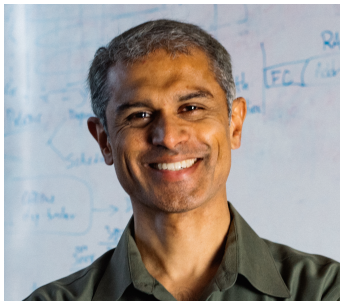
Vesper: Measuring Time-to-Interactivity for Web Pages

Ravi Netravali[*]

Vikram Nathan[*]

James Mickens[‡]



Hari Balakrishnan[*]

[*]MIT CSAIL [‡]Harvard University

The Importance of Page Load Time

Slow page loads → lost revenue and low search rank

Research: Site Speed Is Hurting ~~Your~~ Everyone's Revenue

IAN LURIE // MAY 9 2014

Site speed, site speed, site speed. Everyone around me is sick of hearing me rant abo

because I've pushed it on every client Portent's had since, oh, 2008.

How One Second Could Cost Amazon $1.6 Billion In Sales

ts suggests that if this sentence takes longer than a second
l have clicked elsewhere already. If you've got the patience
for more shocking data on not dawdling.

Google Rank Website On Loading Time of the Page

By: Harsh Agrawal | In: SEO | Last Updated: 18/03/2015

Few months back Google webmaster team indicated that they will start ranking websites
e loading time. Websites which take ages to load slows down the
re considering this factor seriously. Apart from other parameters like
scriptions, Google will also consider Page load time as one of the
your website search engine ranking.

Google Play will now downrank poorly performing apps

Posted Aug 3, 2017 by Sarah Perez (@sarahintampa)

It's Official: Google Now Counts Site Speed As A Ranking Factor

Matt McGee on April 9, 2010 at 2:00 pm

Google has kept a promise it made last year: Site speed is now a ranking factor in Google's algorithm, and is already in place for U.S. searchers. But Google also cautions web site owners not to sacrifice relevance in the name of faster web pages, and even says this new ranking factor will impact very few queries. More on that below, but first the background on today's announcement from Google Fellow Amit Singhal and Matt Cutts, head of Google's web spam team.

Why Page Speed Matters

The first warning that site speed was on Google's radar came last November, when Cutts said there

Here's Everything You Need to Know About Google's 'Speed Update'

BY TONY PALAZZO

Business.com / Marketing Solutions / Last Modified: March 28, 2018

SHARE THIS

Website loading speeds on mobile devices will now be used as ranking factors in Google search results. Here's how to take action and protect your ranking.

Earlier this year, Google announced a big change to the way they factor search rankings. For the first time, website loading speeds on mobile devices will be used as ranking factors in search results.

How Website Speed Actually Impacts Search Ranking

On-page SEO

The author's views are entirely his or her own (excluding the unlikely event of hypnosis) and may not always reflect the views of Moz.

Google uses a multitude of factors to determine how to rank search engine resu
factors are either related to the content of a webpage itself (the text, its URL, th

Everyone agrees that web pages should load quickly…

Everyone agrees that web pages should load quickly…

…but how should page load time be defined?

Contributions

1. **Ready Index (RI)**: analytical definition of page time-to-interactivity in terms of visibility and functionality
2. **Vesper**: system that automatically measures RI
3. **Optimizing pages for RI**: framework to optimize page loads for time-to-interactivity
4. **User studies**: interactive users strongly prefer pages that optimize for RI

Outline

- How pages load today
- Existing Metrics
- Ready Index (RI)
  - Definition
  - Measurement system (Vesper)
- Evaluation
  - RI vs. preexisting metrics
  - Optimizing pages for RI (time-to-interactivity)
  - User studies: how does RI capture user experience?

Page Loads Today

Page Loads Today

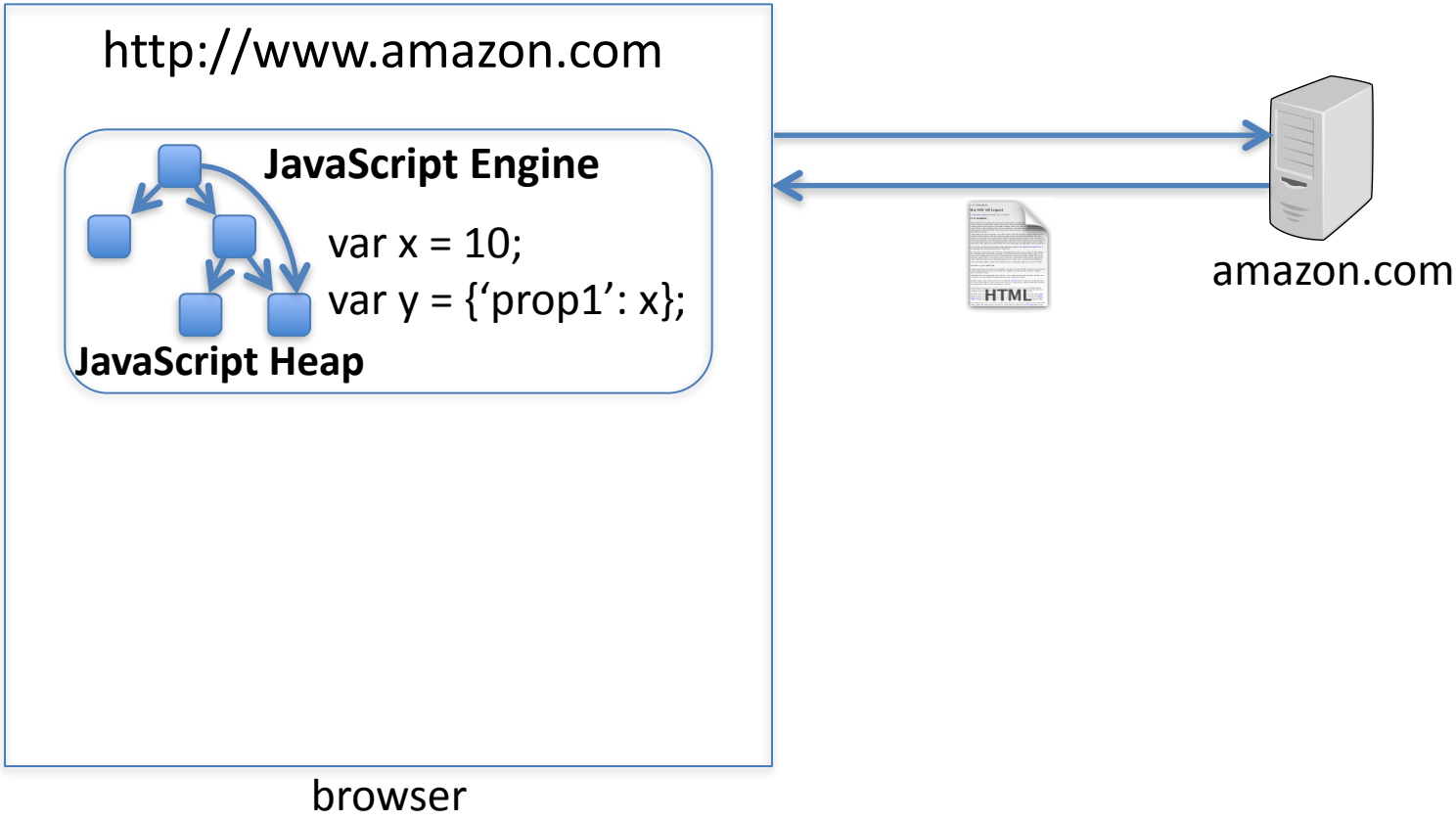
Page Loads Today

browser

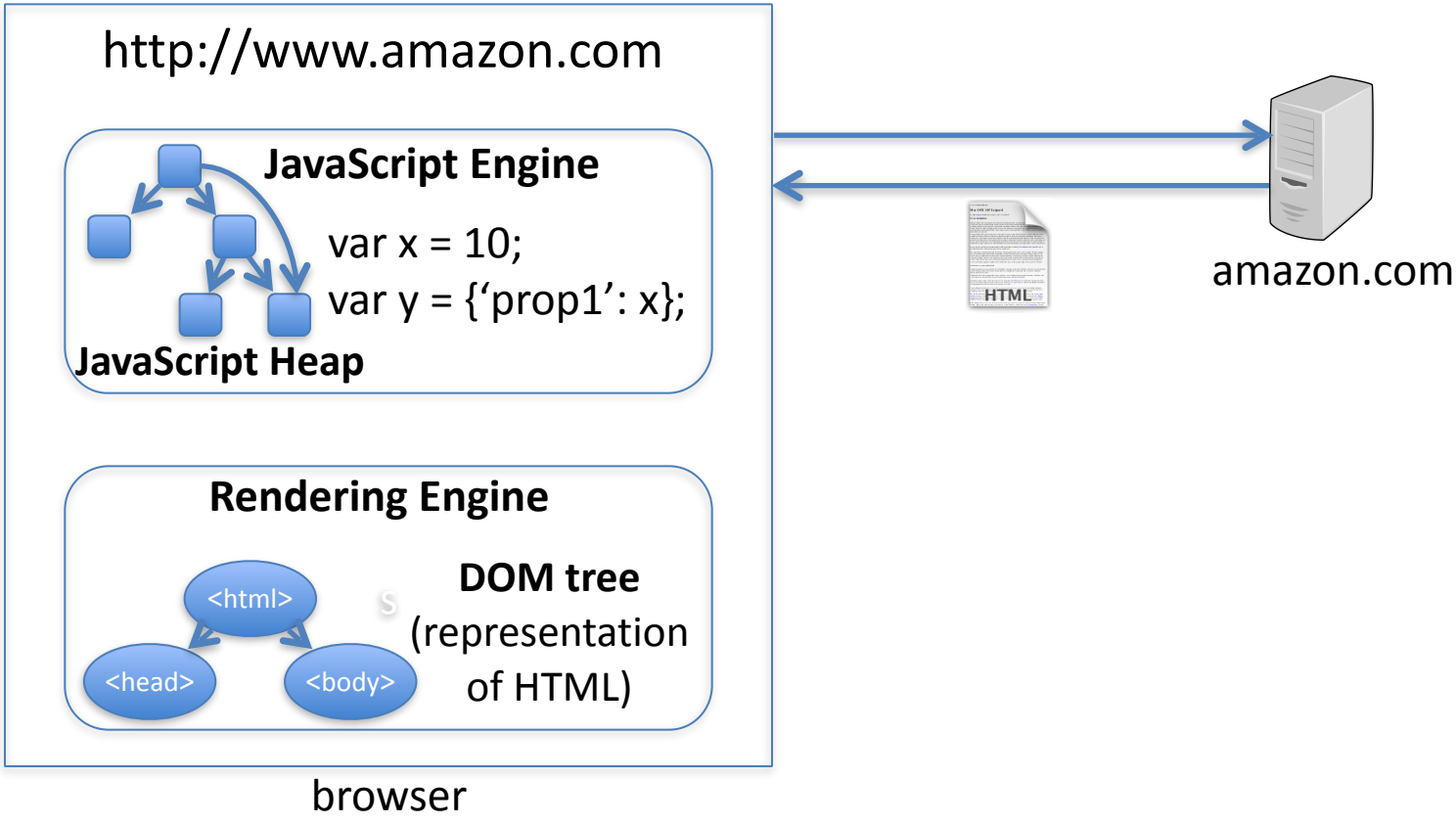
Page Loads Today

Page Loads Today

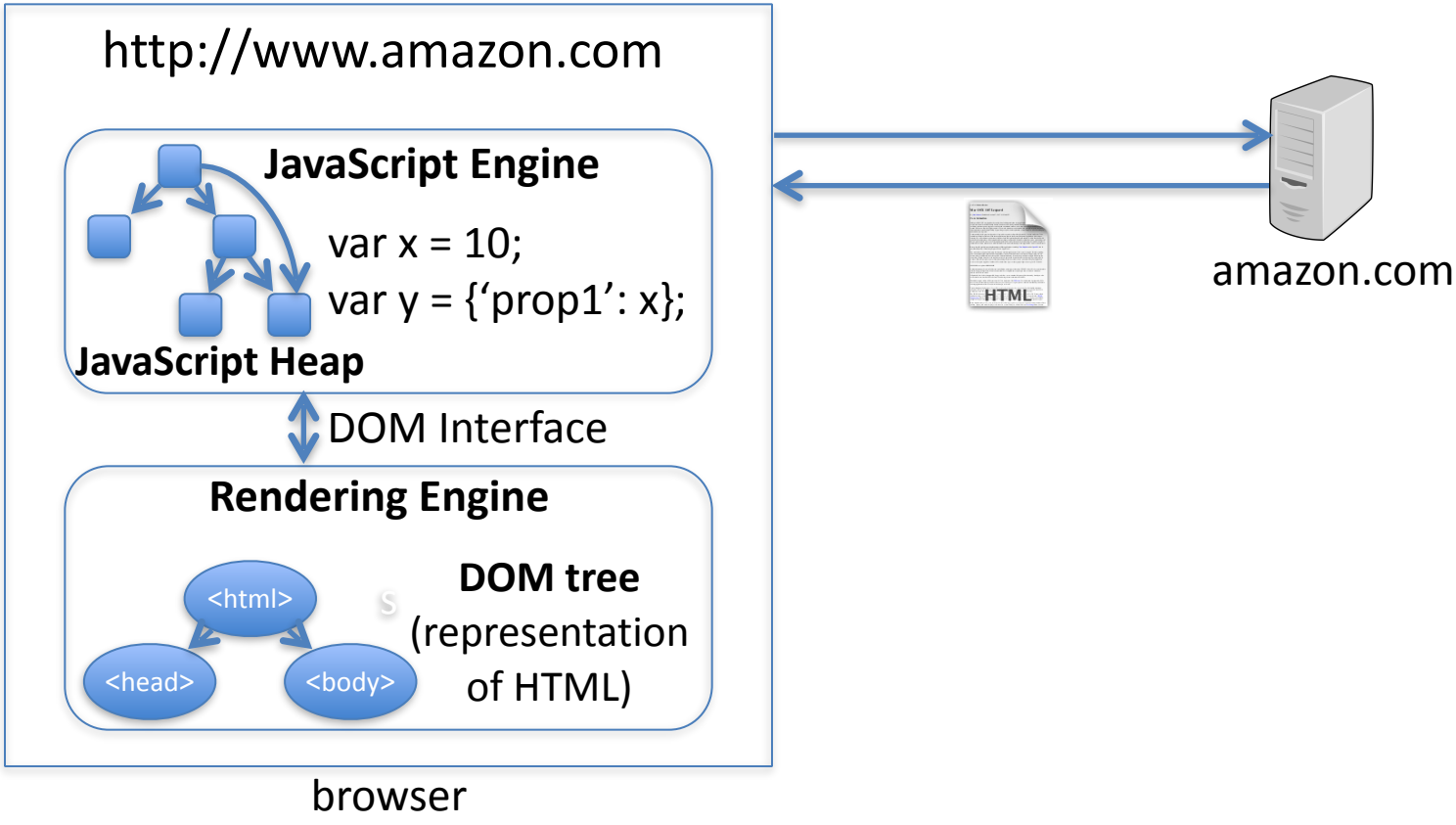
Page Loads Today



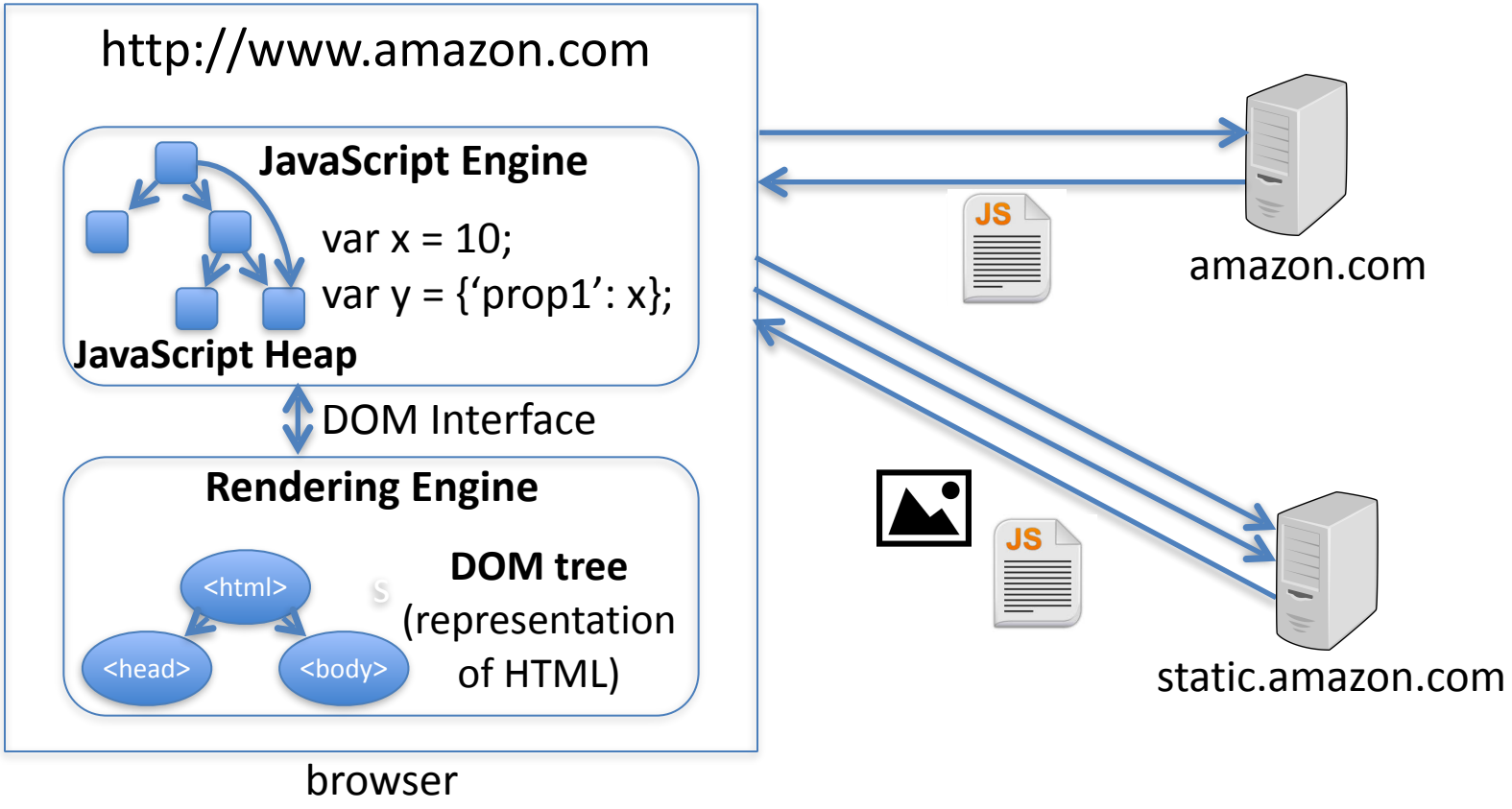
Page Loads Today



Page Loads Today

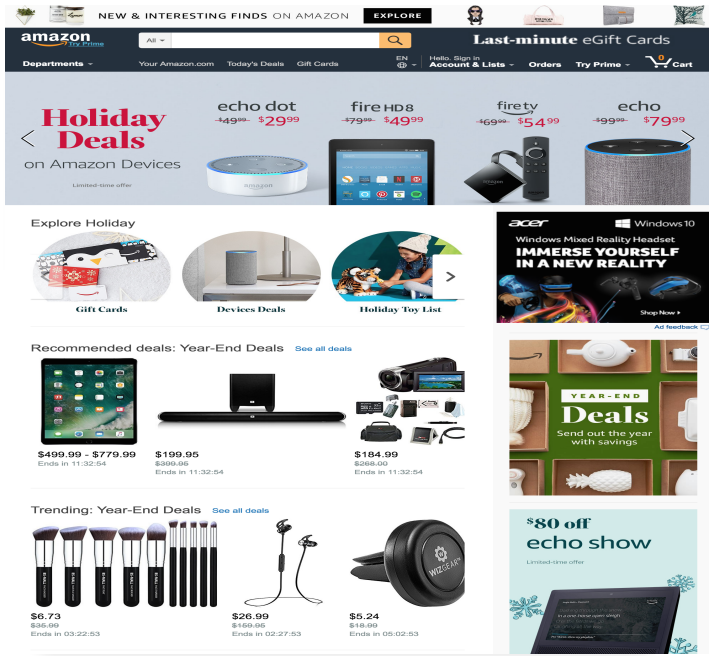


Page Loads Today



Page Loads Today

http://www.amazon.com



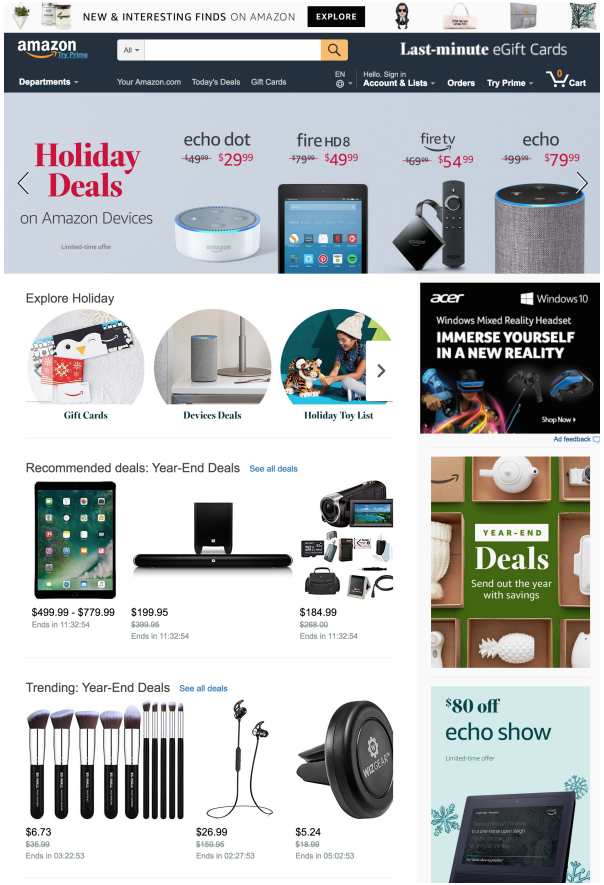
browser

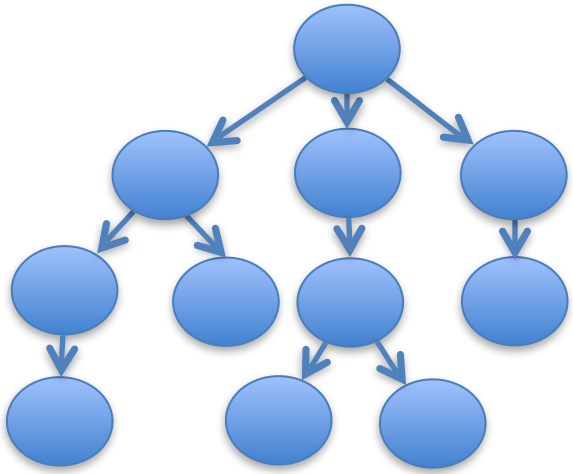
amazon.com

static.amazon.com

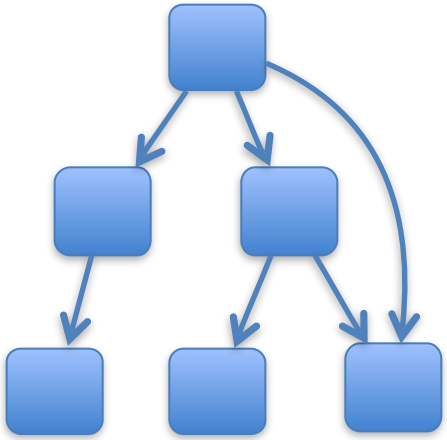
Existing Metrics

Existing Metrics



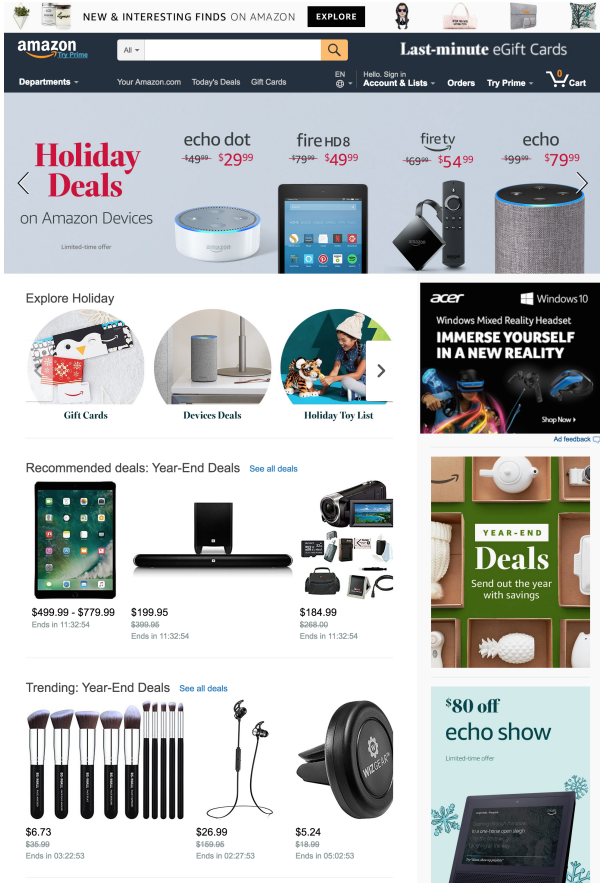


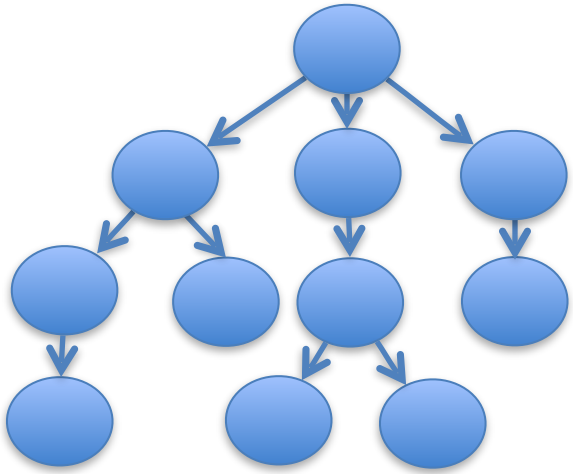
DOM Tree



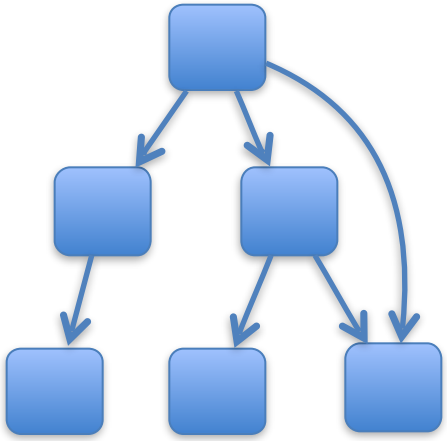
JavaScript heap

Existing Metrics





DOM Tree



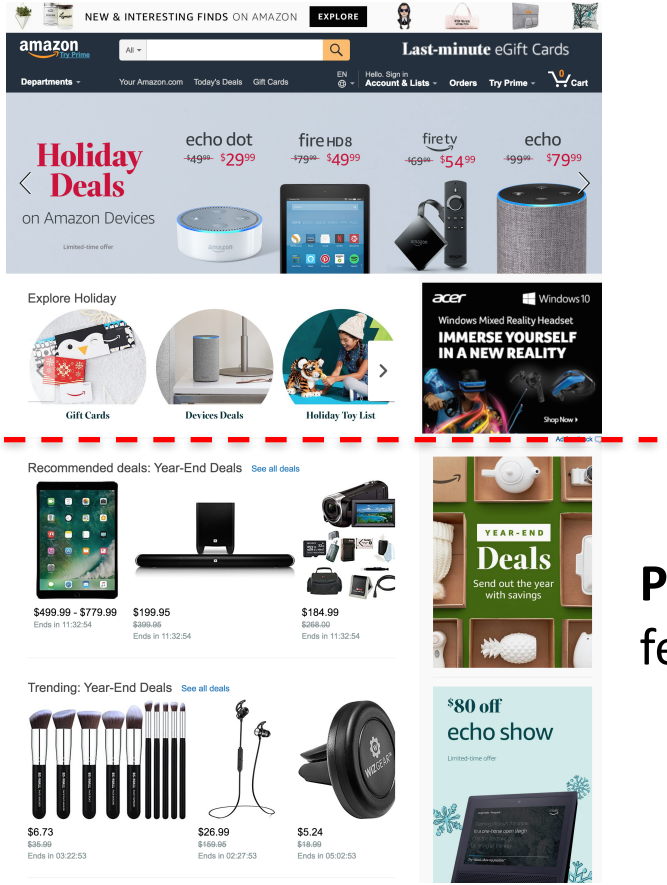
JavaScript heap

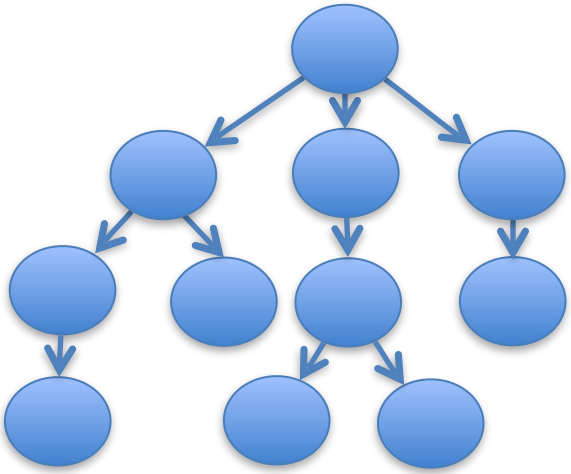
Page load time (PLT): time until all objects are fetched and evaluated

Existing Metrics

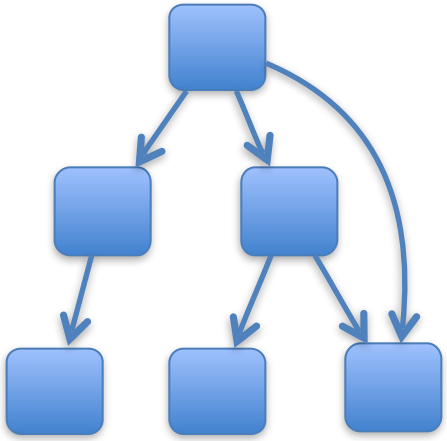
Above The Fold

Below The Fold





DOM Tree



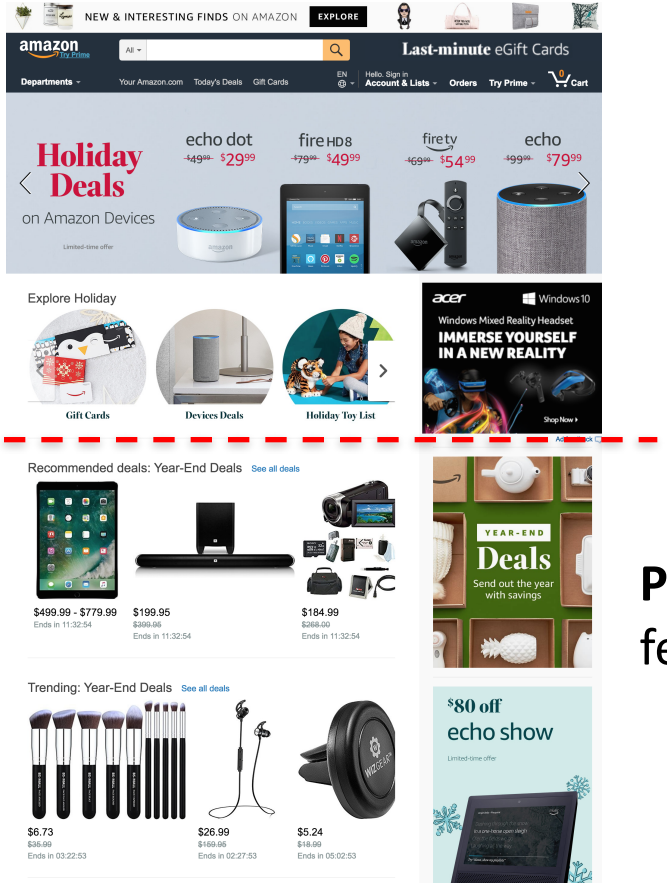
JavaScript heap

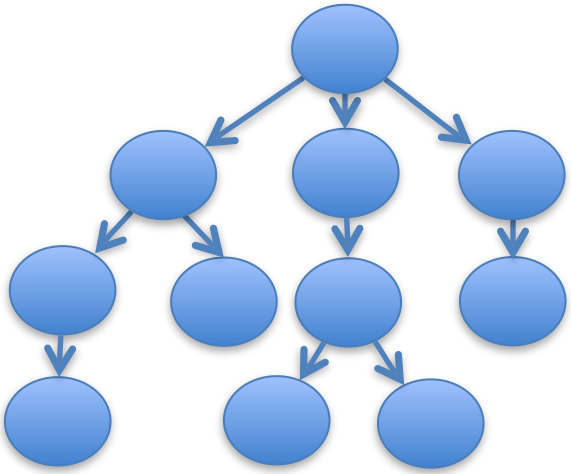
Page load time (PLT): time until all objects are fetched and evaluated

Existing Metrics

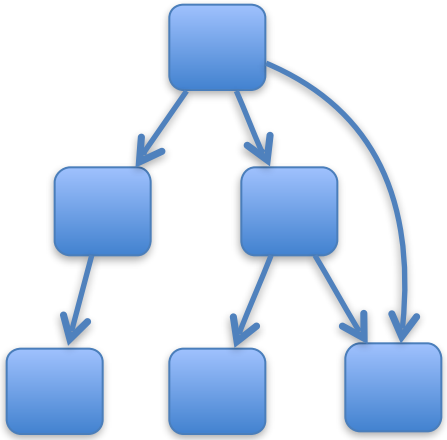
Above The Fold

Below The Fold





DOM Tree



JavaScript heap

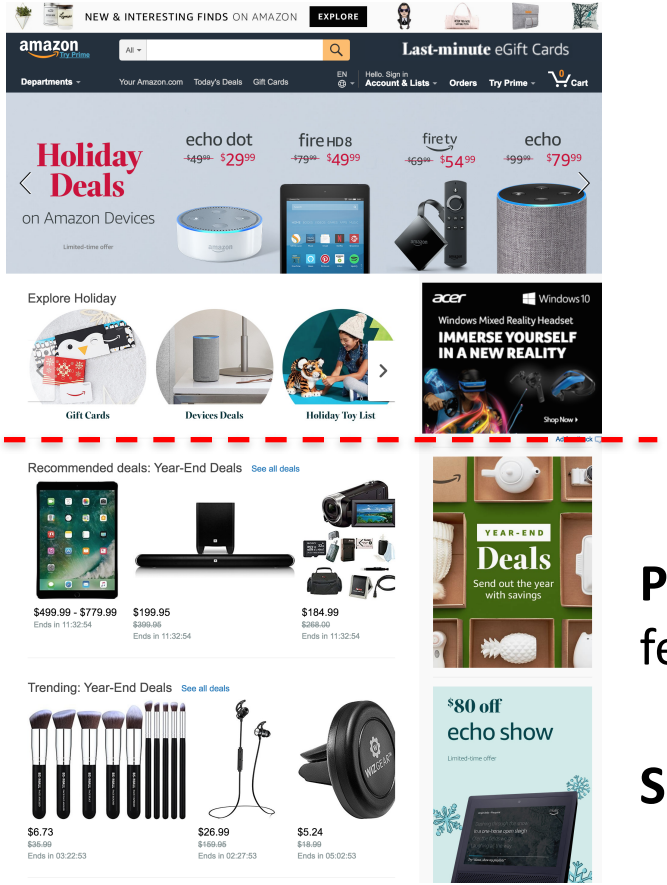
Page load time (PLT): time until all objects are fetched and evaluated

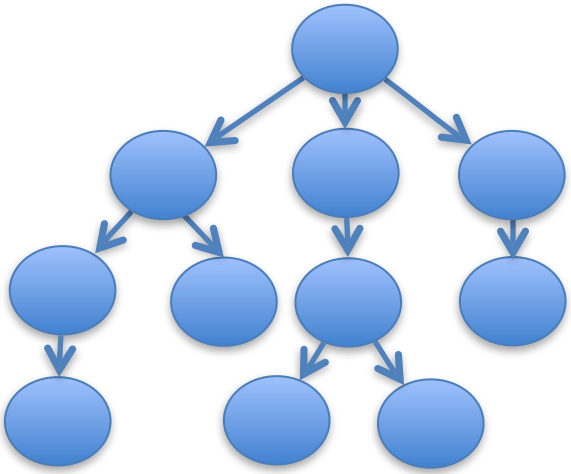
- Too conservative

Existing Metrics

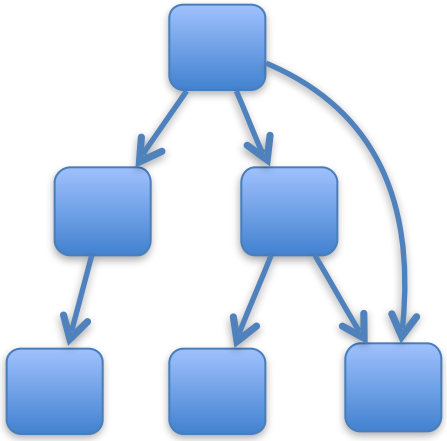
Above The Fold

Below The Fold





DOM Tree



JavaScript heap

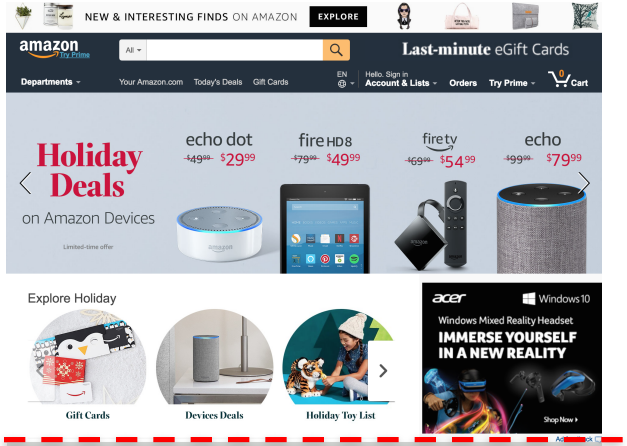
Page load time (PLT): time until all objects are fetched and evaluated

- Too conservative

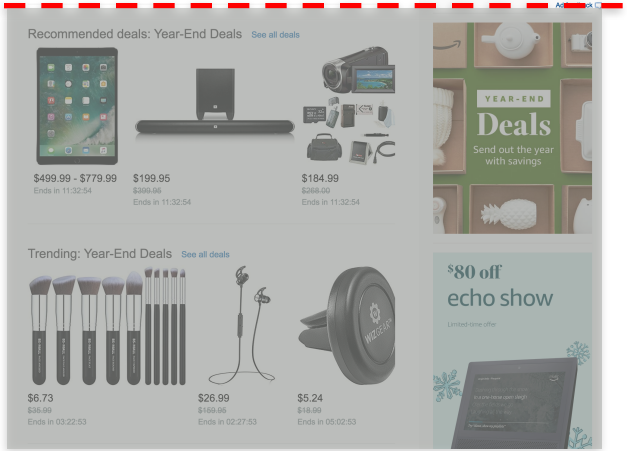
Speed Index (SI): time to render above-the-fold

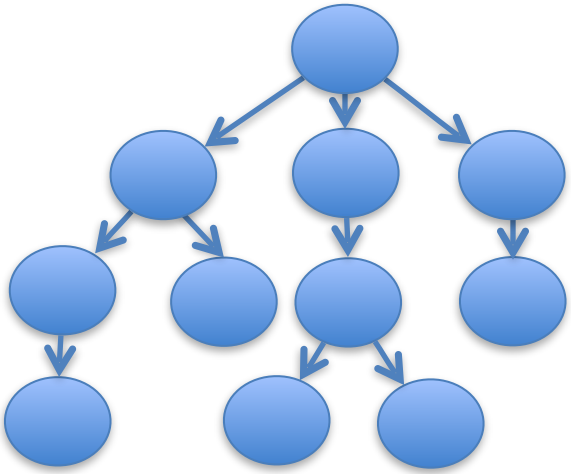
Existing Metrics

Above The Fold

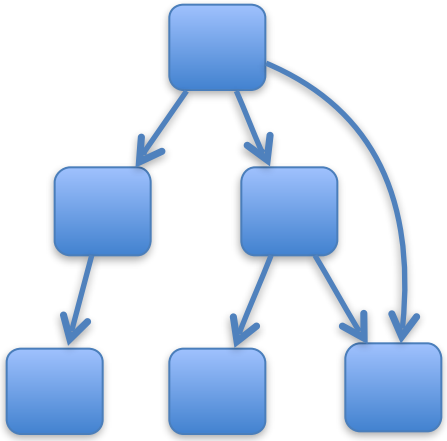


Below The Fold





DOM Tree



JavaScript heap

Page load time (PLT): time until all objects are fetched and evaluated

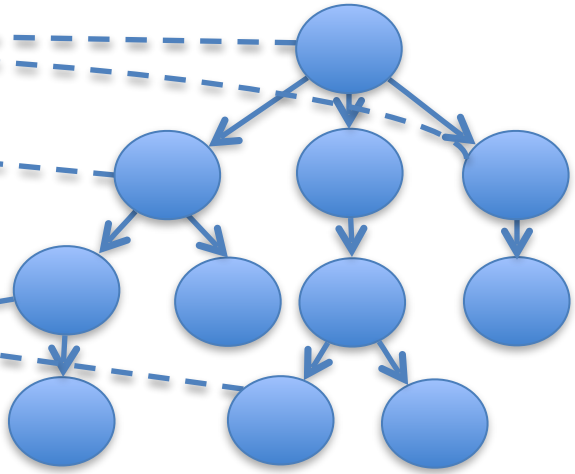
- Too conservative

Speed Index (SI): time to render above-the-fold

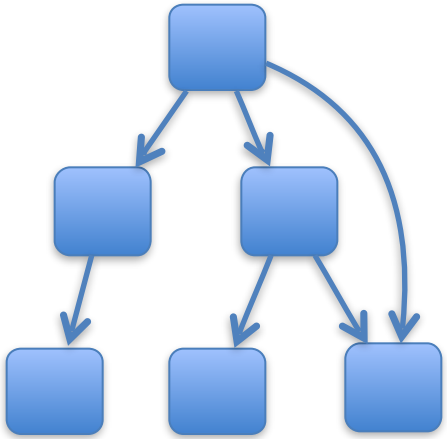
Existing Metrics

Above The Fold

Below The Fold



DOM Tree



JavaScript heap

Page load time (PLT): time until all objects are fetched and evaluated

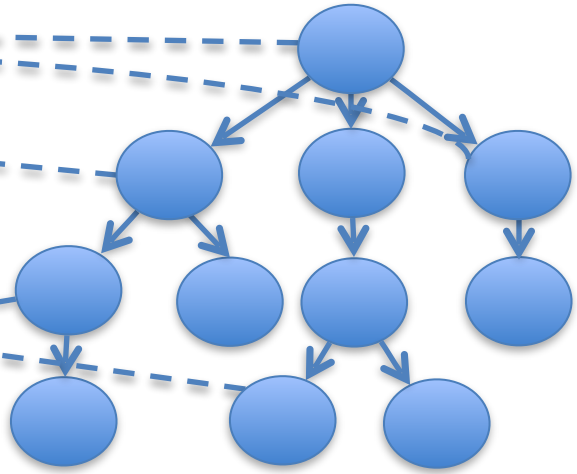
- Too conservative

Speed Index (SI): time to render above-the-fold

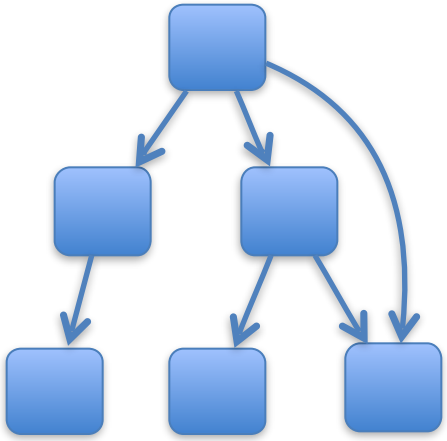
Existing Metrics

Above The Fold

Below The Fold



DOM Tree



JavaScript heap

Page load time (PLT): time until all objects are fetched and evaluated

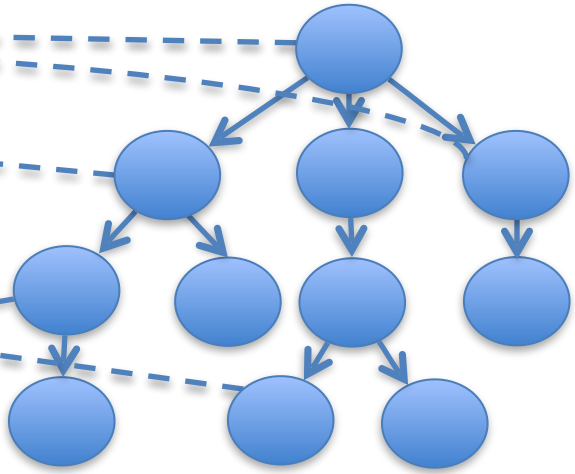
- Too conservative

Speed Index (SI): time to render above-the-fold

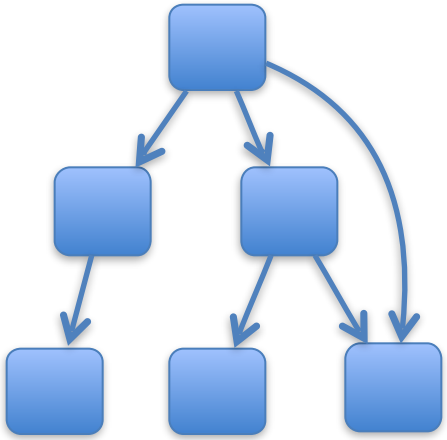
Existing Metrics

Above The Fold

Below The Fold



DOM Tree



JavaScript heap

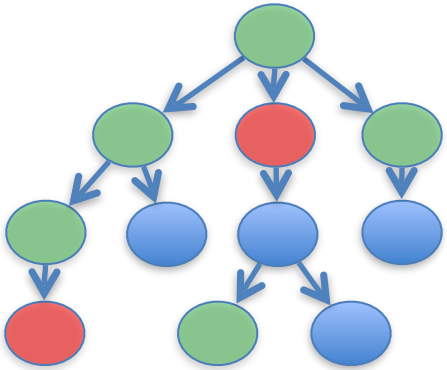
Page load time (PLT): time until all objects are fetched and evaluated

- Too conservative

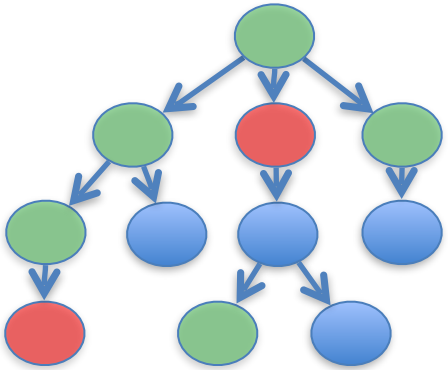
Speed Index (SI): time to render above-the-fold

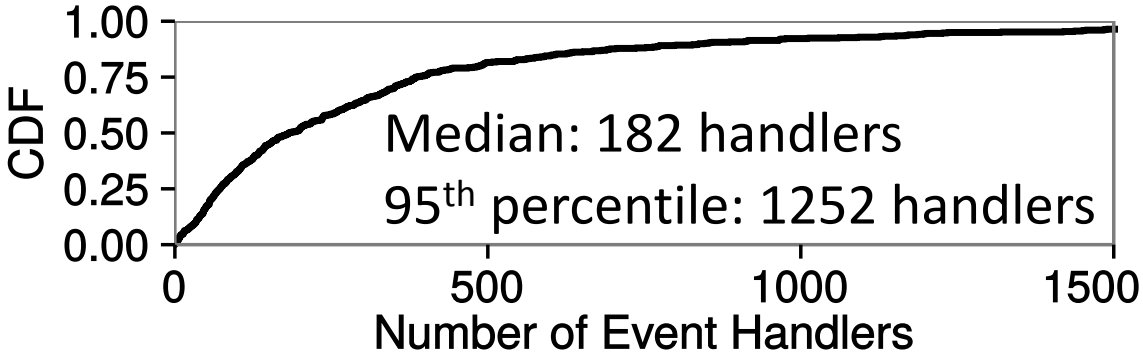
- Ignores JavaScript that supports functionality

Interactive page: above-the-fold is visually ready and fully functional

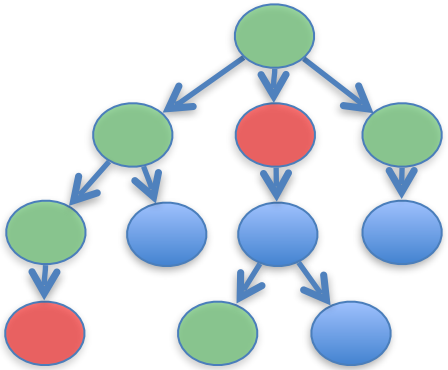


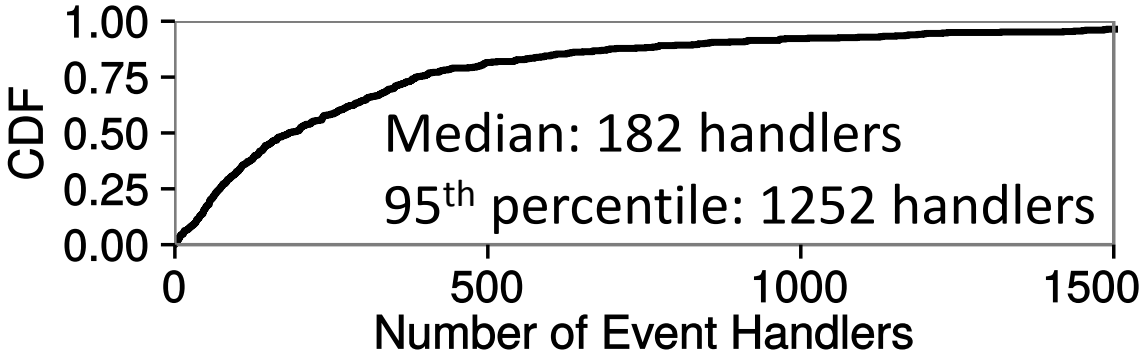
Interactive page: above-the-fold is visually ready and fully functional





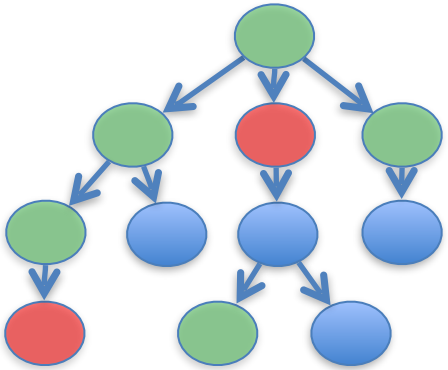
Interactive page: above-the-fold is visually ready and fully functional

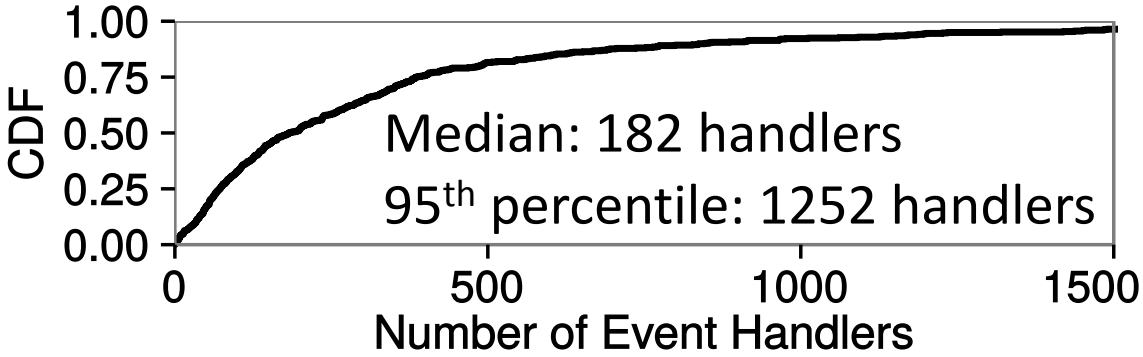




Challenge: nobody knows a good way to automatically identify that interactive state

Interactive page: above-the-fold is visually ready and fully functional





Challenge: nobody knows a good way to automatically identify that interactive state

1. Identify page's interactive state in DOM/JS
2. Analytically define rate at which state is visible and functional

Outline

- How pages load today
- Existing Metrics
- **Ready Index (RI) + Vesper**
- Evaluation

Ready Index

Ready Index

Functionality: $F(e,t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases}$

above-the-fold element

time when e's JavaScript handlers are registered, and state that handlers access when fired is loaded

Ready Index

Functionality: $F(e,t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases}$

- e = above-the-fold element
- t_e = time when e's handlers are registered, and state they access when fired is loaded

Visibility: $V(e,t) = \frac{|P_t(e)|}{|P(e)|}$

$|P_t(e)|$: e's paint events that are finished by time t

$|P(e)|$: paint events that affect e

Ready Index

Functionality: $F(e,t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases}$

Visibility: $V(e,t) = \frac{|P_t(e)|}{|P(e)|}$

Element Readiness: $R(e,t) = \frac{1}{2}F(e,t) + \frac{1}{2}V(e,t)$

- e = above-the-fold element
- t_e = time when e's handlers are registered, and state they access when fired is loaded
- P(e) = paint events that affect e
- $P_t(e)$ = e's paint events that are finished by time t

Ready Index

Functionality: $F(e,t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases}$

Visibility: $V(e,t) = \frac{|P_t(e)|}{|P(e)|}$

- e = above-the-fold element
- t_e = time when e's handlers are registered, and state they access when fired is loaded
- P(e) = paint events that affect e
- $P_t(e)$ = e's paint events that are finished by time t

Element Readiness: $R(e,t) = \frac{1}{2}F(e,t) + \frac{1}{2}V(e,t)$

Page Readiness: $R(t) = \sum_{e \in E} A(e)R(e,t)$

pixel area of e

Ready Index

Functionality: $F(e,t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases}$

Visibility: $V(e,t) = \frac{|P_t(e)|}{|P(e)|}$

- e = above-the-fold element
- t_e = time when e's handlers are registered, and state they access when fired is loaded
- P(e) = paint events that affect e
- $P_t(e)$ = e's paint events that are finished by time t

Element Readiness: $R(e,t) = \frac{1}{2}F(e,t) + \frac{1}{2}V(e,t)$

Page Readiness: $R(t) = \sum_{e \in E} A(e)R(e,t)$

- A(e) = pixel area of e

Ready Index: $RI = \int_0^T 1 - \frac{R(t)}{R(T)}\,d(t)$

loose upper bound on load time

Ready Index

Functionality: $F(e,t) = \begin{cases} 0 & t < t_e \\ 1 & t \geq t_e \end{cases}$

Visibility: $V(e,t) = \frac{|P_t(e)|}{|P(e)|}$

- e = above-the-fold element
- t_e = time when e's handlers are registered, and state they access when fired is loaded
- P(e) = paint events that affect e
- $P_t(e)$ = e's paint events that are finished by time t

Element Readiness: $R(e,t) = \frac{1}{2}F(e,t) + \frac{1}{2}V(e,t)$

Page Readiness: $R(t) = \sum_{e \in E} A(e)R(e,t)$

- A(e) = pixel area of e

Ready Index: $RI = \int_0^T 1 - \frac{R(t)}{R(T)}\,d(t)$

- T = loose upper bound on load time

Ready Time (RT): smallest time when all above-the-fold elements are ready

Measuring Ready Index (RI)

Measuring Ready Index (RI)

- Need to know:

Visible elements and their event handlers

State that handlers access when fired

Effect and timing of browser paint events

Measuring Ready Index (RI)

- Need to know:

Visible elements and their event handlers

State that handlers access when fired

Effect and timing of browser paint events

- Requirements for instrumentation:

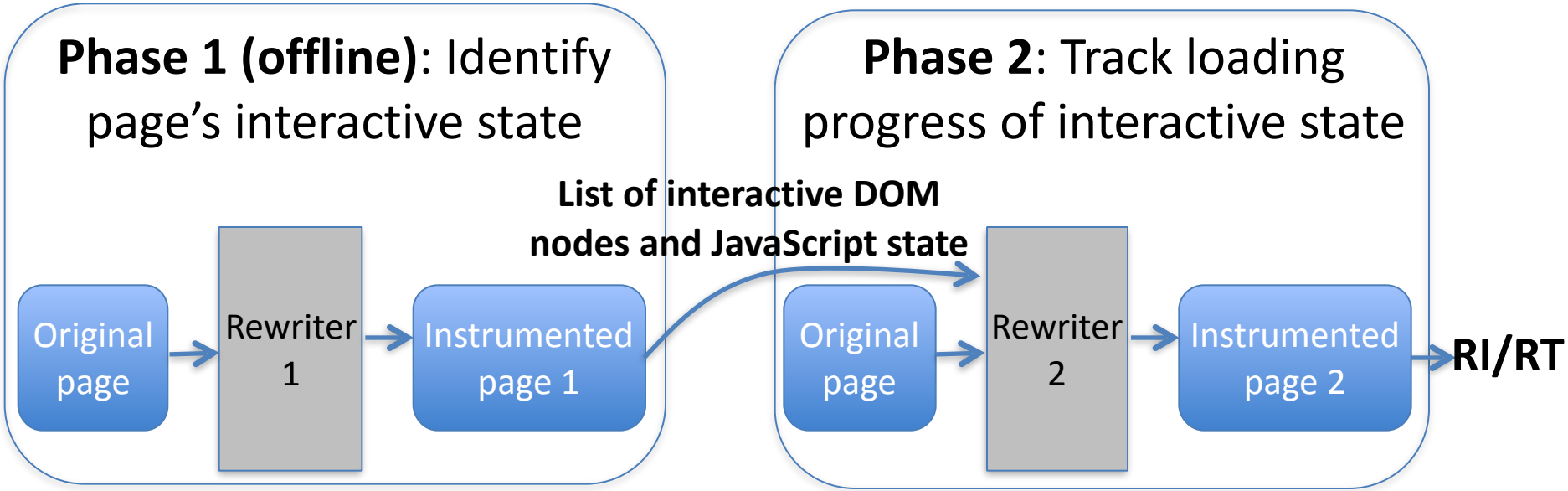
No developer annotations

Low overhead

Generic

Vesper: Overview

Approach: Use two measurement phases to reduce impact of instrumentation



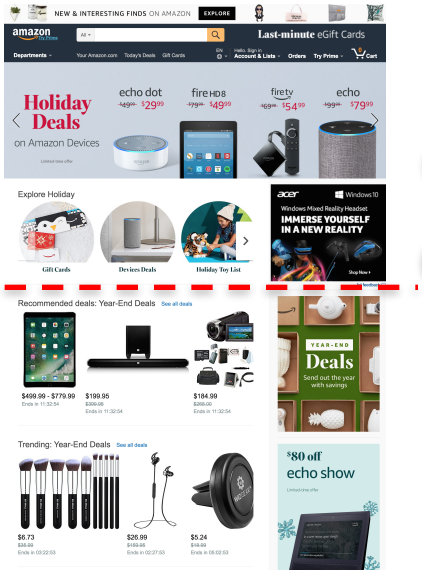
Vesper: Phase 1

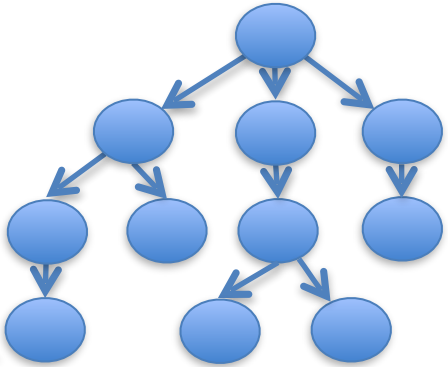
Goal: Identify visible elements, event handlers, and the state handlers access when fired

Vesper: Phase 1

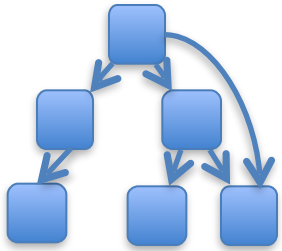
Goal: Identify visible elements, event handlers, and the state handlers access when fired

Above The Fold





DOM Tree

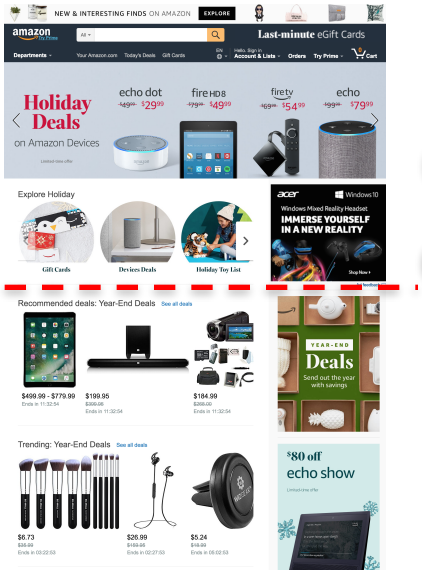


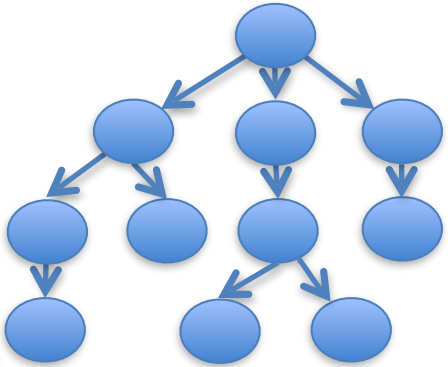
JavaScript heap

Vesper: Phase 1

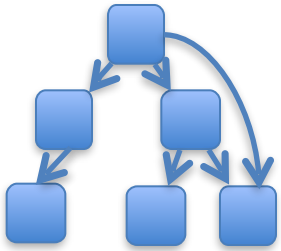
Goal: Identify visible elements, event handlers, and the state handlers access when fired

Above The Fold





DOM Tree



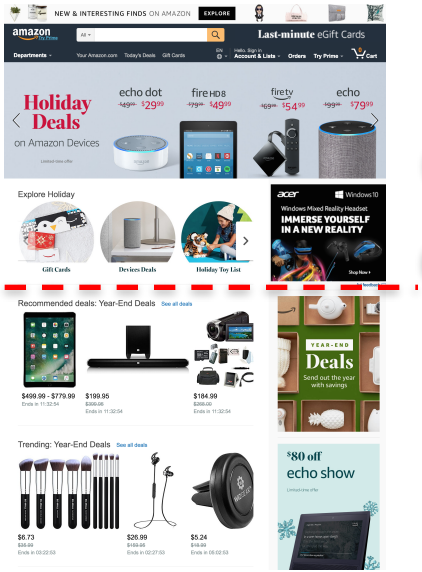
JavaScript heap

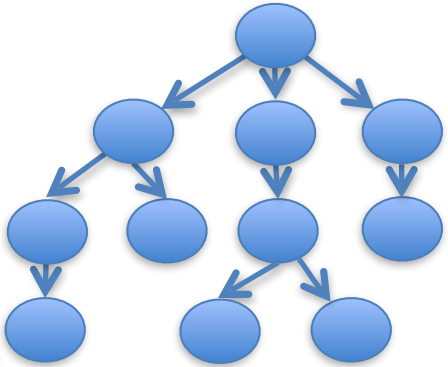
Element visibility: analyze element bounding boxes and CSS rules

Vesper: Phase 1

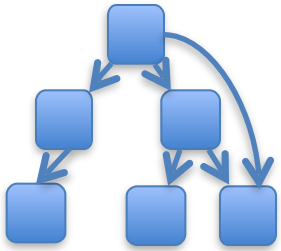
Goal: Identify visible elements, event handlers, and the state handlers access when fired

Above The Fold





DOM Tree



JavaScript heap

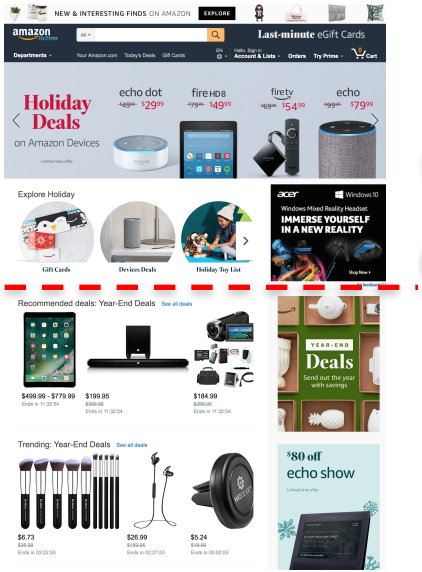
Element visibility: analyze element bounding boxes and CSS rules

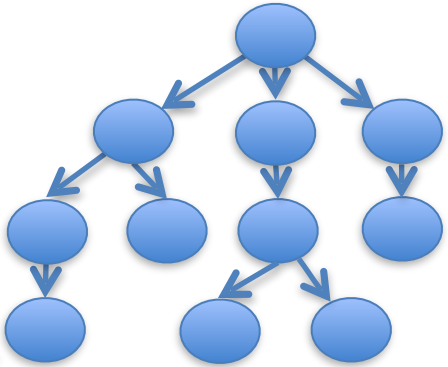
Logging event handlers: shim event handler registration mechanisms

Vesper: Phase 1

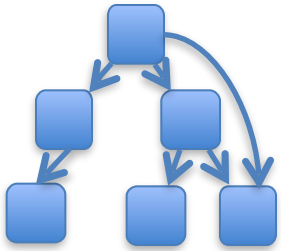
Goal: Identify visible elements, event handlers, and the state handlers access when fired

Above The Fold





DOM Tree



JavaScript heap

Element visibility: analyze element bounding boxes and CSS rules

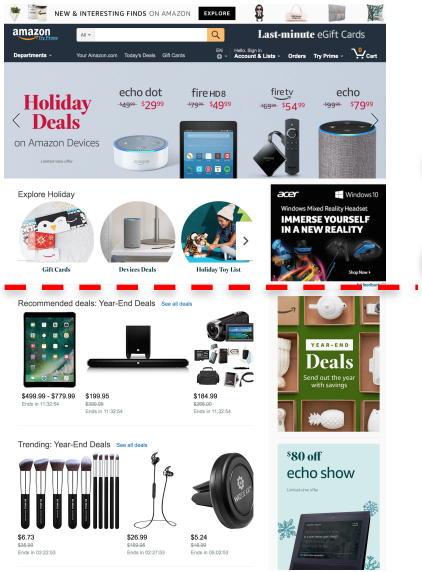
Logging event handlers: shim event handler registration mechanisms

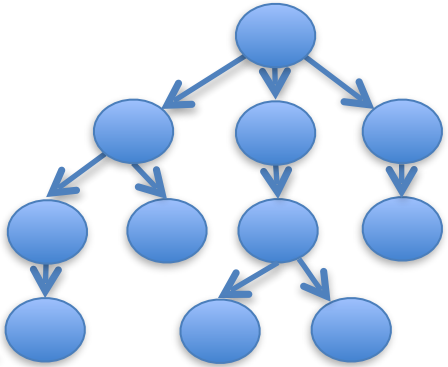
Event handler state: fire handlers and log accessed state with Scout

Vesper: Phase 1

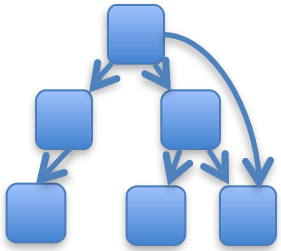
Goal: Identify visible elements, event handlers, and the state handlers access when fired

Above The Fold





DOM Tree



JavaScript heap

Element visibility: analyze element bounding boxes and CSS rules

Logging event handlers: shim event handler registration mechanisms

Event handler state: fire handlers and log accessed state with Scout

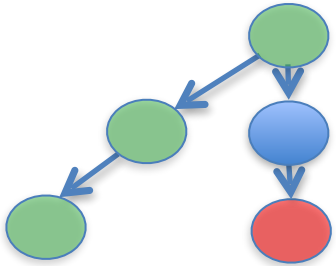
Phase 1: 4.5% overhead

Vesper: Phase 2

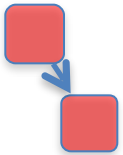
Goal: Track loading progress of interactive state from Phase 1

Vesper: Phase 2

Goal: Track loading progress of interactive state from Phase 1



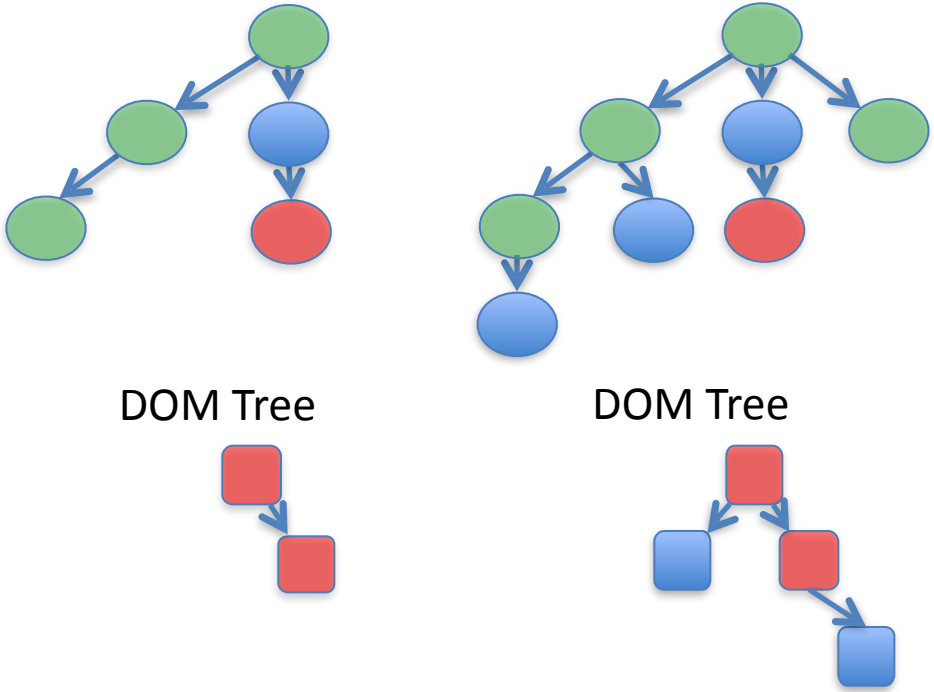
DOM Tree



JavaScript heap

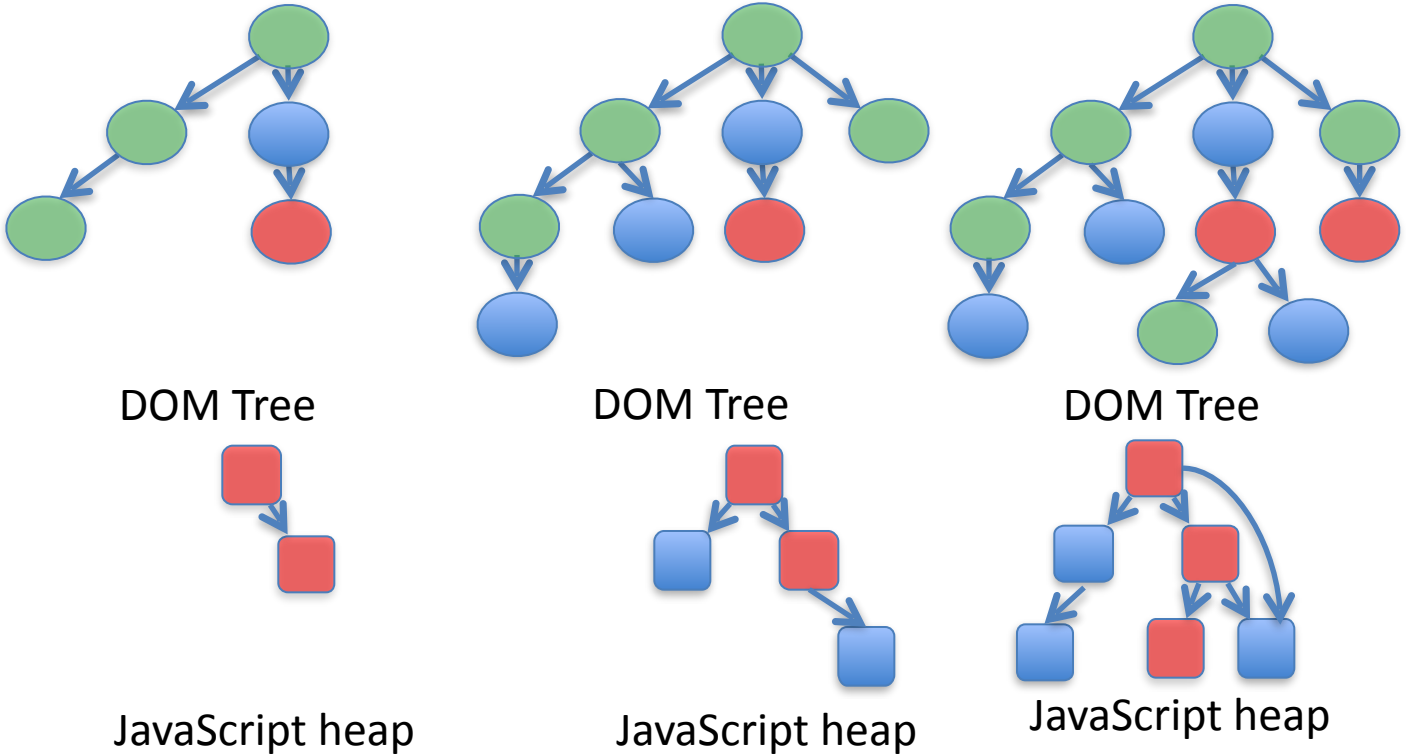
Vesper: Phase 2

Goal: Track loading progress of interactive state from Phase 1



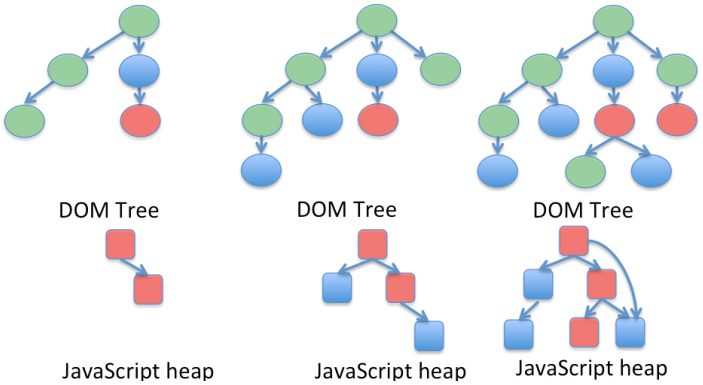
Vesper: Phase 2

Goal: Track loading progress of interactive state from Phase 1



Vesper: Phase 2

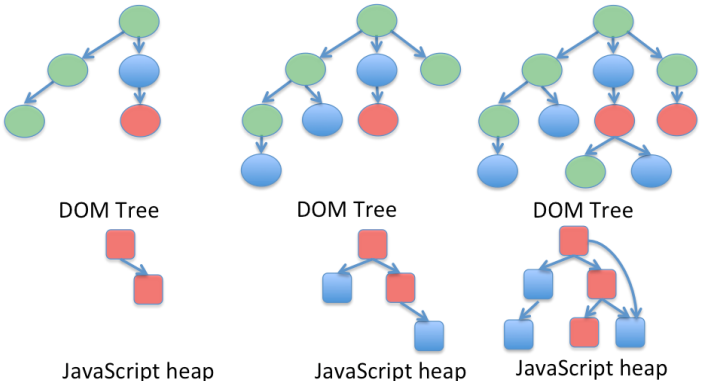
Goal: Track loading progress of interactive state from Phase 1



Vesper: Phase 2

Goal: Track loading progress of interactive state from Phase 1

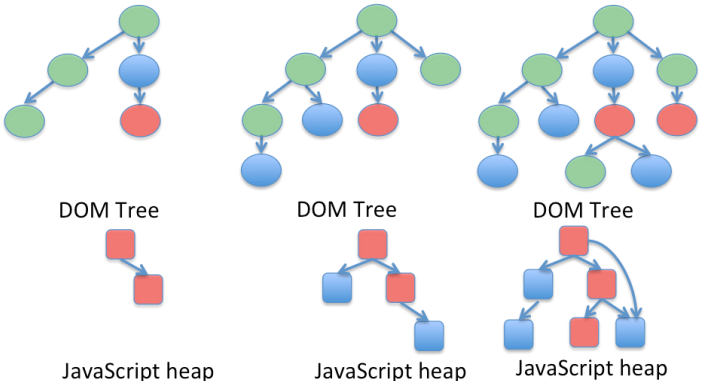
- Log “last writes” for DOM/heap state



Vesper: Phase 2

Goal: Track loading progress of interactive state from Phase 1

- Log “last writes” for DOM/heap state

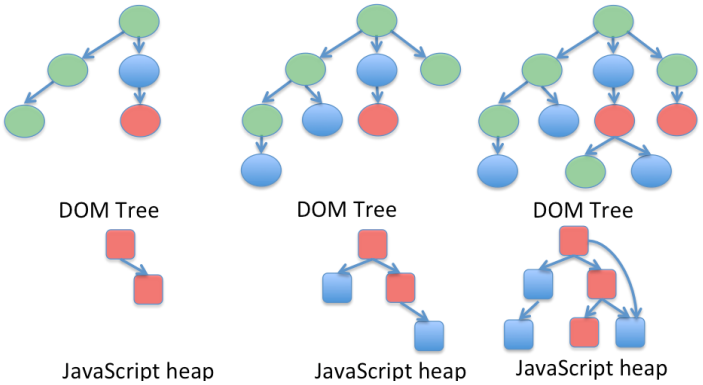



```
var x = 50; var y = 0;
while (y < 50) {
    x = x + 1;
    y = y + 1;
}
x = x + 5;
```

Vesper: Phase 2

Goal: Track loading progress of interactive state from Phase 1

- Log “last writes” for DOM/heap state

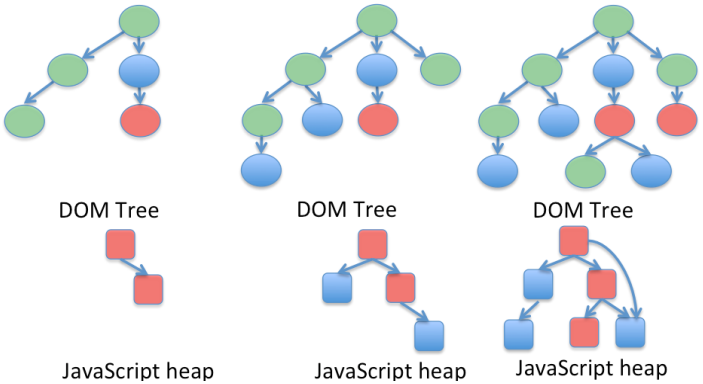



```
var x = 50; var y = 0;
while (y < 50) {
    x = x + 1;
    y = y + 1;
    if (y == 49) {
        vesper_log(y);
    }
}
x = x + 5;
vesper_log(x);
```

Vesper: Phase 2

Goal: Track loading progress of interactive state from Phase 1

- Log “last writes” for DOM/heap state
- Track browser layout/paint events

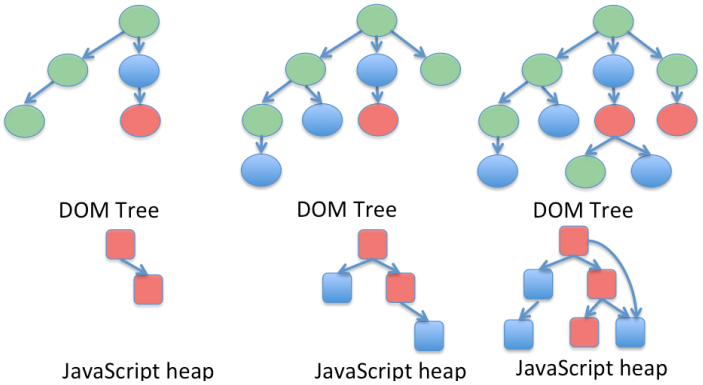



```
var x = 50; var y = 0;
while (y < 50) {
   x = x + 1;
   y = y + 1;
   if (y == 49) {
      vesper_log(y);
   }
}
x = x + 5;
vesper_log(x);
```

Vesper: Phase 2

Goal: Track loading progress of interactive state from Phase 1

- Log “last writes” for DOM/heap state
- Track browser layout/paint events



Phase 2: 1.9% overhead

```
var x = 50; var y = 0;
while (y < 50) {
   x = x + 1;
   y = y + 1;
   if (y == 49) {
      vesper_log(y);
   }
}
x = x + 5;
vesper_log(x);
```

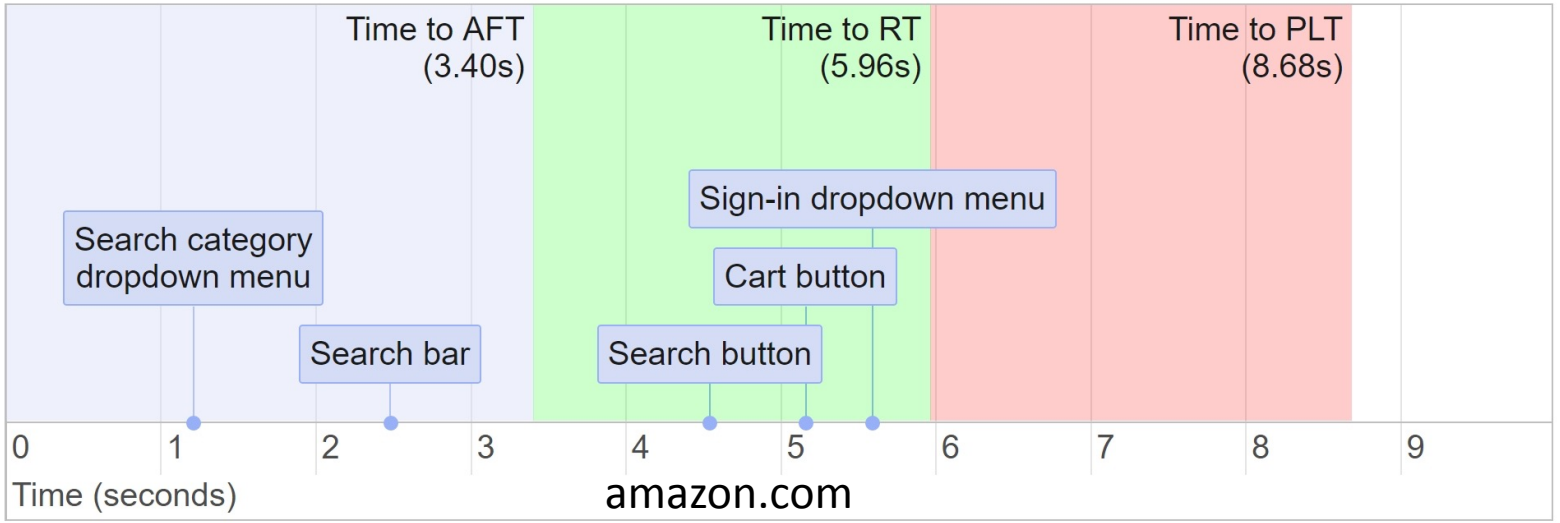
Outline

- How pages load today
- Existing Metrics
- Ready Index (RI) + Vesper
- **Evaluation**

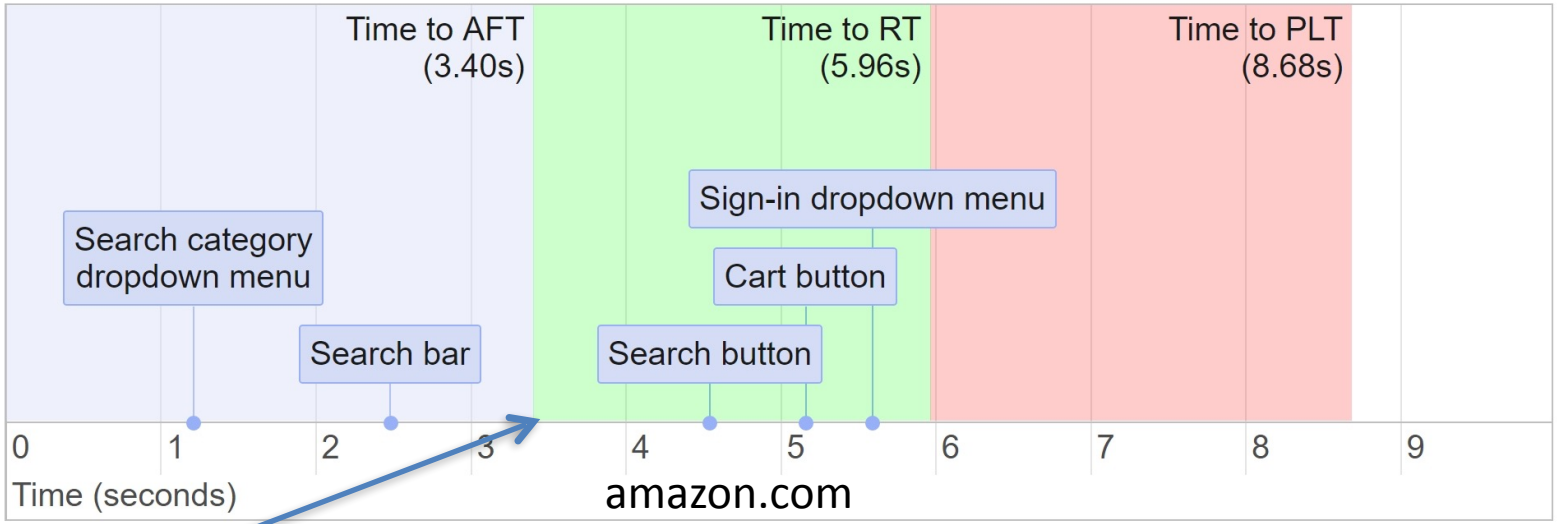
Evaluation Outline

- Are there differences between Ready Index and existing metrics?
- Can we optimize a page load for Ready Index?
- How well does Ready Index capture user experience?

AFT vs. RT vs. PLT

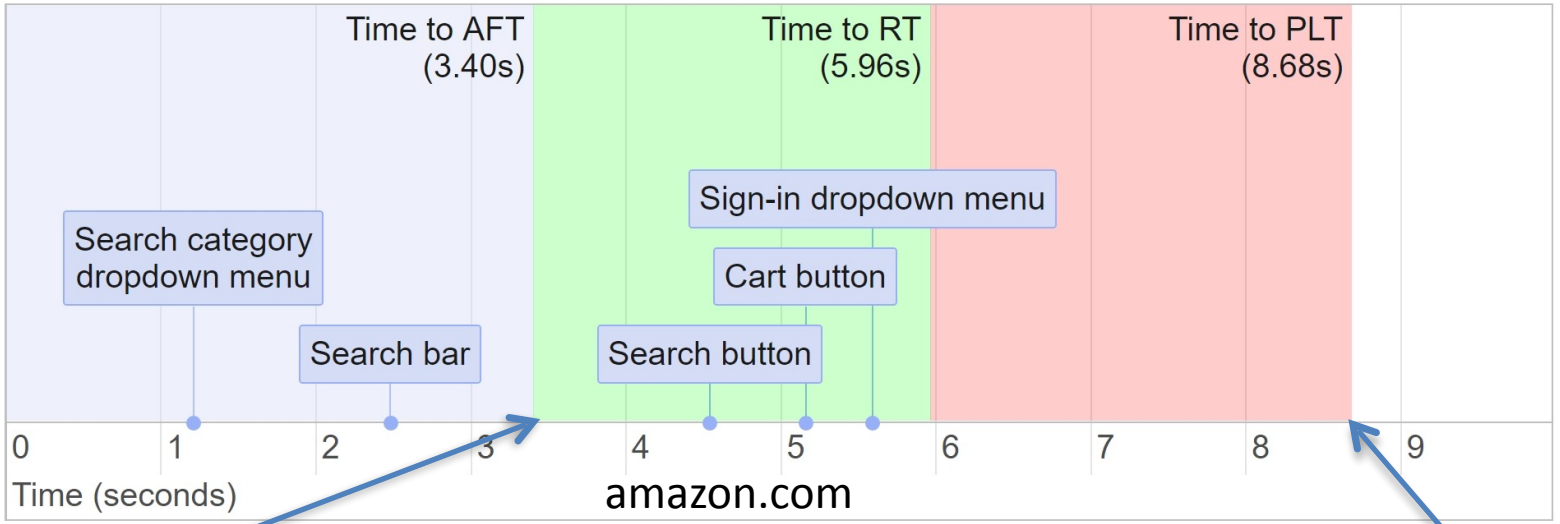


AFT vs. RT vs. PLT



Above-the-fold time (AFT) underestimates ‘interactive time’ by 2.56 seconds!

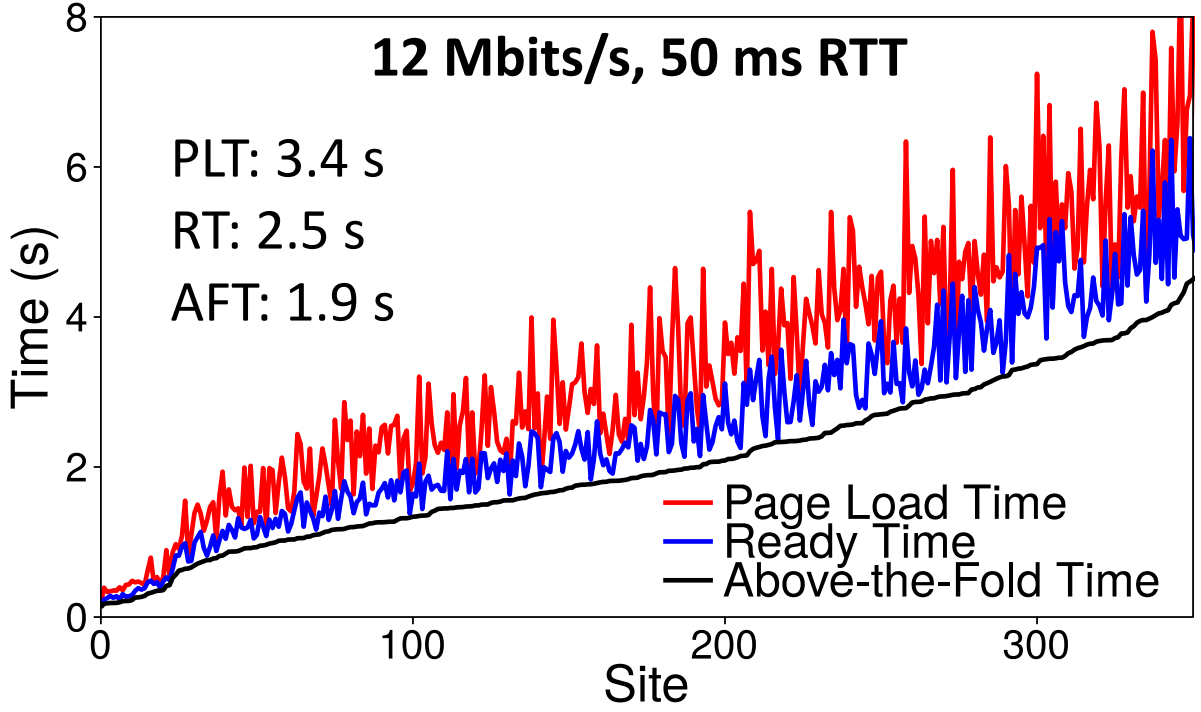
AFT vs. RT vs. PLT



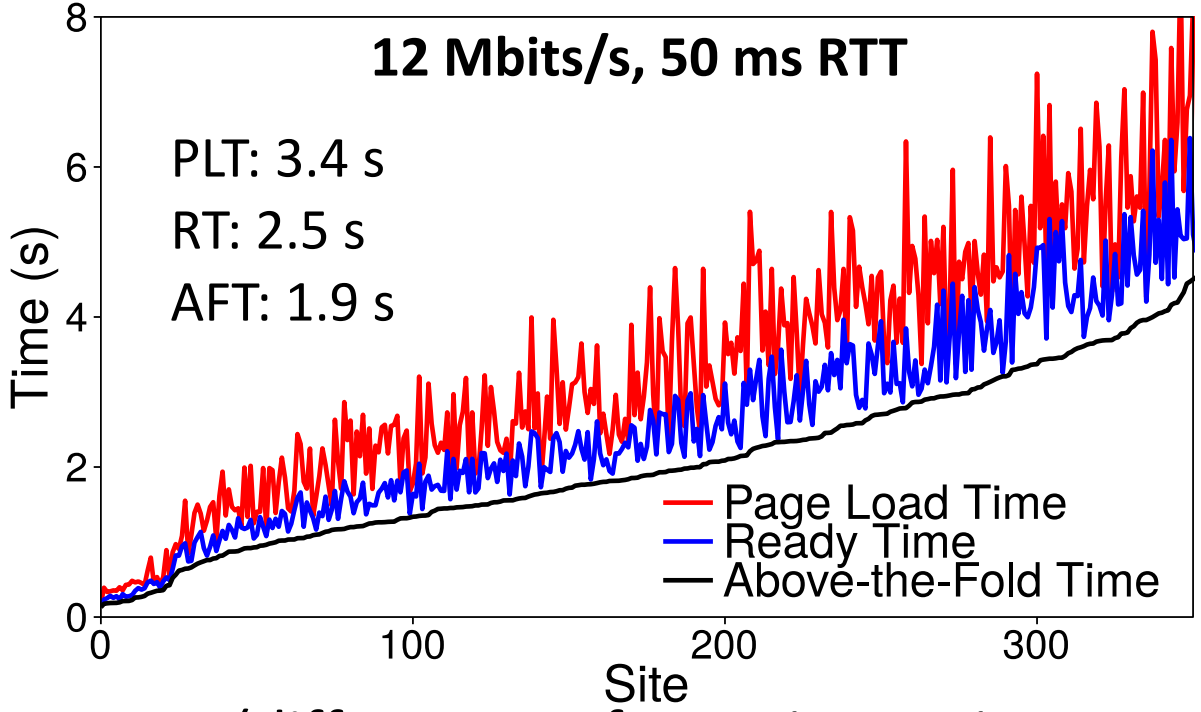
Above-the-fold time (AFT) underestimates 'interactive time' by 2.56 seconds!

Page load time (PLT) overestimates 'interactive time' by 2.72 seconds!

350 Popular Sites: AFT vs. RT vs. PLT

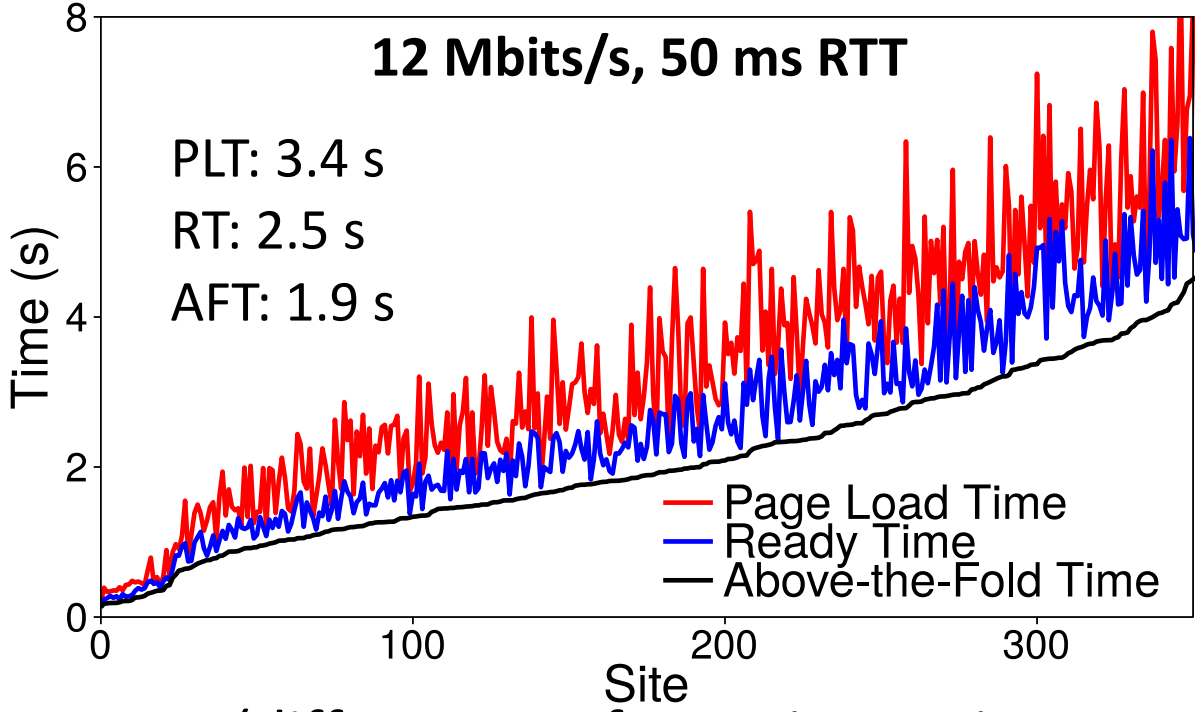


350 Popular Sites: AFT vs. RT vs. PLT



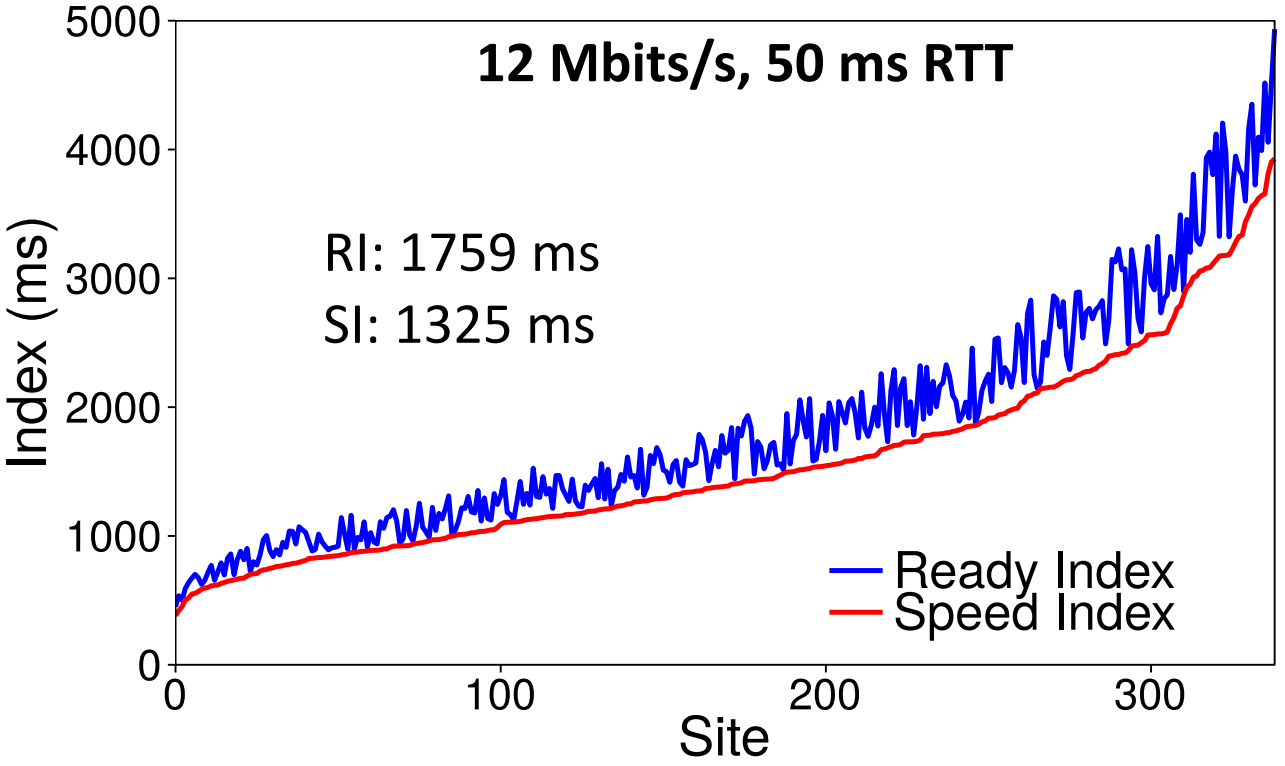
- PLT > RT > AFT (differences of 24.0%-64.3%, 0.3-3.6 seconds)

350 Popular Sites: AFT vs. RT vs. PLT



- PLT > RT > AFT (differences of 24.0%-64.3%, 0.3-3.6 seconds)
- Differences increase as RTTs increase

350 Popular Sites: SI vs. RI

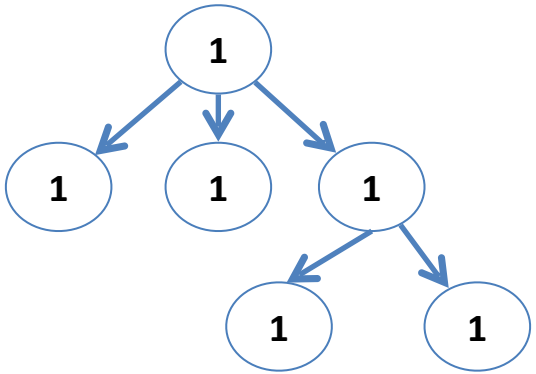


Optimizing for Ready Index

- Vesper: identify objects of importance
- Polaris: optimize loading of important objects
  - Dependency-aware request scheduler that uses dynamic critical path analysis to reduce page load times

Optimizing for Ready Index

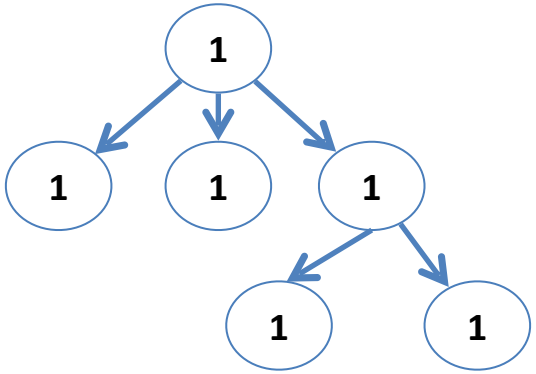
- Vesper: identify objects of importance
- Polaris: optimize loading of important objects
  - Dependency-aware request scheduler that uses dynamic critical path analysis to reduce page load times



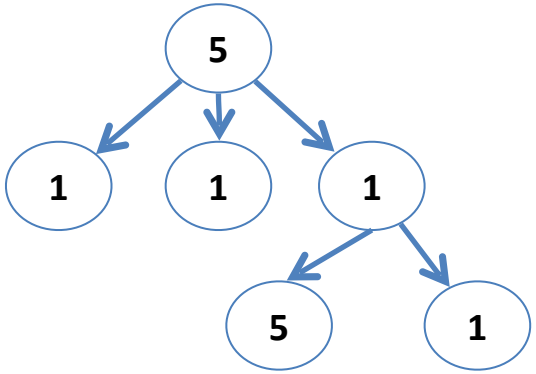
Polaris_PLT

Optimizing for Ready Index

- Vesper: identify objects of importance
- Polaris: optimize loading of important objects
  - Dependency-aware request scheduler that uses dynamic critical path analysis to reduce page load times



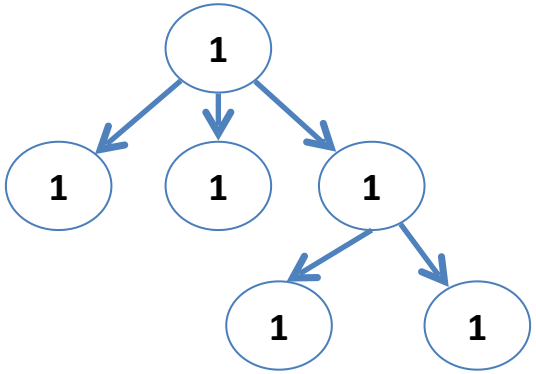
Polaris_PLT



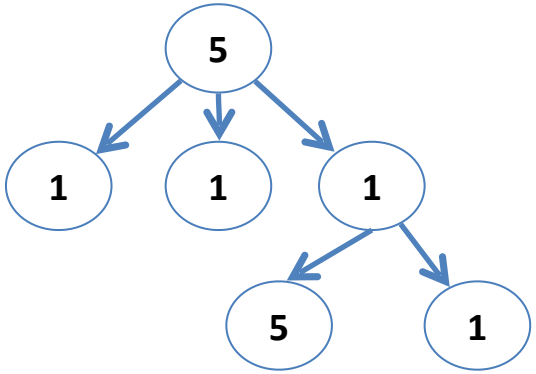
Polaris_SI

Optimizing for Ready Index

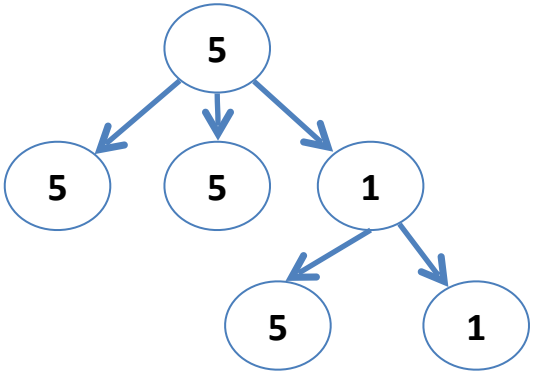
- Vesper: identify objects of importance
- Polaris: optimize loading of important objects
  - Dependency-aware request scheduler that uses dynamic critical path analysis to reduce page load times



Polaris_PLT



Polaris_SI



Polaris_RI

Optimization Results

12 Mbits/s, 100 ms

Weight	PLT	RI	SI
Polaris-PLT	**36%**	**8%**	**-7%**
Polaris-RI	**23%**	**29%**	**12%**
Polaris-SI	**10%**	**14%**	**18%**

Optimization Results

12 Mbits/s, 100 ms

Weight	PLT	RI	SI
Polaris-PLT	**36%**	**8%**	**-7%**
Polaris-RI	**23%**	**29%**	**12%**
Polaris-SI	**10%**	**14%**	**18%**

Targeted metrics improve the most!

User Study 1: Interactivity

- Perform interactive task with Polaris-PLT, Polaris-RI, Polaris-SI: which is fastest?
- 5 sites, 85 users

User Study 1: Interactivity

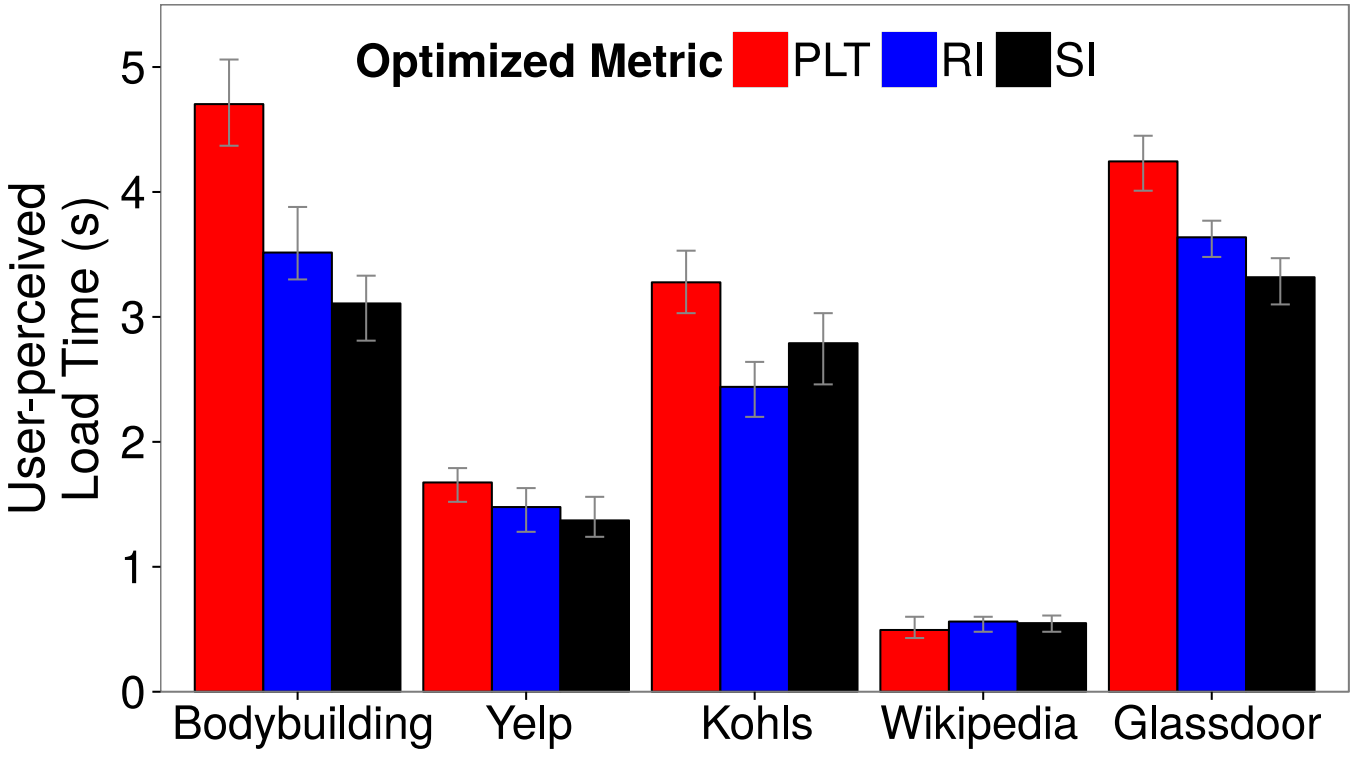
- Perform interactive task with Polaris-PLT, Polaris-RI, Polaris-SI: which is fastest?
- 5 sites, 85 users

Scheduling policy	Preference percentage
Polaris-RI	**83%**
Polaris-SI	**4%**
Polaris-PLT	**7%**
None	**6%**

Takeaway: interactive users strongly prefer pages optimized for RI

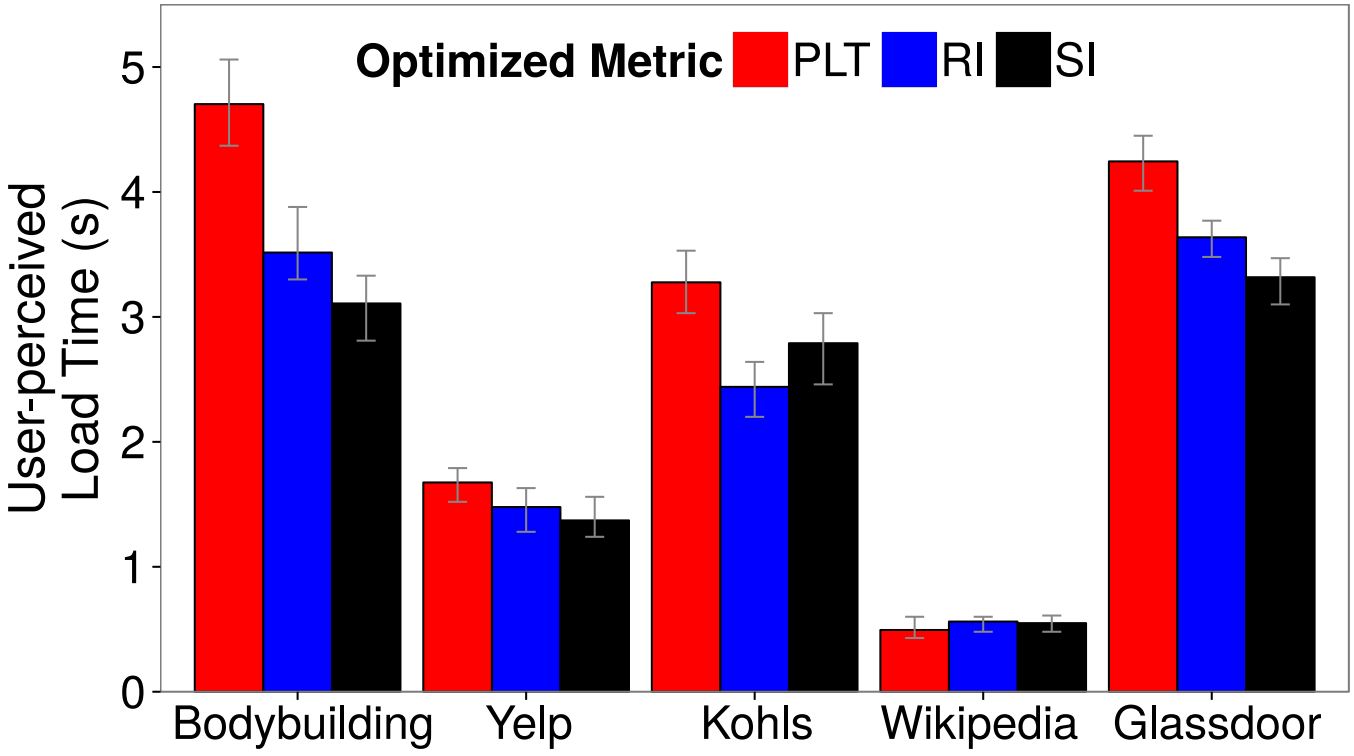
User Study 2: Rendering

- 15 sites, 73 users



User Study 2: Rendering

- 15 sites, 73 users



Takeaway: Polaris-SI is best for rendering, but Polaris-RI is comparable

Conclusion

- Existing web performance metrics ignore page interactivity
  - Over or underestimate time-to-interactivity by 24%-64%
- Ready Index (RI): analytical definition of page time-to-interactivity
- Vesper: system to automatically measure RI by identifying and tracking loading of page's interactive state
  - Helps reduce time-to-interactivity by 32%