3. Building a Contacts App
4. Displaying Data
    a. Components
    b. Styling
    c. Behavior
5. Deleting Data
6. Editing Data
7. Creating Data
    a. Form Validation
8. Login



1. JavaScript is THE problem
2. It is not your fault
3. There is a solution

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS

Expand view

■ First Contentful Paint 2.1 s	● Time to Interactive 2.2 s
● Speed Index 2.1 s	● Total Blocking Time 10 ms
■ Largest Contentful Paint 2.6 s	● Cumulative Layout Shift 0

Captured at Aug 15, 2022, 3:50 PM PDT
Initial page load
Emulated Moto G4 with Lighthouse 9.6.5
Slow 4G throttling
Single page load
Using HeadlessChromium 102.0.5005.115 with lr

But... is this really a problem?

Every 100ms faster → 1% more conversions
For Mobify, every 100ms decrease in homepage load speed worked out to a 1.11% increase in session-based conversion, yielding an average annual revenue increase of nearly $380,000.

20% faster → 10% more conversions
Retailer Furniture Village audited their site speed and developed a plan to address the problems they found, leading to a 20% reduction in page load time and a 10% increase in conversion rate.

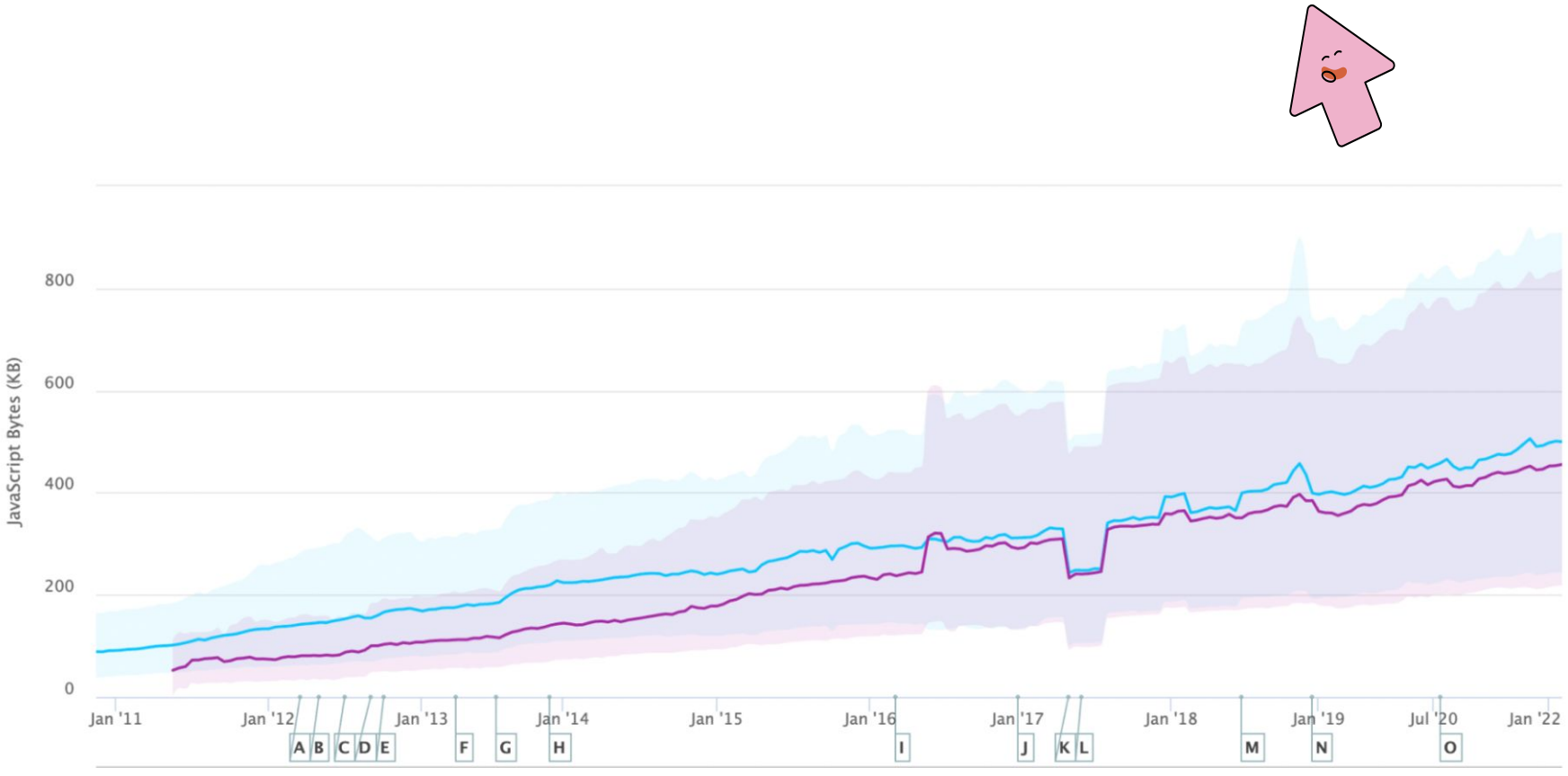
850ms faster → 7% more conversions
COOK reduced average page load time by 850 milliseconds which increased conversions by 7%, decreased bounce rates by 7%, and increased pages per session by 10%.

50% faster → 12% more sales
When AutoAnything reduced page load time by half, they saw a boost of 12% to 13% in sales.

40% faster → 15% more sign-ups
Pinterest reduced perceived wait times by 40% and this increased search engine traffic and sign-ups by 15%.

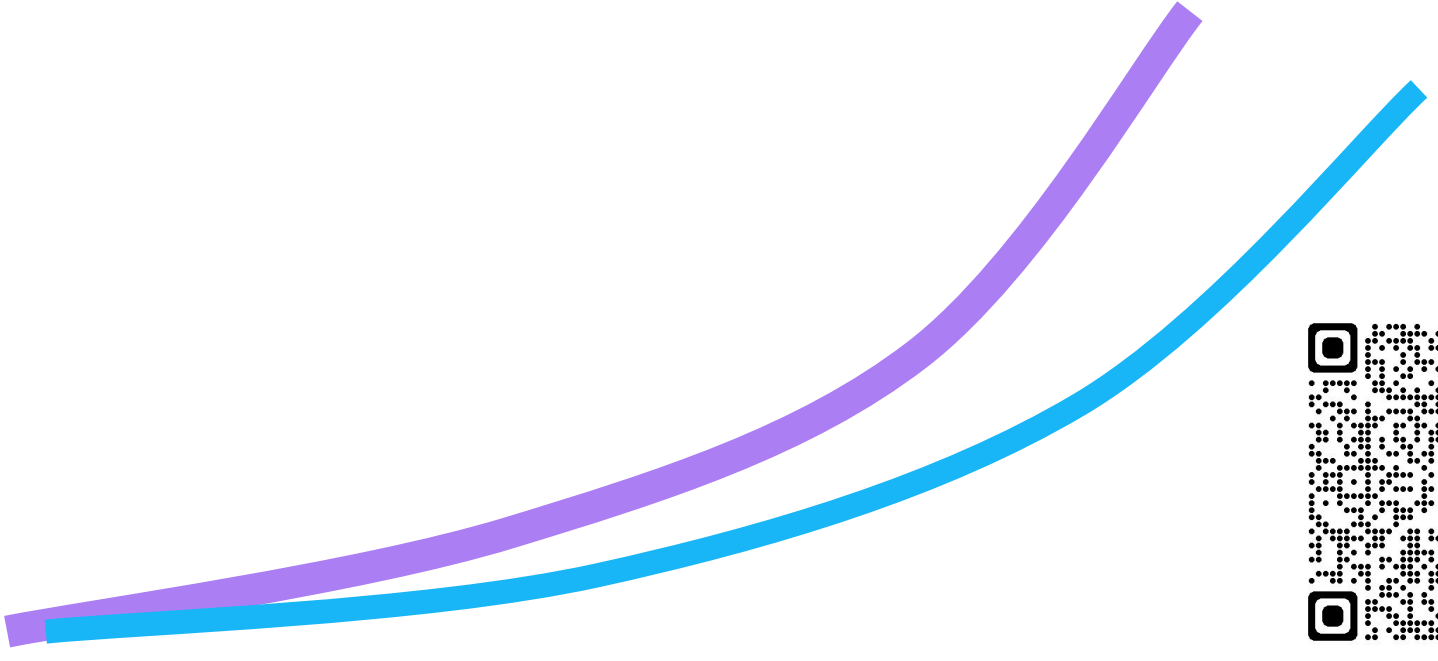
1 seconds slowness → 10% less users
The BBC found they lost an additional 10% of users for every additional second their site took to load.

https://web.dev/tags/case-study/


JavaScript Bytes (KB)
800
600
400
200
0
Jan '11
Jan '12
Jan '13
Jan '14
Jan '15
Jan '16
Jan '17
Jan '18
Jan '19
Jul '20
Jan '22
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O

Existing correlation between:

- Sites becoming more interactive
- Amount of JS

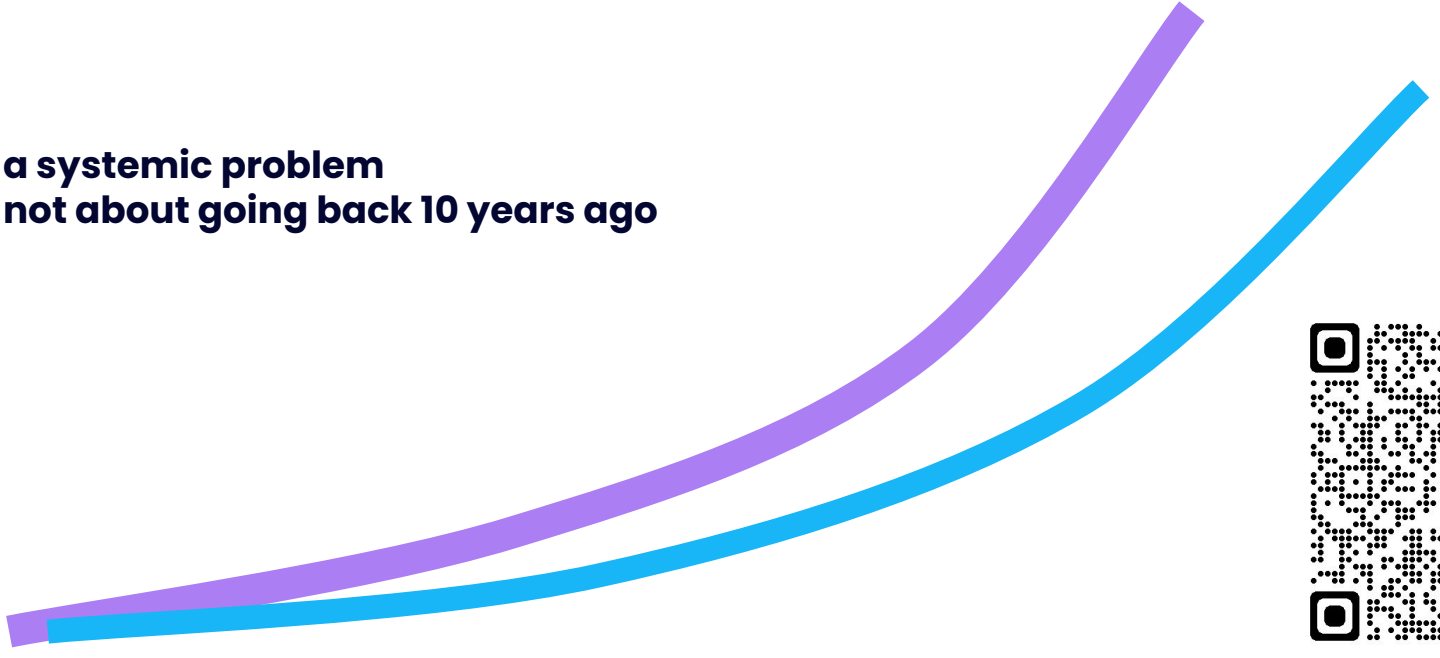


Existing correlation between:

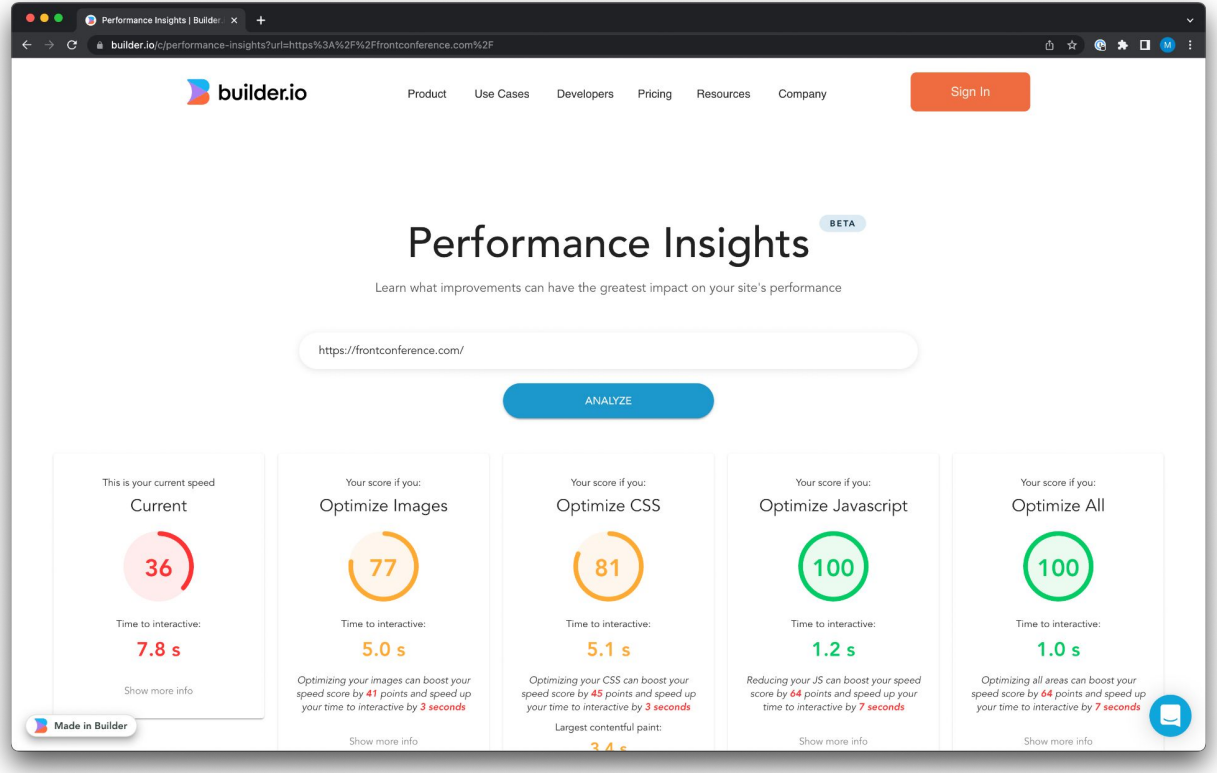
- Sites becoming more interactive
- Amount of JS

And...

- It's a systemic problem
- It's not about going back 10 years ago



https://www.builder.io/c/performance-insights

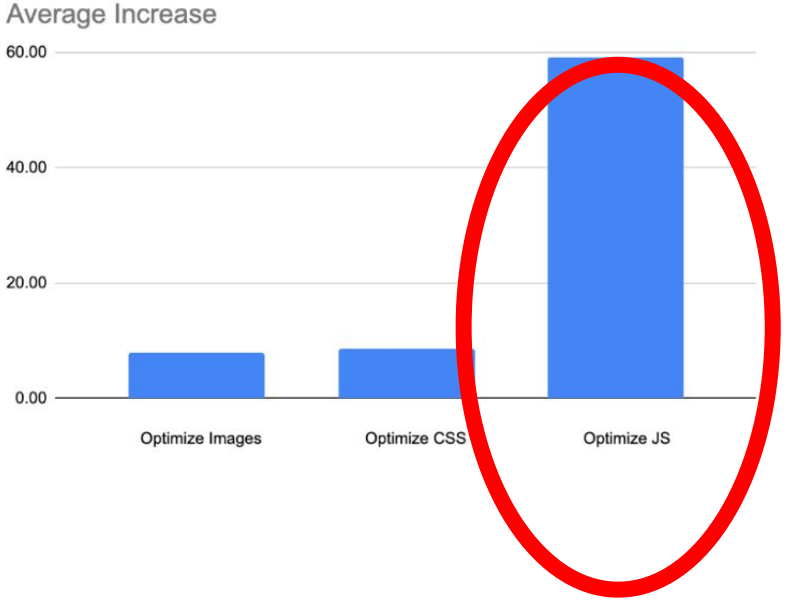


Site	Current Score	Optimize Images	Optimize CSS	Optimize JS	Optimize all
amazon.com	50	55	60	95	99
walmart.com	37	38	37	96	98
apple.com	48	80	94	81	97
target.com	28	44	33	96	85
homedepot.com	15	24	23	85	94
bestbuy.com	16	16	17	85	100
wayfair.com	22	33	22	100	100
carvana.com	x	x	x	x	x
costco.com	6	31	18	74	95
chewy.com	19	19	19	64	82
lowes.com	33	33	33	93	99
macys.com	19	19	19	91	96
kroger.com	x	x	x	x	x
samsclub.com	28	38	41	85	94
kohls.com	x	x	x	x	x
shein.com	8	9	12	76	93
gap.com	10	18	19	89	93
nike.com	21	21	22	68	97
grainger.com	21	23	25	88	97
qvc.com	x	x	x	x	x
nordstrom.com	x	x	x	x	x
bedbathandbeyond.com	x	x	x	x	x
overstock.com	32	32	32	80	90
staples.com	71	71	71	91	100
safeway.com	16	24	21	59	92
officedepot.com	6	16	17	72	91
vroom.com	20	56	59	99	100
dickssportinggoods.com	1	5	11	71	96
sephora.com	18	18	18	99	100
stitchfix.com	38	38	38	96	100
walgreens.com	4	23	24	86	95
potterybarn.com	x	x	x	x	x
jcpenney.com	16	16	16	89	96
ulta.com	17	17	17	94	95
dell.com	30	30	30	96	96
victoriassecret.com	13	25	33	83	89
bathandbodyworks.com	3	15	3	40	75
apmex.com	81	81	87	93	97
lululemon.com	10	10	10	72	100
ikea.com	71	85	82	98	99
nordstromrack.com	x	x	x	x	x
ae.com	4	25	32	34	70
build.com	8	20	25	63	98
adidas.com	43	43	43	88	98
newegg.com	21	53	63	98	98
sweetwater.com	26	26	36	87	100
zara.com	27	27	30	98	99
menards.com	3	24	17	67	94
sears.com	x	x	x	x	x
zulily.com	18	18	18	87	93
Average	**23.85**	**31.68**	**32.37**	**83.07**	**94.63**

Base ↓ | Optimize Images ↓ | Optimize CSS ↓ | Optimize JS ↓ | Optimize All ↓

Average Lighthouse Score	**23.85**	**31.68**	**32.37**	**83.07**	**94.63**

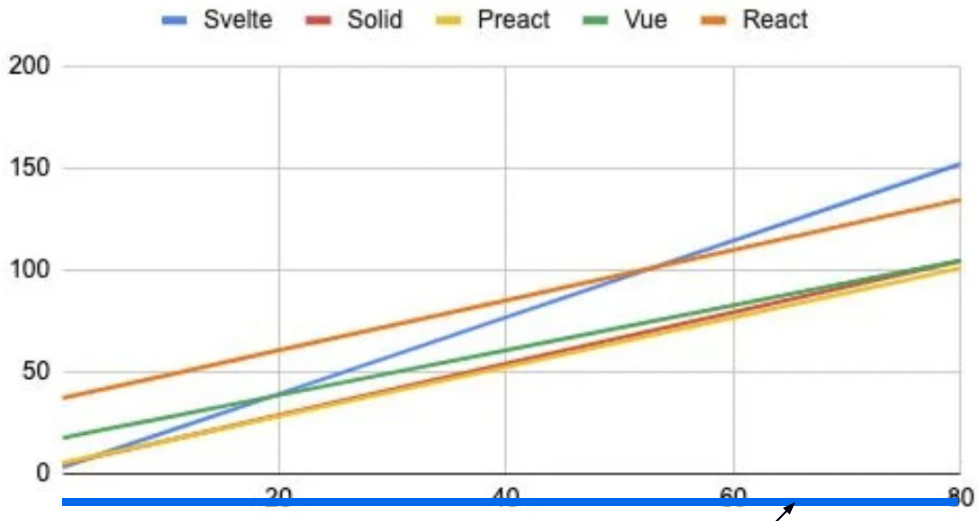
Average TTI	**26.43**	**19.96**	**20.33**	**2.42**	**1.82**

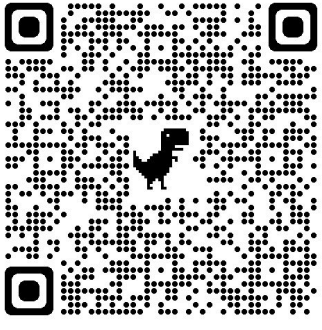


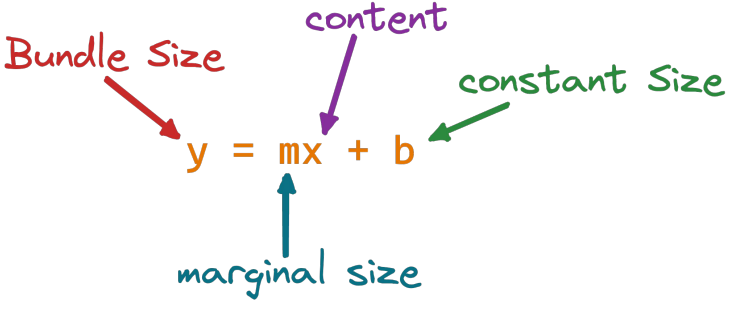
The bottleneck is JS

Big O() notation for frameworks

	1	5	10	20	40	80
Svelte	3.73kb	11.25kb	20.65kb	39.45kb	77.05kb	152.25kb
Solid	5.12kb	10.16kb	16.46kb	29.06kb	54.26kb	104.66kb
Preact	5.60kb	10.44kb	16.49kb	28.59kb	52.79kb	101.19kb
Vue	17.99kb	22.39kb	27.89kb	38.89kb	60.89kb	104.89kb
React	37.45kb	42.37kb	48.52kb	60.82kb	85.42kb	134.62kb

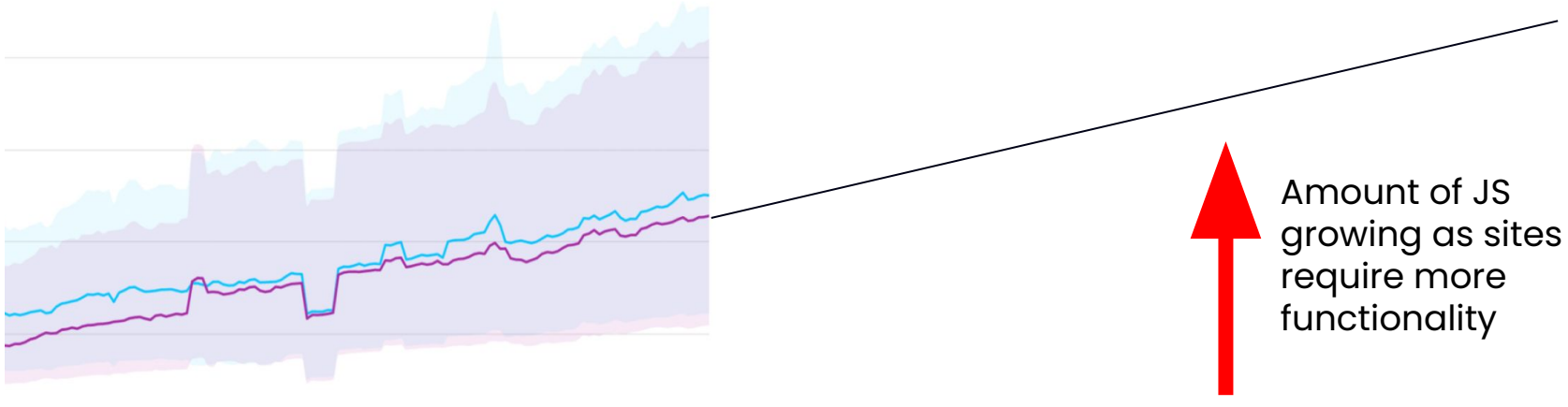




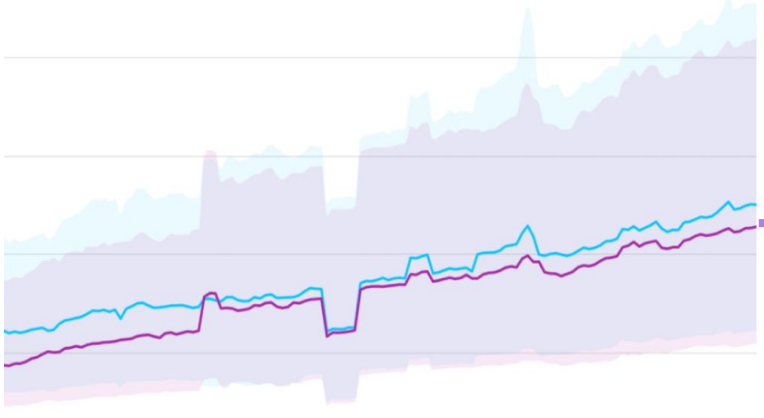


Resumable

Current trend is linear because current frameworks are O(n)



What if frameworks were O(1)?



Correlation broken
between functionality
and amount of JS

Past

Rendering and interactivity separated

Rendering done at the server by:

Interactivity:

Very performant

Slow development, error prone, duplicated logic…

DX ⬆️⬆️

Current

Render and interactivity united into a single framework.

App runs twice in server and client using the same tech:

Unified model, fast development, scales in complexity

Uses hydration, to enable interactivity. App runs twice.

PERF ⬆️⬆️

Future

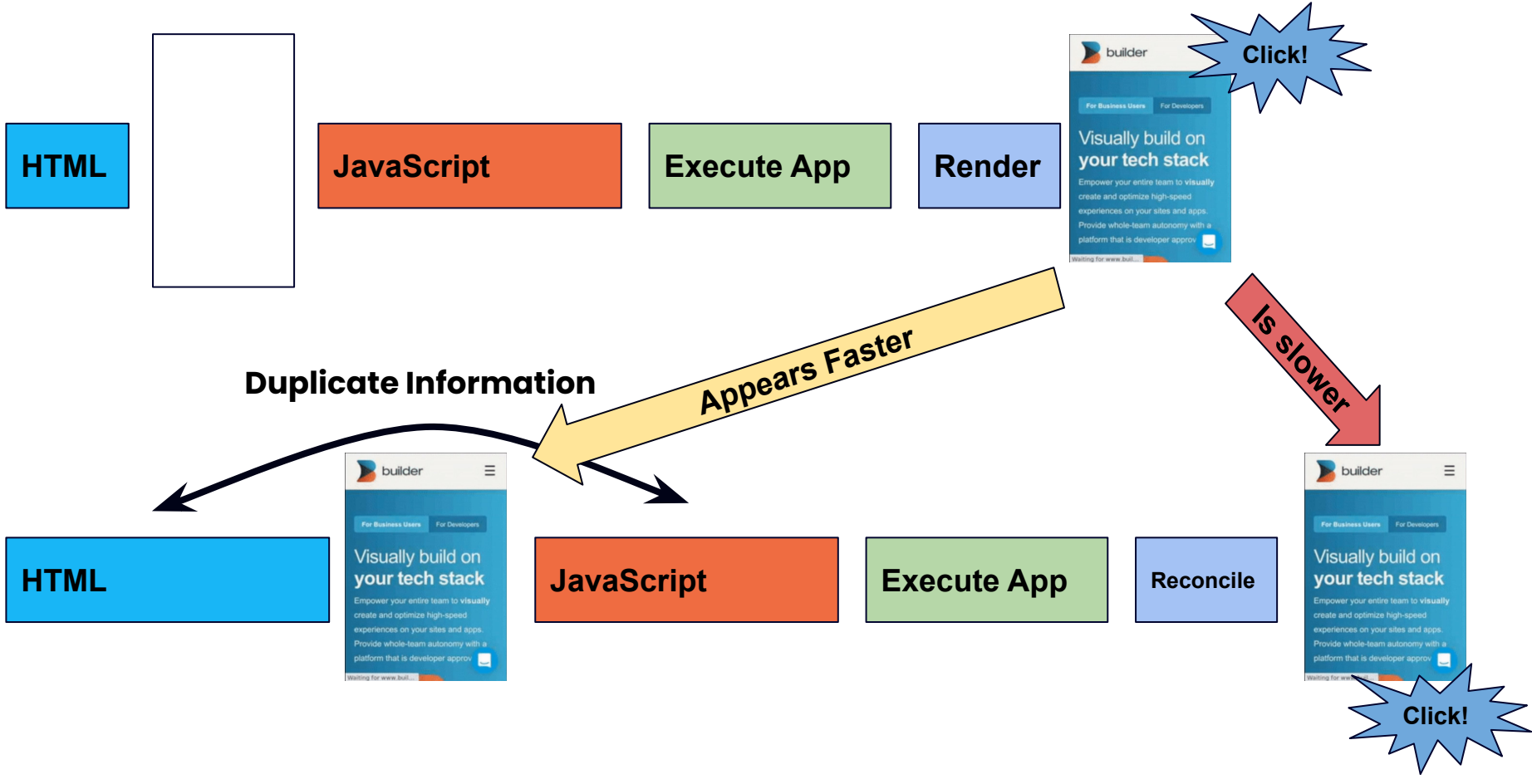
Render and interactivity united into a single framework.

App runs once, client only executes minimal bites of interactivity.

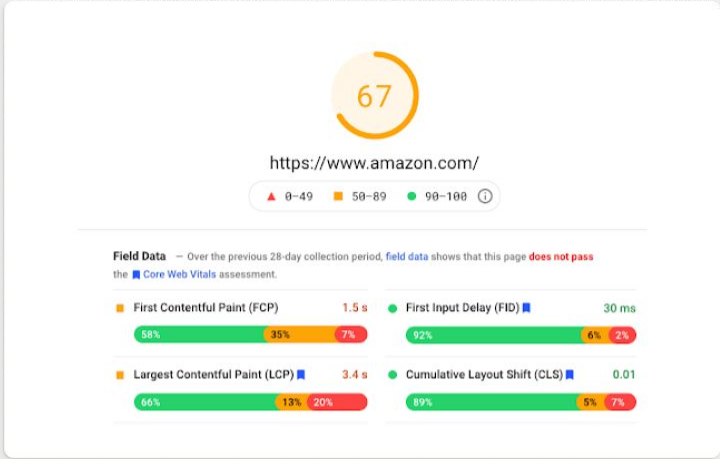
Unified model, fast development, scales in complexity

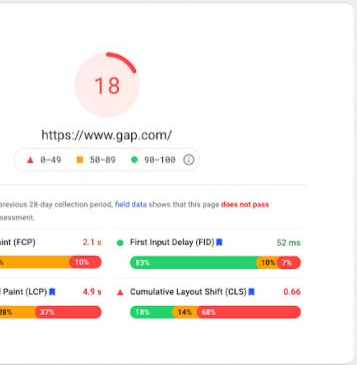
Uses resumability, performance scales well. O(1)

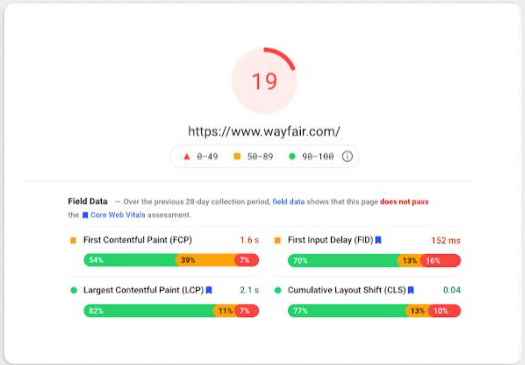
Hydration is a workaround

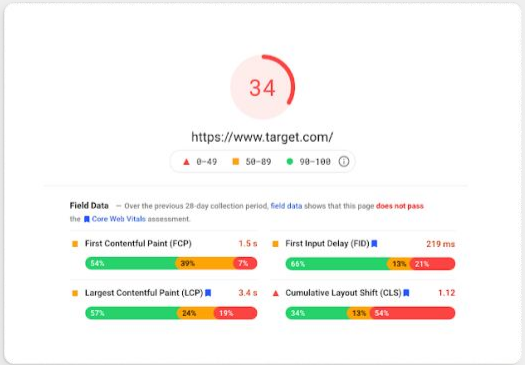


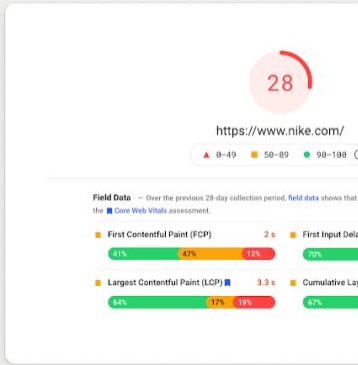
Most sites do not pass



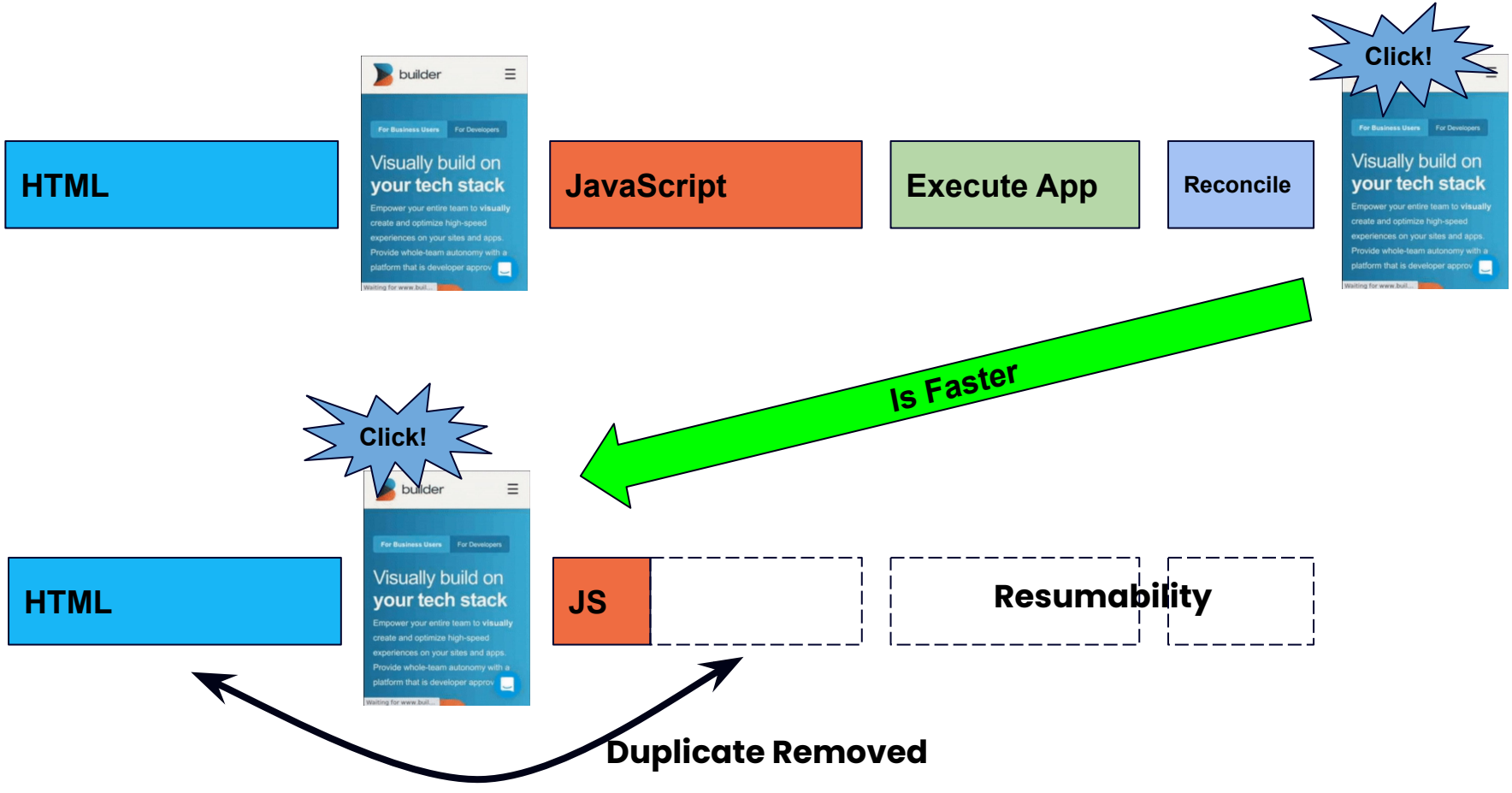








Resumability



www.builder.io?render=next

0.0

www.builder.io

0.0

qwik

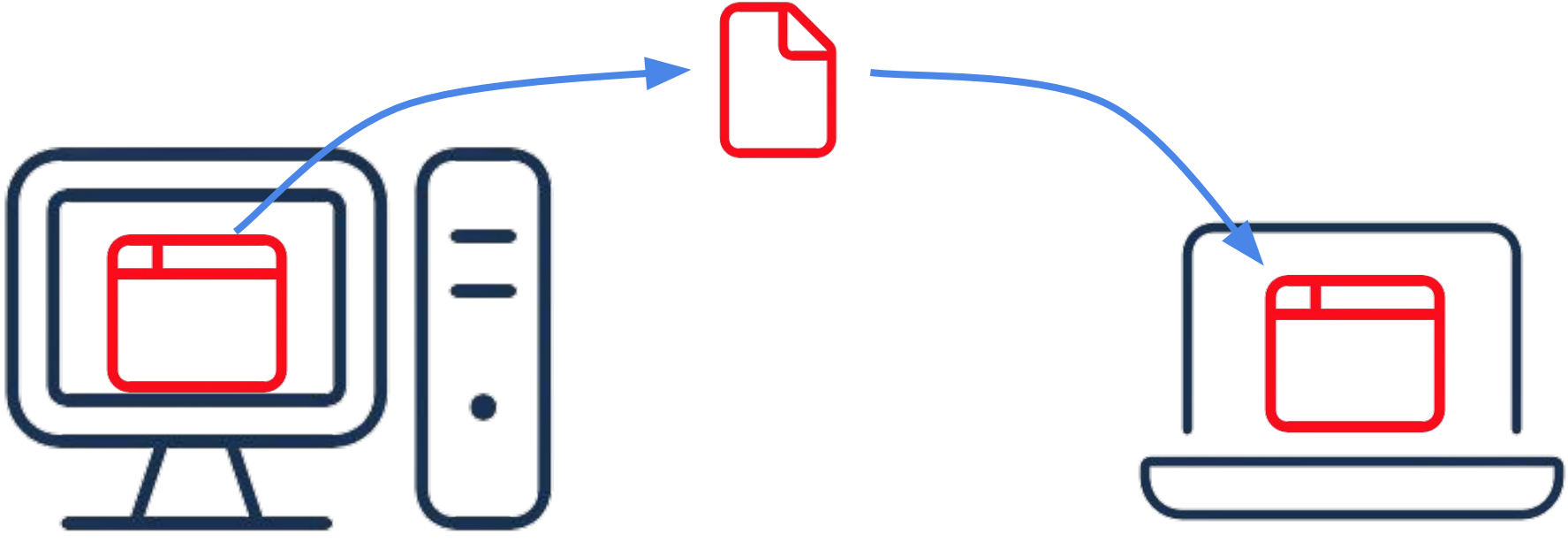
1

Resumable

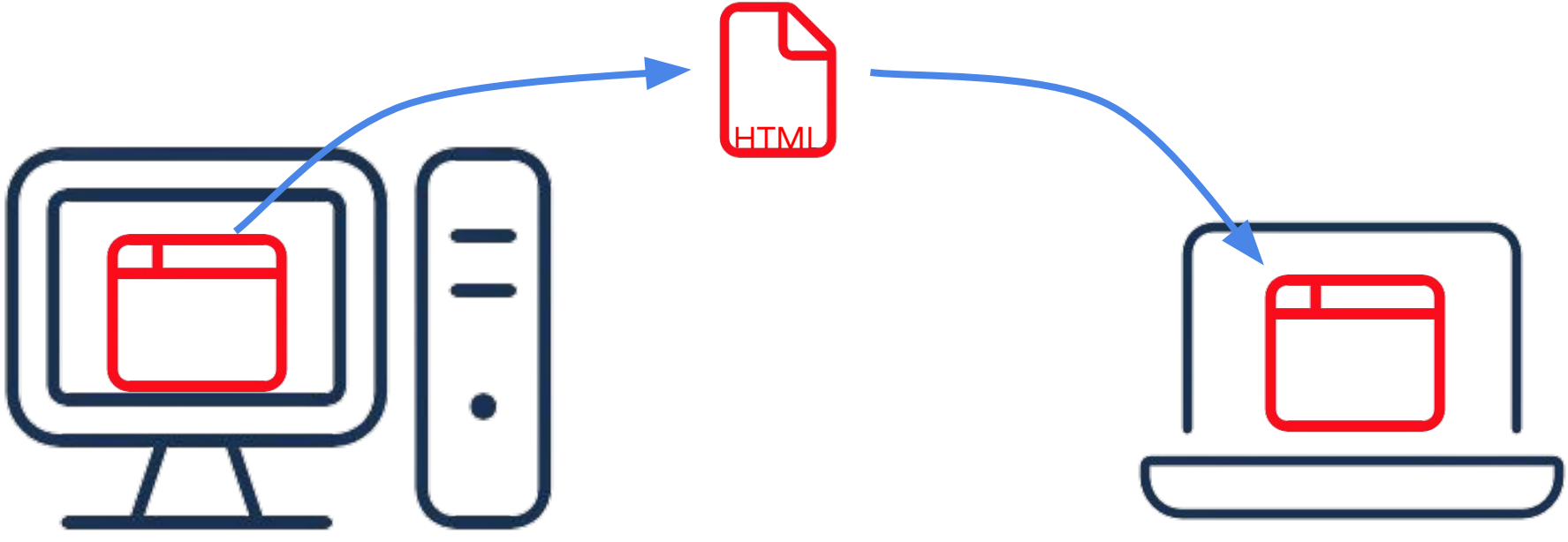
2

Progressive

Virtual Machines are awesome

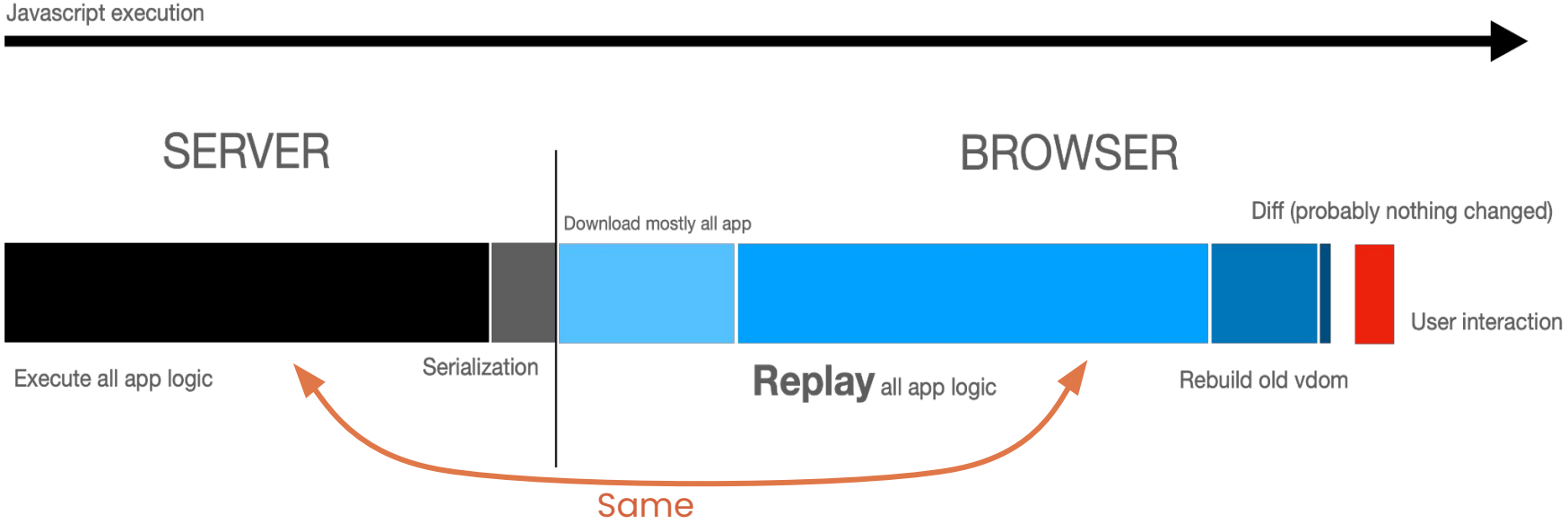


Resumability
(just like VMs)



Hydration = replay app

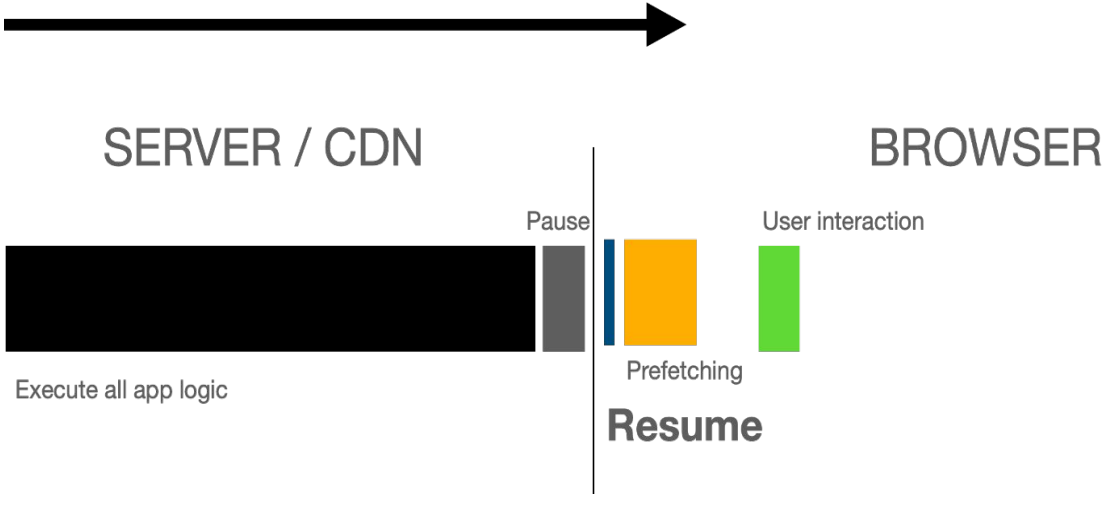
Rendering runs twice. All logic needs to be downloaded.



Resumability = resume app

Leverage what already ran in the server.

No initialization code, rendering, diffing, setting up listeners needs to run in the client.
Server serializes app state then it resumes.



qwik

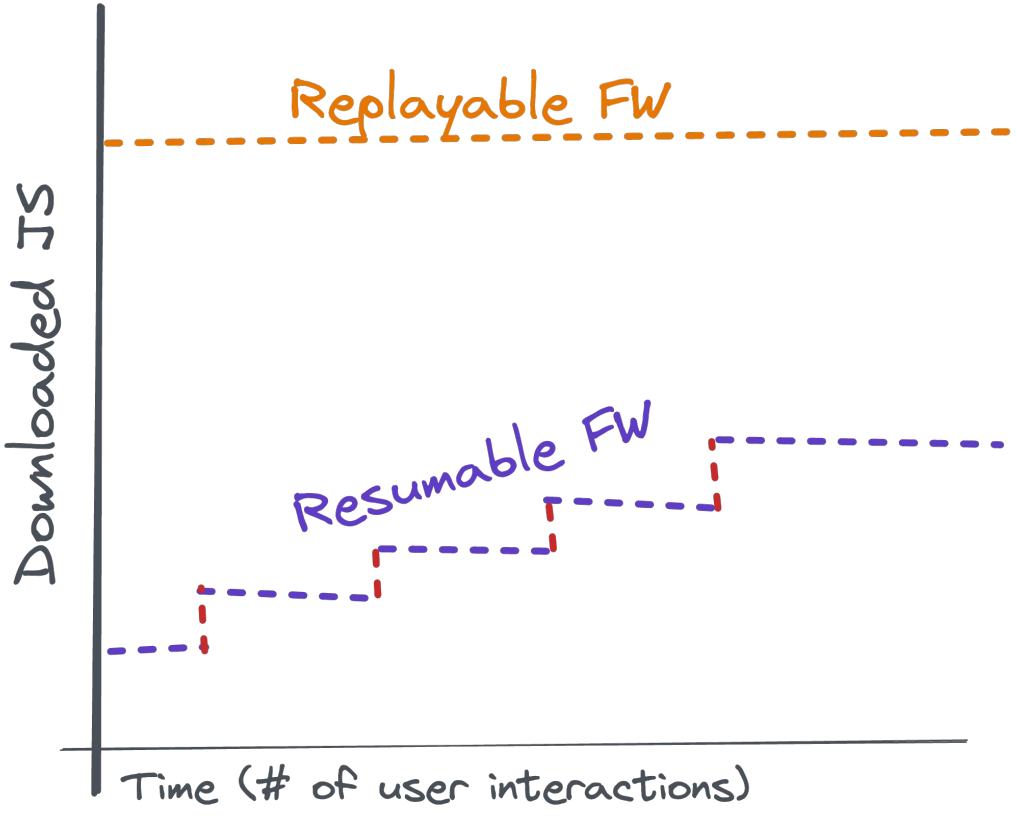
1

Resumable

2

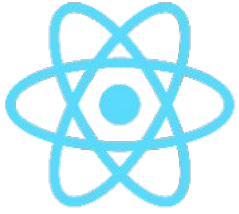
Progressive

Progressiveness



Replayable



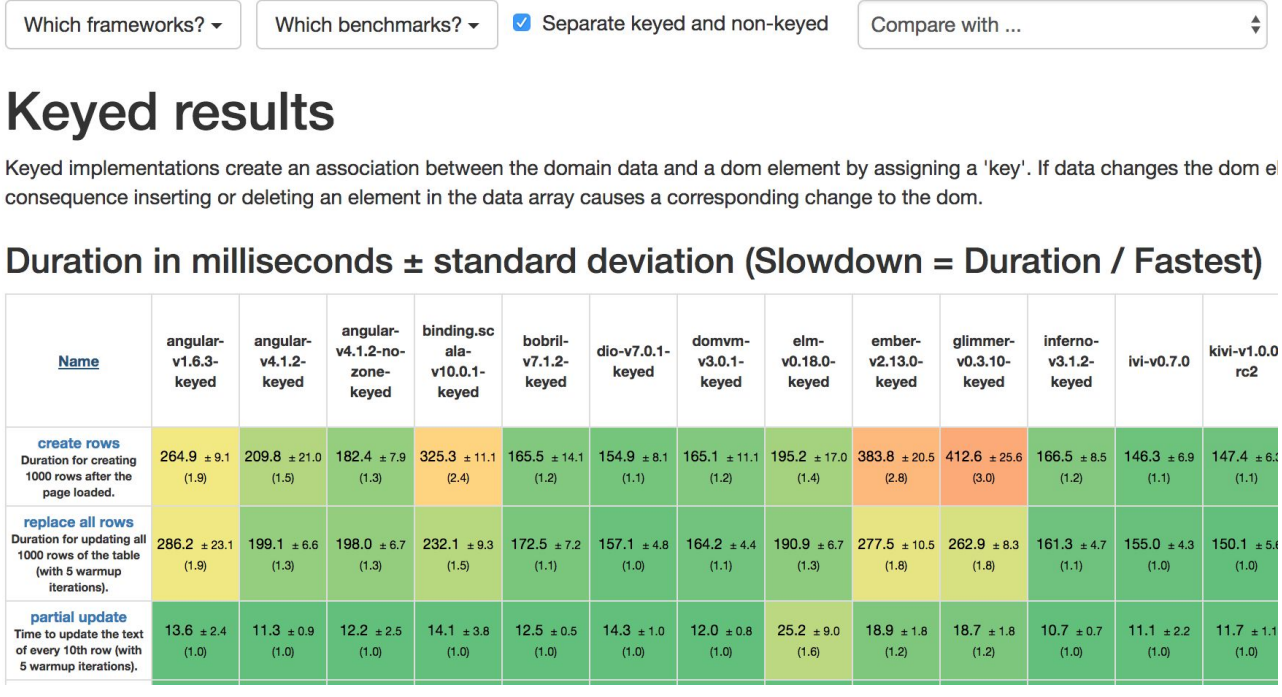


Resumable



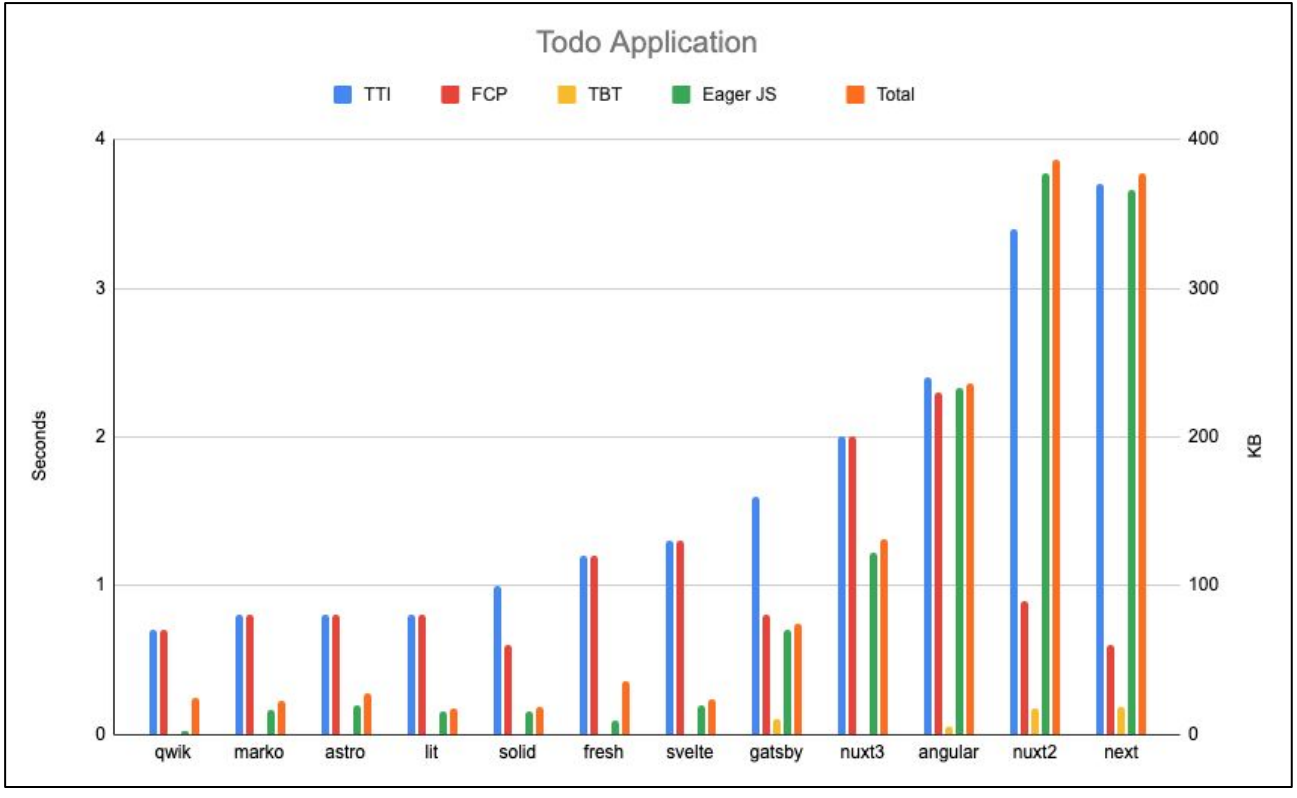
Runtime Perf

- JS downloaded
- Warm JIT
- DOM update (vs user latency)
- Ignore TTI & TBT



Startup Perf

Name	TTI (s)	FCP (s)	TBT (s)	Eager JS (KiB)	Total (KiB)
qwik	0.7	0.7	0.000	2	24
marko	0.8	0.8	0.000	16	22
astro	0.8	0.8	0.000	19	28
lit	0.8	0.8	0.000	15	17
solid	1	0.6	0.000	15	18
fresh	1.2	1.2	0.000	9	36
svelte	1.3	1.3	0.000	19	23
gatsby	1.6	0.8	0.100	70	74
nuxt3	2	2	0.000	122	131
angular	2.4	2.3	0.050	233	236
nuxt2	3.4	0.9	0.170	377	387
next	3.7	0.6	0.180	366	377



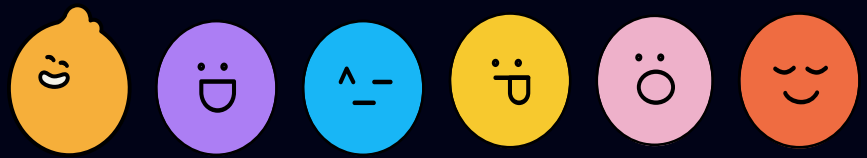
Mitosis


qwik
beta

Demo time

Thank you

Follow me
@mhevery



```
https://qwik.builder.io
npm init qwik@latest
```

BACKUP SLIDES

Why the world needs a new solution?

Who is responsible for slow sites?

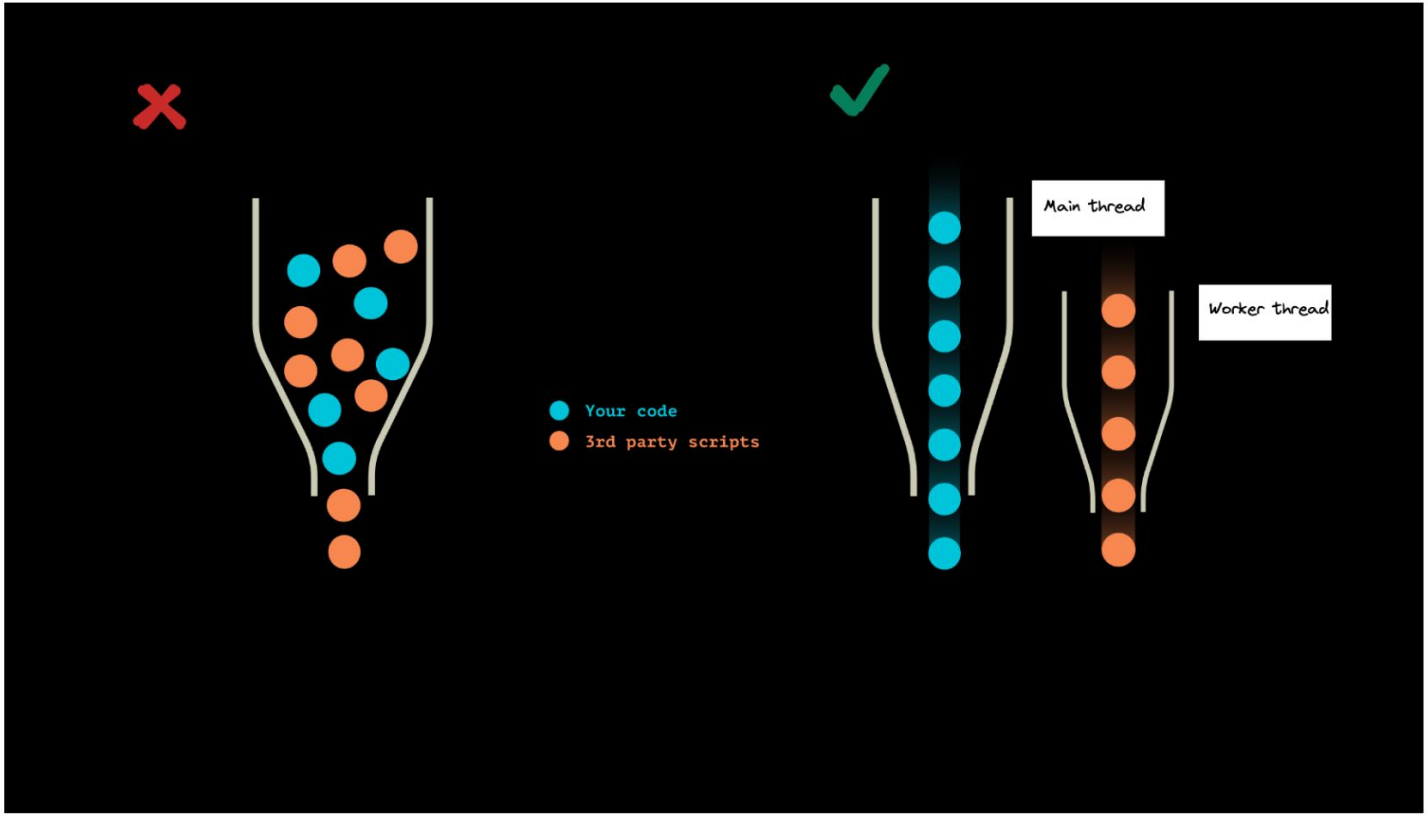
Developers?
Tools?
Frameworks?

Framework
Mental Model

Tools

Developers

Partytown



Component-based development-model

- Developer should write component based applications
- How do we make such code resumable?

```
const MyApp = () => {
  return <Counter value={1} />;
};

const Counter = (props: { value: number }) => {
  const [count, setCount] = useState(props.value);
  return <button onClick={() => setCount(count + 1)}>{count}</button>;
};
```

Resumable Framework

```
<!-- cmp:chunk-xyz.js#CounterRender[0] -->
<button onclick="chunk-abc.js#click[0]">1</button>
<!-- /cmp -->
<script type="STATE">
  [1, ...]
</script>
```

- Don't unnecessarily re-execute code on client.
- Compile time transformation.
- Sufficient information in HTML to "reason" about application.

```
const MyApp = () => {
  return <Counter value={1} />;
};

const Counter = ref('chunk-xyz.js', 'CounterRender');

// FILE: chunk-xyz.js
export const CounterRender = (props: { value: number }) => {
  const [count, setCount] = useState(props.value);
  return (
    <button onClick={ref('chunk-abc.js', 'click', [count, setCount])}>
      {count}
    </button>
  );
};

// FILE: chunk-abc.js
export const click = (count, setCount) => setCount(count + 1);
```

Core Web Vitals Assessment: Passed

Expand view

- Largest Contentful Paint (LCP) — 1.7 s
- First Input Delay (FID) — 32 ms
- Cumulative Layout Shift (CLS) — 0

OTHER NOTABLE METRICS

- First Contentful Paint (FCP) — 1.4 s
- Interaction to Next Paint (INP) — 115 ms
- Time to First Byte (TTFB) — 1 s

Latest 28-day collection period
Various mobile devices
Many samples (Chrome UX Report)
Full visit durations
Various network connections
All Chrome versions

Performance Insights

BETA

Learn what improvements can have the greatest impact on your site's performance

priceline.com

ANALYZE

This is your current speed

Current

Time to interactive:

25.3 s

Show more info

Your score if you:

Optimize Images

Time to interactive:

25.3 s

Optimizing your images has no effect on your speed score

Show more info

Your score if you:

Optimize CSS

Time to interactive:

25.3 s

Optimizing your CSS has no effect on your speed score

Show more info

Your score if you:

Optimize Javascript

Time to interactive:

2.2 s

Reducing your JS can boost your speed score by 60 points and speed up your time to interactive by 23 seconds

Show more info

Your score if you:

Optimize All

Time to interactive:

2.2 s

Optimizing all areas can boost your speed score by 61 points and speed up your time to interactive by 23 seconds

Show more info

Why are we building Qwik?

Site	Current Score
amazon.com	50
walmart.com	37
apple.com	48
target.com	28
homedepot.com	15
bestbuy.com	16
wayfair.com	22
carvana.com	x
costco.com	6
chewy.com	19
lowes.com	33
macys.com	19
kroger.com	x
samsclub.com	28
kohls.com	x
shein.com	8
gap.com	10
nike.com	21
grainger.com	21
qvc.com	x
nordstrom.com	x
bedbathandbeyond.com	x
overstock.com	32
staples.com	71
safeway.com	16
officedepot.com	6
vroom.com	20
dickssportinggoods.com	1
sephora.com	18
stitchfix.com	38
walgreens.com	4
potterybarn.com	x
jcpenney.com	16
ulta.com	17
dell.com	30
victoriassecret.com	13
bathandbodyworks.com	3
apmex.com	81
lululemon.com	10
ikea.com	71
nordstromrack.com	x
ae.com	4
build.com	8
adidas.com	43
newegg.com	21
sweetwater.com	26
zara.com	27
menards.com	3
sears.com	x
zulily.com	18

Every 100ms faster → 1% more conversions
For Mobify, every 100ms decrease in homepage load speed worked out to a 1.11% increase in session-based conversion, yielding an average annual revenue increase of nearly $380,000.

20% faster → 10% more conversions
Retailer Furniture Village audited their site speed and developed a plan to address the problems they found, leading to a 20% reduction in page load time and a 10% increase in conversion rate.

850ms faster → 7% more conversions
COOK reduced average page load time by 850 milliseconds which increased conversions by 7%, decreased bounce rates by 7%, and increased pages per session by 10%.

50% faster → 12% more sales
When AutoAnything reduced page load time by half, they saw a boost of 12% to 13% in sales.

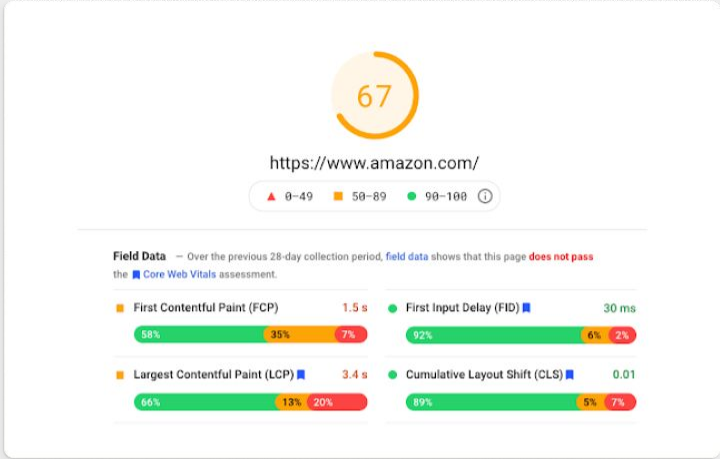
40% faster → 15% more sign-ups
Pinterest reduced perceived wait times by 40% and this increased search engine traffic and sign-ups by 15%.

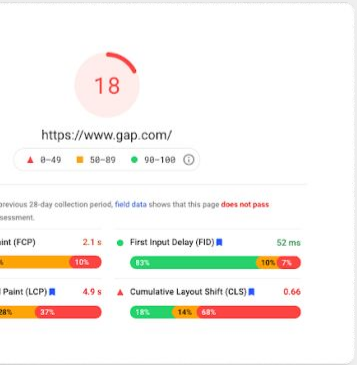
1 seconds slowness → 10% less users
The BBC found they lost an additional 10% of users for every additional second their site took to load.

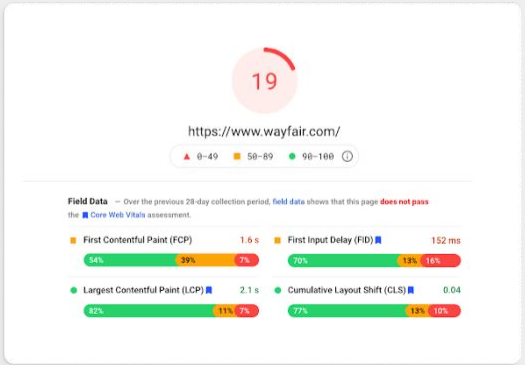
Web Application

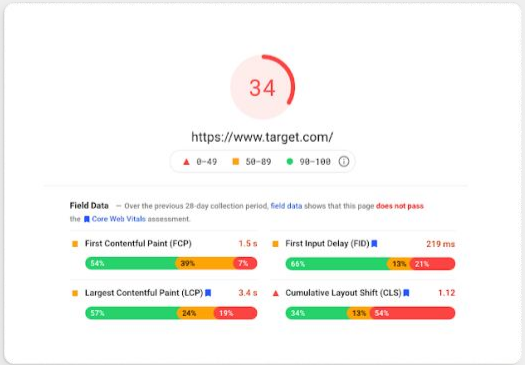
startup performance

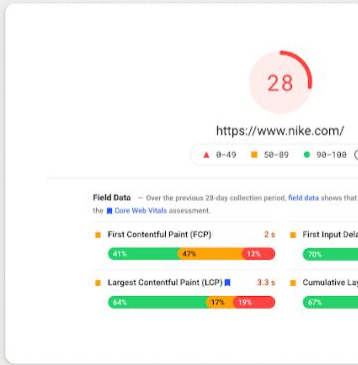
Most sites do not pass



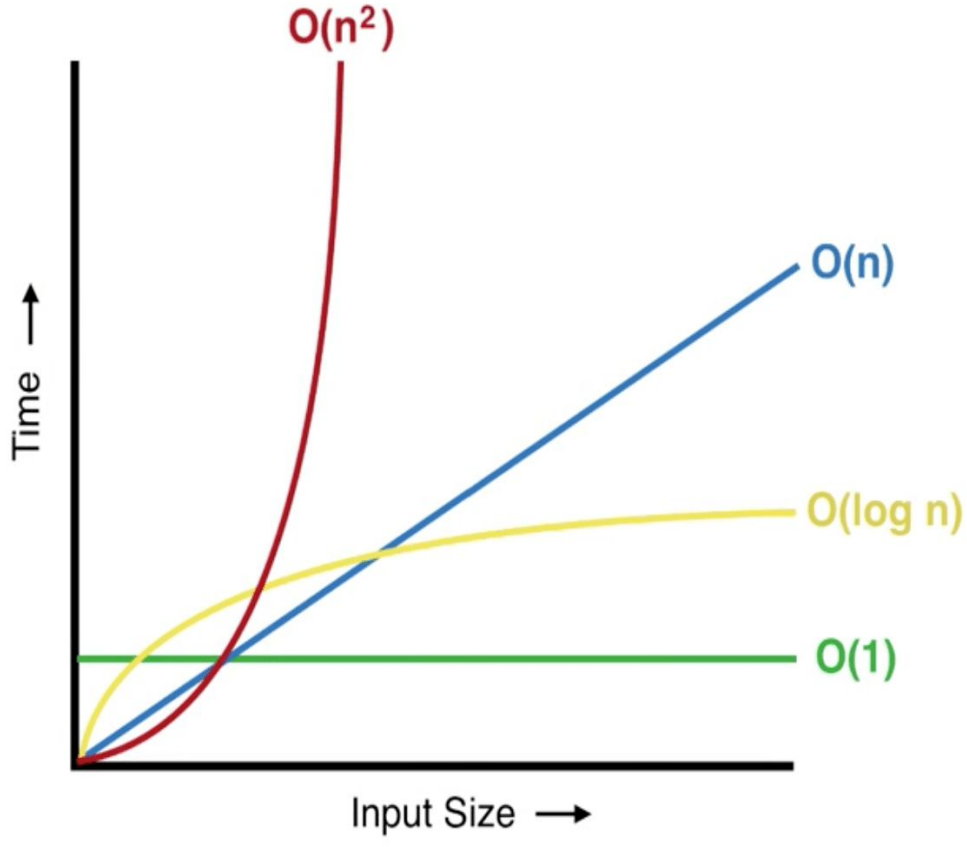




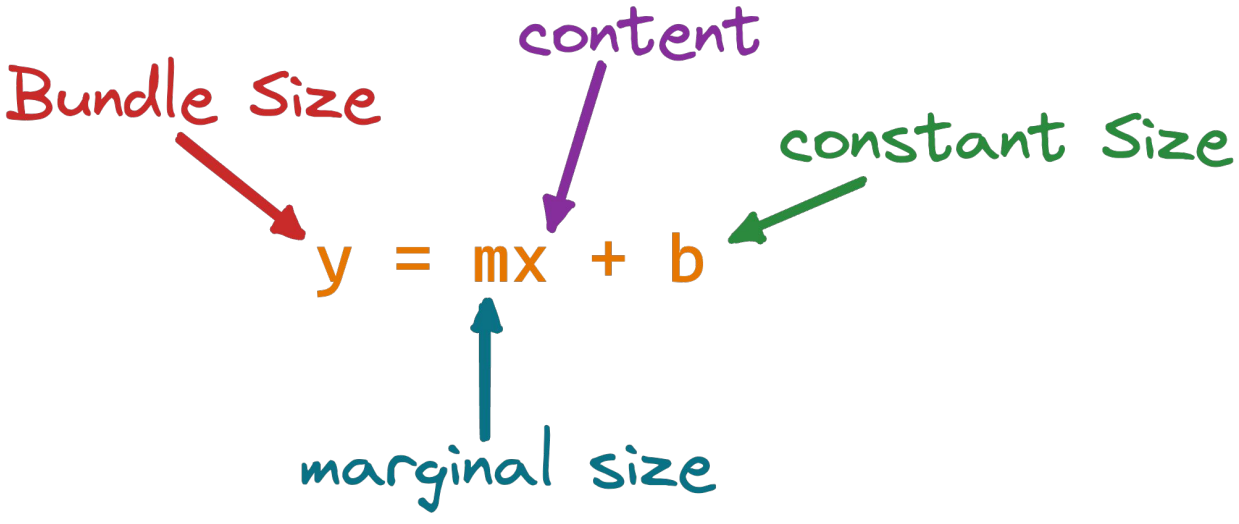




Big O() notation



Let's talk about Frameworks



Replayable

(hydration)

vs

Resumable

O(1) frameworks are not new! but...

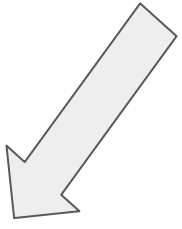
Wiz is an internal framework at Google:

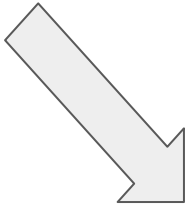
- Powers Google Search, Gmail, Google Photos...
- Around for 8 years!
- Deeply coupled with a backend in Java, business logic may need to be implemented twice, JS and Java
- Lots of manual DOM modifications
- Slow development

Amazon can not use existing frameworks for its main site

- Too slow, does not scale well
- Lots of custom code, allows them to deliver fast website that scale in functionality
- Lots of manual dom modifications
- Slow development

All Frameworks claim that they are FAST





Runtime Perf

Startup Perf

Server-side cake baking analogy

1. Server bakes a cake
2. Server sends a picture of the cake + cooking instructions
3. Client gets the picture + instructions
4. Client goes buy all ingredients
5. Client bakes the cake again
6. User eats the cake

“baking a cake in a datacenter”