Web Performance

```
> 3s load time is not good, 53 % of AR
```

```
500ms speed delay => 26% user frustation
```

```
=> -20% google.com traffic
```

```
lower CR , frustated users, less productive => issue with web performance
```

Web Performance

wpo:

```
web development approach for measuring, optimizing, and improving
web performance and user's perception/satisfaction using best practices
and techniques, essentially increasing conversion and lowering the abandonment
```

BR	Seconds
+32%	1-3
+90%	1-5
+106%	1-6
+123%	1-10

2s page load => 74% CR

Web Performance

WPO in Enterprises

no CR

no AR

targetted audience

ux, satisfaction, server load, productivity

ux
__

not sell pdts.

not index in search engines

not all ntks. and devices are covered

ntk
___

VPNs

Qos

Saturated

SPAs

PWAs

Web Performance

- Performance Goals
  - RAIL Model
    - Response 100ms
    - Animation 10ms
    - Idle 50ms
    - Load 1s

Web Performance

`Perceptual Speed Index`

`WebPagetest tool in 2012`

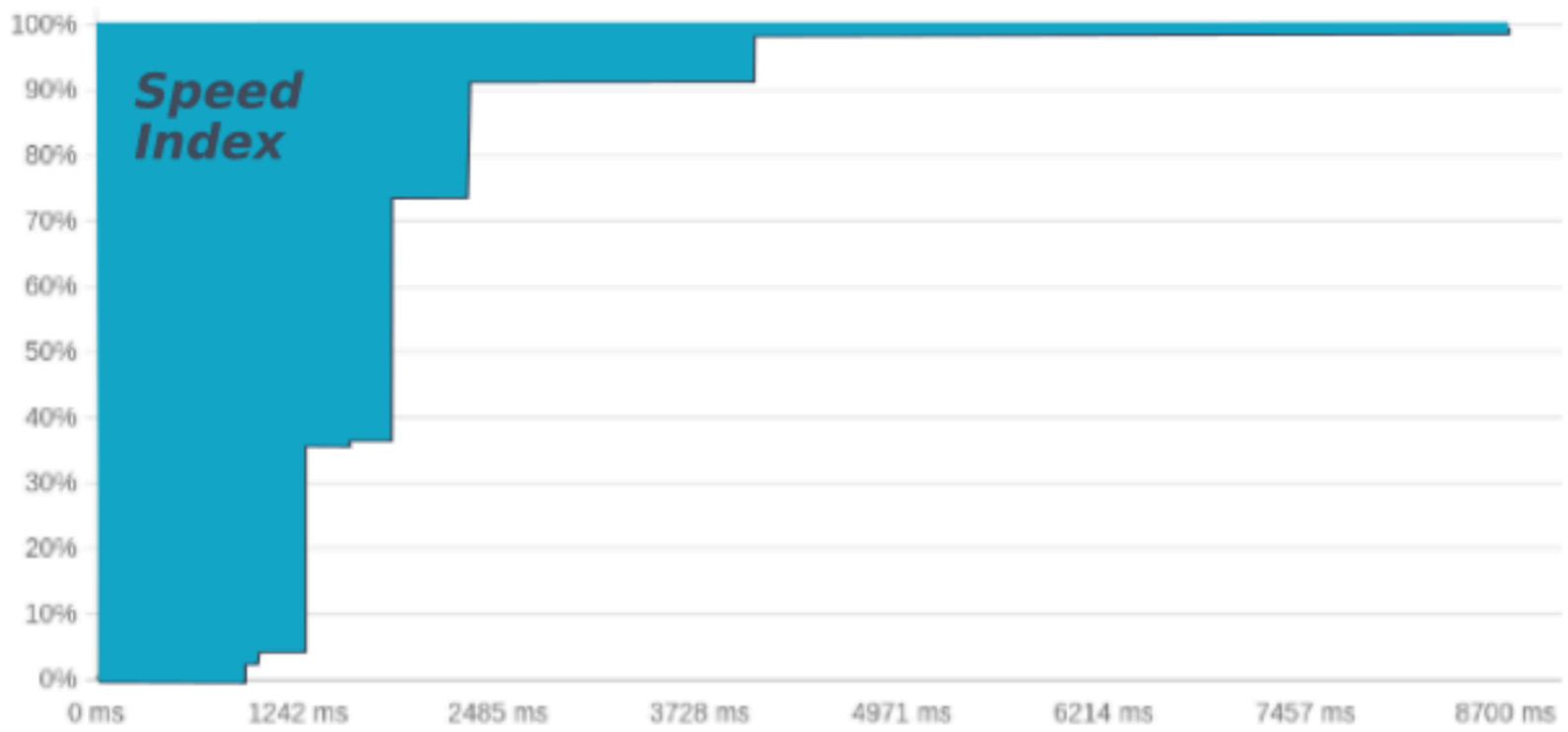
`ATF content`

`blank amount the user is seeing on the screen`

https://blog.dareboost.com/en/2018/02/speed-index-web-performance/

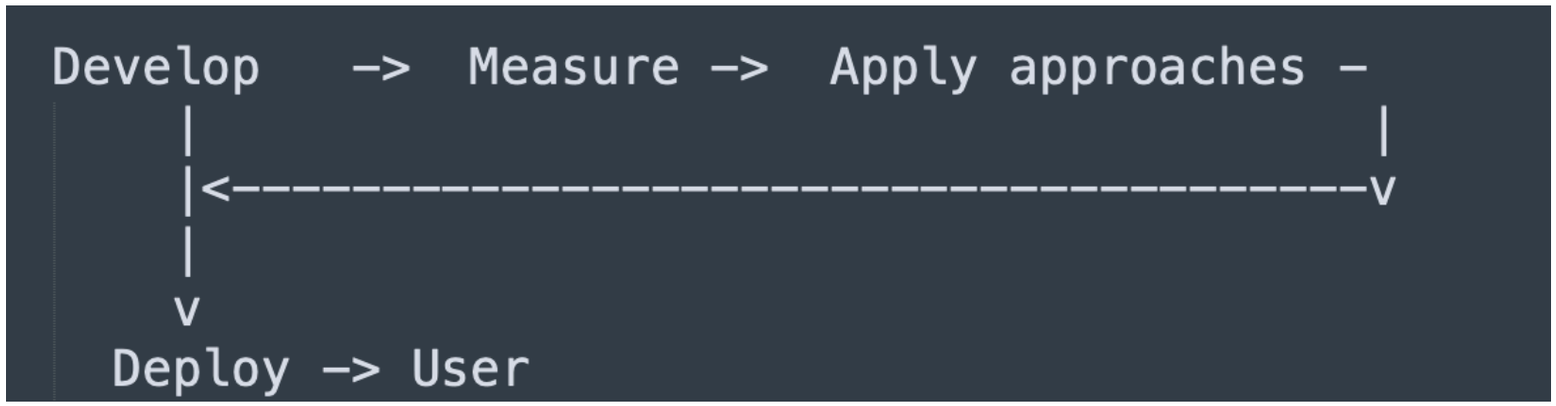
Web Performance

Perceptual Speed Index

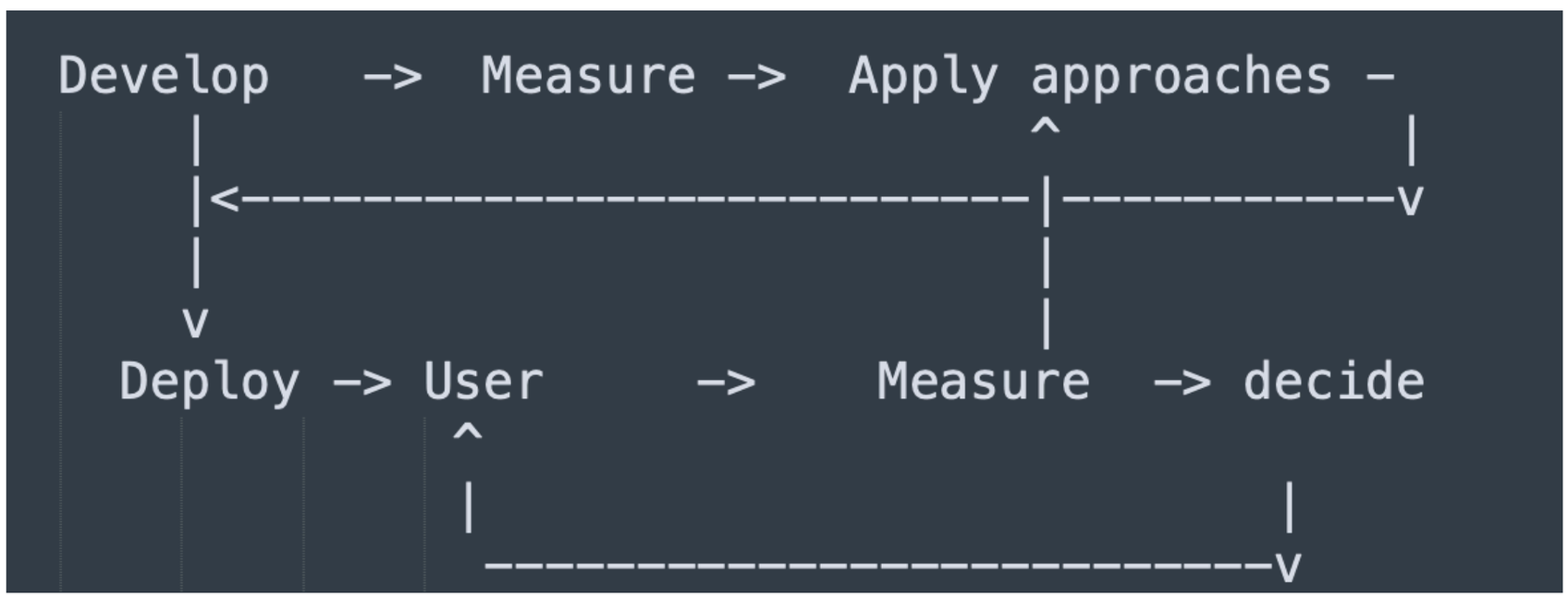


Web Performance

How WPO works



user's context (4G,2G, WiFi, ..., mobile, roaming ..) effects web performance



Web Performance

```
RUM (Real User Monitoring)
```

```
reactive web performance techniques
```

```
from : "You are in a poor connectivity area or in a remote area..I'm helpless"
```

```
context always keeps on changing
```

```
to: "Let me figure it out as i am responsible for ur experience of my site.. "
```

Web Performance

Measure Time

```
const ts1 = new Date().getTime()
//some task
const ts2 = new Data().getTime()

const taskTime = t2 - t1
```

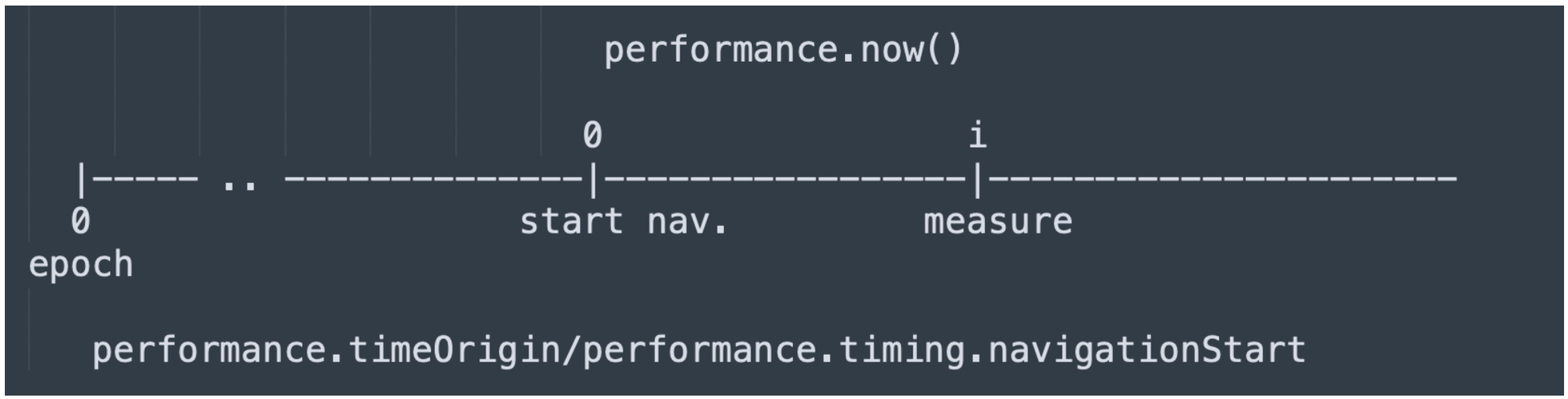
```
const ts1 = performance.now()
//some task
const ts2 = performance.now()
const taskTime = t2 - t1
```

global option

use zero as the moment in which the browser started the navigation to that page. measure is relative to that page load

Web Performance

Measure Time



JS data type: DOMHighResTimeStamp

Web Performance

Navigation Timining API

global object

available in all browsers

API is the base for other measurement specs

also abl. in web workers

3 functionalities:

performance.now()

```
const current = performance.now() //timestamp from the timeOrigin
const zeroMoment = performance.timeOrigin //ms
```

Web Performance

`Navigation Timining API`

`Navigation type`

```
//level1
 const redirects = performance.navigation.redirectCount
```

```
switch (performance.navigtion.type) {
    case: performance.navigation.TYPE_NAVIGATE //normal navigation
    case: performance.navigation.TYPE_RELOAD
    case: performance.navigation.TYPE_BACK_FORWARD

}
```

Web Performance

```
Navigation Timining API
```

```
get timings for the current navigation
```

```
list of timings
```

```
const allNT = performance.timing
```

```
const domNT =  performance.timing.domInteractive
```

Web Performance

```
Navigation Timining API
```

```
const allNT = performance.getEntriesByType("navigation")
```

```
const domNT = allNT[0].domInteractive
```

```
new props:

  - redirectCount
  - type
  - nextHopProtocol (HTTP2 ..)
  - transferSize (bytes of the whole response including headers),
    encodedBodySize (transfer package without headers), decodedBodySize (bytes after decoding)

  - workerStart: 0 //if no service worker, or timestamp before fetch event in SW
```

Web Performance

`Navigation Timining API`

```
prepare -> DNS -> TCP -> SSL -> HTTP Request -> Server Wait -> HTTP Response -
-> HTML Parsing -> CSS/JS Parsing -> Images ..
```

`prepare`

```
navigationStart, unloadEventStart, unloadEventEnd,
redirectStart, redirectEnd, fetchStart
```

`DNS`

```
fetchStart, domainLookupStart,
domainLookupEnd, connectStart
```

Web Performance

Navigation Timining API

TCP

connectStart, connectEnd

SSL

```
connectEnd, secureConnectionStart //for security reasons we will not get information on
                                  //secureConnectionEnd
, requestStart
```

HTTP Request

requestStart,

Web Performance

`Navigation Timining API`

```
Server Wait

..responseStart


HTTP Response

responseStart, responseEnd, domLoading

HTML Parsing

domLoading, domInteractive

CSS/JS parsing

domInteractive, domContentLoadedEventStart, domContentLoadedEventEnd

Images ..

domContentLoadedEventEnd, domComplete, loadEventStart, loadEventEnd
```

Web Performance

```
Navigation Timining API
```

```
Time to first byte = responseStart - fetchStart (~ 600ms on an avg. conn.)
```

```
Bandwidth (download) = responseStart, responseEnd,
                       calculate based on the size of the file
```

```
DOM load time = domInteractive - fetchStart
```

```
Parsing and Rendering time = loadEventEnd - domLoading
```

```
Page Load time = domComplete - fetchStart
```