

React Performance Playbook

Introduction to React Performance Optimization



Kamran Ayub

Helping React developers create snappy apps

@kamranayub | www.kamranayub.com

Version Check



Live Video



Live Video



**Web performance
optimization is no longer a
choice.**



Live Video



100ms = **1%**
latency fewer sales



1s = **7%**
latency loss in
conversion



1s = **7%**
latency loss in
conversion

For an e-commerce store doing \$100,000 per day in revenue,
that's **\$2.5M** lost per year



Percentage of mobile device website traffic in the United States from 1st quarter 2015 to 1st quarter 2023



20%

Reduction in
abandonment rates

since Google made page speed a ranking factor



When page load time
increases from...

1s to 3s



+32%

1s to 5s



+90%

1s to 6s



+106%

1s to 10s



+123%



**Better performance means
better SEO visibility**



1s = **4.6%**
more load time fewer views



3s = **7.9%**
more load time fewer views



1s = **10%**
more load time fewer visitors



Live Video



**Use data-driven decision
making to justify
performance optimizations.**



SIGN UP: FROM THE US EDITOR
The essential US newsletter, packed with exclusive news, analysis and opinion



UKRAINE: THE LATEST
Listen to our daily podcast invasion

warned over
plot to kill Sikh
on US soil

ld pneumonia
ported in China

entists discover the
can't stop itching

FING

Changing of
K-pop as
plays Gangnam

Jackpink also
after South Korean band
MBEs by the King at
Palace

ince of
th Korean
utfit is a
age of

MAS WAR

Terror victims group
ourt to block hostage deal

health workers carry a body into a
in Gaza



WORLD

Russian actress
and soldiers killed
in HIMARS strike
on awards
ceremony



War latest | Ukrainian army facing
'difficult' winter, says Zelensky

Russian satanic cannibal pardoned
after fighting in Ukraine war

UK NEWS

◆ Live | Autumn
Statement | Tax
burden to hit
record high
despite Jeremy
Hunt's cuts



Winners and losers of 2023
Autumn Statement

Jeremy Hunt offered precious
little for struggling middle class
families

JANET DALEY



South Korea suspends
military pact with North
after spy satellite launch

Duke of York agrees
settlement with
millionaire after being
dragged into complex
legal case

Watch | The video that
proves some airlines
don't care about disabled
passengers

OPINION

ROBERT CLARK

The West can no longer ignore
murderous alliance with Iran

Their relationship has already
caused destruction in Europe and
will likely lead to increased
threats in the Middle East too

DAVID AXE

Xi Jinping's window of opportunity
against the US Navy is closing

ROGER KIMBALL

Liberals are realising that Trump will
probably win. And they are terrified

MADELINE GRANT

Universities have
become incubators for
hatred

DON HUFFINES

Open borders have
brought America to its
knees

ISMAHANE I
Science ou
fight and b
insecurity

MATT



In the Autumn Statement
every voter will get a
small tax cut, a paper
hat and a joke'

MORE STORIES

Unseen Robin
Williams footage
could appear in
'new
documentary'



Buddhist leader arrested after
trying to control followers' minds
with mercury-laced 'medicines'

woman disfigured in attempted suicide bomb could be
hostage deal

Optimizely and Telegraph Performance Case Study

Stakeholders wanted to keep adding third-party tracking scripts

Engineering team wanted to model how performance affects revenue

Artificially slowed down page load through A/B testing



A page load delay of...

4s

.....►

decreased page views by...

-11.02%



A page load delay of...

decreased page views by...

4s



-11.02%

8s



-17.52%

16s



-20.53%

20s



-44.19%



Live Video



| KPI | Dealer Masters (Built on Gatsby) | Average Competitor | Percent Difference |
|---------------------|-------------------------------------|--------------------|--------------------|
| Bounce Rate | 28.75% | 47.85% | -39.92% |
| Overall Load Speed | 3.43s | 7.43s | -53.81% |
| CPC Bounce Rate | 41.4% | 58.37% | -29.04% |
| Organic Bounce Rate | 23.11% | 29.94% | -22.82% |



1s = **1.6%**

more load time higher bounce rate



Live Video



Business Impact of Performance Optimization

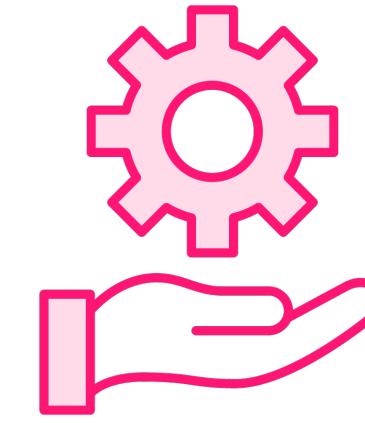


Better brand perception

Business Impact of Performance Optimization



Better brand perception



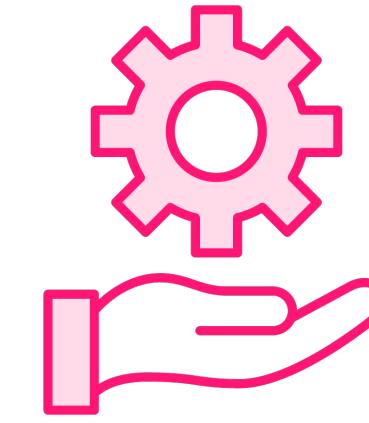
Lower tech debt



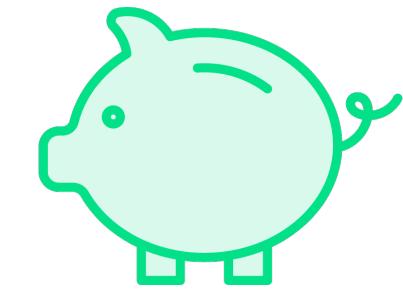
Business Impact of Performance Optimization



Better brand perception



Lower tech debt



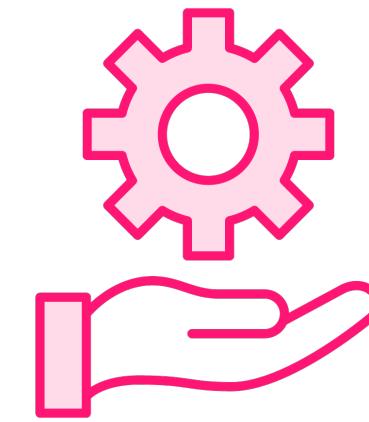
Lower operational cost



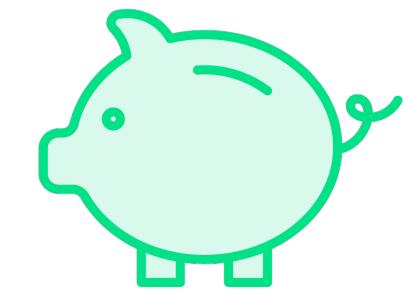
Business Impact of Performance Optimization



Better brand perception



Lower tech debt



Lower operational cost



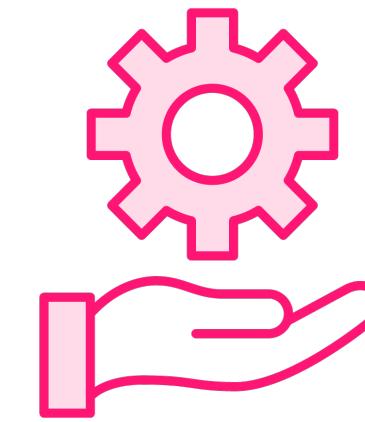
Deeper engagement



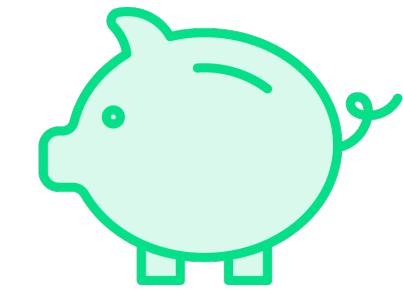
Business Impact of Performance Optimization



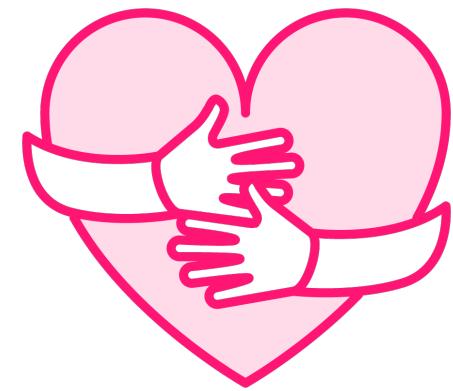
Better brand perception



Lower tech debt



Lower operational cost



Deeper engagement



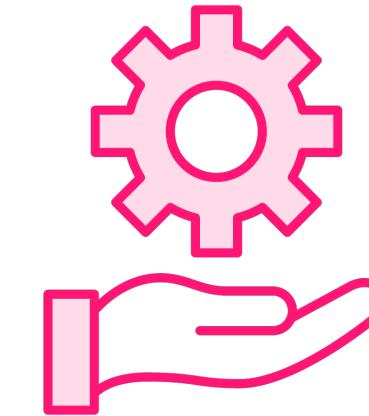
Higher conversion



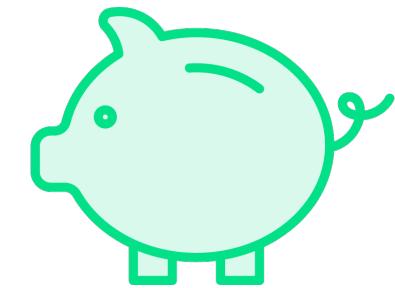
Business Impact of Performance Optimization



Better brand perception



Lower tech debt



Lower operational cost



Deeper engagement



Higher conversion

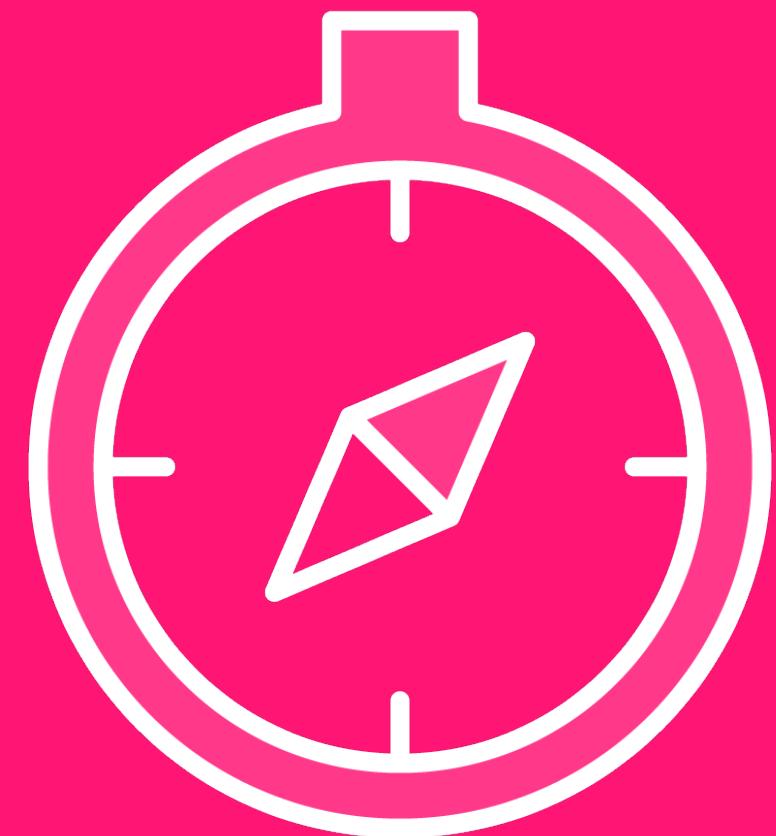


Higher revenue



Live Video: Building faster React apps is good for business

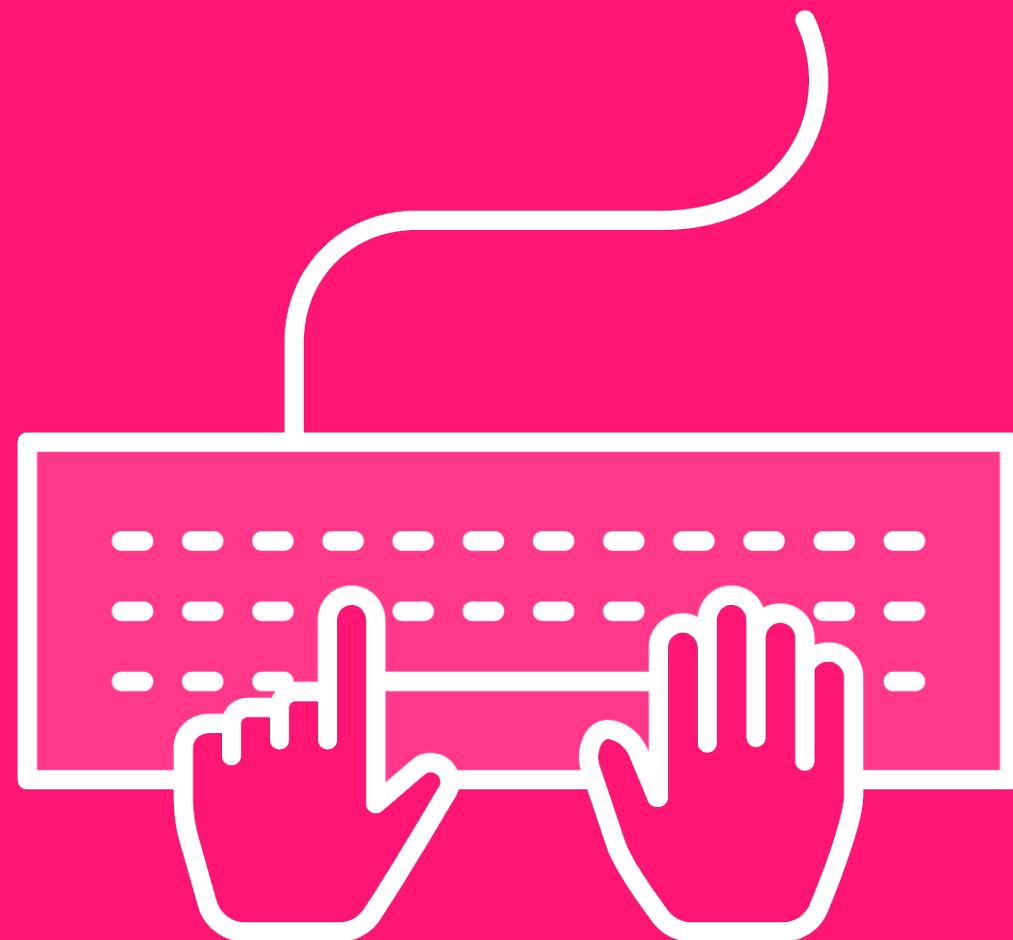




What to Expect

Feel free to jump around. Each module starts simple and ends with more advanced concepts.





Performance in Action

Access the full demo app at:
bit.ly/PSReactPerfDemo



Where to find the demo files

Inside the Pluralsight app, under "Exercise Files"

On GitHub, at bit.ly/PSReactPerfRepo (case-sensitive)



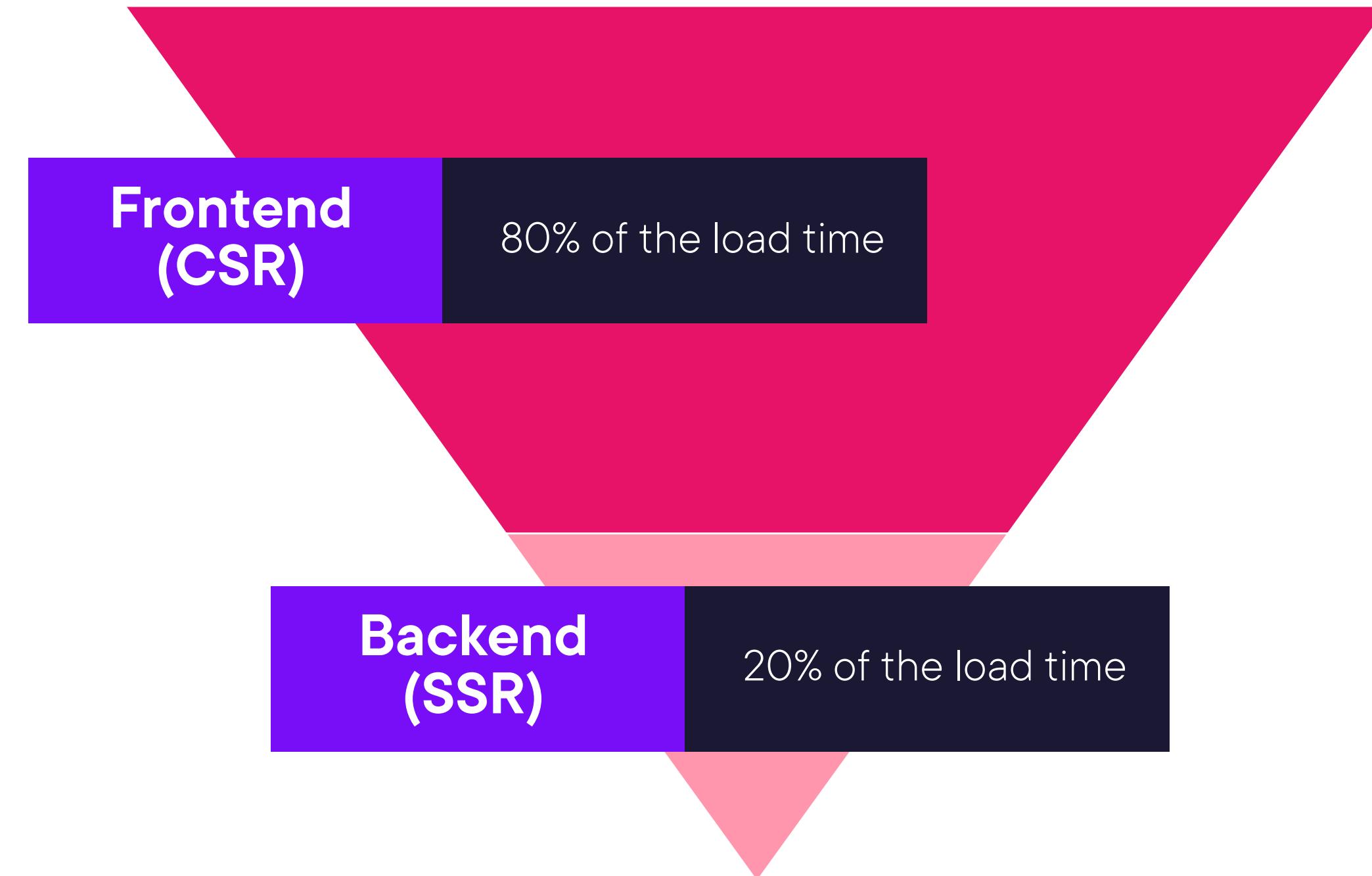
Live Video Outro



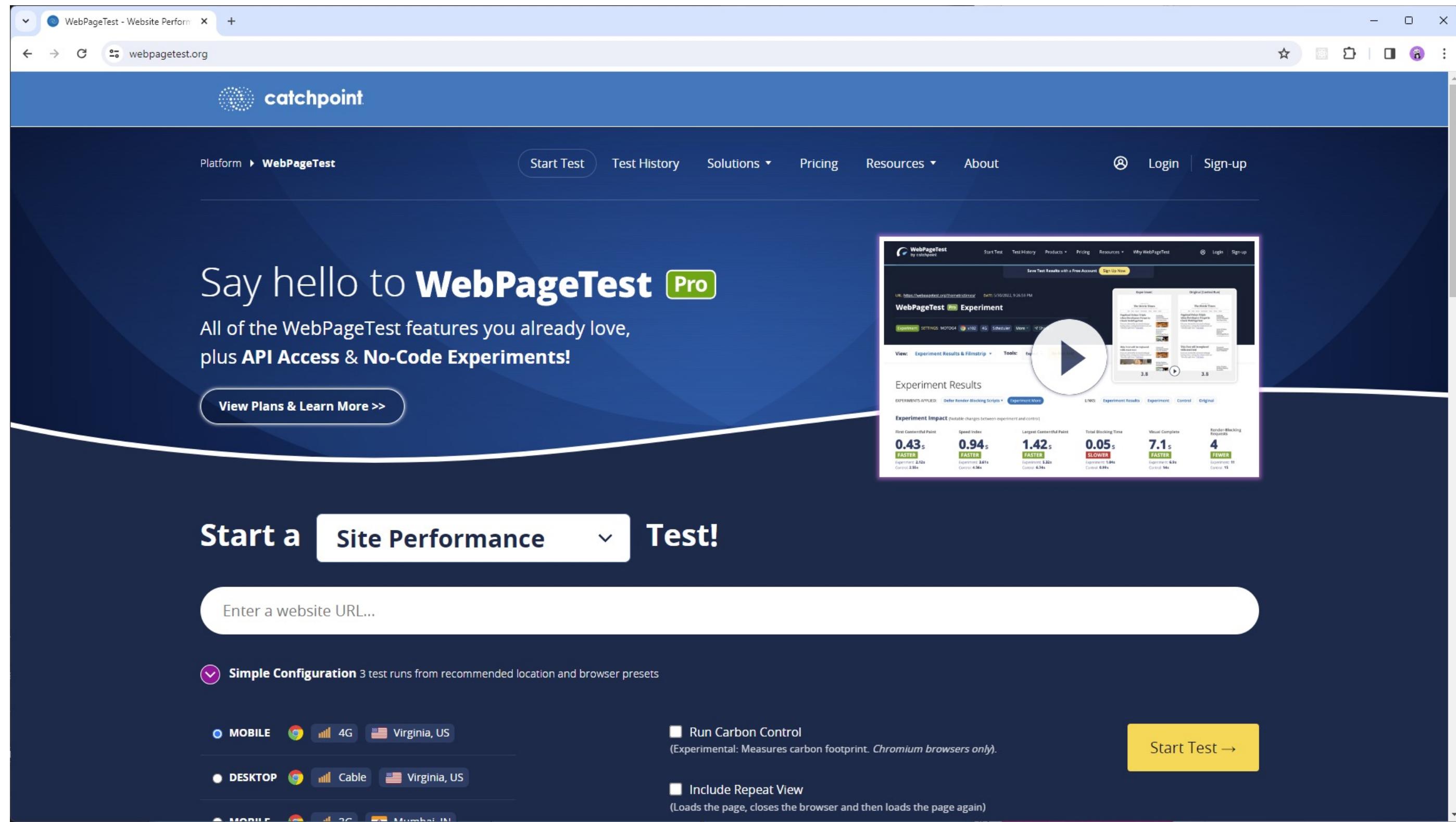


Avoiding the Pitfalls of Performance

80/20 Performance Rule of Thumb?

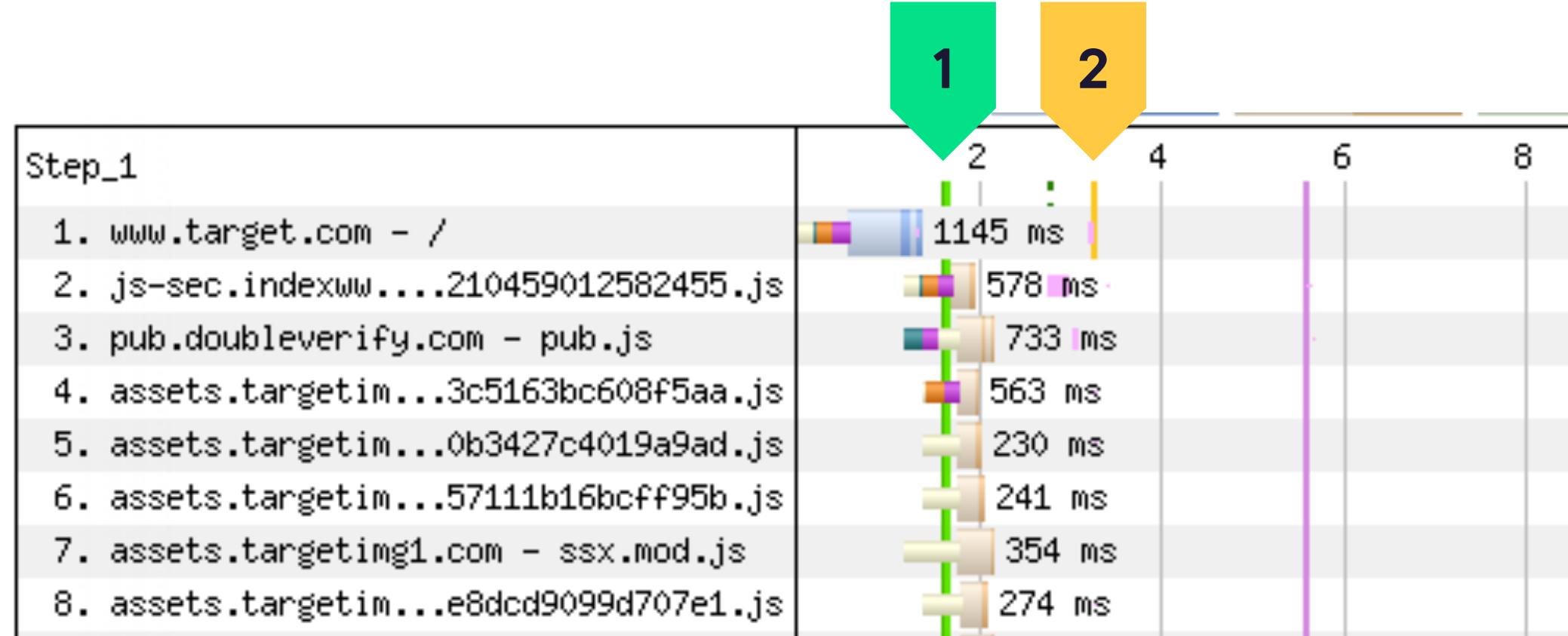


WebPageTest B-Roll Footage



The screenshot shows the main interface of the WebPageTest website. At the top, there's a navigation bar with links for "Start Test", "Test History", "Solutions", "Pricing", "Resources", and "About". Below the navigation is a large banner with the text "Say hello to **WebPageTest Pro**". It highlights "All of the WebPageTest features you already love, plus API Access & No-Code Experiments!". A "View Plans & Learn More >>" button is located below this text. To the right of the banner, there's a small thumbnail preview of a test results page. The main area features a large "Start a Site Performance Test!" button with a dropdown menu above it. Below this, there's a search bar placeholder "Enter a website URL...". Underneath the search bar, there are configuration options for "Simple Configuration" (3 test runs from recommended location and browser presets), "MOBILE" (selected) and "DESKTOP" device types, and "Virginia, US" location. There are also checkboxes for "Run Carbon Control" (experimental, for Chromium browsers only) and "Include Repeat View" (loads the page, closes the browser, and then loads the page again). A prominent yellow "Start Test →" button is located at the bottom right.





Target.com

1

Time-to-First Byte (TTFB)

1.1s

2

Largest Contentful Paint (LCP)

2.7s

Frontend
(CSR)

60/40

Backend
(SSR)

Target.com



Time-to-First Byte (TTFB)



1.1s



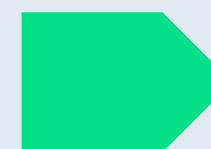
Largest Contentful Paint (LCP)



2.7s

Frontend
(CSR) → 70/30 ← Backend
(SSR)

Airbnb.com



Time-to-First Byte (TTFB)



3s



Largest Contentful Paint (LCP)

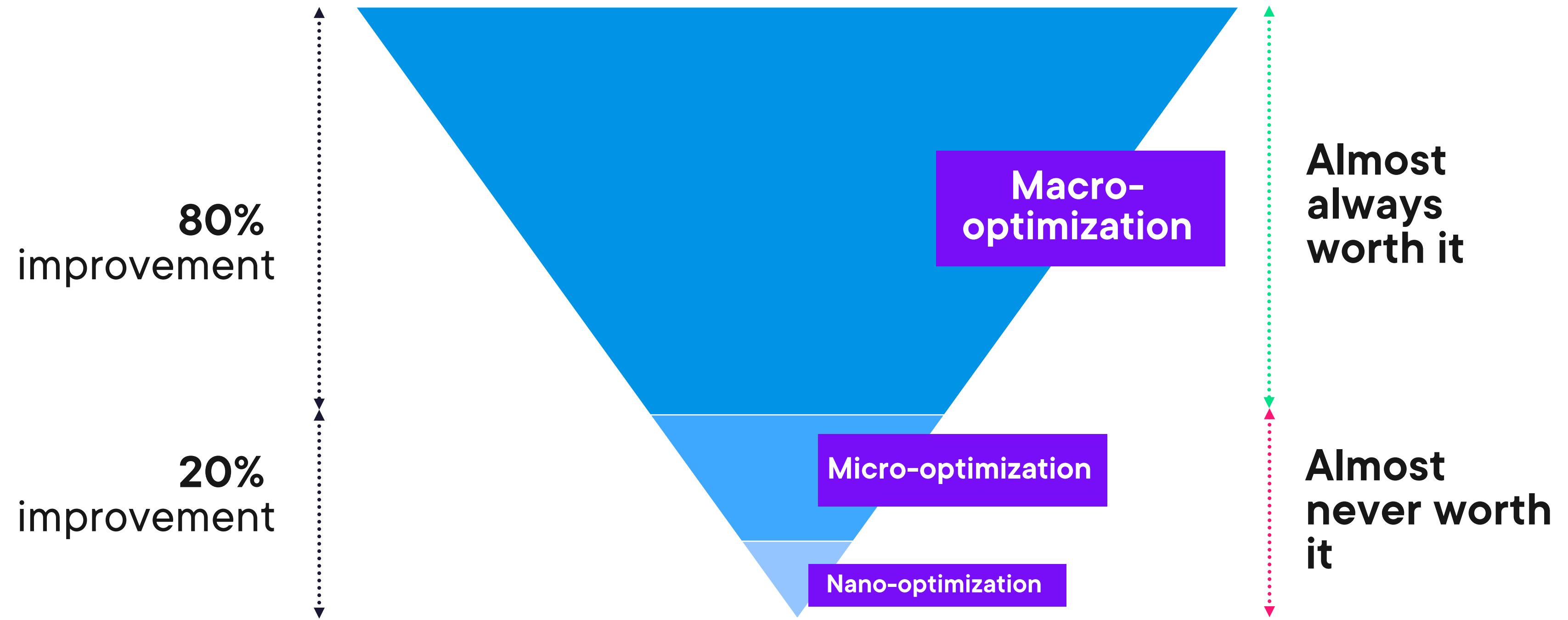


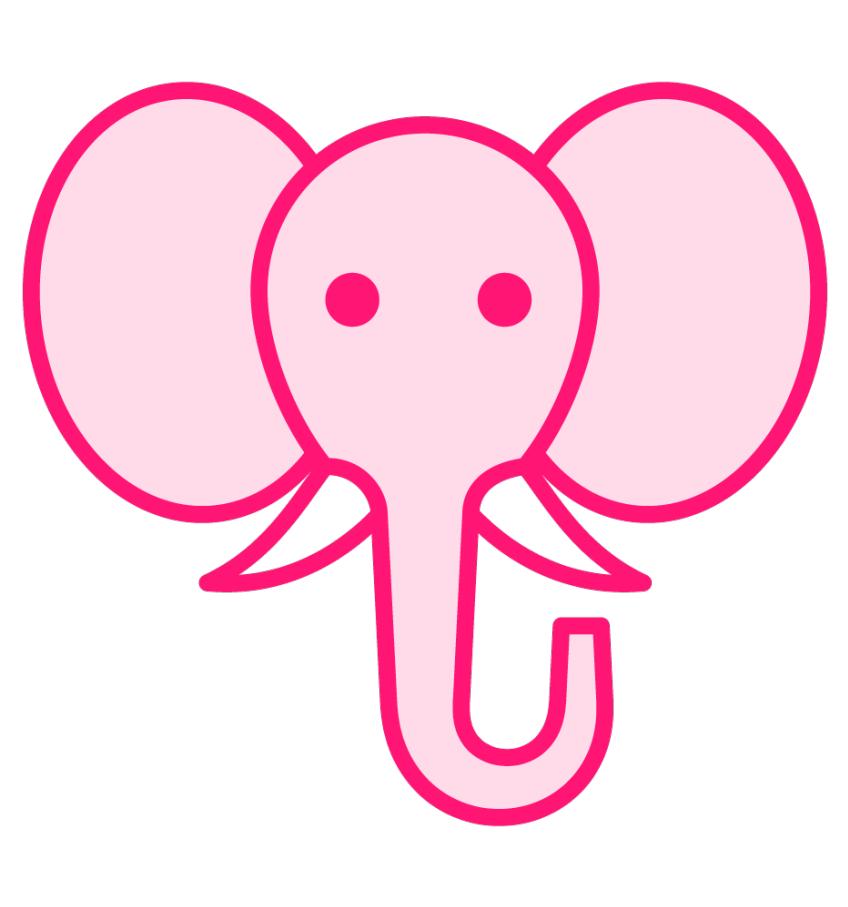
10s

Don’t just rely on “rules of thumb.” Measure your application’s performance to decide where to focus.



The Performance Optimization Hierarchy





Macro optimizations

Contributes a significant performance improvement (on the order of seconds)

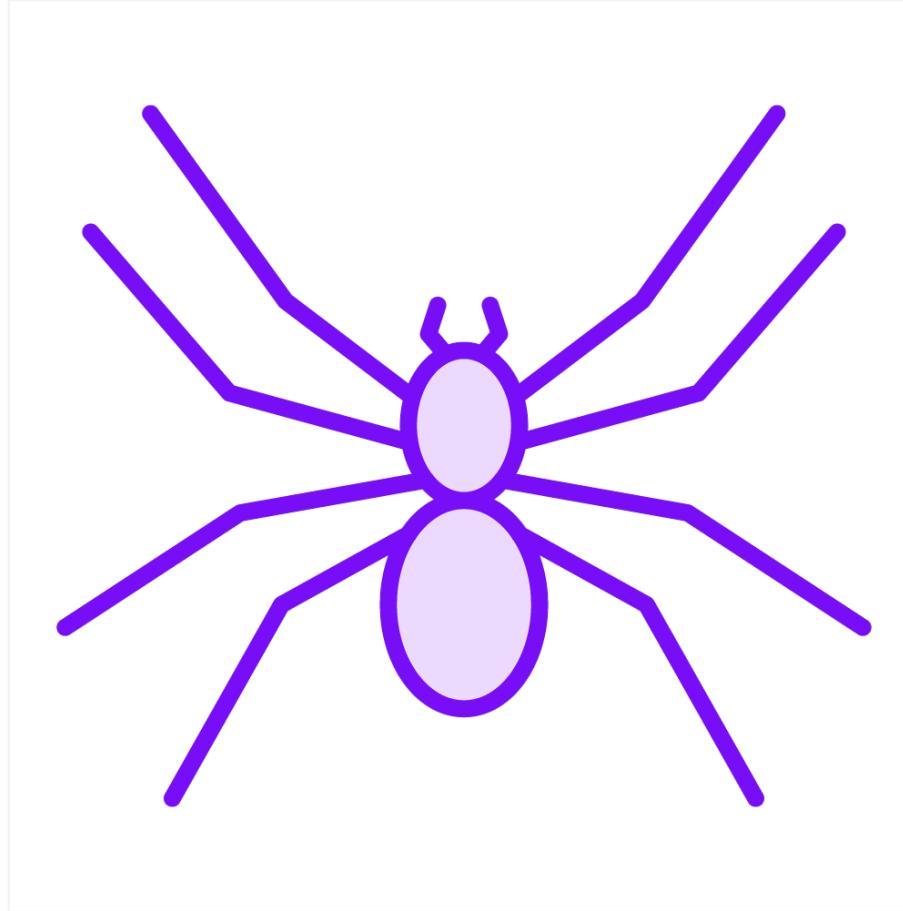




Micro optimizations

Contributes a modest improvement (on the order of 100s of milliseconds)

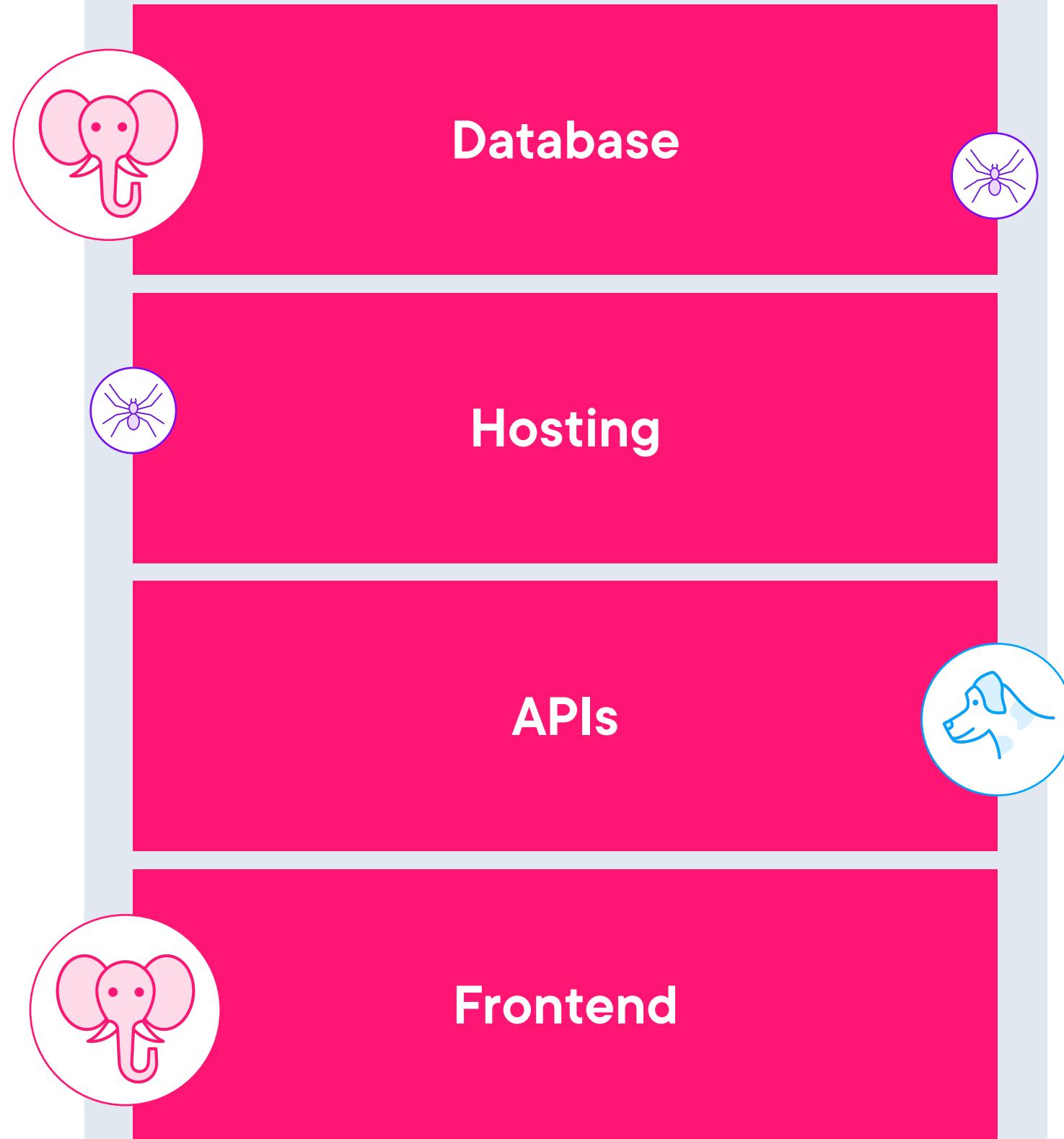




Nano optimizations

Contributes a minor improvement (on the order of 10s of milliseconds)





Example: The Tale of The Slow Spread Operator

```
export const selectProductAttributes = createSelector(  
  selectAttributes, selectProducts, (attr, products) => {  
    // complex merging logic  
  });
```



Example: The Tale of The Slow Spread Operator

```
export const selectProductAttributes = createSelector(  
  selectAttributes, selectProducts, (attr, products) => {  
  
  const mergedProduct = {  
    ...prevAttributes[productId],  
    ...newAttributes[productId],  
    ...commonAttributes  
  };  
  
}) ;
```



Example: The Tale of The Slow Spread Operator

```
export const selectProductAttributes = createSelector(  
  selectAttributes, selectProducts, (attr, products) => {  
  
  const mergedProduct = Object.assign({},  
    prevAttributes[productId],  
    newAttributes[productId],  
    commonAttributes  
  );  
  
}) ;
```

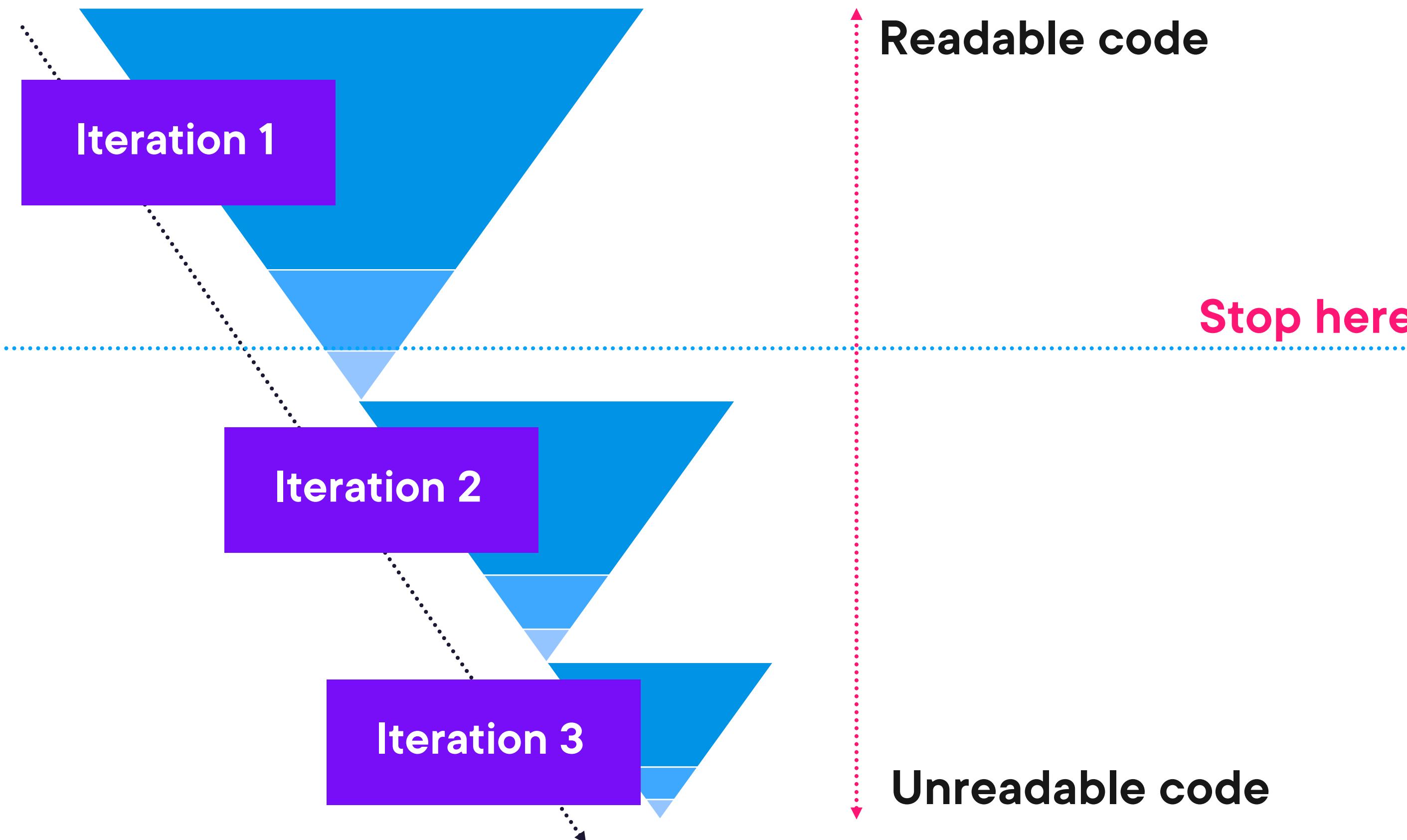


Example: The Tale of The Slow Spread Operator

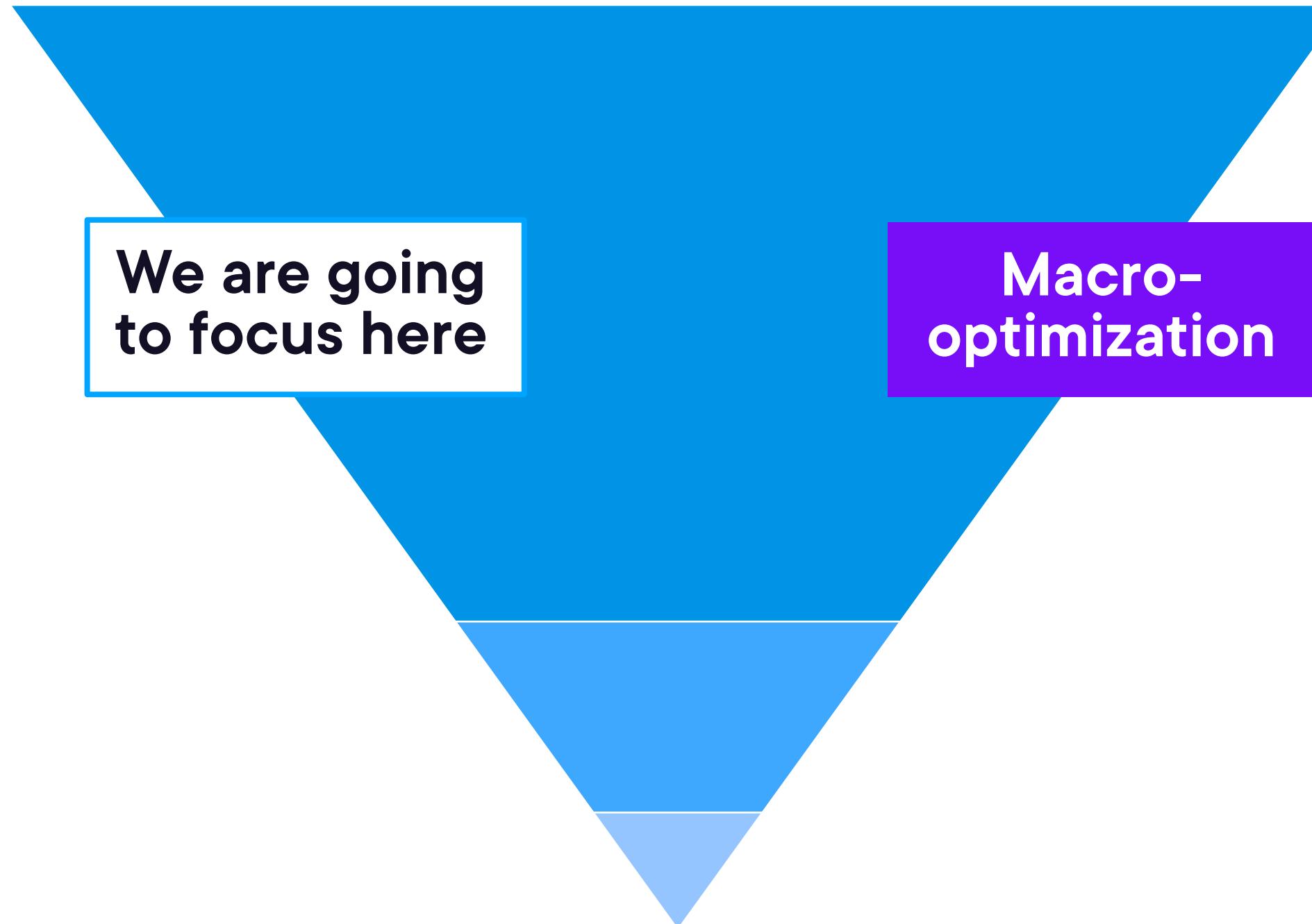
```
export const selectProductAttributes = createSelector(  
  selectAttributes, selectProducts, (attr, products) => {  
  
    // WARNING: Do not use spread (...) operator due to performance issue  
    // See: GH-1234 for more details  
    const mergedProduct = Object.apply({},  
      prevAttributes[productId],  
      newAttributes[productId],  
      commonAttributes  
    );  
  
  }) ;
```



Law of Diminishing (Performance) Returns



The Performance Optimization Hierarchy



**All potential performance
improvements need to take
their larger context into
account**





The Usual Suspects: Get to Know Common React Bottlenecks



Live Video



Changes Triggering Unnecessary Re-renders

The app is slow or janky even when seemingly trivial changes occur.

MyComponent.jsx

```
// Bad Practice
function MyComponent({ list }) {
  return list.map((item, index) =>
    <ChildComponent key={index} item={item} />);
}
```

State or prop changes causing a cascade of unnecessary re-renders in child components.



DOM Thrashing and Manipulation

The UI freezes or lags noticeably, especially during animations or transitions.

useMyAnimation.jsx

```
// Bad Practice
const element =
document.getElementById("myElement");
element.style.height = "100px";
element.style.width = "100px";
```

Directly manipulating the DOM in quick succession, forcing multiple reflows/repaints.



Rendering Large Tables or Lists

Scrolling through a table or list is slow and choppy.

LargeTable.jsx

```
// Bad Practice
function LargeTable({ rows }) {
  return (
    <table>
      {rows.map((row, index) => (
        <tr key={index}>{/* ... */}</tr>
      ))}
    </table>
  );
}
```

Rendering a table with hundreds or thousands of rows without using virtualization.



Inefficient Data Fetching

Data-dependent UI components take too long to render.

UserProfile.jsx

```
// Bad Practice
function UserProfile({ userId }) {
  const [user, setUser] = useState(null);

  // Fetches data every time userId changes,
  // no caching
  useEffect(() => { fetch(`/api/users/${userId}`)
    .then(response => response.json())
    .then(data => setUser(data));
  }, [userId]);
}
```

Fetching data within a component's render method without caching or error handling.



Slow App Loading Time

Users see a blank screen or a loading spinner for a longer than acceptable time.

App.jsx

```
// Bad Practice: No Code Splitting
import ComponentA from './ComponentA';
import ComponentB from './ComponentB';

function App() {
  return (
    <div>
      <ComponentA />
      <ComponentB />
    </div>
  );
}
```

Bundling all your JavaScript together, leading to a large file size and slow initial loading times.



Poorly Optimized State Management

Slow re-renders or unresponsive UI during state updates.

my-reducer.js

```
// Bad Practice
const myReducer = (state, action) => {
  // ...Recalculates everything even if not needed
  return newState;
};
```

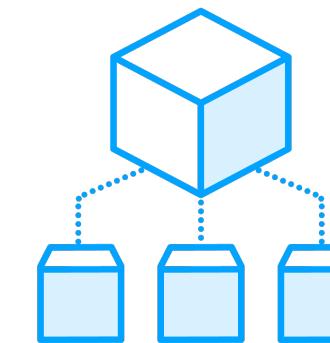
Using Redux without proper memoization, causing unnecessary recalculations and re-renders.



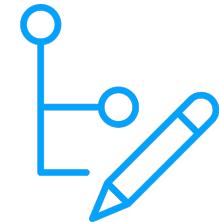
Common React Bottlenecks



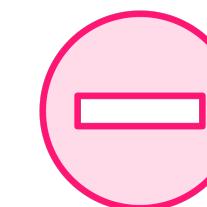
Changes triggering unnecessary re-renders



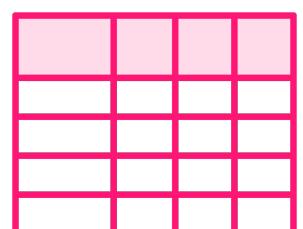
Inefficient data fetching



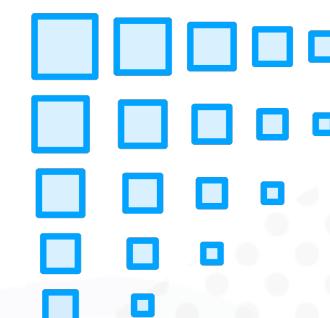
DOM thrashing and manipulation



Slow app loading time



Rendering large lists or tables



Poorly optimized state management



Summary

Today's users expect fast applications which is why speed equals revenue

There are diminishing returns to performance improvements which is why it's important to measure first, then prioritize

React bottlenecks usually stem from unnecessary updates or ineffective memoization

